VARCHART
XNet

.NET Edition 5.2
User's and
Reference Guide

NETRONIC

VARCHART XNet .NET Edition

Version 5.2

User' s Guide

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

Last Revision: 27 April 2020

Table of Contents

1 Introduction

1.1 VARCHART XNet at a Glance

VARCHART XNet is an interactive chart component for visualizing graph related data. VARCHART XNet lets you display, edit and print your data in scenarios like business process modeling and work flow design. The powerful built-in layout algorithm and the integrated scheduling module are ideal for precedence network diagrams in project management. It can also be used to build class and entity-relationship diagrams. Benefit from the wide range of design options and implement an initial graphical representation of your data in a matter of minutes.

> Short Feature Overview

- **Annotation Boxes**

In addition to the graph structure, information boxes that hold annotations and pictures can be positioned freely in the chart.

- **Automatic Layout**

The automatic layout of nodes maintains clarity in large and complicated structures.

- **Calendar**

Calendars consists of a continuous sequence of work and non-work periods. You can compose individual calendars for nodes. Calendars are used for scheduling to take work free periods into account.

- **Clustering**

To handle larger graph structures comfortably, nodes can be grouped in clusters. They are areas which can be collapsed and expanded. You can define the appearance and the information of the substitute node to be displayed.

- **Data interface**

Use the flexible data interface in order to adapt easily to existing data structures. Different tables that hold selected data fields can be defined as in a relational data model and can be linked to one another. A CSV import filter is available for reading the application data. The data fields of a data table can be used in filters and maps and to annotate nodes.

- **Flow Direction**

Adjust the graph flow direction to your needs: top to bottom or left to right.

- **Filter**

Filters let you select nodes or links that fulfill the criteria defined, e.g. in order to highlight nodes in the diagram.

- **Graphics Export**

Save the chart in your preferred graphics format: PNG, BMP, EMF, GIF, TIF, JPG.

- **In-Flow Grouping**

In-flow grouping places nodes in a chronological or in a criteria based order. It is applicable in both the x direction and the y direction.

- **Intuitive interactions**

Adapt the visualization on the screen and change the basic data based on user interactions. For example, nodes and links can be moved or copied by drag & drop.

- **Language Support**

The product and the documentation are available in English and German. In addition, in run time mode each text item in the chart can be replaced by a term of your choice in any language. Unicode characters are supported. The characters of all languages can be used simultaneously and independently of the operating system that the application is run on.

- **Legend**

The legend of a chart can be positioned outside the chart and becomes visible in prints and exported charts. The layout of the legend can be put in the shape of a well structured matrix.

- **Links**

Annotate links, select port symbols and choose the appropriate appearance for your links. In addition, the four different link types (Start-Start, Start-Finish, Finish-Start, Finish-Finish) can be displayed graphically.

- **Navigation Window**

Navigation in the chart is easy due to the integrated worldview.

- **Node appearance**

Represent your nodes through different shapes, colors, color gradients and your own bitmaps. Nodes can be marked and supplemented by individual tooltip texts. The appearance of the nodes can be dynamically based on your data by creating and applying filters and assignment tables. The labeling of nodes can consist of different data fields that may be positioned within or beyond the node limits.

- **Node position**

If desired, the position of a node can also be set manually. Also, positions automatically set can be retrieved from data fields.

- **Printing**

Select the page layout and preview it in the integrated print preview. Specify diagram parts to be repeated on each page, and set the number of pages on which it should be printed.

- **Property Pages**

For any important object a property page exists which dramatically reduces the amount of code to be written.

The property pages allow you to intuitively customize nearly every aspect of the component and the powerful API offers further options at run time. Events let your application react to your users’ interactions (for example, to validate data) in a certain way.

- **Scheduling**

The integrated scheduler lets you calculate the Early Start, Early Finish, Late Start, Late Finish, Total Float and Free Float. The calculation is based on the duration of the activities, their logical dependencies and the project start or project end.

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 2010 and upwards 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 Service Pack 3 upwards can be used.

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

By default, the control and ist associated files will be stored below the folder

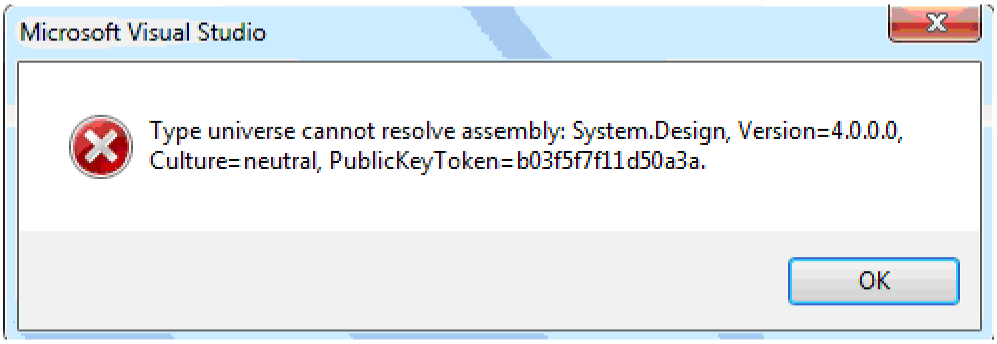
c:\Program Files\NETRONIC (32bit-Windows) or

c:\Program Files (x86)\NETRONIC (64bit-Windows).

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; in other development environments the procedure is similar:

1. In Visual Studio create a new project of the type **Windows 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.XNet.dll** from the installation directory. After confirming by **OK** the icon of VARCHART XNet .NET will be added to the toolbox.
4. Important for the users of **Visual Studio 2010**: **Before** you drag the control to the form, you have to change the target framework from **.NET Framework Client Profile** to **.NET Framework 4** in the **Application Settings** (C#) or **Advanced Compiler Settings** (VB) since the former lacks the System.Design.dll, which is required by the property pages at design-time. If you don't change the framework, the following error message will pop up when you try to drag the control onto the form:



Type universe cannot resolve assembly: System.Design, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a.

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

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

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

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

1.3 Licensing

1.3.1 Developer Licenses

For licensing the VARCHART XNet .NET control please click the icon and draw the control onto the form.

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

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

By clicking on the button **Request license information from NETRONIC** the according dialog will open.

Three items are needed for the registration:

- the license number
- your name
- the name of the company

Please fill in the information needed. You will find the license number "NXnnnn" on the delivery note of your order or on the wrapper of your CD.

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

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

1.4 Delivery

If you wish to deliver to a customer an application developed by yourself having used XNet .NET, the following files need to be delivered with the application. All other files belonging to VARCHART XNet .NET are only used during the phase of development and must **not** be passed on to your customers.

> Framework .NET 2.0/3.0/3.5

- **In the according processor version for x86 or x64**

NETRONIC.XNet.dll

NETRONIC.XNetd.dll (if you want to use the German version)

NETRONIC.XNetc.dll (if you want to use the Chinese version)

mfc80u.dll

mfcm80u.dll

msvcp80.dll

msvcr80.dll

In order to install the libraries *mfc80u.dll, msvcp80.dll, mfcm80u.dll* and *msvcr80.dll* please use the setup file *vcredist_vs2005sp1_x86.exe* or *vcredist_vs2005sp1_x64.exe* respectively. You will find these files in the installation folder of XNet .NET in the subfolder **redist**.

For further information please see:

msdn2.microsoft.com/en-us/library/ms235285(VS.80).aspx.

> Framework .NET 4.0/4.5

- **In the according processor version for x86 or x64**

NETRONIC.XNet.dll

NETRONIC.XNetd.dll (if you want to use the German version)

NETRONIC.XNetc.dll (if you want to use the Chinese version)

mfc100u.dll

mfcm100u.dll

msvcp100.dll

msvcr100.dll

In order to install the libraries *mfc100u.dll*, *msvcp100.dll*, *mfcm100u.dll* and *msvcr100.dll* you can either copy them directly to the Windows system directory or you can use the setup file *vcredist_vs2010_x86.exe* or *vcredist_vs2010_x64.exe* respectively. You find these files in the installation folder of XNet .NET in the subfolder **redist**.

VARCHART XNet .NET can be run on the the below platforms:

- Windows 8
- Windows 7
- Windows Server 2008
- Windows Vista
- Windows Server 2003
- Windows XP SP3 or later

using the .NET framework 2.0 at least (for further information, see

msdn.microsoft.com/netframework/technologyinfo/sysreqs/default.aspx)

Tip:

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.

1.5 Usage of the German version

The VARCHART XNet .NET Edition is available in German and in English. When installing the German version, the resource assembly NETRONIC.XNet**d**.dll is copied to the installation directory in addition to the control assembly NETRONIC.XNet.dll.

Usage at design time

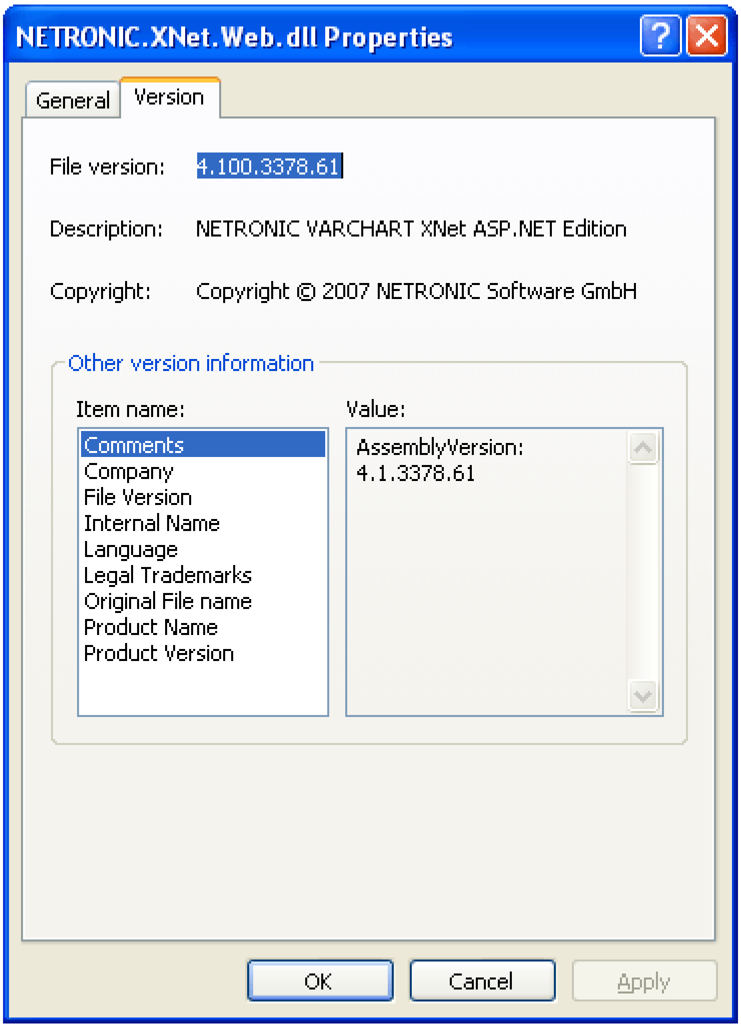
If the **Regional Options** (Control Panel, Regional and Language Options) were set to **German**, the resource assembly is loaded from the installation directory and the German dialogs and property pages are available at design time.

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**).



File version: 4.100.3378.61

Description: NETRONIC VARCHART XNet ASP.NET Edition

Copyright: Copyright © 2007 NETRONIC Software GmbH

Other version information

Item name:	Value:
Comments Company File Version Internal Name Language Legal Trademarks Original File name Product Name Product Version	AssemblyVersion: 4.1.3378.61

1.6 Support and Advice

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

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

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

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

NETRONIC Software GmbH
Pascalstr. 15
52076 Aachen
Germany
Phone +49-2408-141-0
Fax +49-2408-141-33
Email support@netronic.com
www.netronic.com

...by the way: you may order our support and maintenance service 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 XNet releases for development and runtime versions.

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

2 Tutorial

2.1 Overview

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

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

- **Property pages and dialogs**

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

- **Elements of the user interface**

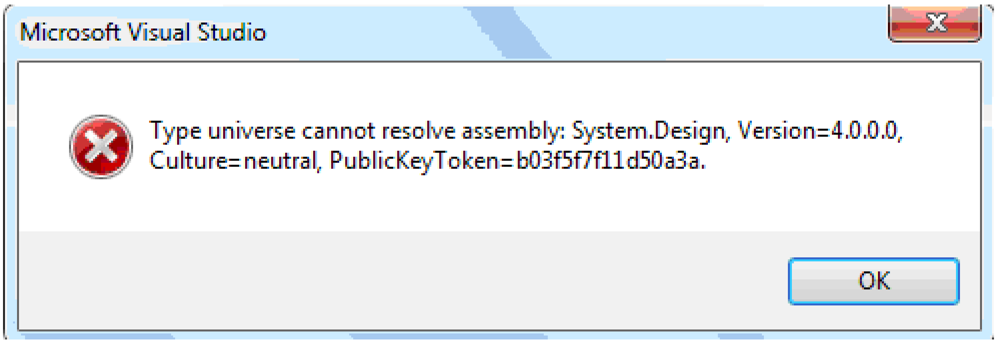
  In the chapter quoted above the interactions which are available in the diagram are described. Details of the user interface can be fitted or changed individually.

- **API Reference**

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

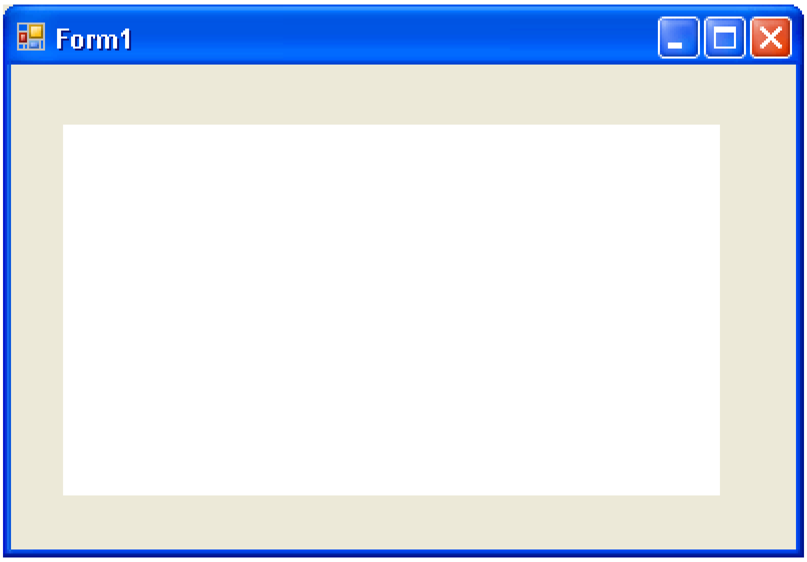
2.2 Placing the Control on a Form

Important for the users of **Visual Studio 2010!: Before** you drag the control to the form, you have to change the target framework from **.NET Framework Client Profile** to **.NET Framework 4** in the **Application Settings** (C#) or **Advanced Compiler Settings** (VB) since the former lacks the System.Design.dll, which is required by the property pages at design-time. If you don't change the framework, the following error message will pop up when you try to drag the control onto the form:



Type universe cannot resolve assembly: System.Design, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a.

To place the VARCHART XNet control on the form, please select its icon in the toolbox and draw a frame at the position in the form where you want it to appear. The size of the VARCHART XNet control can be readjusted by mouse.



2.3 Automatic Scaling of VARCHART XNet

If you wish the bottom and right-hand side of the VARCHART XNet control to be adjusted to the full size of the window during runtime, add the below code:

Example Code VB.NET

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    VcNet1.Width  = ClientSize.Width - VcNet1.Left
    VcNet1.Height = ClientSize.Height - VcNet1.Top
End Sub
```

Example Code C#

```
private void Form1_Load(object sender, System.EventArgs e)
   {
   vcNet1.Width  = ClientSize.Width - vcNet1.Left;
  vcNet1.Height = ClientSize.Height - vcNet1.Top;
   }

Private void Form1_Resize(object sender, System.EventArgs e)
   {
   vcNet1.Width  = ClientSize.Width - vcNet1.Left;
   vcNet1.Height = ClientSize.Height - vcNet1.Top;
   }
```

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.

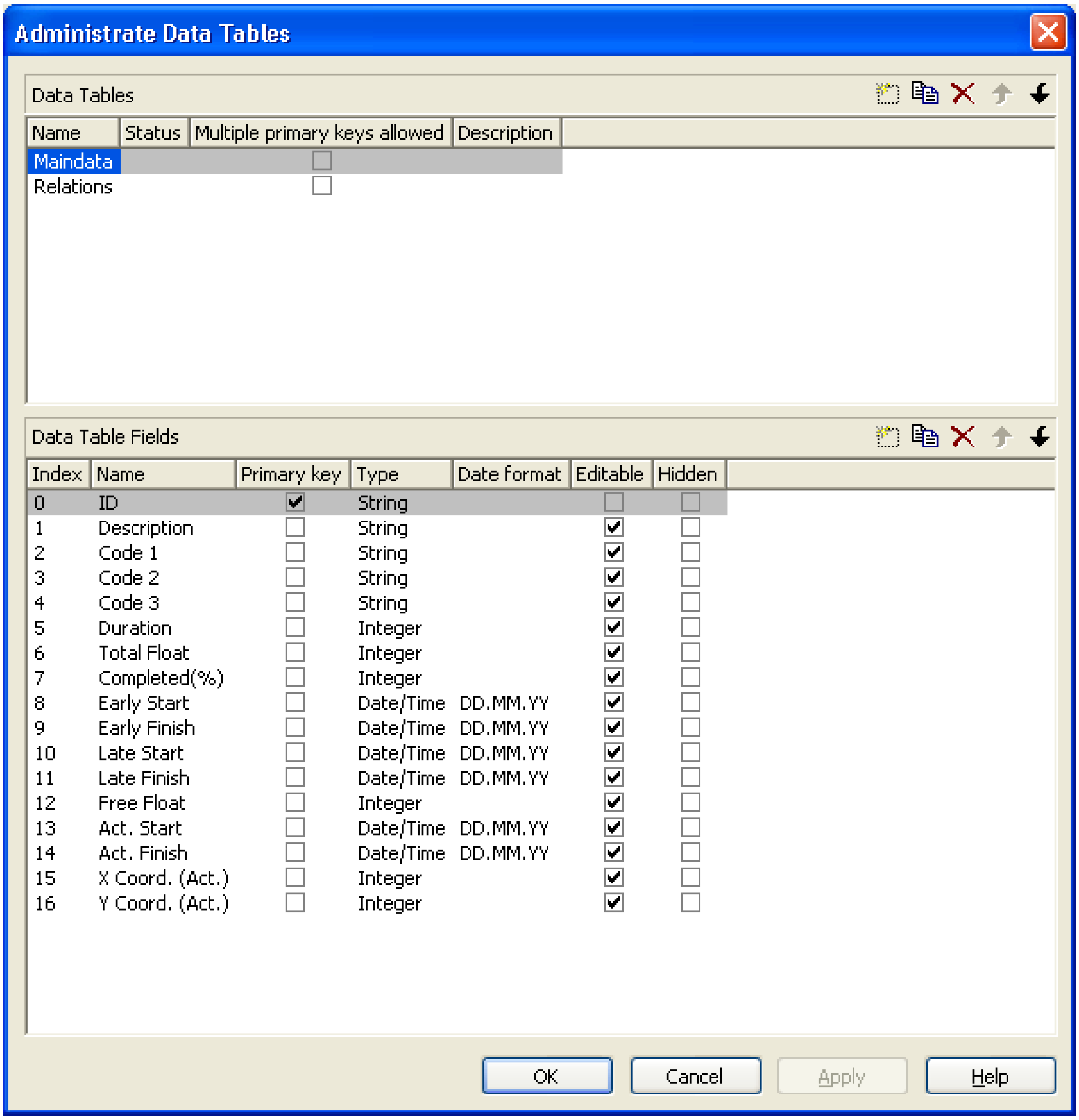
VB: Imports NETRONIC.XNet

C#: using NETRONIC.XNet;

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

2.4 Preparing the Interface

Prepare the interface now by defining the data fields of the **Maindata** and the **Relations** table. For this, please open the **Administrate Data Tables** dialog.



Data Tables

Name	Status	Multiple primary keys allowed	Description
Maindata		□	
Relations		□	

Data Table Fields

Index	Name	Primary key	Type	Date format	Editable	Hidden
0	ID	☑	String		□	□
1	Description	□	String		☑	□
2	Code 1	□	String		☑	□
3	Code 2	□	String		☑	□
4	Code 3	□	String		☑	□
5	Duration	□	Integer		☑	□
6	Total Float	□	Integer		☑	□
7	Completed(%)	□	Integer		☑	□
8	Early Start	□	Date/Time	DD.MM.YY	☑	□
9	Early Finish	□	Date/Time	DD.MM.YY	☑	□
10	Late Start	□	Date/Time	DD.MM.YY	☑	□
11	Late Finish	□	Date/Time	DD.MM.YY	☑	□
12	Free Float	□	Integer		☑	□
13	Act. Start	□	Date/Time	DD.MM.YY	☑	□
14	Act. Finish	□	Date/Time	DD.MM.YY	☑	□
15	X Coord. (Act.)	□	Integer		☑	□
16	Y Coord. (Act.)	□	Integer		☑	□

Please select the **Maindata** table, where you can define the data fields of a node record.

To adapt your interface for the tutorial, please replace "ID" by "Number" and select the data type **Integer**. Disable the option **editable** to avoid overwriting the number in the **Edit Data** default dialog.

The name can be edited via a double-click or when it is marked via a left-click. The data type can be selected from a select box that appears after clicking on the type.

You can create a new data field by clicking and editing the new field in the last row.

Fields of the Maindata table:

Field	Name	Type
0	Number	Integer
1	Structure code	String
2	Level	Integer
3	Parent node	String
4	Name	String
5	Group code	String
6	Code1	Integer
7	Group name	String
8	Duration	Integer
9	Float	Integer
10	completed (%)	Integer
11	Early start	Date/Time
12	Early finish	Date/Time
13	Late start	Date/Time
14	Late finish	Date/Time
15	Free float	Integer
16	Calculated Start	Date/Time
17	Calculated Finish	Date/Time
18	X-Coord. (node)	Integer
19	Y-Coord. (node)	Integer
20	Auxiliary node	String

For the fields "Calculated Start" and "Calculated Finish" please tick the check box **Hidden** to hide them from the user in the **Edit Data** dialog.

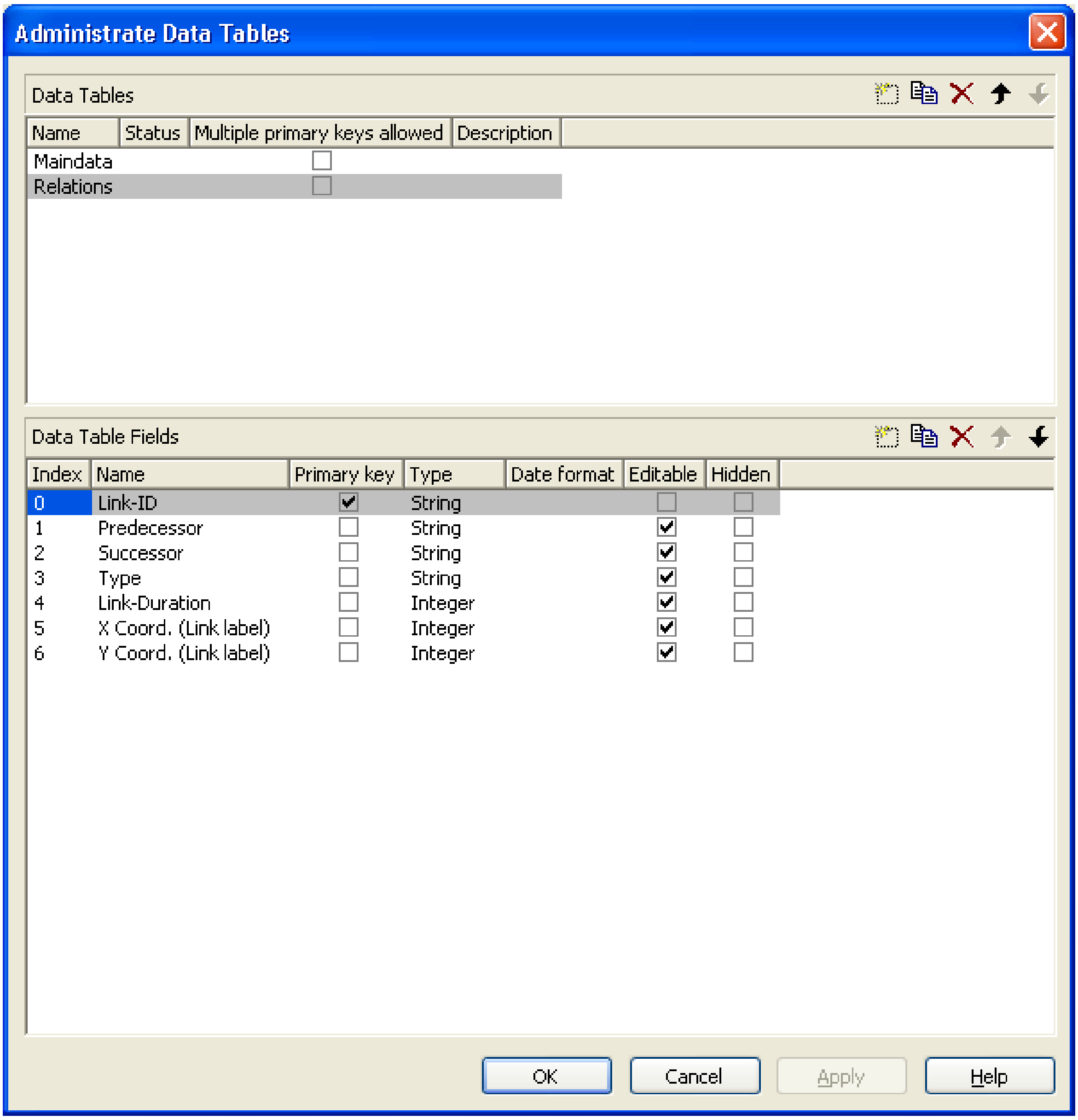
The **Date/Time** fields allow to enter a format. Please select "DD.MM.YY".

Now define a primary key for the nodes. Please select **Primary key** for the field "Number".

Please change to the **Relations** table.

Fields of the Relations table:

Index	Name	Type
0	Link ID	String
1	Predecessor	Alphanumeric
2	Successor	Alphanumeric
3	Link type	Alphanumeric
4	Time interval	Integer
5	X-Coord. (Link label)	Integer
6	Y-Coord. (Link label)	Integer



Administrate Data Tables

Data Tables

Name	Status	Multiple primary keys allowed	Description
Maindata		□	
Relations		□	

Data Table Fields

Index	Name	Primary key	Type	Date format	Editable	Hidden
0	Link-ID	☑	String		□	□
1	Predecessor	□	String		☑	□
2	Successor	□	String		☑	□
3	Type	□	String		☑	□
4	Link-Duration	□	Integer		☑	□
5	X Coord. (Link label)	□	Integer		☑	□
6	Y Coord. (Link label)	□	Integer		☑	□

Now define a primary key for the links. Please select **Primary key** for the field "Link ID". Disable the option **editable** for this field.

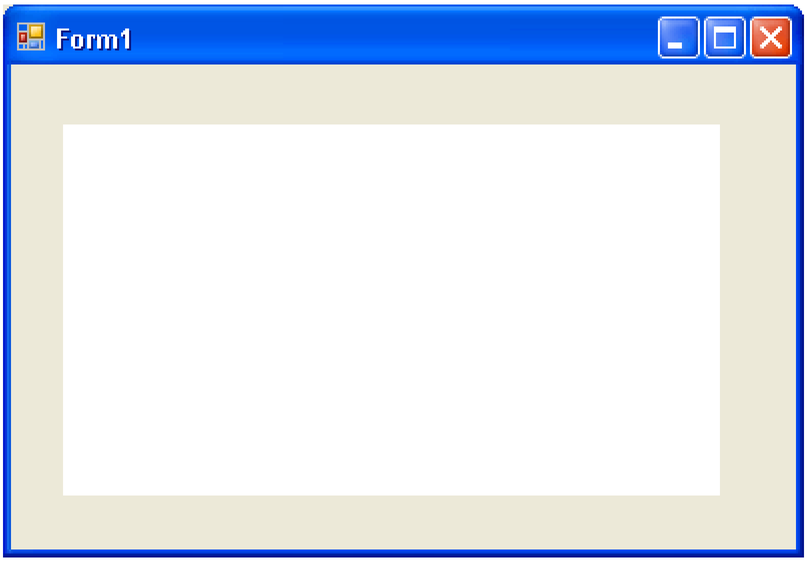
Note: A name that already exists in the table will not be accepted. Instead, a message appears in which you are requested to define another name.

By clicking on the **Apply** button your modifications will be saved.

They will as well be stored by clicking on the **OK** button and by changing to a different property page, thus being available to other property pages immediately.

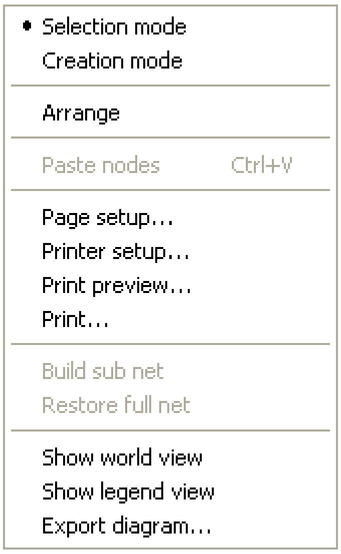
2.5 Your First Run

Start the program via **Run – Start**, the function key F5 or the according icon (▶). The generated form shows an empty chart.

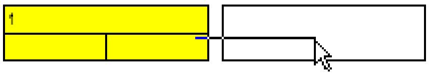


> Creating Nodes and Links

There are two modes available at run time. The **Selection mode** and the **Creation Mode**. Nodes and links can be generated in Creation mode only. To change modes, press the right mouse button on an empty area in the diagram and select the menu item **Creation mode** from the context menu popping up.

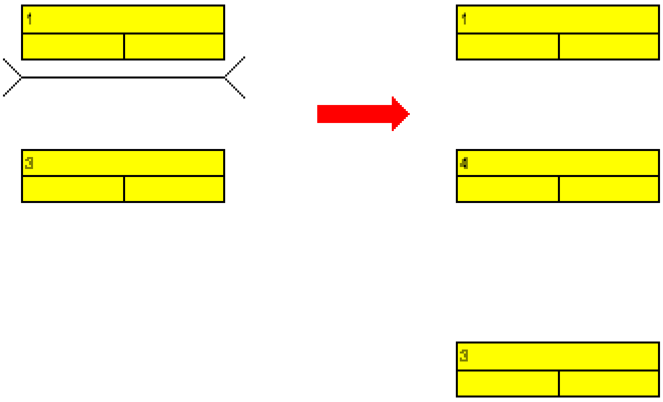


In Creation mode the pointer will transform into a rectangular frame. You can create a node by pressing the left mouse button in an empty area of the diagram. A link you can generate by dragging the mouse from one node to another while keeping the left mouse button pressed. While dragging, the pointer looks like an arrow that draws a line.



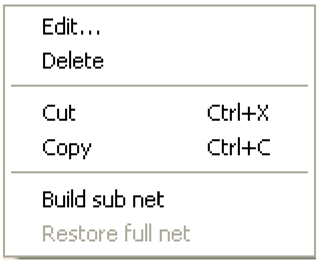
As soon as you release the mouse button, the link will occur. If you drag the mouse between a node and an empty place, both a node and a link will be generated.

If you place the mouse between two nodes that are close together, the pointer will assume the shape of a bone, i.e. a line with an inverted arrow tip at each of its ends. If you click now, the two nodes will shift apart and a node will be inserted in the gap.

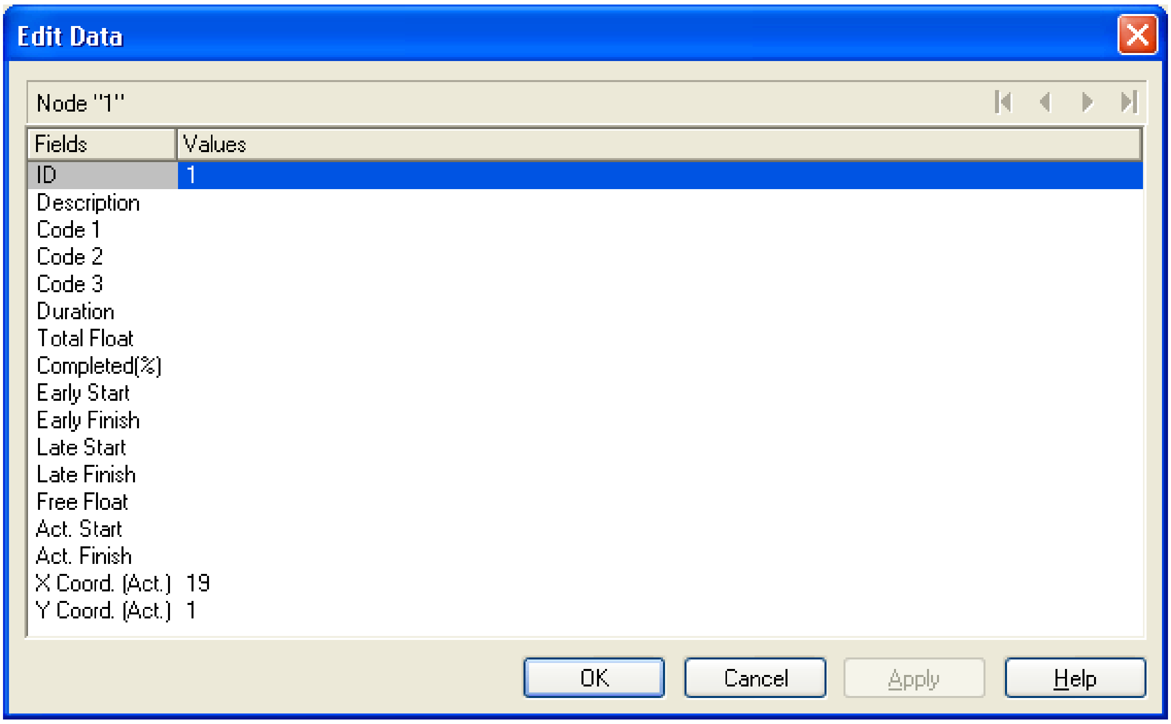


> Editing Nodes

To edit a node, double-click on it. The **Edit Data** dialog will appear. Alternatively, you can click with the right mouse button on the node. The context menu appearing will offer options for editing, cutting, copying or deleting nodes.



Now open the **Edit Data** dialog for a node. You will find the data fields that you defined in the **Administrate Data Tables** dialog. The data fields that were defined as **hidden** will not appear in this dialog. The data fields that were defined as **not editable** cannot be edited in this dialog.



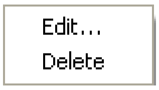
Edit Data

Node "1"

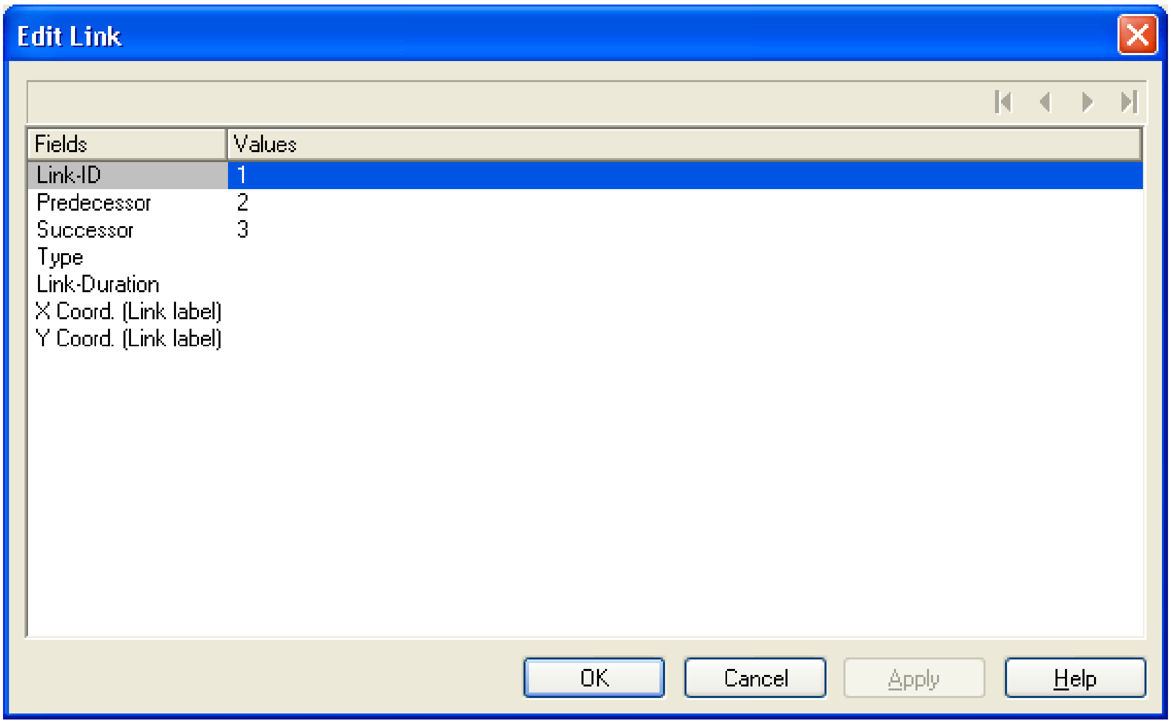
Fields	Values
ID	1
Description	
Code 1	
Code 2	
Code 3	
Duration	
Total Float	
Completed(%)	
Early Start	
Early Finish	
Late Start	
Late Finish	
Free Float	
Act. Start	
Act. Finish	
X Coord. (Act.)	19
Y Coord. (Act.)	1

> Edit Links

You can edit a link either by selecting the item **Edit** from the context menu or by a double-click on the link, popping up the **Edit Link** dialog.



Link context menu



Edit Link

Fields	Values
Link-ID	1
Predecessor	2
Successor	3
Type	
Link-Duration	
X Coord. (Link label)	
Y Coord. (Link label)	

This dialog allows to edit the data of the link.

> Moving Nodes and Links interactively

Nodes and links can be moved with the mouse. For this, switch to Selection mode and place the pointer onto a node or a link. The pointer will turn into a little square and four arrows. You can now move the desired node or link with the mouse by keeping the left mouse button pressed. When moving a node, joining links will follow automatically.

> Back to Design Mode

Finish the first run by closing the form.

2.6 Loading Data from a File

To feed data into VARCHART XNet, load the file *tutorial.net*. You can do this automatically on the start. *Tutorial.net* is a CSV-formatted file, that your interface is customized to (if you wish to modify this, please see "Tutorial: Preparing the Interface"). To load the file, react to the **Form_Load** event:

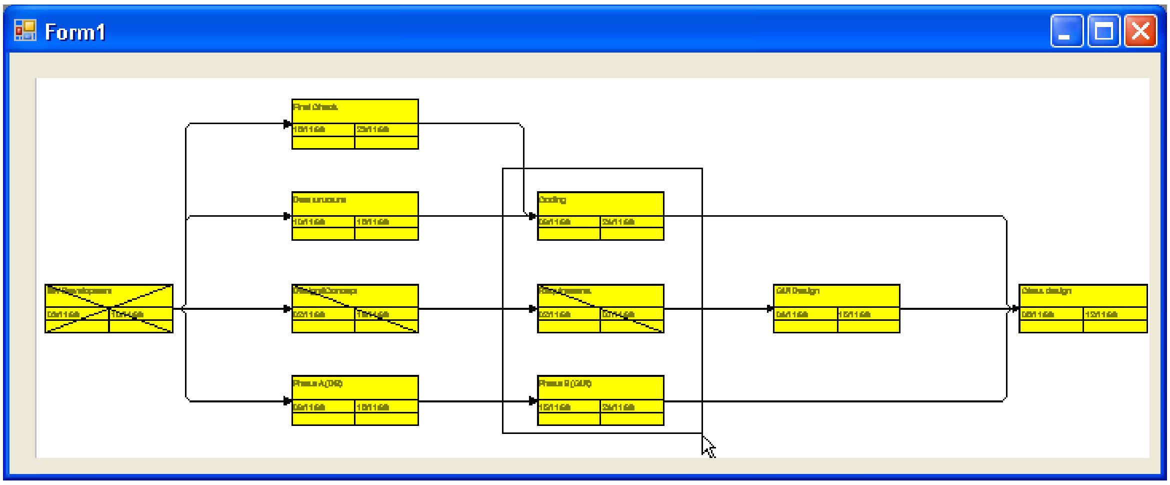
Example Code

```
Private Sub Form_Load()
    VcNet1.Open "C:\Programs\Varchart\xnet\tutorial.net"
End Sub
```

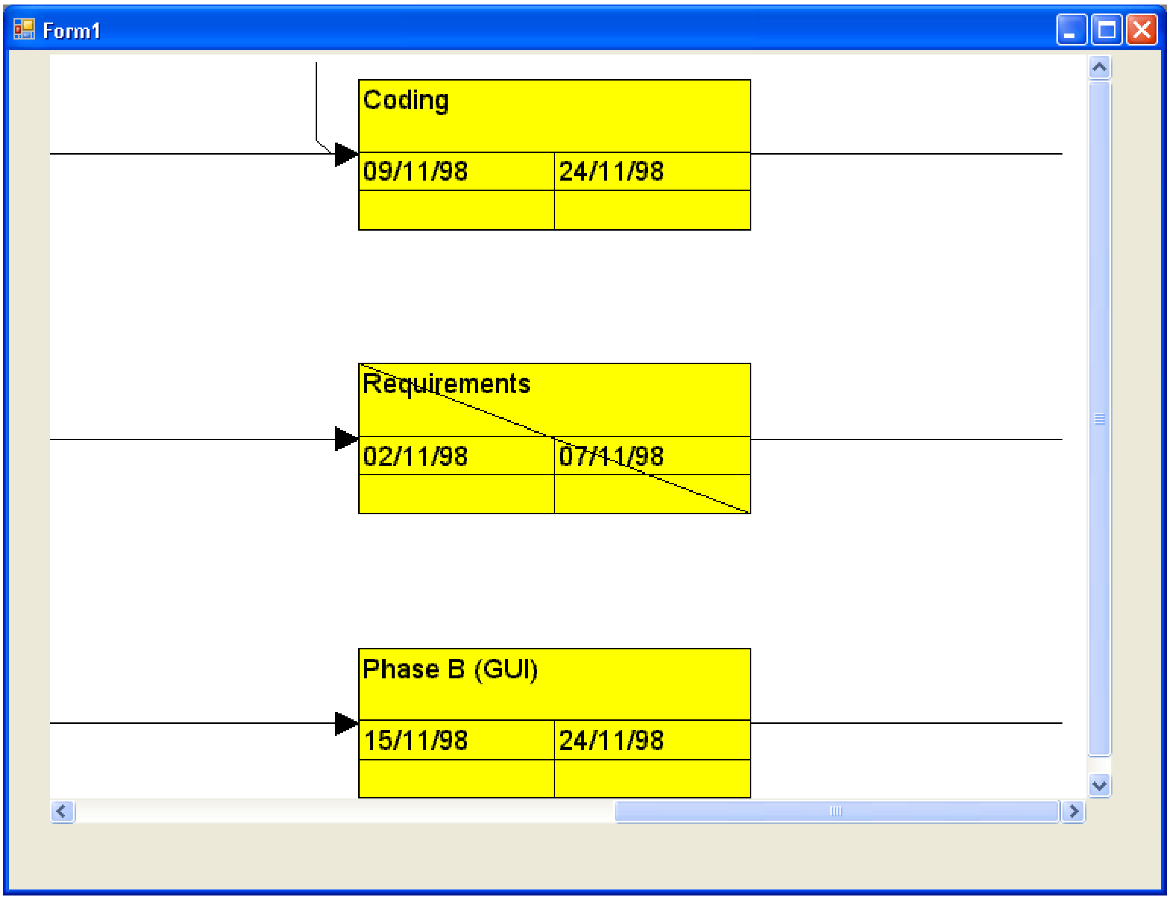
The path depends on the installation of your program. Please save the project now. If you start the program, the nodes and links of the project will be displayed.

VARCHART XNet will display a network diagram completely.

You can mark a section of your diagram and display it in full screen size. Mark the section to be zoomed, keep the left mouse button depressed and in addition press the right mouse button.



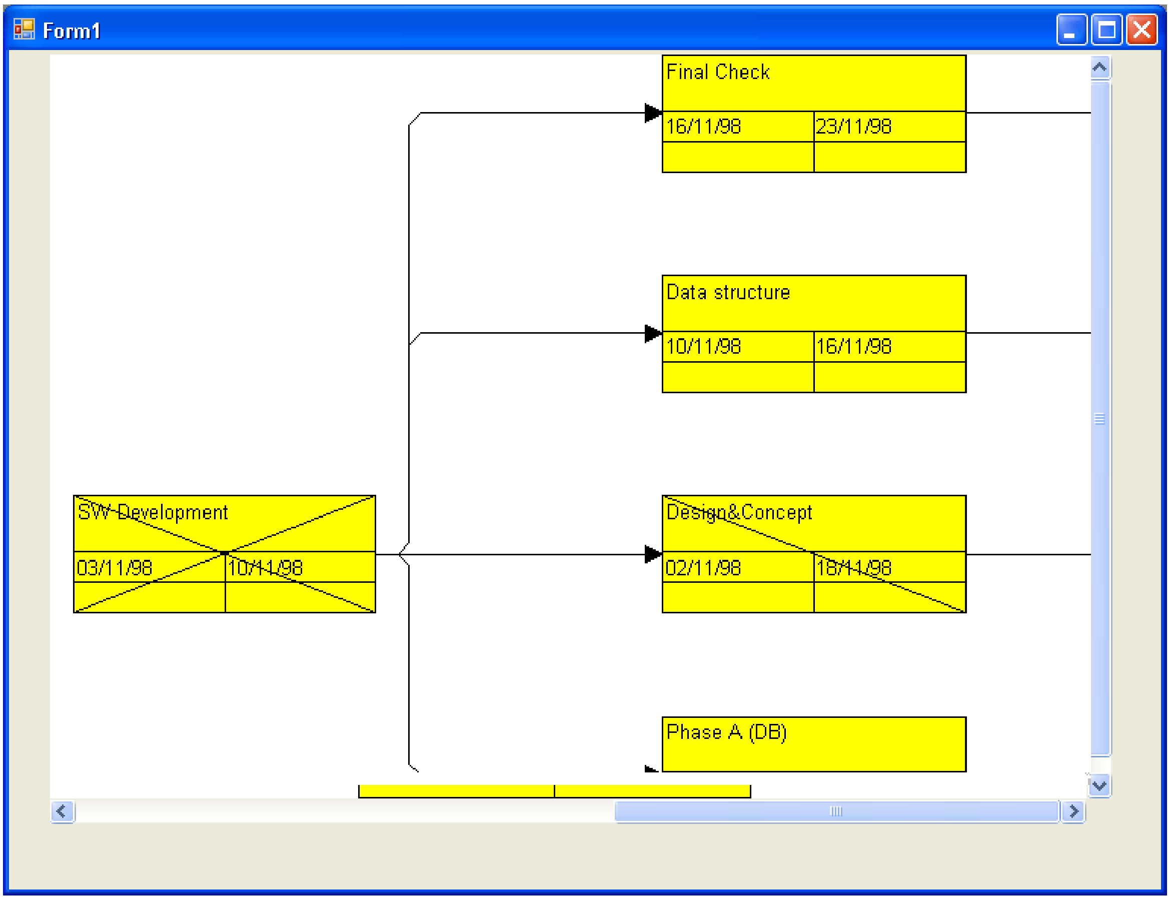
The marked section will be zoomed to full screen size. Use the scrollbars to move through the section and to other parts of the diagram magnified to the same scale.



Return to design mode. Add the code below to set vertical and horizontal scroll bars. Whether or not scrollbars appear depends on the zoom factor selected.

Example Code

```
Private Sub Form_Load()
    VcNet1.Open "C:\Programs\Varchart\xnet\tutorial.net"
    VcNet1.Zoomfactor = 150
End Sub
```

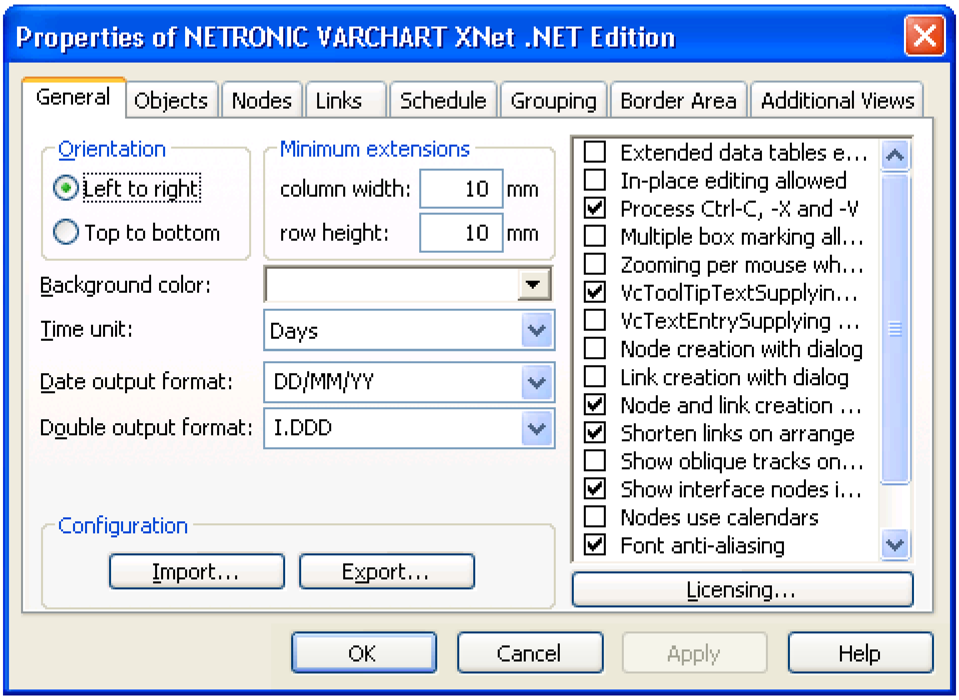


Return to design mode. If you want VARCHART XNet to completely cover the form, please verify the following:

- Make sure that the properties **Top** and **Left** are set to 0. This will position VARCHART XNet to the top left corner of the form.
- Set the VARCHART XNet properties **Width** and **Height** to the form values **ScaleWidth** and **ScaleHeight**. In case you are having VARCHART XNet rescaled automatically, as described above, the latter becomes obsolete.

2.7 Setting the Orientation of a Diagram

On the **General** property page, you can adjust the basic and general settings of VARCHART XNet.



Orientation
- ● Left to right
- ○ Top to bottom

Minimum extensions
- column width: 10 mm
- row height: 10 mm

Background color:

Time unit: Days

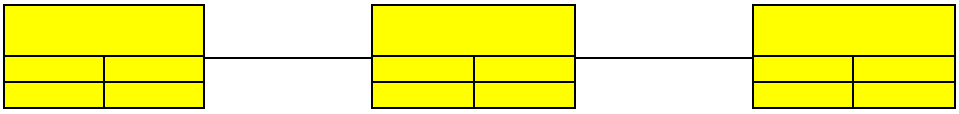
Date output format: DD/MM/YY

Double output format: I.DDD

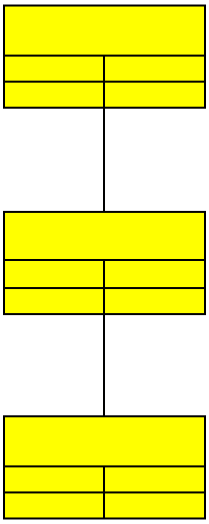
Configuration

- □ Extended data tables e...
- □ In-place editing allowed
- ☑ Process Ctrl-C, -X and -V
- □ Multiple box marking all...
- □ Zooming per mouse wh...
- ☑ VcToolTipTextSupplyin...
- □ VcTextEntrySupplying ...
- □ Node creation with dialog
- □ Link creation with dialog
- ☑ Node and link creation ...
- ☑ Shorten links on arrange
- □ Show oblique tracks on...
- ☑ Show interface nodes i...
- □ Nodes use calendars
- ☑ Font anti-aliasing

At first, please select via the **Orientation** whether the nodes should be directed from left to right or from top to bottom. Try both orientations on the sample file *tutorial.net*, which you can do in design mode. Browse for the file *tutorial.net* in the field **Temporary data file** further down.



Left-to-right orientation



Top-to-bottom orientation

To enable creating nodes, the check box **Allow creation of nodes and links** needs to be ticked. To make the program produce an empty chart when starting, transform the corresponding source code line into a comment:

Example Code

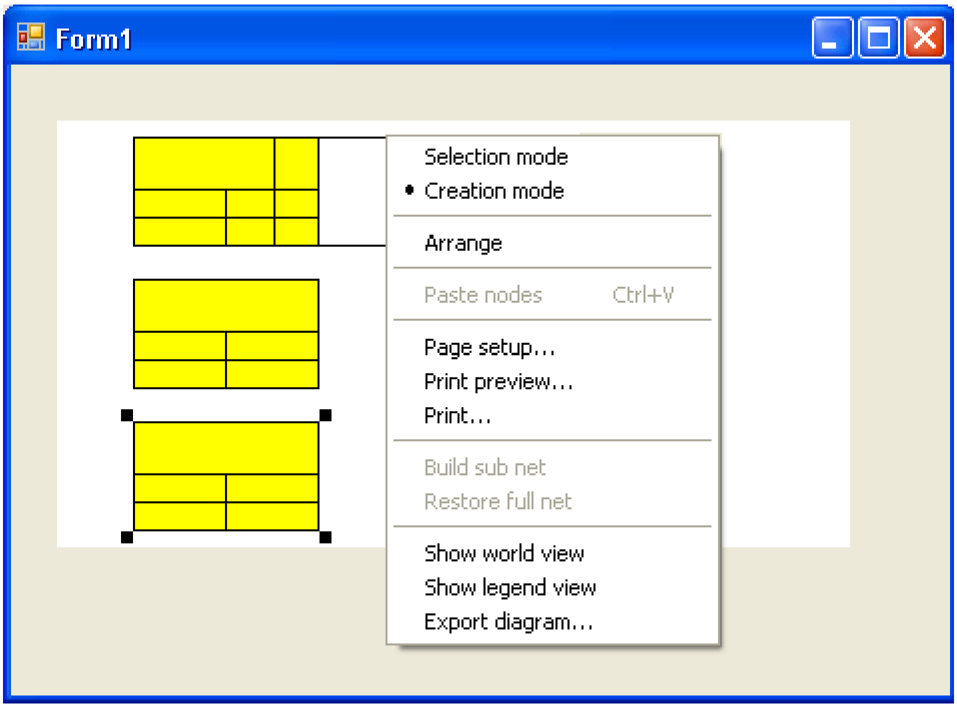
```
Private Sub Form_Load()
    ' VcNet1.Open "C:\Programs\Varchart\xnet\tutorial.net"
    VcNet1.Zoomfactor = 100
End Sub
```

Please start the program again. An empty network diagram will appear. Press the right mouse button to make the context menu appear and select the creation mode. Generate some nodes in a row and a column and connect them by links. Make the context menu appear by pressing the right mouse button on an empty spot of the diagram and select the menu item **Arrange**. VARCHART XNet will arrange the nodes according to the orientation set.

2.8 Generating and Editing Nodes and Links

On the **General** property page by the option **Allow creation of nodes and links** you can enable the user to create new nodes interactively via a mouse click. If in addition you tick the options **Edit new nodes** and **Edit new links**, the **Edit Data** dialog will open as soon as the mouse button has been released by the user. The data of the node or link is displayed and you can edit them.

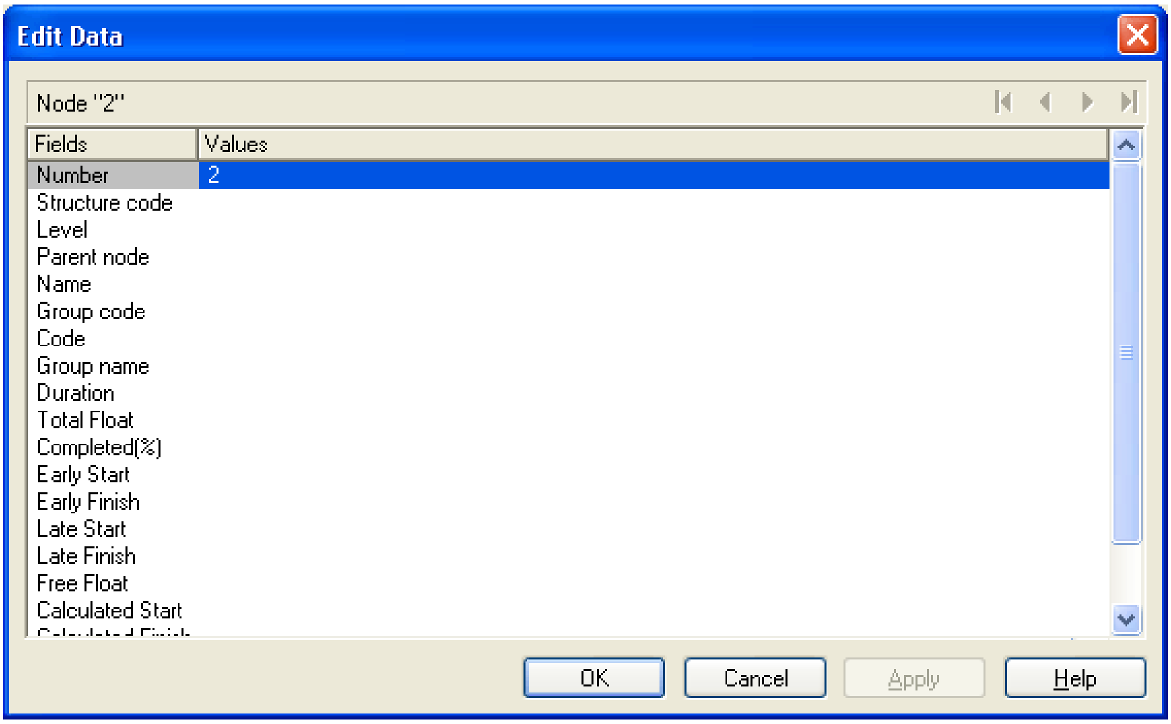
On the **General** property page, please activate the option **Allow creation of nodes and links** and deactivate the **Edit new nodes** and **Edit new links** check boxes. Start the application by the F5 key, open the context menu via the right mouse button and select **Creation Mode**. Then use the left mouse button to click on an empty space in the diagram. Each mouse click will generate another node.



If you deactivate the option **Node and link creation allowed**, the user will be unable to generate nodes and links, even when the creation mode is switched on. Nodes and links can then only be loaded via the API.

Please activate the options **Node creation with dialog** and **Link creation with dialog** now.

Start the application and change to creation mode. Click the right mouse button and choose the **Edit** menu item to open the **Edit Data** dialog where the data fields that were defined in the **Administrate Data Tables** dialog (not defined as hidden) are displayed. You can edit the data fields and the values of the node record. When you click on **OK**, the node will be generated from the values set.



Fields	Values
Number	2
Structure code	
Level	
Parent node	
Name	
Group code	
Code	
Group name	
Duration	
Total Float	
Completed(%)	
Early Start	
Early Finish	
Late Start	
Late Finish	
Free Float	
Calculated Start	
Calculated Finish	

Please generate some nodes and links as described in the chapter "Tutorial: The First Run". You can edit nodes in immediate sequence.

Please mark some nodes by simultaneously keeping the Ctrl key depressed. Then click the right mouse button on one of the marked objects. This will open the context menu of the node. Click on the **Edit** menu item to open the **Edit Data** dialog where you can edit the data of all nodes right away.

In the head line of the **Edit Data** dialog you will find the ID of the node, as well as the number of the current node out of the total number of nodes being edited (Activity n of m).

When opening the dialog, the data and values of the first node are displayed. With the help of the arrow buttons you can navigate in the nodes.

Please close the form now and return to design mode.

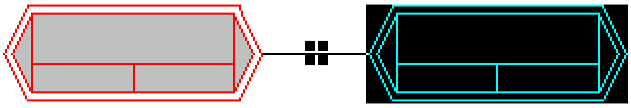
2.9 Marking Nodes and Links

On the **Nodes** and the **Links** property pages you can set a pattern to mark nodes and links, respectively. Just select an option from the **Marking type** select box.

Start the program, switch to the creation mode and generate some nodes and links for marking.

You can mark nodes or links by clicking on them with the left mouse button. By simultaneously pressing the Ctrl key you can mark and toggle several nodes or links. By marking a single object and then using the Shift key to mark a second one, all objects between them will be marked.

Try different options of marking nodes and links. The picture below shows marking a node by inverted colors and marking a link by pickmarks:

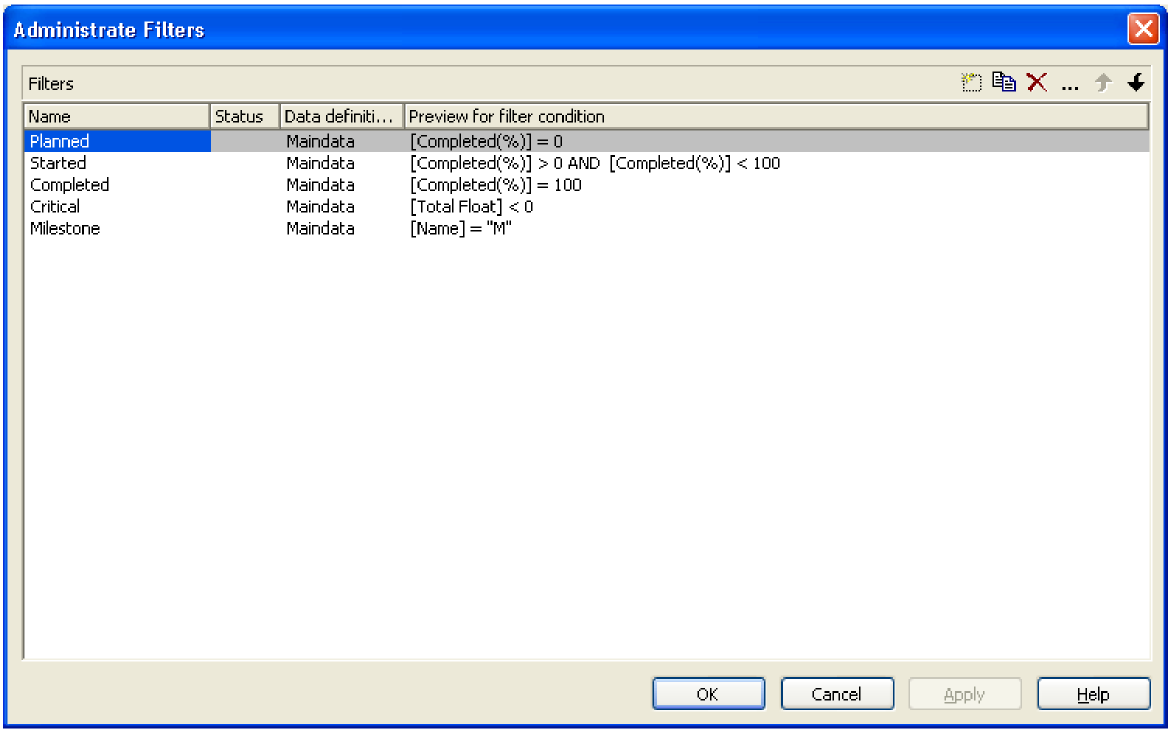


2.10 Setting Filters for Nodes

A filter consists of criteria to select for defined data, for example for data of nodes and links.

When using a filter in a node appearance, only those nodes will show the features defined in the appearance that match the filter conditions.

Please click on the **Filters** button of the **Objects** property page to open the **Administrate Filters** dialog box. Here you can rename create, copy, edit or delete filters.



Name	Status	Data definiti...	Preview for filter condition
Planned		Maindata	[Completed(%)] = 0
Started		Maindata	[Completed(%)] > 0 AND [Completed(%)] < 100
Completed		Maindata	[Completed(%)] = 100
Critical		Maindata	[Total Float] < 0
Milestone		Maindata	[Name] = "M"

> Buttons in the "Administrate Filters" dialog box

Add filter

Copy filter

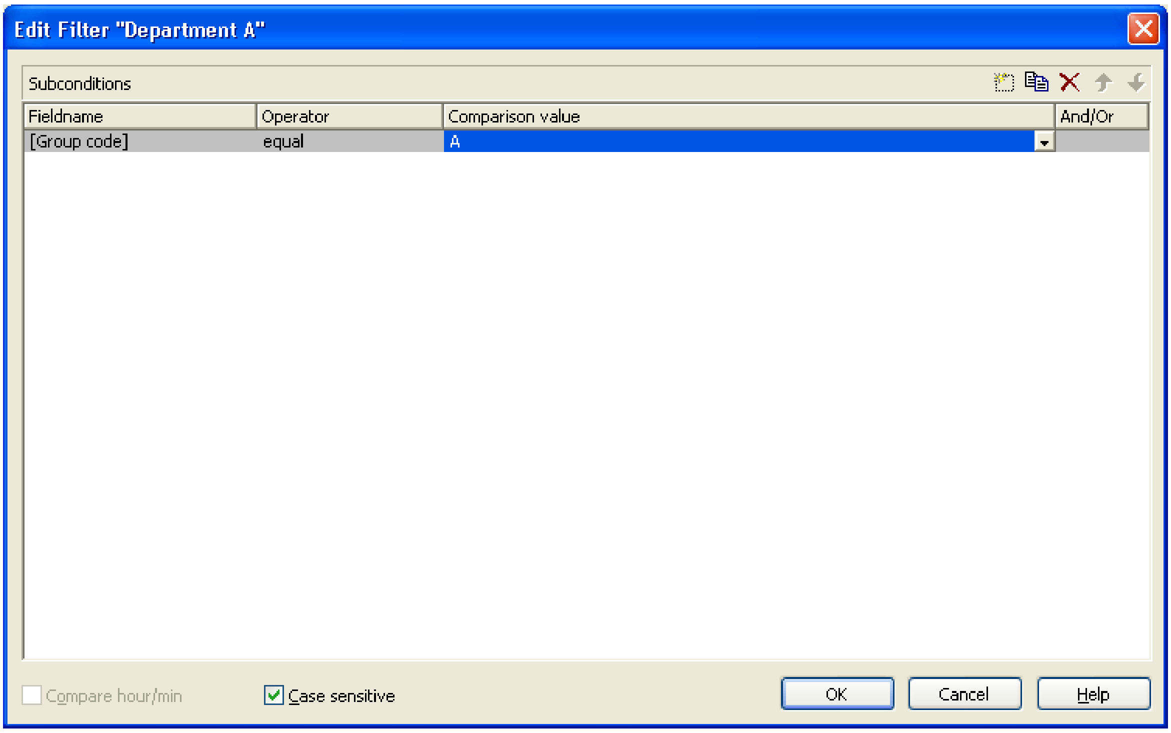
✕ Delete filter

... Edit filter

> Creating and editing filters

Now create new filters and edit them. Click on the **Add filter** button. The new filter appears at the end of the list. Rename it to "Department A".

Now edit the new filter. Click on the **Edit filter** button to reach the **Edit Filter** dialog box. Specify the following:



Edit Filter "Department A"

Subconditions

Fieldname	Operator	Comparison value	And/Or
[Group code]	equal	A	

□ Compare hour/min ☑ Case sensitive

The head line indicates the name of the current filter.

The **Code name** field displays the data field whose value is compared with the **Comparison value**. Please select the field "Group name".

The **Operator** field displays the current operator. The type of operator available depends on the type of data field selected. Please select the operator "equal" now.

The entry in the **Comparison value** field is a value that the **Code name** entry will be compared with. Therefore it needs to be of the same data type as the **Code name** entry. Please select "A".

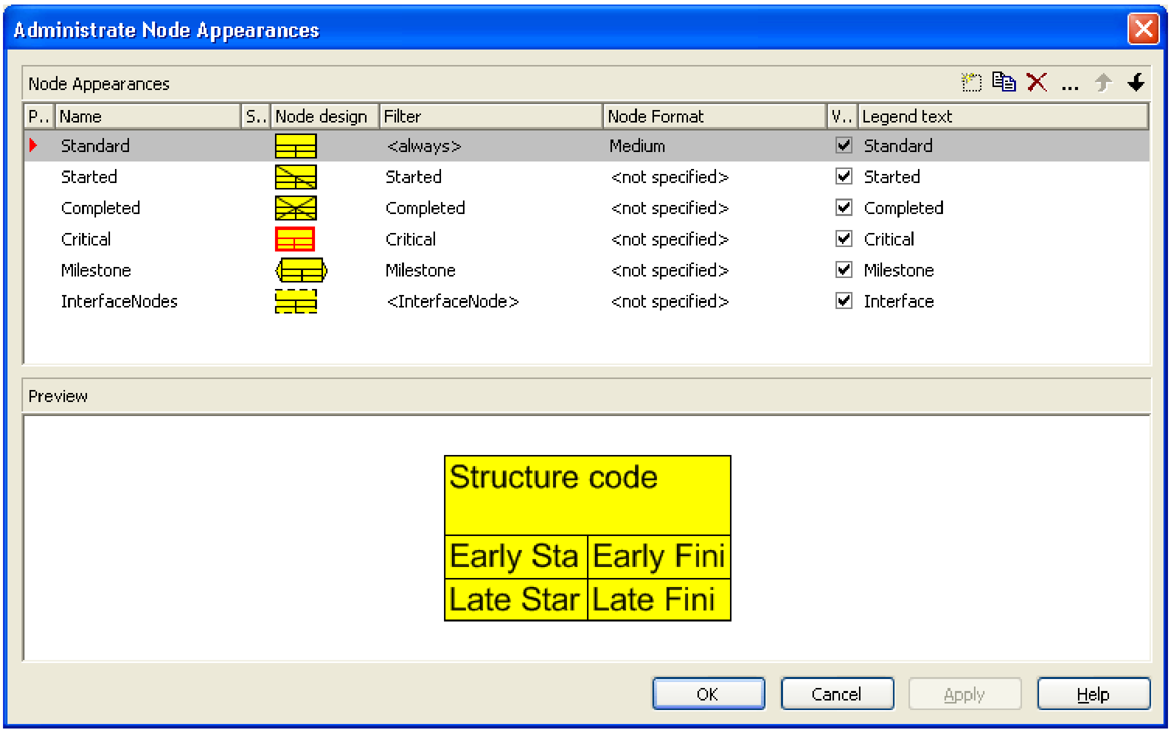
In the **And/Or** column you can choose the operators to combine the condition of the current row with the one in the row following, if necessary.

Leave the **Edit Filter** dialog box by **OK** and return to the **Administrate Filters** dialog box.

2.11 Setting Node Appearances

VARCHART XNet offers a variety of options to modify node appearances. You can define the appearance of a node depending on its data. For example, critical activies may show a double feature and a red background, finished activities may be struck through etc. A defined set of graphical attributes is called an appearance. A node may have several appearances of different priorities.

Please click on the **Objects** property page and then on the **Node Appearances** button to get to the **Administrate Node Appearances** dialog.



P..	Name	S..	Node design	Filter	Node Format	V..	Legend text
▶	Standard			<always>	Medium	☑	Standard
	Started			Started	<not specified>	☑	Started
	Completed			Completed	<not specified>	☑	Completed
	Critical			Critical	<not specified>	☑	Critical
	Milestone			Milestone	<not specified>	☑	Milestone
	InterfaceNodes			<InterfaceNode>	<not specified>	☑	Interface

Here the available node appearances are listed. Please mark them one by one to display their shapes in the preview window.

A node appearance always is associated with a node format and a filter (except the "Standard" node appearance which is not associated with a filter).

A filter consists of conditions that have to be fulfilled by a node for the appearance to apply. For example, the appearance "Marked" is associated with the filter "Marked", that selects all marked nodes.

If a node fulfils the criteria of several appearances, all of them will apply to the node. Each appearance is of a different priority. The appearance assigned last is inserted at the bottom of the column and will override all others. The list therefore represents an inverted hierarchy, with the bottom appearance being of top priority.

Usually, the "Standard" appearance at the top of the list is of lowest priority. It is not associated with a filter and applies to all nodes.

↑ ↓ You can modify the order of working off the node appearances with the help of the arrow buttons.

> Creating, copying, deleting and editing node appearances

In the **Administrate Node Appearances** dialog box you can create, copy, delete and edit node appearances via the following buttons:

Add node appearance

Copy node appearance

✕ **Delete node appearance**

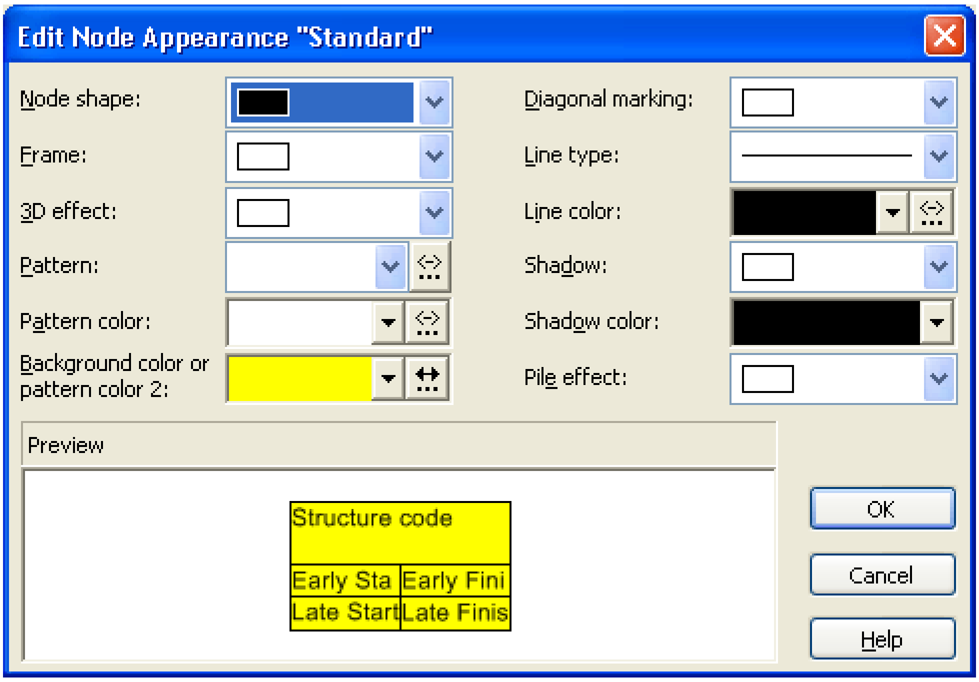
... **Edit node appearance**

Note: You can delete all node appearances except the default node appearances. Before a node appearance is actually deleted, you have to confirm it.

> Using node appearances and filters

This paragraph is about handling node appearances and their associated filters. Please assign to the node appearance "Finished" the top priority by placing it at the bottom. Place the "Started" node appearance right above it to receive second place priority.

Please edit the node appearances now. For this, mark one of them in the **Administrate Node Appearances** dialog and click on the **Edit node appearance** button. You will get to the **Edit Node Appearance** dialog. In the head line the name of the current node appearance is indicated. In this dialog you can modify its graphical attributes.



Edit Node Appearance "Standard"

Node shape:

Frame:

3D effect:

Pattern:

Pattern color:

Background color or pattern color 2:

Diagonal marking:

Line type:

Line color:

Shadow:

Shadow color:

Pile effect:

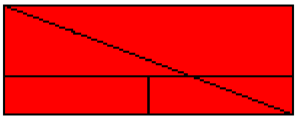
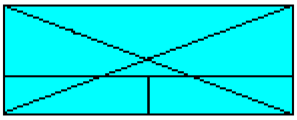
Preview

Structure code

Early Sta | Early Fini

Late Start | Late Finis

Please enter the below settings:

Node appearance	Started	Completed
Filter	Started	Completed
Filter criterion	completed (%) larger than 0 and smaller than 100	completed (%) = 100
Background color	red	blue
Diagonal marking	downward	crossed lines
Appearance		

Please confirm your settings by **OK** and run the program. Create a node, click on it twice and edit its data in the **Edit Data** dialog as follows:

- Please enter "0" into "Completed(%)": The node will show the "Standard" node appearance.
- Next, please enter a figure smaller than 100 and larger than zero into "Completed(%)": The node will show the "Started" node appearance with a red background and a downward strike-through pattern.
- Finally, please enter "100" into "Completed(%)": The node will show the "completed" appearance, that has a crossed-lines strike-through pattern and a blue blackground.

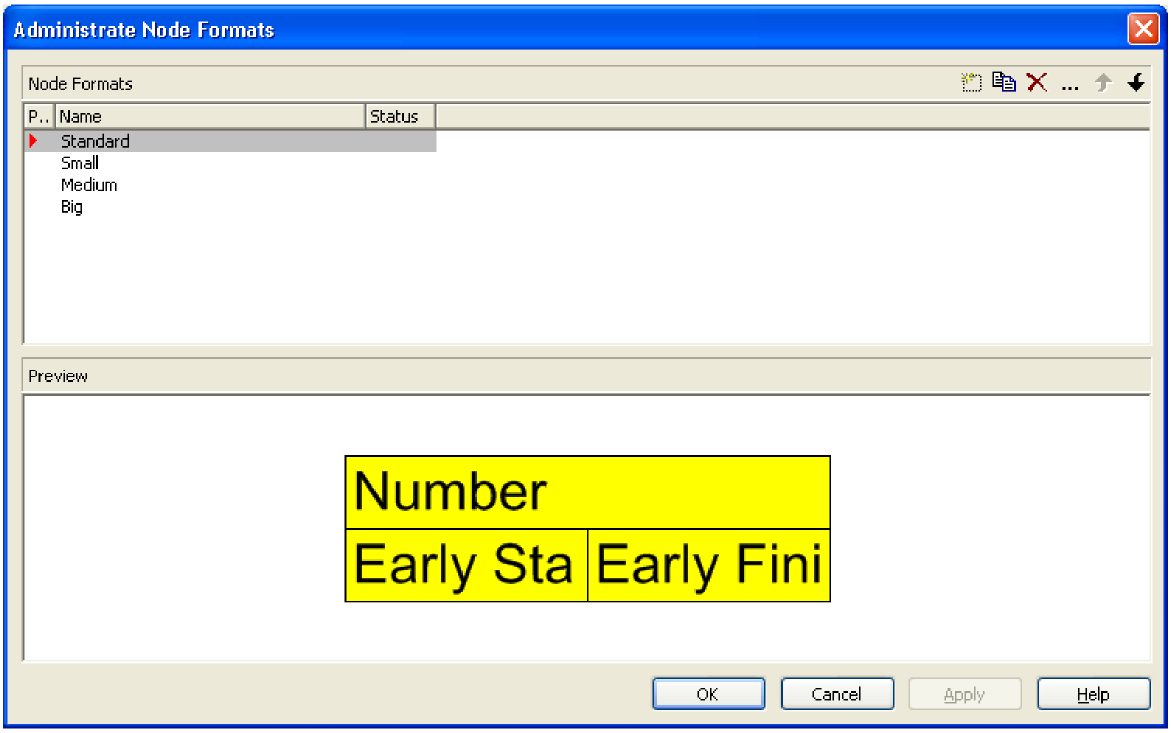
> Specifying the node appearance data dependant

For each node appearance you can assign the pattern, pattern color, background color and the link colur data dependant by means of a map. For details, please see the chapter "Important Concepts: Maps".

2.12 Setting Node Formats

A node appearance always is combined with a node format. The latter you can define yourself.

Please click on the **Node Formats** button of the **Objects** property page. You will get to the **Administrate Node Formats** dialog.



Administrate Node Formats

Node Formats

P..	Name	Status
▶	Standard	
	Small	
	Medium	
	Big	

Preview

The **Node Formats** table contains the node formats available. Mark each one of them in order to view their appearance in the preview window.

In the **Administrate Node Formats** dialog box you can create, copy, delete and edit node formats via the following buttons:

Add node format

Copy node format

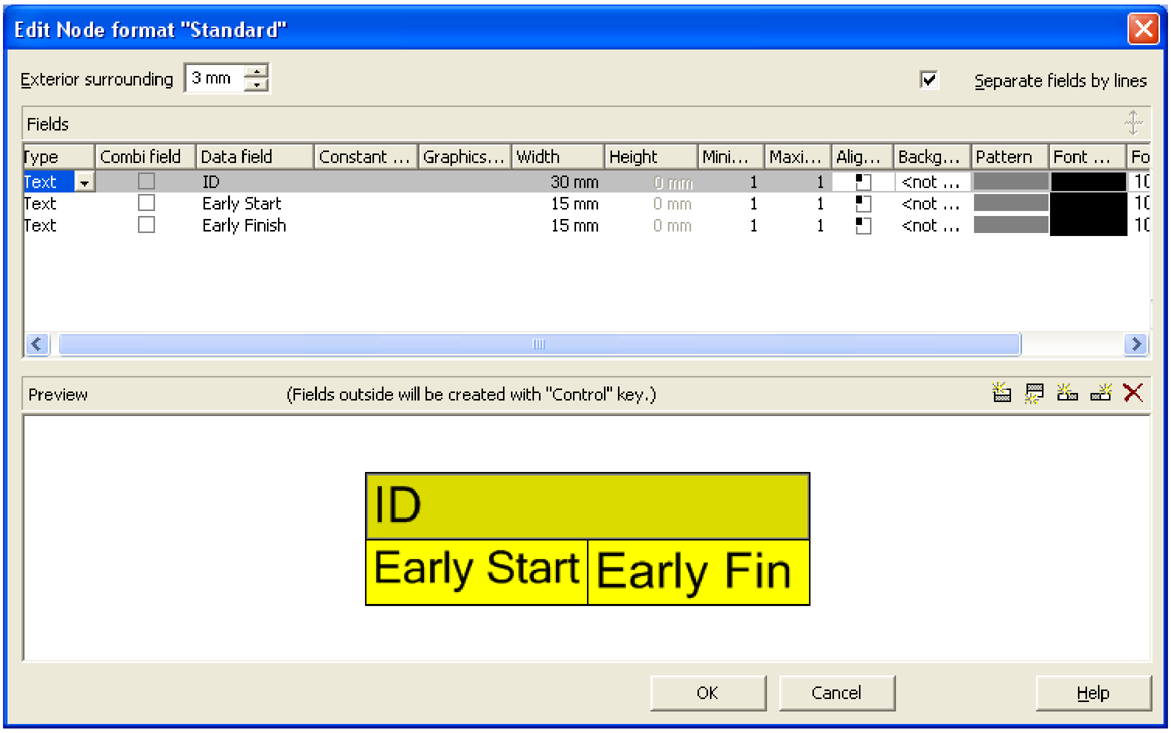
✕ **Delete node format**

... **Edit node format**

Note: You cannot delete the "Standard" node format. The same is valid for node formats used in node appearances. Before a node format is deleted, you have to confirm it.

> Editing Node Formats

To edit a node format, mark it in the list and click on the **Edit node format** button. The dialog **Edit Node Format** will appear.



Edit Node format "Standard"

Exterior surrounding 3 mm

☑ Separate fields by lines

Fields

Type	Combi field	Data field	Constant ...	Graphics...	Width	Height	Mini...	Maxi...	Alig...	Backg...	Pattern	Font ...	Fo
Text	□	ID			30 mm	0 mm	1	1	◰	<not ...	▨	■	10
Text	□	Early Start			15 mm	0 mm	1	1	◰	<not ...	▨	■	10
Text	□	Early Finish			15 mm	0 mm	1	1	◰	<not ...	▨	■	10

Preview (Fields outside will be created with "Control" key.)

In this dialog box you can specify the following:

- whether the node fields are to be separated by lines
- the margins (distance between nodes or between a node and the margin of the chart. Unit: 1/100 mm)
- the type: text or graphics
- for the type text: a data field whose content is to be displayed in the current field or a constant text
- for the type graphics: the name and directory of the graphics file that will be displayed in the current field
- the width and height of the marked field
- how many lines of text can be displayed in the current field
- alignment of the text/graphics of the current field
- the background color of the current field
- the pattern of the current field
- the font attributes of the current field

> Displaying graphics in node fields

For each format field of the type graphics you can specify the graphics file to be displayed.

... To select a graphics file, click on the first button. Then the Windows dialog box **Choose Graphics File** will open.

↔... To configure a mapping from data field entries to graphics files, click the second button. Then the **Configure Mapping** dialog box will open.

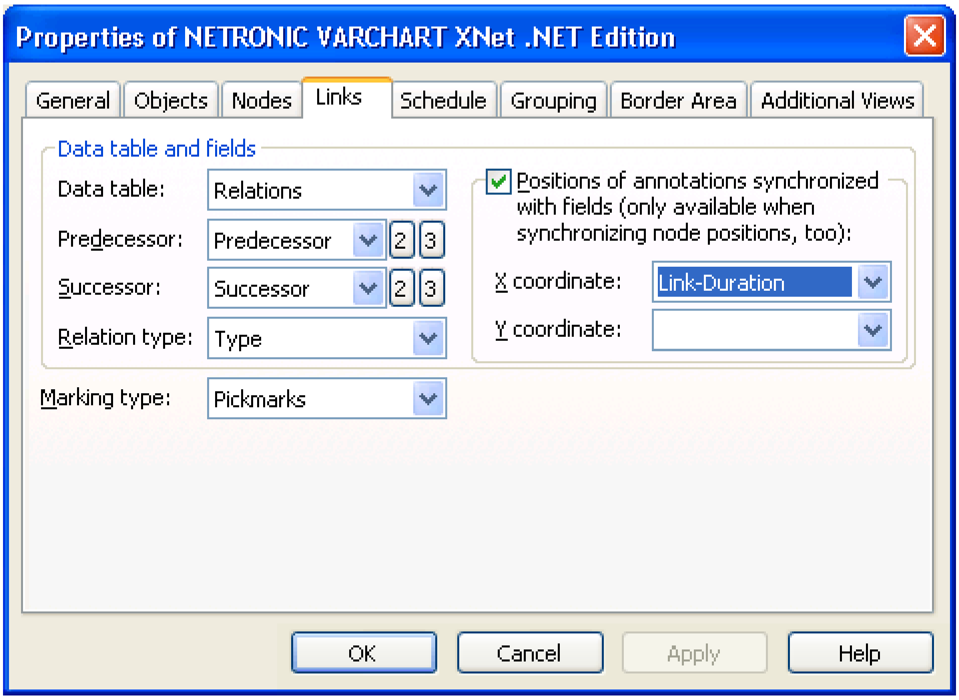
If a mapping has been configured, a symbol is displayed besides the symbol file name (↔...).

For further details please see the chapters "Property Pages and Dialog Boxes" and "Important Concepts: Maps".

2.13 Setting the Link Appearances

In the above paragraphs you learned how to apply filters to nodes. Filters can also be applied to link appearances. It is possible, for example, to assign certain link appearances to different types of links (for example green lines to start-start links and blue lines to finish-finish links).

To set the appearance of a link, please open the **Links** property page.



Properties of NETRONIC VARCHART XNet .NET Edition

Data table and fields

Data table: Relations

Predecessor: Predecessor 2 3

Successor: Successor 2 3

Relation type: Type

☑ Positions of annotations synchronized with fields (only available when synchronizing node positions, too):

X coordinate: Link-Duration

Y coordinate:

Marking type: Pickmarks

In the table in the lower section of the table you can define new appearances or edit already existing ones.

Please double-click on the entry "New..." in the last line to define a new appearance named "FF link". As soon as you click on a different field, the appearance will be created and added to the list. By default, the appearance has the same line attributes and filters as the preceding one.

For selecting the filter used with a link appearance, click on an entry in the **Filter** column. Open the appearing select box by clicking on its arrow-down button. Up to now, only the default filter "always" exists.

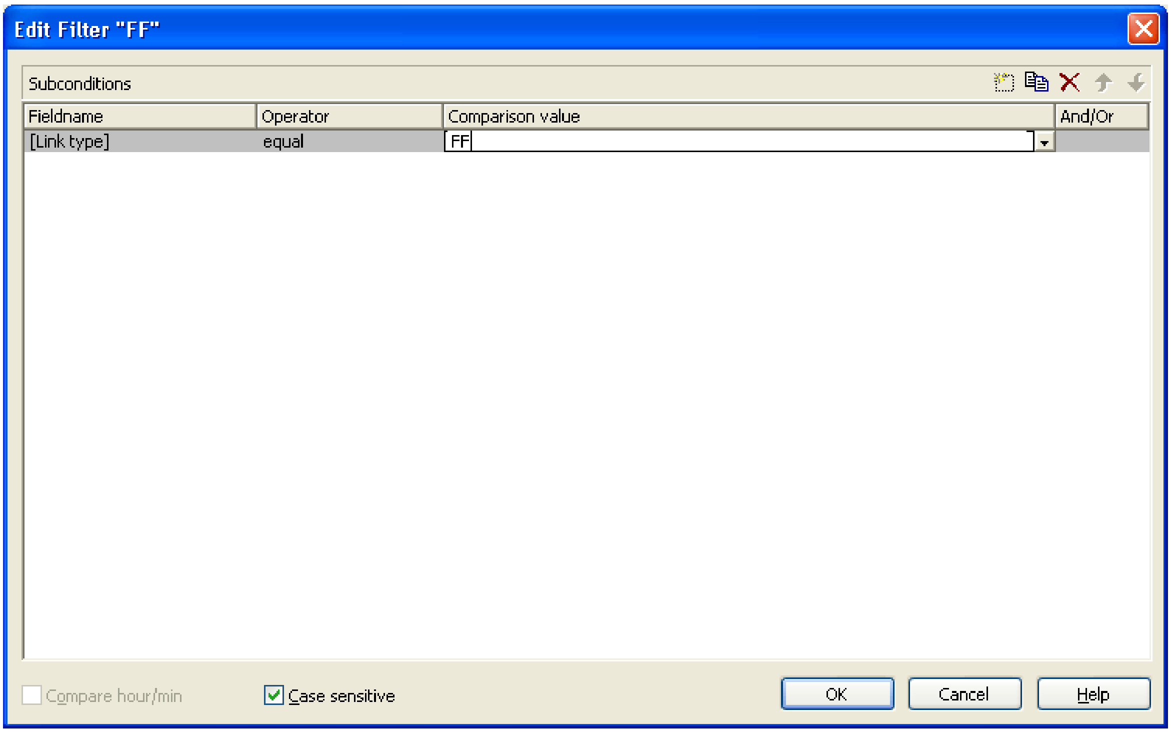
Create a new filter for link appearances now. Click on the **Filter** button to open the **Administrate Filters** dialog wher you can create, copy, edit and delete filters. Click on the "Add filter" button to create a new filter. Rename this filter into "FF". It is designed to select finish-finish links.

Now edit the filter "FF". Click on the **Edit filter** button to open the **Edit Filter** dialog box.

Please define the criterion "Link type equals FF". All links of the type "FF" will now adopt the link appearance associated with this filter.

Modifications of a filter will be valid for this filter in general an not only for the current link appearance.

Click on the **OK** button in this dialog box and in the **Administrate Filters** dialog box.



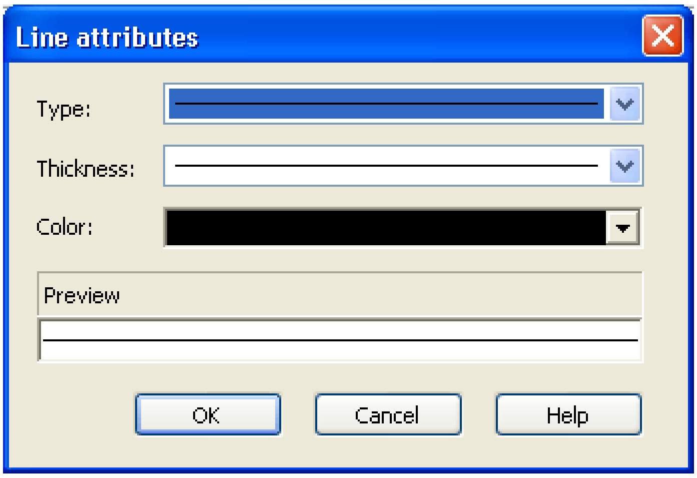
Edit Filter "FF"

Subconditions

Fieldname	Operator	Comparison value	And/Or
[Link type]	equal	FF	

□ Compare hour/min ☑ Case sensitive

Please set the line attributes for the "FF link" link appearance now. Click on the **Edit** button near the **Line type** to open the **Line Attributes** dialog where you can set the color, type and thickness of the line.



Line attributes

Type:

Thickness:

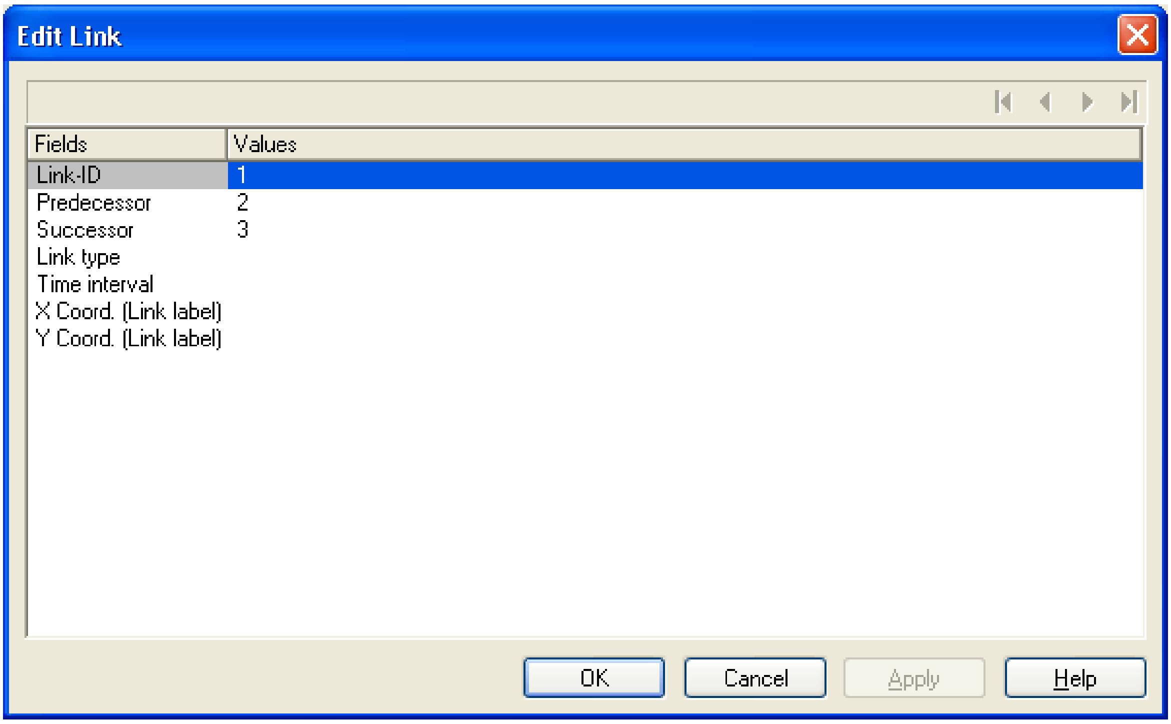
Color:

Preview

Select a blue color for the links that fulfil the criterion of the "FF link" filter, i.e. that are of the FF type.

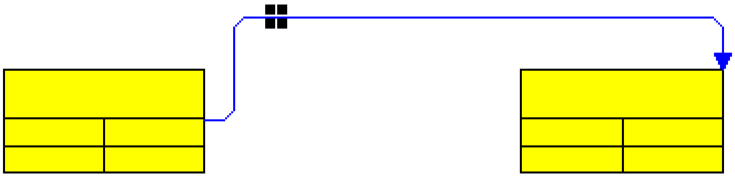
Start the program by the F5 key and generate two nodes and a link between them.

Right click on the link and select **Edit** from the context menu to open the **Edit Link** dialog.

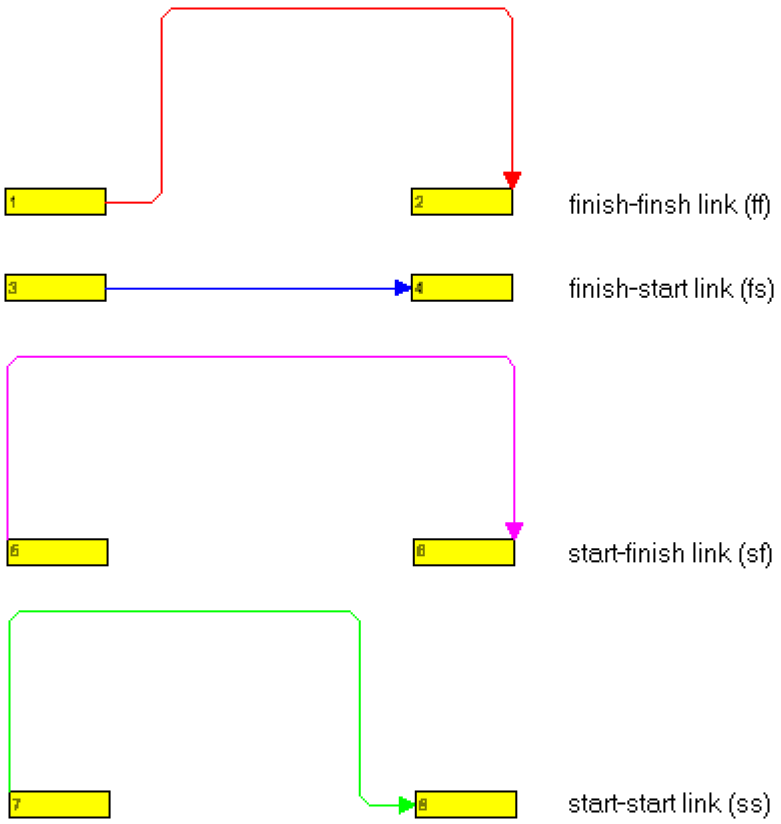


Fields	Values
Link-ID	1
Predecessor	2
Successor	3
Link type	
Time interval	
X Coord. (Link label)	
Y Coord. (Link label)	

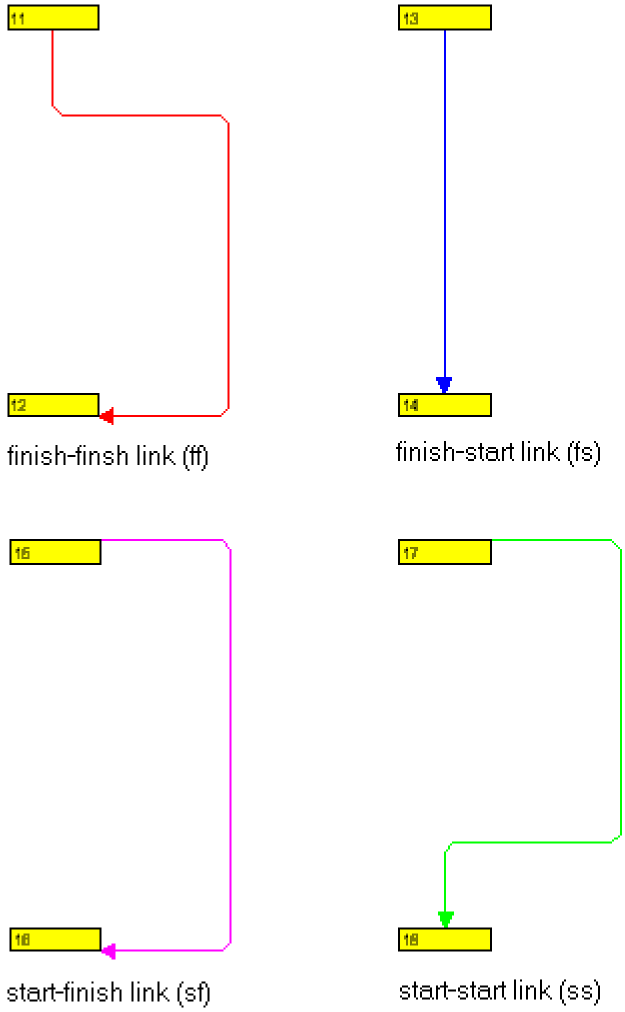
Enter "FF" into the **Type** and confirm by clicking on the **OK** button. The marked link will turn into a finish-finish relation and will be immediately diplayed as a blue line.



How different link types are displayed in different orientations is shown by the pictures below:



Left-to-right orientation



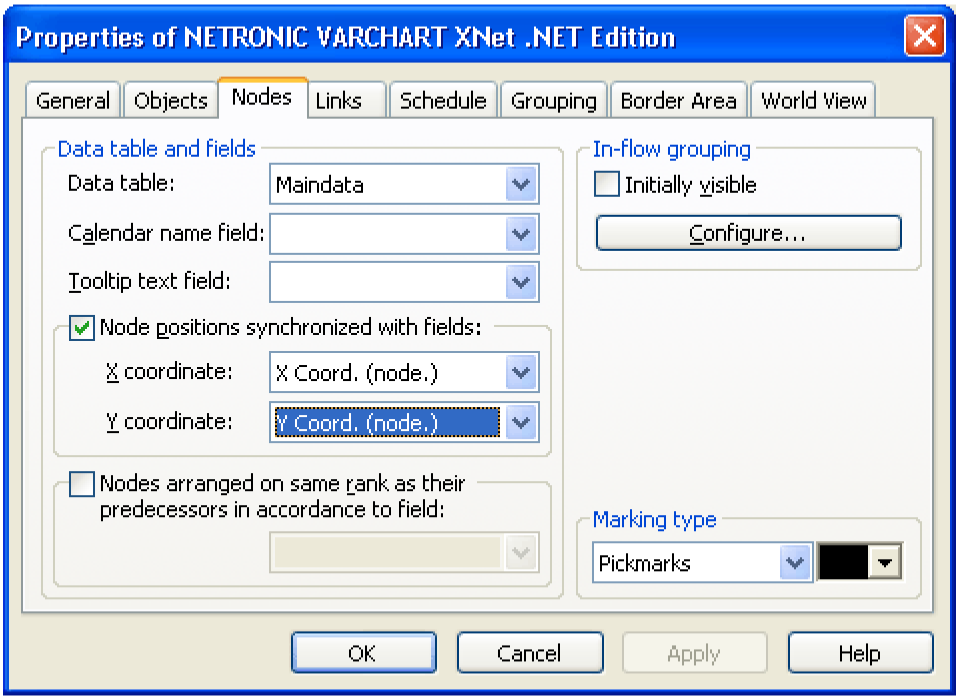
Top-to-bottom orientation

2.14 Saving Positions of Nodes and Link Annotations

Synchronizing the positions of nodes and link annotations with data fields is required if these positions have to be restored after closing the project.

To synchronize node positions with their data fields, please activate the check box **Node positions synchronized with data fields** on the **Nodes** property page and select the following data fields:

- for the X coordinate: "X Coord. (node)"
- for the Y coordinate: "Y Coord. (node)"



Properties of NETRONIC VARCHART XNet .NET Edition

Data table and fields

Data table: Maindata

Calendar name field:

Tooltip text field:

☑ Node positions synchronized with fields:

X coordinate: X Coord. (node.)

Y coordinate: Y Coord. (node.)

□ Nodes arranged on same rank as their predecessors in accordance to field:

In-flow grouping

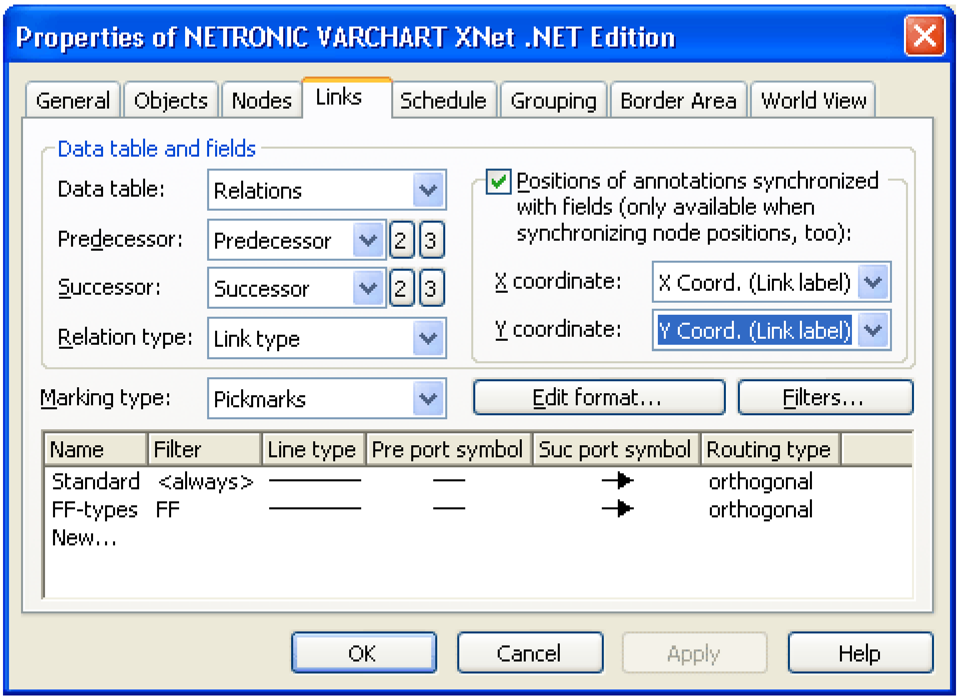
□ Initially visible

Marking type

Pickmarks

To synchronize the link annotation positions with their data fields, on the **Links** property page activate the check box **Annotation positions synchronized with data fields** and select the following data fields:

- for the X coordinate: "X Coord. (Link label)"
- for the Y coordinate: "Y Coord. (Link label)"



Data table and fields

Data table: Relations

Predecessor: Predecessor 2 3

Successor: Successor 2 3

Relation type: Link type

☑ Positions of annotations synchronized with fields (only available when synchronizing node positions, too):

X coordinate: X Coord. (Link label)

Y coordinate: Y Coord. (Link label)

Marking type: Pickmarks

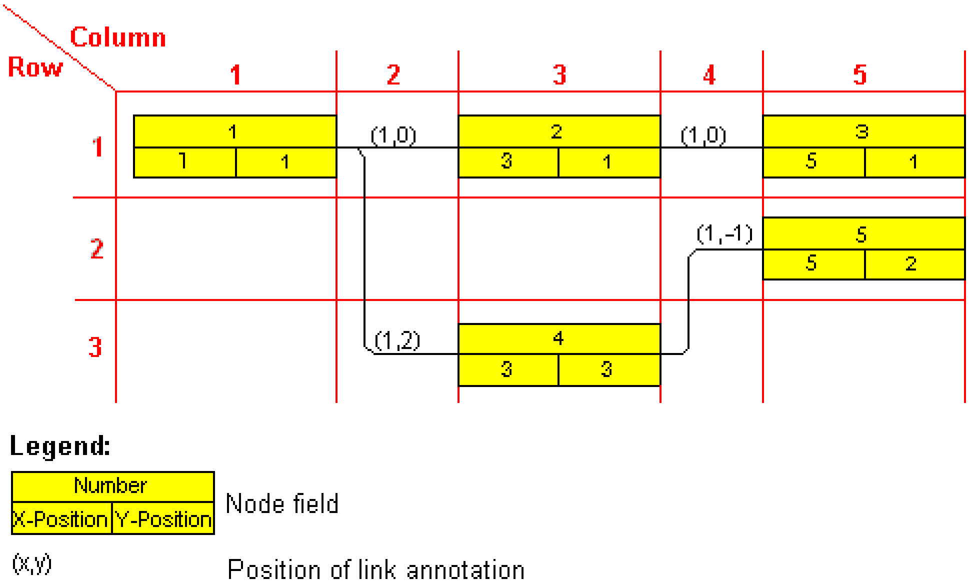
Name	Filter	Line type	Pre port symbol	Suc port symbol	Routing type
Standard	<always>	———	—	→	orthogonal
FF-types	FF	———	—	→	orthogonal
New...					

Positions of nodes and of link annotations are stored as coordinates in a matrix.

- The X and Y coordinates of a node represent the absolute position of the node in the matrix.
- In contrast, the X and Y coordinates of a link annotation refer to the position of the predecessor node.

The top left postion of the matrix is defined as (X,Y) = (1,1) and is reserved for nodes. All other node coordinates are generated by continuously adding 1 to the coordinates of the top left position. Except for the top left position any position may contain a node or a link annotation.

Node coordinates, that represent absolute values, always show positive figures, whereas link annotation coordinates, that represent relative values may show negative figures. Link annotation coordinates cannot be placed in the (0,0) position.



If you wish to save and reload the node positions of a diagram, please supply the below code for the **FormClosing** event:

Example Code

```
Private Sub Form1_FormClosing(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.FormClosingEventArgs) Handles MyBase.FormClosing
   VcNet1.SaveAsEx("C:\test.csv", VcEncoding.vcUnicodeEncoding)
End Sub
```

Example Code

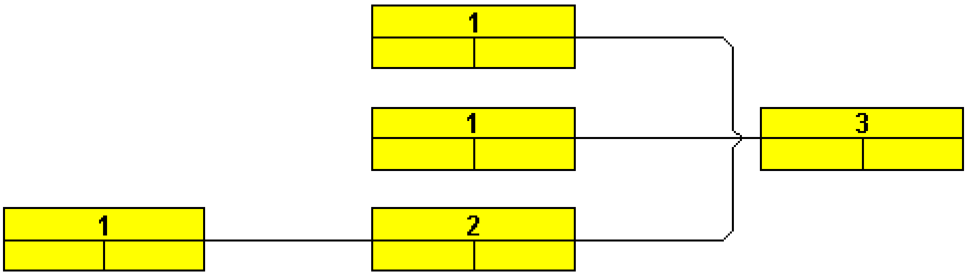
```
private void Clustering_FormClosing(object sender, FormClosingEventArgs
e)
{
   vcNet1.SaveAsEx(@"....", VcEncoding.vcUnicodeEncoding);
}
```

2.15 Positioning Auxiliary Nodes

In some applications it may be useful not to keep all nodes in the same orientation. In a left-to-right orientation you can put nodes above or below their predecessors, in a top-to-bottom orientation you can place them left or right of their predecessors. The way to do this is to diminish the rank number of a node.The rank of a node is a figure defined according to the following rules: The rank of an unpreceded node equals 1. The rank of a node that has predecessors equals 1 plus the rank number of the predecessor of the top rank. This definition avoids cyclic structures to occur in a network diagram.

Examples:

- The rank of a node, the predecessor of which is unpreceded equals 1+1=2.
- The rank of a node that has three predecessors of the ranks 1, 1 and 2 equals 1+2=3 (see sketch).



Ranks of nodes in a left-to-right orientation

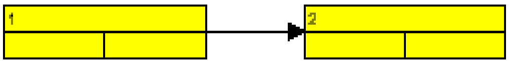
This is how ranks of nodes work:

- In a left-to-right orientation the top rank of all nodes in a node column equals the column number (link annotation columns not included).
- In a top-to-bottom orientation the top rank of all nodes in a node row equals the row number (link annotation rows not included).

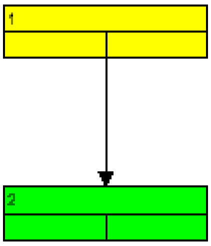
Ranks are calculated by clicking on the **Arrange** item of the diagram context menu. They serve as a base to the layout algorithm to position the nodes in the overall orientation. If cyclic structures exist in the chart, VARCHART XNet will identify them by a separate algorithm and ignore them temporarily. The links ignored will appear as returning links. At the same time, the layout aims at differing as little as possible from a layout that lacks returning links.

In some applications it may be useful to place a node in the same rank as its predecessor, for example, if the node is an auxiliary node of its predecessor. The rank of such a node can be diminished by 1.

In a left-to-right arrangement the auxiliary node, the rank number of which was diminished by 1, is placed below or above its predecessor instead of left or right of it.

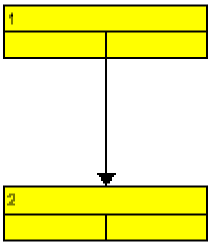


Rank 1 and rank 2 holding a node each

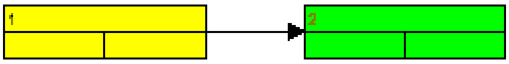


*The rank number of the second node was diminished by 1. Then **Arrange** was invoked.*

In a top-to-bottom arrangement the auxiliary node, the rank number of which was diminished by 1, is placed left or right of its predecessor instead of below or above it.



Rank 1 and rank 2 holding a node each



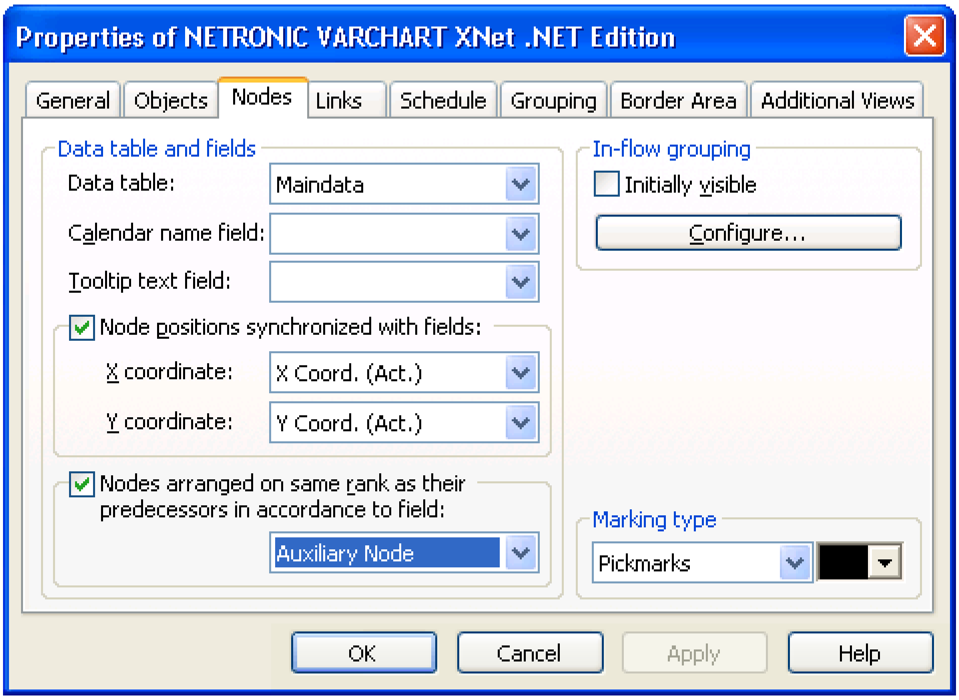
*The rank number of the second node was diminished by 1. After this, **Arrange** was invoked.*

The "Auxiliary node" data field serves to store modifications of the node rank. The entry into the "Auxiliary node" data field will set the position of the node, allowing the values **0, 1, 2** or **3**.

Value in the field "Auxiliary nodes"	Top-to-bottom orientation	Left-to-right orientation
0	The rank number of the auxiliary node is not diminished.	The rank number of the auxiliary node is not diminished.
1	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears left or right of its predecessor instead of below.	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears above or below its predecessor instead of left or right of it.
2	The rank number of the auxiliary node is diminished by 1. The	The rank number of the auxiliary node is diminished by 1. The

Value in the field "Auxiliary nodes"	Top-to-bottom orientation	Left-to-right orientation
	auxiliary node appears to the left of its predecessor.	auxiliary node appears above its predecessor.
3	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears to the right of its predecessor.	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears below its predecessor.

Please follow the below example of placing auxiliary nodes in the same rank as their predecessors. Select the **Left to right** orientation on the **General** property page. Then tick the check box **Nodes arranged on same rank as their predecessor in accordance to field** on the **Nodes** property page. Select the "Auxiliary node" data field from the select box. The entry of the "Auxiliary node" field controls, whether or not a node is placed in the same rank as its predecessor.



Data table: Maindata

Calendar name field:

Tooltip text field:

☑ Node positions synchronized with fields:

X coordinate: X Coord. (Act.)

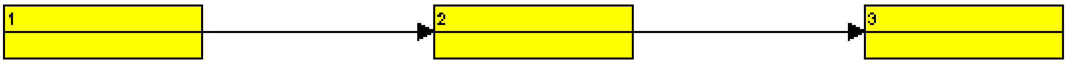
Y coordinate: Y Coord. (Act.)

☑ Nodes arranged on same rank as their predecessors in accordance to field: Auxiliary Node

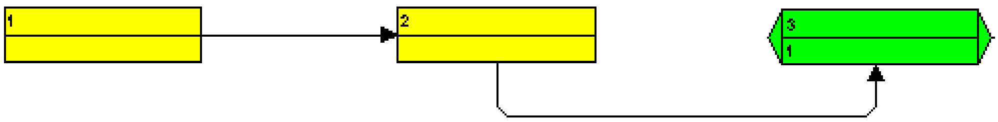
☐ Initially visible

Pickmarks

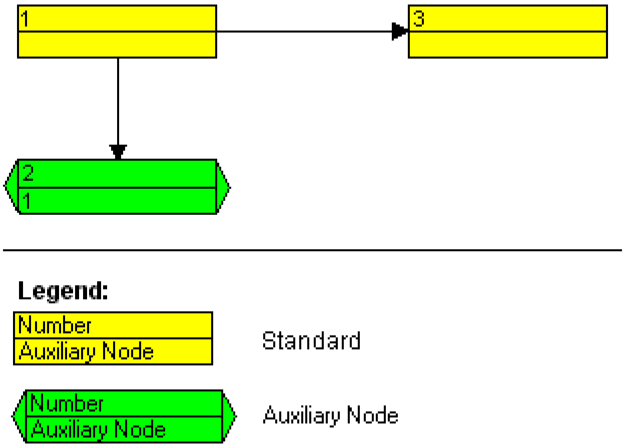
Run the program, generate some nodes and link them as shown in the below picture:



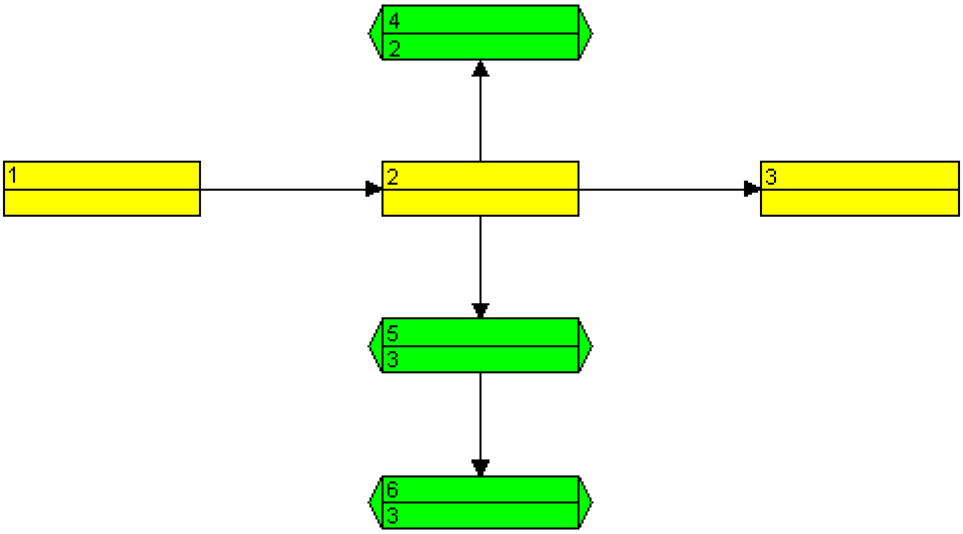
Double-click on the third node and enter the value "1" into the "Auxiliary node" field of the **Edit Data** dialog. This will be the result:



Please pop up the diagram context menu and select the item **Arrange**. This will be the result:

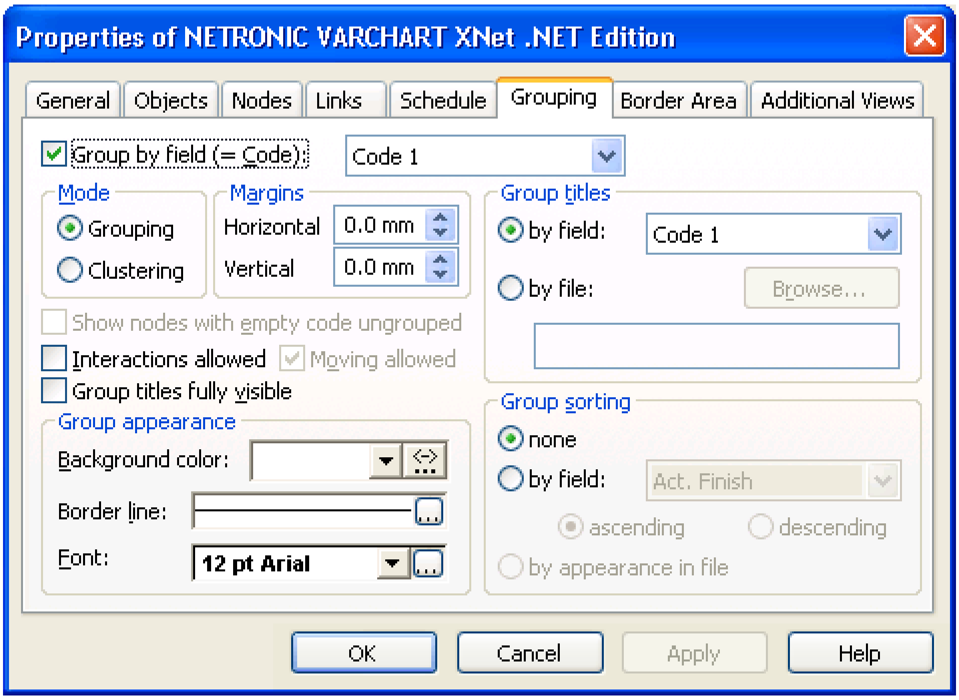


The node the rank of which was reduced, will be placed below instead of right of its predecessor. The picture below shows the ranking of different auxiliary nodes (left-to-right orientation):



2.16 Grouping Nodes

Often, you can improve the layout ot a network diagram by grouping nodes and highlighting groups. You can specify the grouping options on the **Grouping** property page.



☑ Group by field (= Code): Code 1

Mode
● Grouping
○ Clustering

Margins
Horizontal 0.0 mm
Vertical 0.0 mm

□ Show nodes with empty code ungrouped
□ Interactions allowed ☑ Moving allowed
□ Group titles fully visible

Group appearance
Background color:
Border line:
Font: 12 pt Arial

Group titles
● by field: Code 1
○ by file:

Group sorting
● none
○ by field: Act. Finish
● ascending ○ descending
○ by appearance in file

You can group nodes by the field that you select from the select box combined with the **Group by field (= Code)** check box. The field selected will be called **Group code**. Nodes that show the same entry in the **Group code** field will form a group. Please select the field named "Group code".

To generate the titles of groups, there are two options: You can either have them loaded from a file or from data fields. Activate the radio button **by field** to have the group title loaded from a data field, and select a field from the select box. Although the selected field does not necessarily need to be the group code field, the entries of the **Group code** field and of the **Group title** field should correspond in order to give sensible group headings.

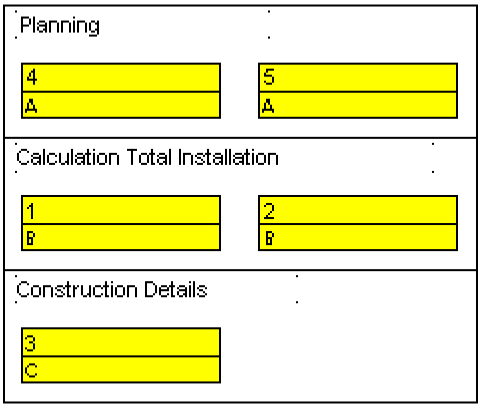
For generating and editing nodes during runtime, as an example in this tutorial please use the tables

Group code	Group name
A	Planning
B	Calculation
C	Details

You need to decide now, whether or not the groups are to be sorted, and if so, what criteria they are to be sorted by. You can either sort them according to a criterion defined by a data field or according to their occurrence in the file. The **Group sorting** section lets you enter the settings for sorting the groups. Please set the radio button to **by field**. Then select the field that the groups are sorted by "Group code", and the sorting order descending. The groups will be sorted according to the group code in descending order now.

The **Group appearance** fields let you set the color, thickness and the type of the line that the groups are framed by, the background color and the font features of the group. Please select some nice settings and run the program.

Generate some nodes and enter values into the data fields "Group code" and "Group name". Please note in which way the two fields correspond. You will receive a picture that more or less looks like this:

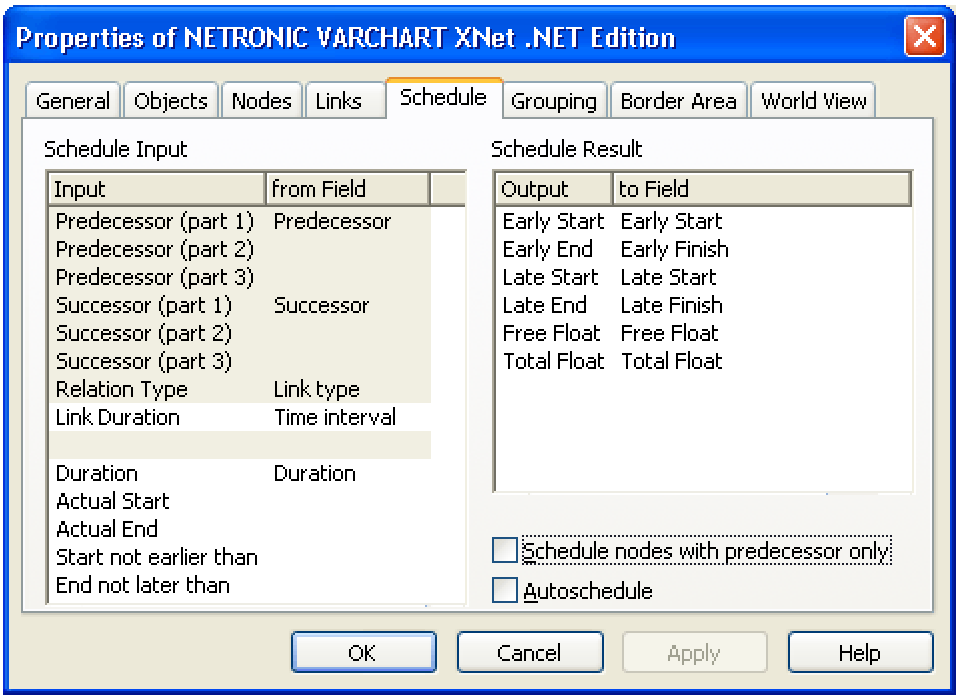


Please move a node to a different group. The value in the "Group code" field will automatically change. You can verify this by invoking the **Edit Data** dialog of the node to view the field.

2.17 Specifying the Scheduling of VARCHART XNet

The VARCHART XNet Scheduler lets you perform simple date calculations, requiring the project start and end dates for parameters.

By the **Schedule** property page you can adapt VARCHART XNet´s date calculation settings to your interface by specifying the data fields you want to use for the input (**Schedule Input**) and output (**Schedule Result**) of the scheduler. Beside, you can set the time unit to be used in the fields that receive the results.



Schedule Input

Input	from Field
Predecessor (part 1)	Predecessor
Predecessor (part 2)	
Predecessor (part 3)	
Successor (part 1)	Successor
Successor (part 2)	
Successor (part 3)	
Relation Type	Link type
Link Duration	Time interval
Duration	Duration
Actual Start	
Actual End	
Start not earlier than	
End not later than	

Schedule Result

Output	to Field
Early Start	Early Start
Early End	Early Finish
Late Start	Late Start
Late End	Late Finish
Free Float	Free Float
Total Float	Total Float

□ Schedule nodes with predecessor only

□ Autoschedule

Please select in the **Schedule Input** table for each item of the **from Field** column a field from the select box that appears as soon as you click in the field. The data will be taken from the fields chosen. Select your settings as shown in the picture.

The scheduler uses data fields of the Maindata and Relations table as input fields for calculating dates.

The key data for calculating the dates are the durations of the various activities, their logical dependencies and the project start. The **Predecessor**, **Successor** and **Relation type** fields cannot be edited in the **Schedule Input** table. They merely show the settings that have been arranged on the **Links** property page.

Please select in the **Schedule Result** table for each item of the **to Field** column a field from the select box that appears as soon as you click in the field. The results will be written to the fields selected, that are fields of the main data table only and were defined by the data definition.

The output data is written to data fields of the interface. Available output options are: **Early Start**, **Early Finish**, **Late Start**, **Late Finish**, **Total Float** and **Free Float**. Please select for each of the output options a field of the list defined by the data definition (as shown in the picture).

There are several options to initialize the scheduler:

1. You can set a project start via the API, by invoking the VcNet method **ScheduleProject**:

   VcNet1.ScheduleProject "04.05.2000", 0

   The method **ScheduleProject** lets you perform a forward and a backward calculation of the project. If you pass the start date, first a forward calculation will be performed, followed by a backward calculation. If you pass the final date, first a backward calculation will be performed, followed by a forward calculation. You can pass both dates, which will add the corresponding float to the activities.

   Setting Parameters to the "ScheduleProject" method:

Start	Finish
Date 1	0
0	Data 2
Date 1	Date 2

2. If you enter current start or end dates, the nodes will become static and cannot be moved.
3. You may enter reference dates for the conditions "Start not earlier than" and "End not later than". For these, select the corresponding data fields in the **Schedule Input** table on the **Schedule** property page. The reference date will be loaded from the fields selected. Then the earliest start of an activity will never be put before and the latest end of an activity will never be put after its reference date.

Please run the scheduler now. Before, please define three buttons ("Command1", "Command2" und "Command3") to execute the scheduler during runtime. Name the buttons "Start of project", "End of project" and "Start and end of project" and add the code:

Example Code

```
Private Sub Command1_Click()
VcNet1.ScheduleProject "01.01.2000", 0
```

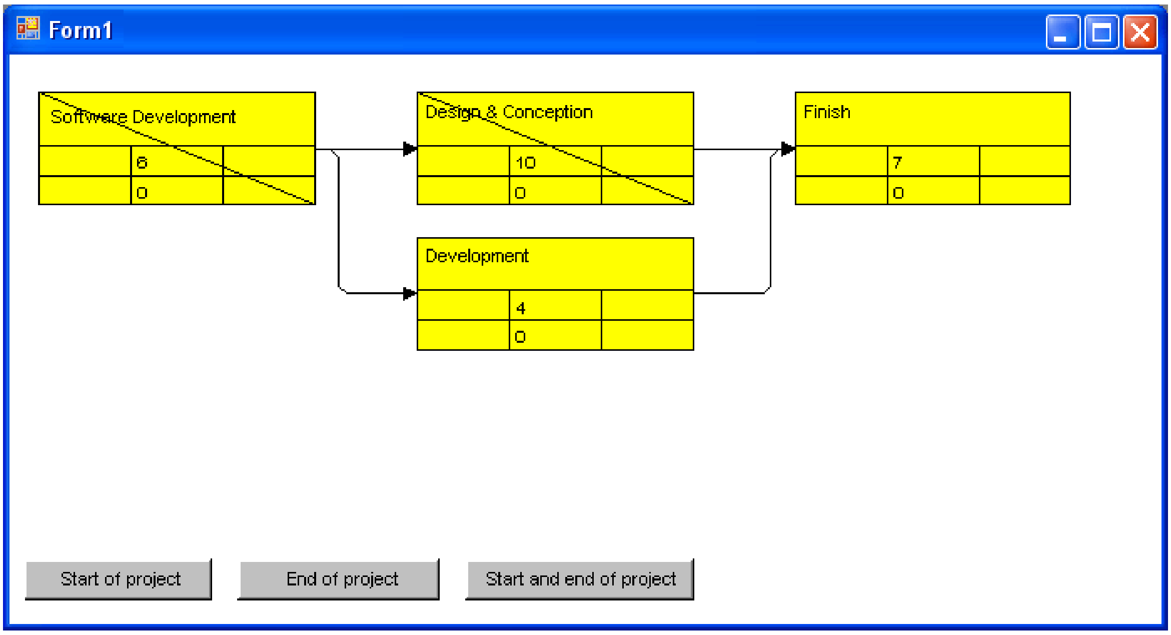
```
End Sub

Private Sub Command2_Click()
VcNet1.ScheduleProject 0, "01.02.2000"
End Sub

Private Sub Command3_Click()
VcNet1.ScheduleProject "01.01.2000", "01.02.2000"
End Sub
```

Please enter the code below, to load some nodes and links on the program start.

Example Code

```
Private Sub Form_Load()

    VcNet1.InsertNodeRecord ("1;1.;;;Software Development;A;; _
                                       Group A;6;0;10;;;;0;;")
    VcNet1.InsertNodeRecord ("2;1.2;;;Design & Conception;C;; _
                                       Group A;10;0;50;;;;0;;")
    VcNet1.InsertNodeRecord ("3;1.2.2;;;Finish;B;; _
                                       Group A;7;0;0;;;;0;;")
    VcNet1.InsertNodeRecord ("4;1.2.4;;;Development;B;; _
                                       Group A;4;0;0;;;;0;;")

    VcNet1.InsertLinkRecord ("1;1;2;;;")
    VcNet1.InsertLinkRecord ("2;1;4;;;")
    VcNet1.InsertLinkRecord ("3;2;3;;;")
    VcNet1.InsertLinkRecord ("4;4;3;;;")

    VcNet1.EndLoading

End Sub
```

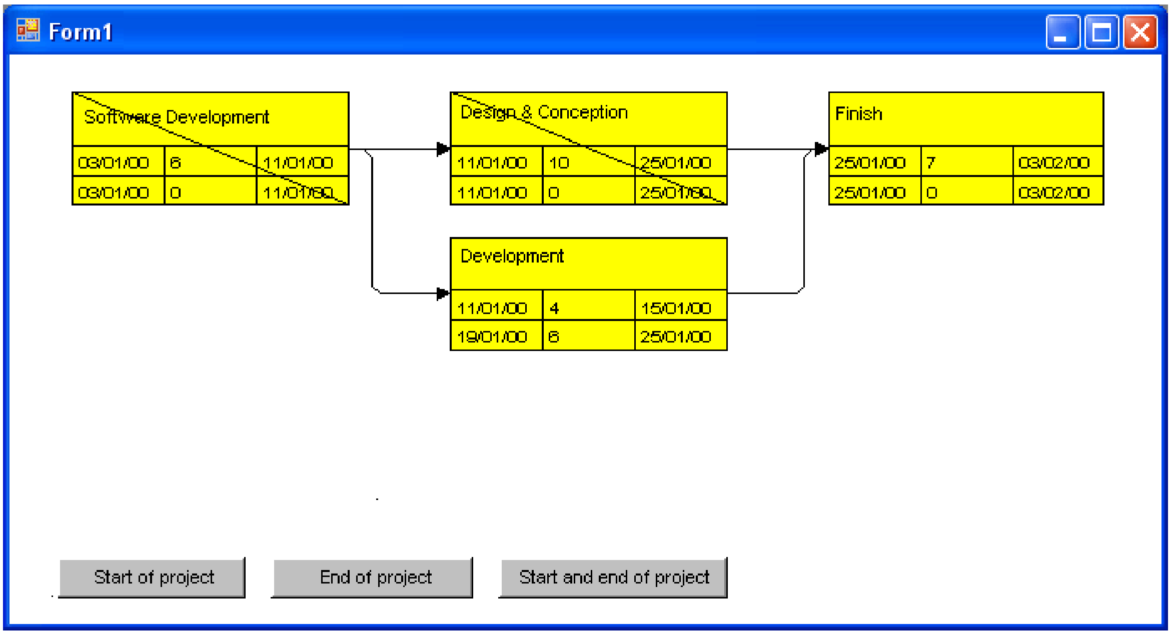
Please start the program now. The nodes and links loaded by the API are displayed:



Node format used: "Big"

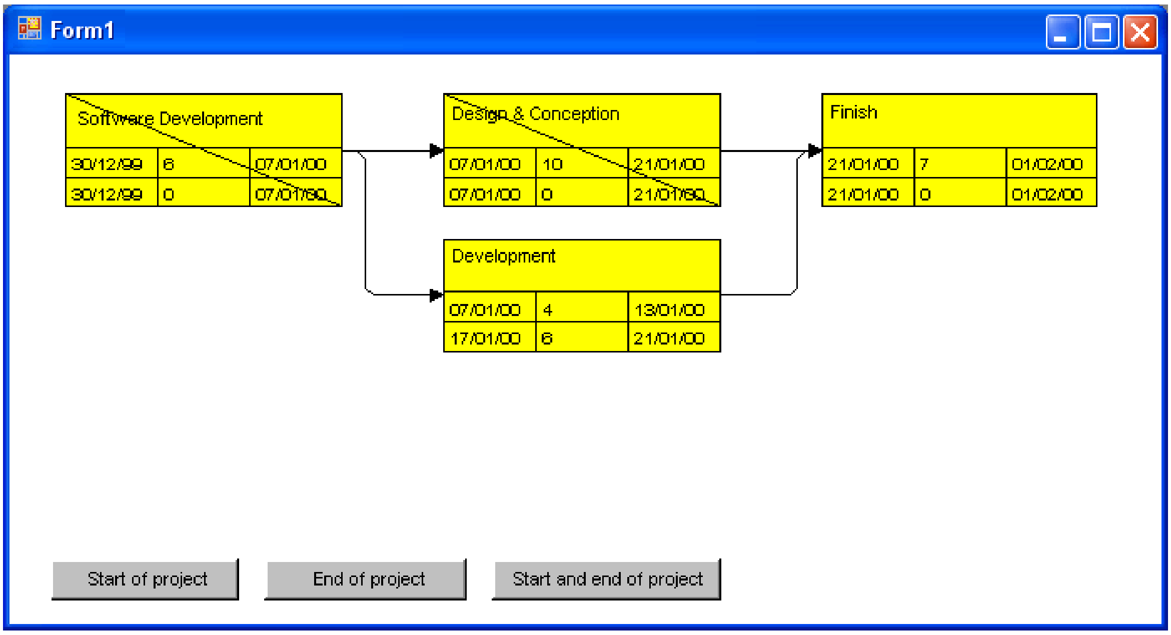
Name		
Early start	Duration	Early finish
Late start	Float	Late finish

Please click on the "Start of project" button. The dates calculated will be based on the project start.



Please note that an internal calendar will be considered where weekends represent work-free periods. The internal calendar is used if you ticked the **Scheduler uses internal calendar** check box on the **General** property page.

Please click on the "End of project" button. The dates calculated will be based on the project end.



Software Development

30/12/99	6	07/01/00
30/12/99	0	07/01/00

Design & Conception

07/01/00	10	21/01/00
07/01/00	0	21/01/00

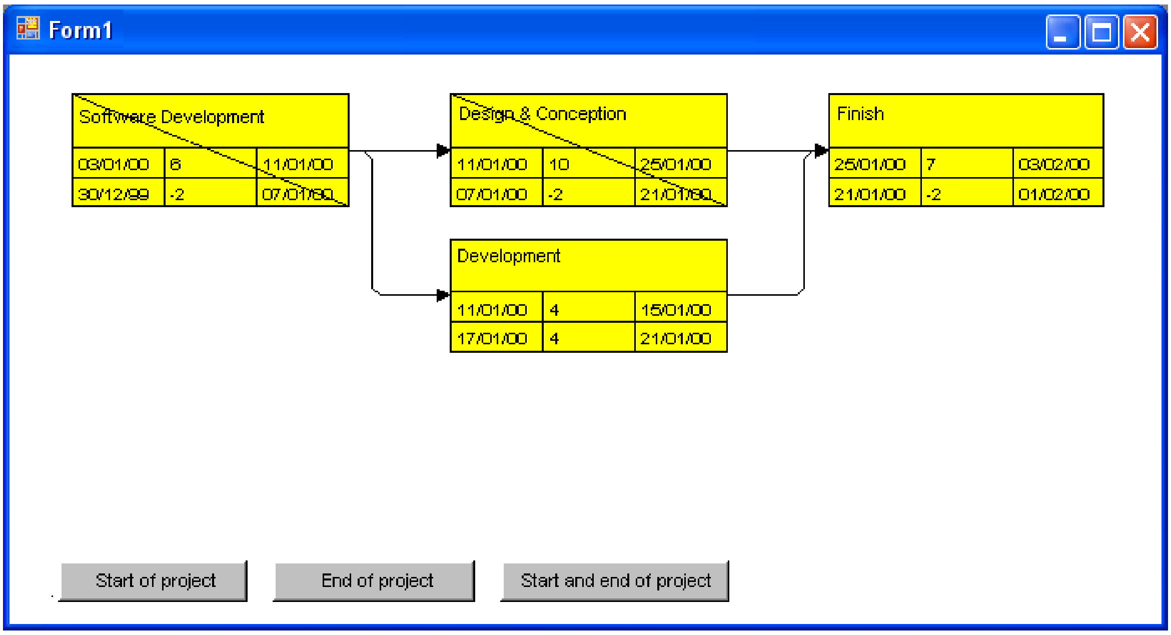
Finish

21/01/00	7	01/02/00
21/01/00	0	01/02/00

Development

07/01/00	4	13/01/00
17/01/00	6	21/01/00

Please click on the "Start and end of project" button. The calculations will consider both, start and end dates.



Software Development

03/01/00	6	11/01/00
30/12/99	-2	07/01/00

Design & Conception

11/01/00	10	25/01/00
07/01/00	-2	21/01/00

Finish

25/01/00	7	03/02/00
21/01/00	-2	01/02/00

Development

11/01/00	4	15/01/00
17/01/00	4	21/01/00

As this example shows, negative floats will occur when both dates are taken into account.

2.18 Printing the Diagram

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

You also can use the method **ShowPrintDialog** of the object VcNet to trigger the printing of the diagram.

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

The method **PrintEx** of the object VcNet lets you print the diagram directly. A dialog box will not be displayed.

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

You can also use the method **ShowPageSetupDialog** of the object VcNet to open the corresponding dialog.

In the **Page Setup** dialog you can set e.g. the scaling, whether the pages shall be numbered, the margins, the alignment etc. For further information please see chapter 5.14 "Setting up Pages".

2.19 Exporting a Diagram

You can export a diagram into a graphics file. There are two different ways to this:

- Please select the menu item **Export graphics** from the default context menu. From there you can get to the Windows dialog **Save as**, that lets you save the diagram as a graphics file.
- Use the API method **ShowExportGraphicsDialog** or **ExportGraphicsToFile**.

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

2.20 Saving the Configuration

All settings made on the property pages at design time are added to your project as a resource. Changes come into operation only after saving your project, since only then the embedded resource will be updated.

Tip: For this reason, you should activate in Microsoft Visual Studio .NET 2005 the Option **Save all changes** in **Tools > Options > Environment > Projects and Solutions > Build and Run**, 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.

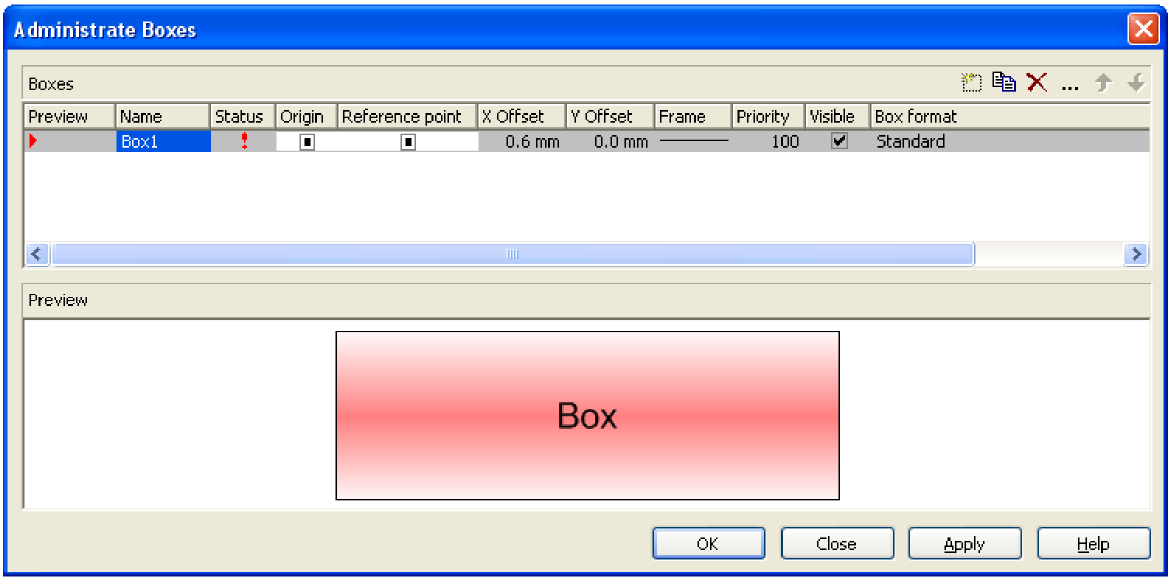
How to load a stored configuration:

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

3 Important Concepts

3.1 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.



Preview	Name	Status	Origin	Reference point	X Offset	Y Offset	Frame	Priority	Visible	Box format
	Box1	!			0.6 mm	0.0 mm		100	☑	Standard

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 is independent of the current diagram size.

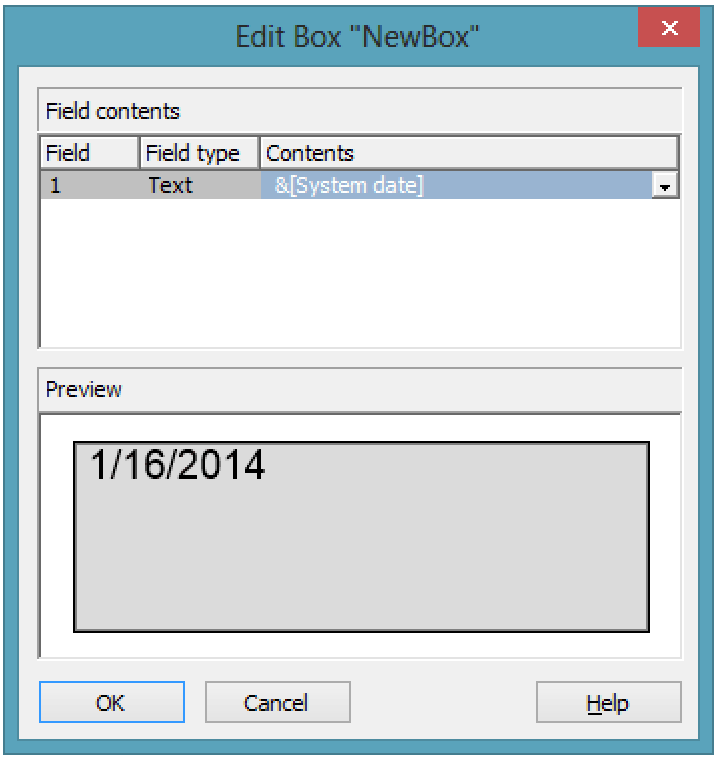
For each box you can specify

- its name
- whether the box can be moved in the diagram at run time
- 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

> Editing boxes

The **Edit Box** dialog lets you specify the contents of the fields. This dialog box will appear at design time when you click the **Edit box** button in the **Administrate Boxes** dialog box. At run time it will appear when you double-click the box to be edited. You also can edit the texts of boxes directly at run time after having selected **In-place editing allowed** on the property page **General**



Field	Field type	Contents
1	Text	&[System date]

1/16/2014

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

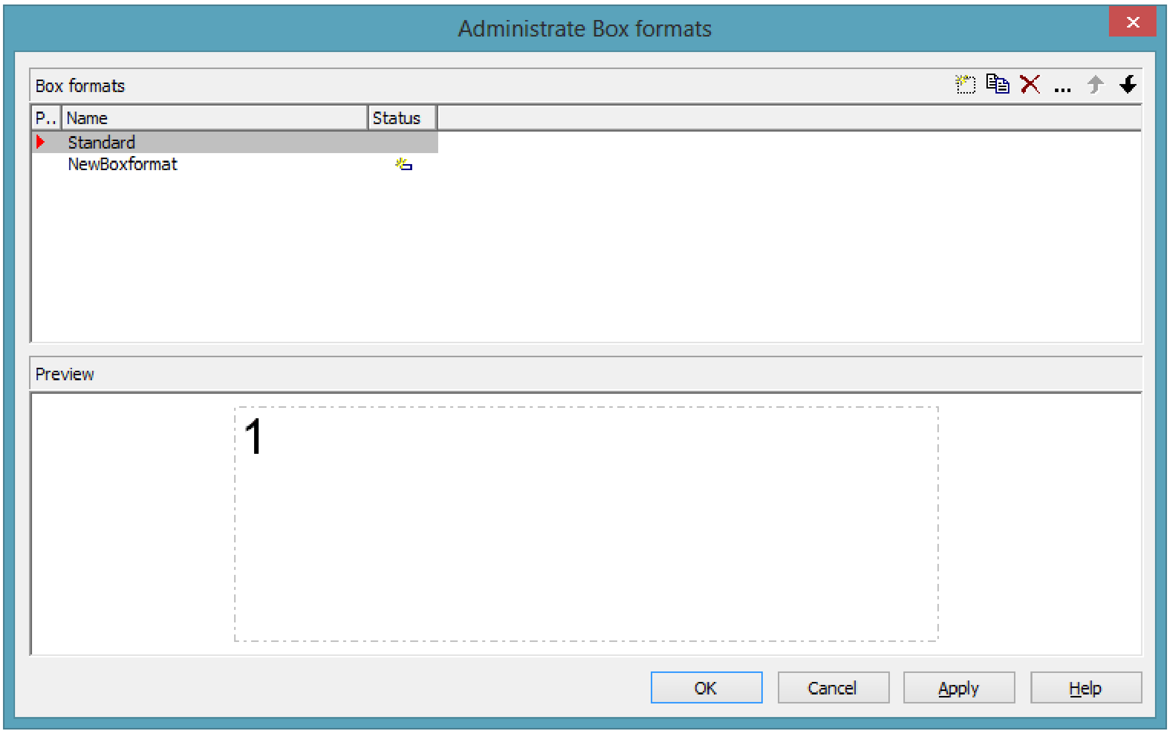
The **Field type** column displays the field types (text or graphics).

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

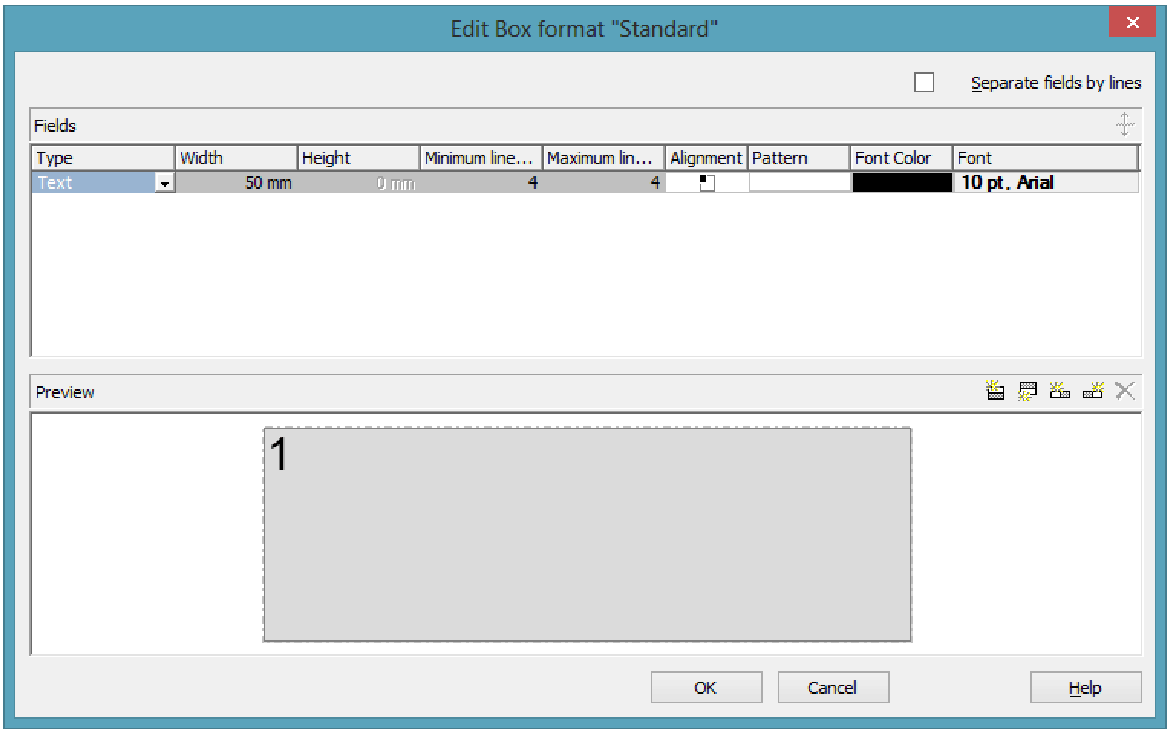
For each box you can select a box format, and you can specify the box formats.

In the **Administrate Box Formats** dialog box you can add, copy, delete or edit box formats. Click the corresponding button on the **Objects** property page to open this dialog.



P..	Name	Status
▶	Standard	
	NewBoxformat	✳

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.



Edit Box format "Standard"

□ Separate fields by lines

Fields

Type	Width	Height	Minimum line...	Maximum lin...	Alignment	Pattern	Font Color	Font
Text	50 mm	0 mm	4	4	◰		■	10 pt, Arial

Preview

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

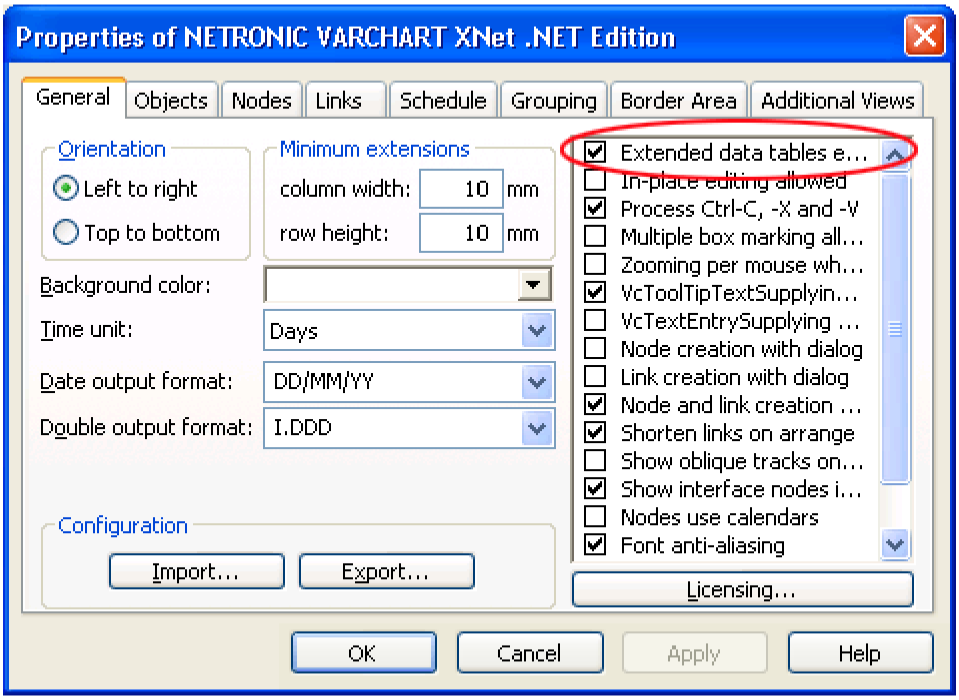
Furthermore, the following items can be specified for each box :

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

3.2 Data Tables

As a data base for the graphical display of Net charts VARCHART XNet 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 XNet.

For reasons of compatibility to existing applications VARCHART XNet 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**:



● Left to right
○ Top to bottom

column width: 10 mm
row height: 10 mm

Background color:
Time unit: Days
Date output format: DD/MM/YY
Double output format: I.DDD

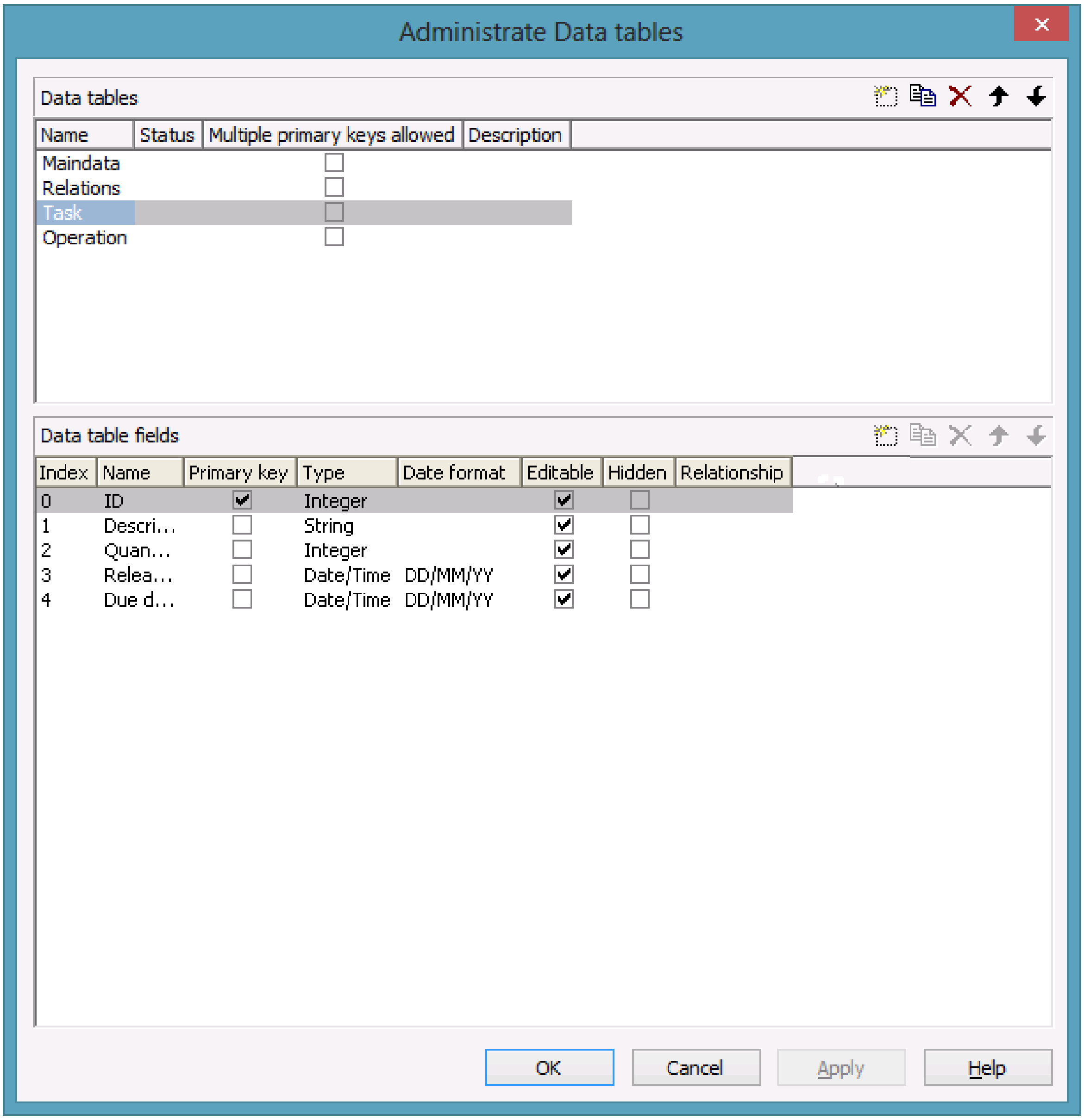
☑ Extended data tables e...
□ In-place editing allowed
☑ Process Ctrl-C, -X and -V
□ Multiple box marking all...
□ Zooming per mouse wh...
☑ VcToolTipTextSupplyin...
□ VcTextEntrySupplying ...
□ Node creation with dialog
□ Link creation with dialog
☑ Node and link creation ...
☑ Shorten links on arrange
□ Show oblique tracks on...
☑ Show interface nodes i...
□ Nodes use calendars
☑ Font anti-aliasing

In the programming interface, the extended data tables are switched on at runtime by setting the VcNet 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 **Administrate Data Tables**. Generating new data tables requires to have switched on the **Extended data tables** mode before.

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.


Administrate Data tables

Data tables

Name	Status	Multiple primary keys allowed	Description
Maindata		□	
Relations		□	
Task		□	
Operation		□	

Data table fields

Index	Name	Primary key	Type	Date format	Editable	Hidden	Relationship
0	ID	☑	Integer		☑	□	
1	Descri...	□	String		☑	□	
2	Quan...	□	Integer		☑	□	
3	Relea...	□	Date/Time	DD/MM/YY	☑	□	
4	Due d...	□	Date/Time	DD/MM/YY	☑	□	

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 version 4.0 of VARCHART XNet 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

> Creating and modifying data records

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

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection

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

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(Main_ID) = "1"
dataRecVal(Main_Name) = "Node 1"
dataRecVal(Main_Start) = DateSerial(2013, 1, 8)
dataRecVal(Main_Duration) = 8
```

Example Code C#

```
VcDataTable dataTable =
vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

Object [] dataRecVal = new
object[dataTable.DataTableFieldCollection.Count];

VcDataRecord dataRec1;
VcDataRecord dataRec2;

dataRecVal[Main_ID] = "1";
dataRecVal[Main_Name] = "Node 1";
dataRecVal[Main_Start] = "08.01.2013";
dataRecVal[Main_Duration] = 8
```

A data record can be added by the method **Add()** of the object **DataRecordCollection**, the object array being passed as parameter.

Example Code VB.NET

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

Example Code C#

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

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

Example Code VB.NET

```
dataRecCltn.Add("2;Node 2;15.01.13;;9")
```

Example Code C#

```
dataRec2.AllData = "2;Activity Y;15.01.13;;9";
```

If a data value contains a semicolon, the character string has to be enclosed in double quotes.

Example Code VB.NET

```
dataRec2 = dataRecCltn.Add("2;""Node 2;"";15.01.13;;9")
```

Example Code C#

```
dataRec2 = dataRecCltn.Add("2;\"Node 2;\";15.01.13;;9");
```

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

Example Code VB.NET

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

Example Code C#

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

The contents of single data fields of a data record may be easily modified by using the indexed property **DataField()**. In order to replace all data field contents of a record you can use the property **AllData**.

Example Code VB.NET

```
dataRec1.DataField(Main_ID) = 1
dataRec1.DataField(Main_Name) = "Activity X"
dataRec1.DataField(Main_Start) = DateSerial(2013, 1, 4)
dataRec1.DataField(Main_Duration) = 12
dataRec1.Update()

dataRec2.AllData = "2;Activity Y;18.01.13;;5"
dataRec2.Update()
```

Example Code C#

```
dataRec1.set_DataField(Main_ID, 1);
dataRec1.set_DataField(Main_Name, "Activity X");
dataRec1.set_DataField(Main_Start, "04.01.2014");
dataRec1.set_DataField(Main_Duration, 12);
dataRec1.Update();

dataRec2.AllData = "2;Activity Y;18.01.14;;5";
dataRec2.Update();
```

A modification of a record can only be displayed after the method **Update()** of the object **DataRecord** was called.

Loading the values by using **Alldata** is suitable for quickly displaying all data values at design time and for transferring the data record contents to the record of a different table. You may also use this data format also for information exchange with OLE Drag & Drop.

Example Code VB.NET

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

Example Code C#

```
content = dataRec1.AllData + "\r\n" + dataRec2.AllData + "\r\n" +
dataRec1.get_DataField(Main_Name);
MessageBox.Show(content);
```

Note: In order to improve the legibility for data field access, you can define global constants that have names rather than index numbers, which are more descriptive. Below please find the code in its context:

Example Code VB.NET

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

'...

Dim dataRec1 As VcDataRecord
Dim dataRec2 As VcDataRecord

Dim content As String

VcNet1.TimeScaleEnd = DateSerial(2014, 1, 1)
VcNet1.TimeScaleStart = DateSerial(2013, 1, 1)

VcNet1.ExtendedDataTablesEnabled = True
dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(Main_ID) = "1"
dataRecVal(Main_Name) = "Node 1"
dataRecVal(Main_Start) = DateSerial(2013, 1, 8)
dataRecVal(Main_Duration) = 8
dataRec1 = dataRecCltn.Add(dataRecVal)
```

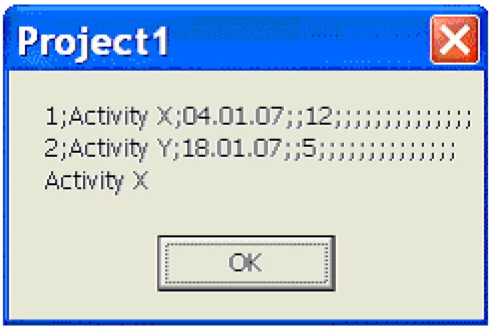
```
dataRecCltn.Add("2;Node 2;15.01.13;;9")

VcNet1.EndLoading()

'...

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

dataRec1.DataField(Main_ID) = 1
dataRec1.DataField(Main_Name) = "Activity X"
dataRec1.DataField(Main_Start) = DateSerial(2013, 1, 4)
dataRec1.DataField(Main_Duration) = 12
dataRec1.Update()

dataRec2.AllData = "2;Activity Y;18.01.13;;5"
dataRec2.Update()

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

'...


dataRec2.AllData = "2;""Activity Y;Z"";18.01.13;;5"
dataRec2.Update()
content = dataRec1.AllData & vbCr & dataRec2.AllData
MsgBox(content)
```

Example Code C#

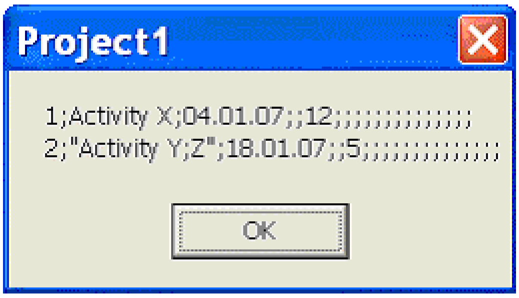
```
const int Main_ID = 0;
const int Main_Name = 1;
const int Main_Start = 2;
const int Main_Duration = 4;

//...

VcDataRecord dataRec1;
VcDataRecord dataRec2;

string content;

vcNet1.TimeScaleEnd = Convert.ToDateTime("01.01.2014");
vcNet1.TimeScaleStart = Convert.ToDateTime("01.01.2013");

vcNet1.ExtendedDataTablesEnabled = true;
VcDataTable dataTable =
vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
Object [] dataRecVal = new
object[dataTable.DataTableFieldCollection.Count];)

dataRecVal[Main_ID] = "1";
dataRecVal[Main_Name] = "Node 1";
dataRecVal[Main_Start] = "08.01.2013";
dataRecVal[Main_Duration] = 8;
dataRec1 = dataRecCltn.Add(dataRecVal);
```

```
dataRecCltn.Add("2;Node 2;15.01.13;;9");

vcNet1.EndLoading();

//...

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

dataRec1.set_DataField(Main_ID, 1);
dataRec1.set_DataField(Main_Name, "Activity X");
dataRec1.set_DataField(Main_Start, "04.01.2013");
dataRec1.set_DataField(Main_Duration, 12);
dataRec1.Update();

dataRec2.AllData = "2;Activity Y;18.01.13;;5";
dataRec2.Update();

content = dataRec1.AllData + "\r\n" + dataRec2.AllData + "\r\n" +
dataRec1.get_DataField(Main_Name);
MessageBox.Show(content);

//...

dataRec2.AllData = "2;Activity Y;Z;18.01.13;;5";
dataRec2.Update();
content = dataRec1.AllData + "\r\n" + dataRec2.AllData;
MessageBox.Show(content);
```

The following output will be created:



```
1;Activity X;04.01.07;;12;;;;;;;;;;;;;;;
2;Activity Y;18.01.07;;5;;;;;;;;;;;;;;;
Activity X
```

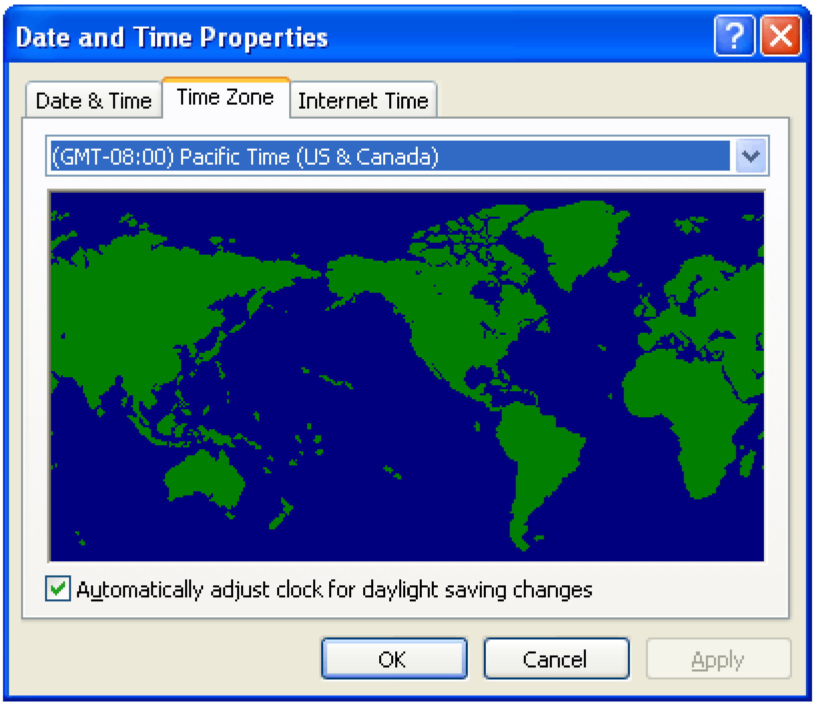


```
1;Activity X;04.01.07;;12;;;;;;;;;;;;;;;
2;"Activity Y;Z";18.01.07;;5;;;;;;;;;;;;;;;
```

3.3 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**.



(GMT-08:00) Pacific Time (US & Canada)

☑ Automatically adjust clock for daylight saving changes

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.

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.

3.4 Drag & Drop

Apart from moving or copying nodes within an instance of the VARCHART XNet component, a user can also move or copy activities beyond the limits of an instance (source component) to a different instance (target component). This chapter introduces subjects that are important to the developer to program the latter type of interaction.

Whereas shifting a node within the same instance entails an alteration of the node's data, its dates do not change if the node is shifted between different instances (they certainly could by a subsequent shift within the target instance).

Shifting a node between different instances splits into two steps: leaving the source component and entering the target component. Each step requires a permission from the corresponding component.

VARCHART XNet allows to move or copy several nodes by a single interaction. If a user presses the left mouse button while the cursor is on a node, internally an object of the type **System.Windows.Forms.DataObject** is generated and filled with the data of the node in CSV format (i.e. by text or by the data type **System.String**). After that, the event **VcDragStarting** is triggered immediately so that the application can control permitted actions (copy and/or move) by itself. By default, both actions are possible, depending on the the status of the <Ctrl> key: by pressing it while releasing the mouse button, the object will be copied, otherwise it will be moved.

After this, the event VcDragCompleting ist triggered to inform the application of the action taken (copy, move or cancellation) and to enable it to probably react.

Then, in the source component the events **Control.GiveFeedback** and **Control.QueryContinueDrag** are triggered. In the target component the events **Control.DragEnter**, **Control.DragOver** and **Control.DragLeave** are triggered.

For further information about the .NET drag&drop routines please refer to the description of the .NET framework. In addition, five more properties exist that influence the behavior of drag&drop:

> Control.AllowDrop

This Boolean property of the base class **Control** allows to set whether objects that were dragged onto the control can be dropped. The property applies only to objects from the outside; objects dragged within the VARCHART control are not affected (i.e., they can always be dropped).

> VcNet.LeavingControlWhileDraggingAllowed

This Boolean property of the VcNet object allows to set whether nodes can be dragged beyond the limits of the source control. This allows to move or copy nodes between two different VARCHART controls, to different controls of the same application or even to controls of different applications.

> VcNet.NodeCreationAtDroppingEnabled

This Boolean property of the VcNet object allows to set whether the target component automatically should generate a node after an object was dropped on it.

> VcNet.PhantomDrawingWhileDraggingEnabled

This Boolean property lets you set to the target component whether the default phantom of the VARCHART component should be generated.

> VcNet.InbuiltMouseCursorWhileDraggingEnabled

This Boolean property lets you set to the target component whether the mouse cursor typical of the VARCHART component should be displayed. If it is not displayed, the drag&drop mouse cursor (arrow and a little square or prohibitory sign) will be displayed, or even a cursor specific of the application.

3.5 Events

Events are the elements that pass information on the user's interactions with the VARCHART control to the application. Each time a user interacts with the VARCHART control, 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, via the **returnStatus** parameter you can deactivate all context menus offered in VARCHART control (and replace them by your own, if you want) plus 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
vcRetStatNoPopup	4	revoking the popup menu

3.6 Filters

A filter consists of conditions that are to be fulfilled by nodes or links. Filters let you select nodes or links that fulfil the criteria defined, e.g. in order to highlight them in the diagram.

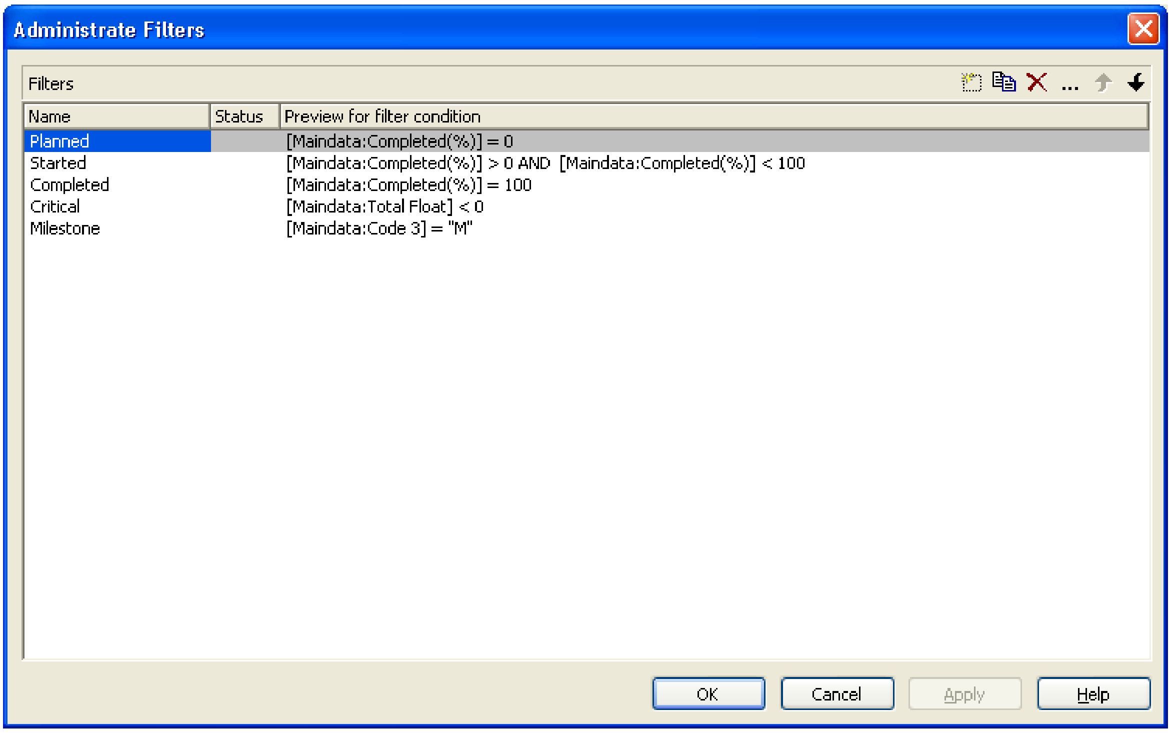
When you apply a filter, the data of the activity or link is compared with the criteria of the filter. Those activities that fulfil the filter criteria will be selected.

For example, you can define a filter that specifies "All nodes starting after January 2001".

Filters can only be handled in design mode.

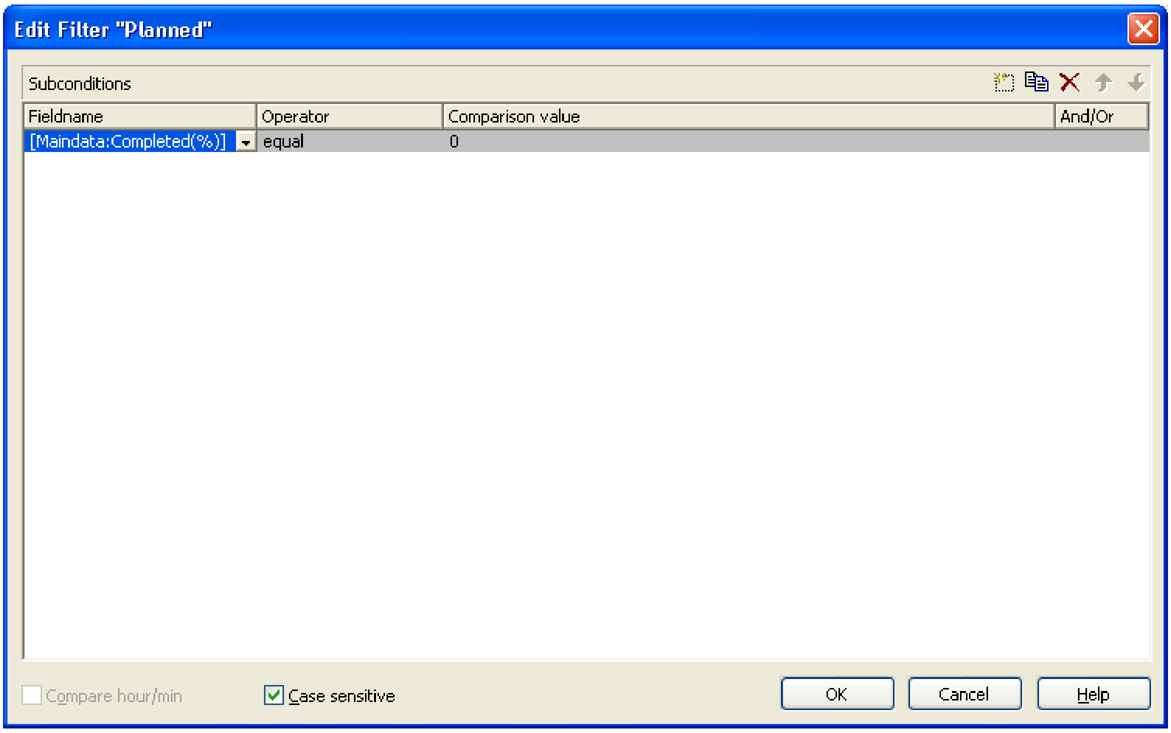
You can get to the **Administrate Filters** dialog via the **Objects** property page, and for filters for links also via the **Links** property page.

With the help of the **Administrate Filters** dialog box you can rename, create, copy, delete or edit filters.



Name	Status	Preview for filter condition
Planned		[Maindata:Completed(%)] = 0
Started		[Maindata:Completed(%)] > 0 AND [Maindata:Completed(%)] < 100
Completed		[Maindata:Completed(%)] = 100
Critical		[Maindata:Total Float] < 0
Milestone		[Maindata:Code 3] = "M"

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



Edit Filter "Planned"

Subconditions

Fieldname	Operator	Comparison value	And/Or
[Maindata:Completed(%)]	equal	0	

□ Compare hour/min ☑ Case sensitive

3.7 Graphics Formats

VARCHART supports the below graphics formats, which is important to exporting charts, affecting mainly the calls **VcNet1.ShowGraphicsExportDialog and VcNet1.ExportGraphics**.

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

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

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

> WMF (Windows Metafile Format)

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

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

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

For the description of the format please see:

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

On the limitations of the format see:

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

> EMF (Enhanced Metafile Format)

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

In spite of 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 2003.

For the format description please see:

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

> GIF (Graphics Interchange Format)

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

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

> JPEG (Joint Photographic Experts Group)

This bitmap format was developed by the JPEG for compressed storage of photographs, accepting loss. Storing charts and diagrams requires a precise storage of lines, so using this format does not make much sense. This format is only supported 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. This format is only supported 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. This format is only supported 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.

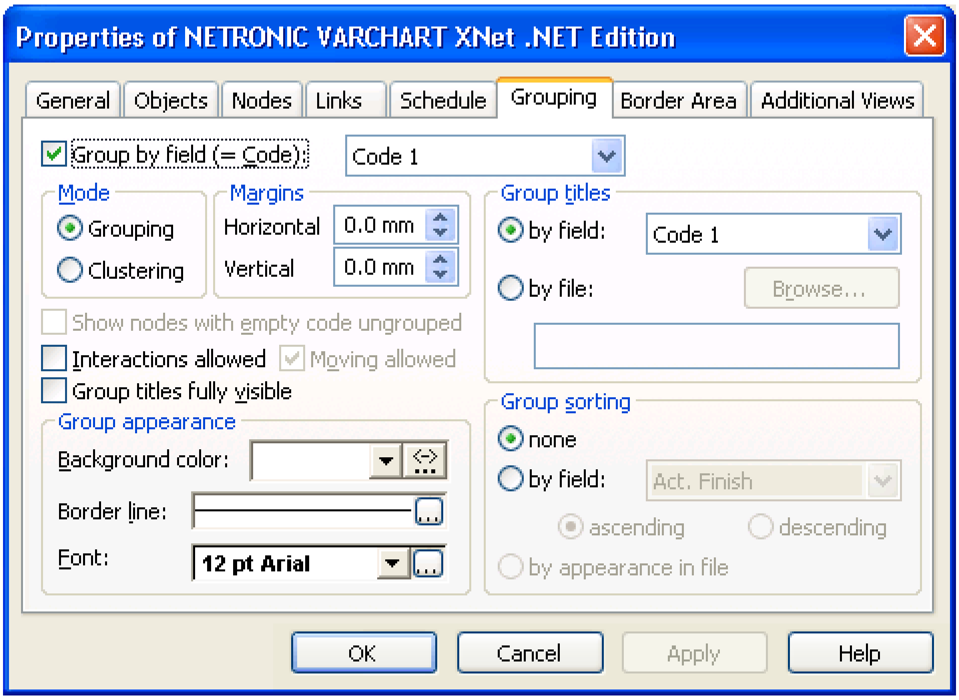
From version 4.2 onward the free library **libpng** is used, in order to set a resolution and thus store bitmaps of any size. It has to be taken into account though that very large PNG files may cause problems when loaded, since usually PNG files get completely unpacked in the memory and then are displayed.

For the format description please see:

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

3.8 Grouping

Many applications require to group nodes and to highlight the groups. For example, nodes can be grouped after different phases of the project, such as the planning phase, the construction phase, the assembly phase etc., or according to different departments, such as construction or accountancy. You can set criteria for grouping on the **Grouping** property page.



☑ Group by field (= Code): Code 1

Mode
● Grouping
○ Clustering

Margins
Horizontal 0.0 mm
Vertical 0.0 mm

☐ Show nodes with empty code ungrouped
☐ Interactions allowed ☑ Moving allowed
☐ Group titles fully visible

Group appearance
Background color:
Border line:
Font: 12 pt Arial

Group titles
● by field: Code 1
○ by file:

Group sorting
● none
○ by field: Act. Finish
● ascending ○ descending
○ by appearance in file

The nodes are grouped by a field that can be selected from the combo box combined with the **Group by field (= Code)** check box. The field selected will be called **Group code**. Nodes that show the same entry in the **Group code** field will form a group.

If a user moves a node to a different group, the value in the data field that is used as the group code will be adapted automatically.

You can either have the group titles loaded from a file or from data fields.

- Activate the radio button **by field** to have the group title loaded from a data field, and select a field from the combo box. Although the selected field does not necessarily need to be the group code field, the entries of the **Group code** field and of the **Group title** field should correspond in order to give sensible group headings.
- If you activate the **by file** radio button, group titles will be loaded from the file you select here. Clicking on the **Browse** button will open the **Choose Group Titles File** dialog where you can choose a file that group titles are to be loaded from. The file needs to be organized like this:

```
A Group A
B Group B
C Group C
```

> Example:

"A" = short text

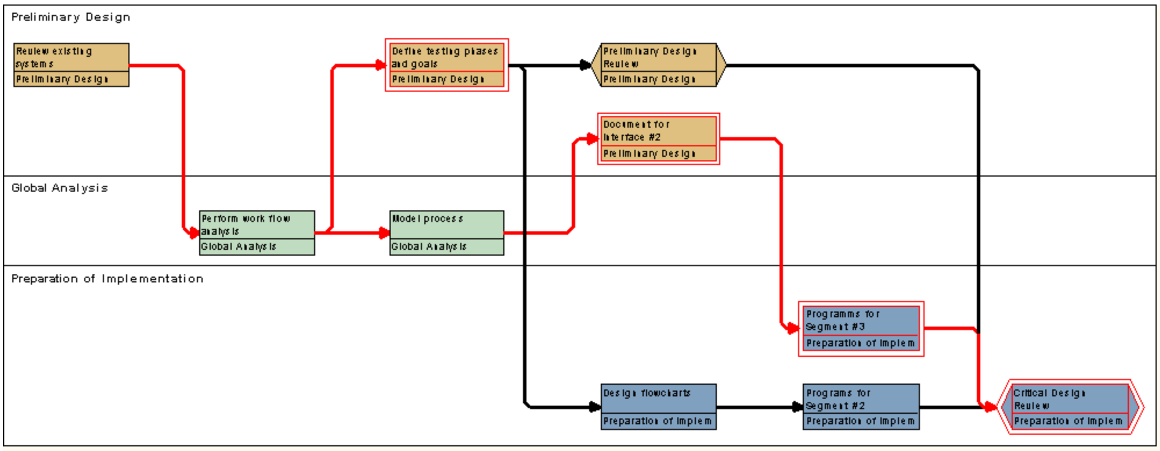
blank = separator

"Group A" = full text

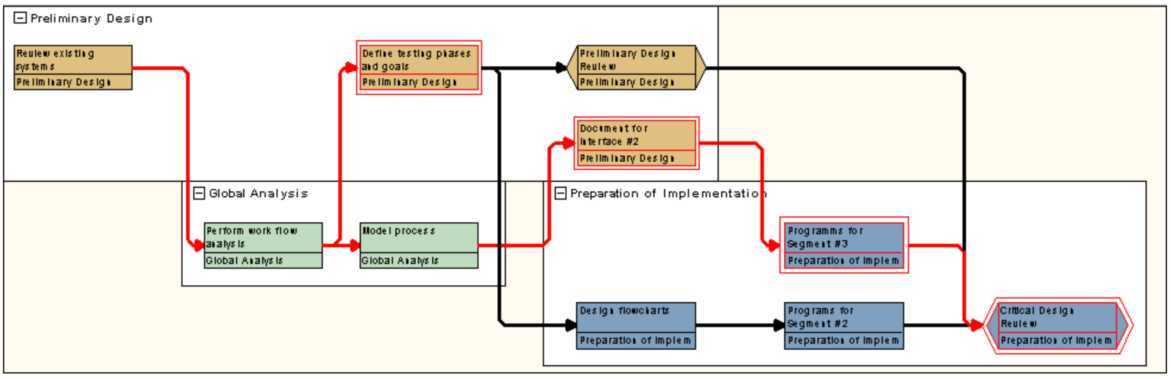
> Modes: Grouping and Clustering

You have the choice between two visualization modes:

- **Grouping:** normal visualization of groups (The width and height of each group is determined by the node positions. Each group needs the full width or height respectively of the net diagram)
- **Clustering:** The nodes are grouped very space-sparing, and the groups are placed freely in the net diagram. In the cluster mode groups can be collapsed or expanded by clicking on the plus or minus symbol (only if the property VcNet.GroupInteractionsAllowed is activated). Collapsed groups can be moved with the mouse like nodes.



Example for grouping mode



Example for cluster mode

In both modi you can move, delete or create nodes interactively.

The visualization mode can be set on the **Grouping** property page (**Mode**) or via API (**VcNet.GroupingType**).

> Sorting of Groups

The **Group sorting** section lets you enter the settings for group sorting.

If you select the **sorting by field** option, you can select the field that the groups are sorted by. In addition, the **ascending** and **descending** options are activated, that let you choose the desired order.

If you select the **by appearance in file** option, the groups will be displayed in the sequence of their occurrence in the file.

> Appearance of Groups

You can specify the group appearance: the background color of the groups, the border lines between the groups and the font of the group titles.

> Events

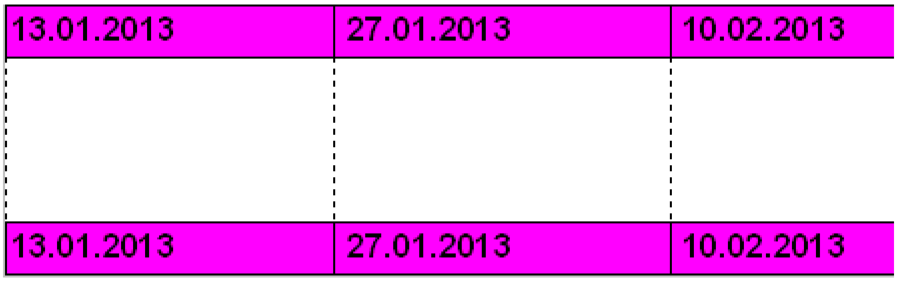
You can react to the events:

- **VcGroupCreated**
- **VcGroupDeleting**
- **VcGroupLeftClicking**
- **VcGroupLeftDoubleClicking**
- **VcGroupModifying**
- **VcGroupModified**
- **VcGroupRightClicking**

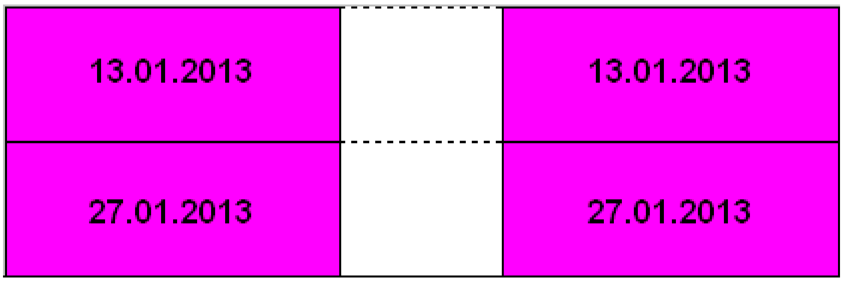
3.9 In-Flow Grouping

If a network has an in-flow grouping, its nodes are displayed in a certain order. This order can be chronological (time-oriented network) or controlled by a certain code (data field).

In time-oriented networks (in-flow grouping by a date field), the nodes are arranged chronologically in x direction (left-to-right orientation) or in y direction (top-to-bottom orientation) and grouped in time intervals. The length of these intervals you can specify yourself.

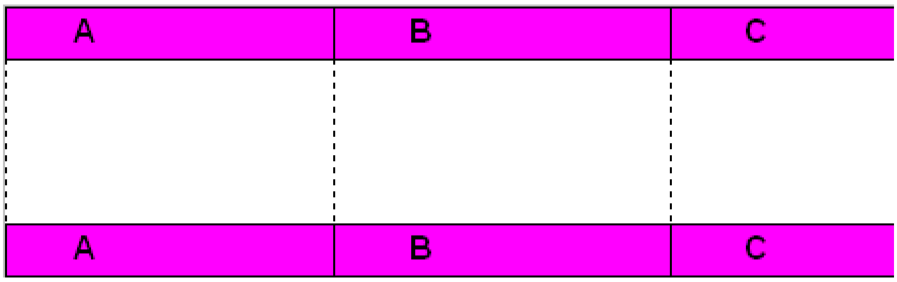


Example for a time-oriented network with a left-to-right-orientation

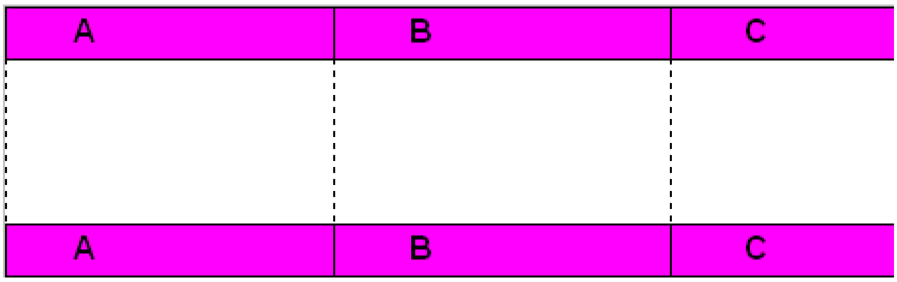


Example for a time-oriented network with a top-to-bottom orientation

Furthermore a grouping by a certain data field is possible:

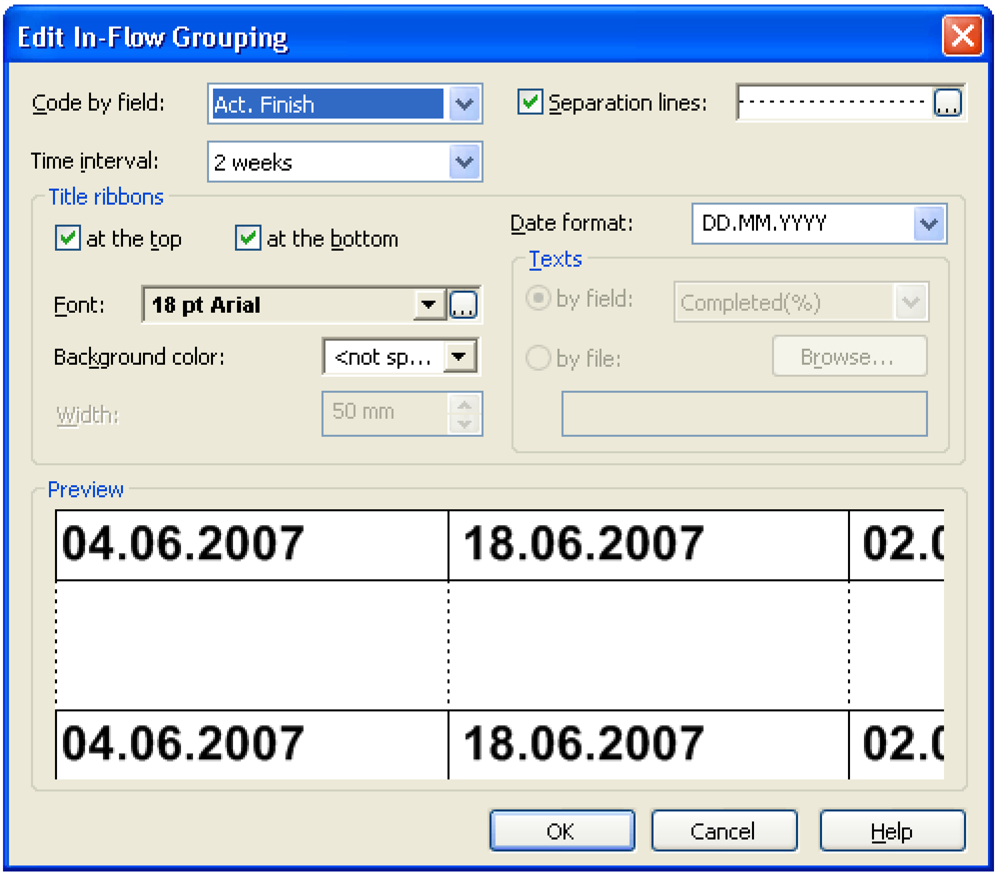


Example for a diagram with an in-flow grouping by a data field in a left-to-right-orientation



Example for a diagram with an in-flow grouping by a data field in a top-to-bottom orientation

In the **Edit In-Flow Grouping** dialog you can define the criteria for the in-flow grouping and the layout.



Edit In-Flow Grouping

Code by field: Act. Finish

☑ Separation lines:

Time interval: 2 weeks

Title ribbons

☑ at the top ☑ at the bottom

Font: 18 pt Arial

Background color: <not sp...

Width: 50 mm

Date format: DD.MM.YYYY

Texts

● by field: Completed(%)

○ by file:

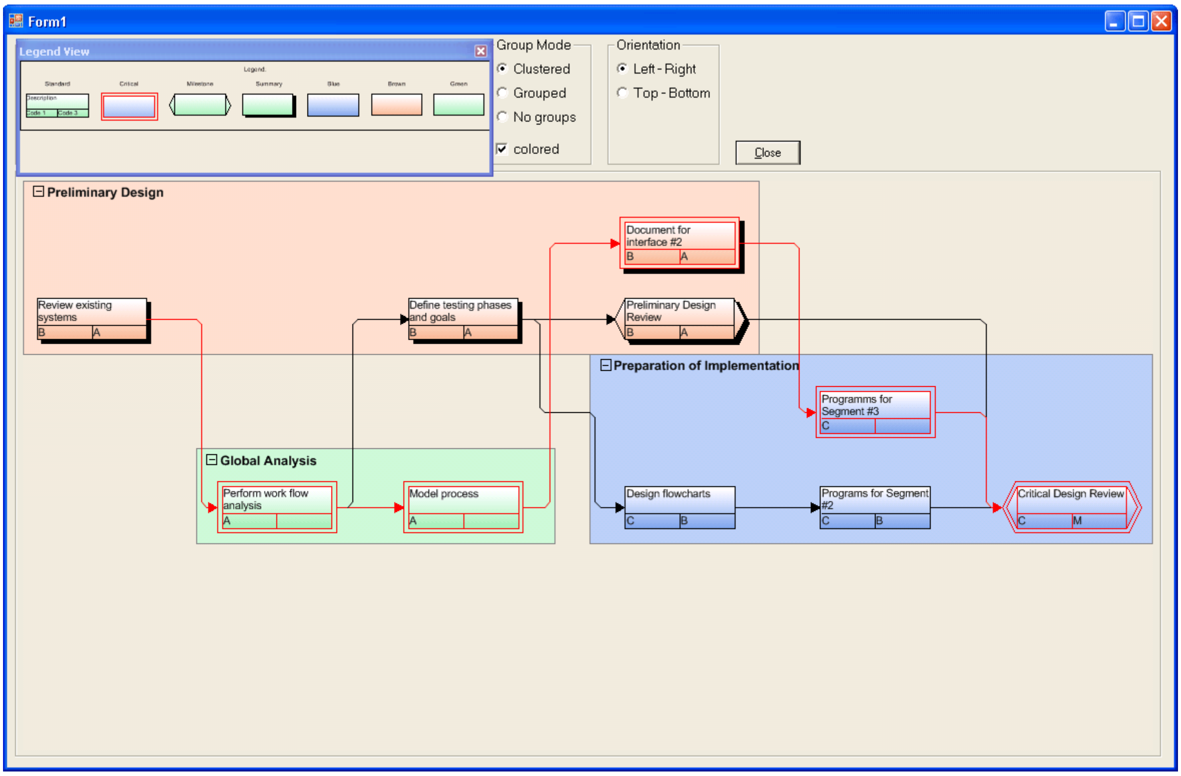
You reach this dialog via the **Nodes** property page.

If you choose an in-flow grouping for a diagram with a left-to-right orientation, you can display an annotated ribbon at the top and/or bottom of the diagram area. For diagrams with a top-to-bottom orientation you can display an annotated ribbon at the left and/or right side of the diagram area.

Furthermore you can specify whether vertical or horizontal separation lines are used to mark the group borders. You can specify the color and the annotation of the ribbons as well as the attributes of the separation lines.

3.10 Legend View

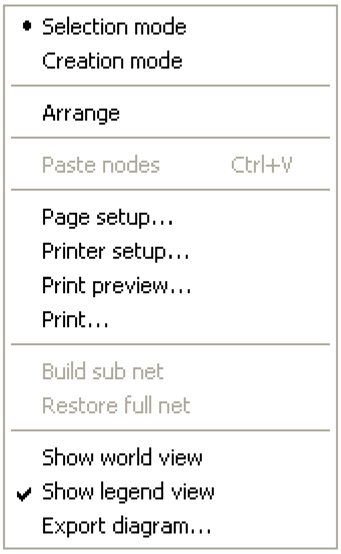
The legend view is an additional window that lets you display a legend on the screen. The layout of the legend can be specified with the legend attributes of **VcBorderBox** or in the dialog **Legend attributes** which can be reached from the **Border area** property page



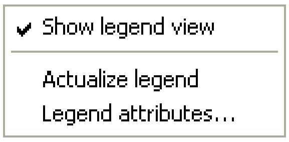
Group Mode
● Clustered
○ Grouped
○ No groups
☑ colored

Orientation
● Left - Right
○ Top - Bottom

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



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



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

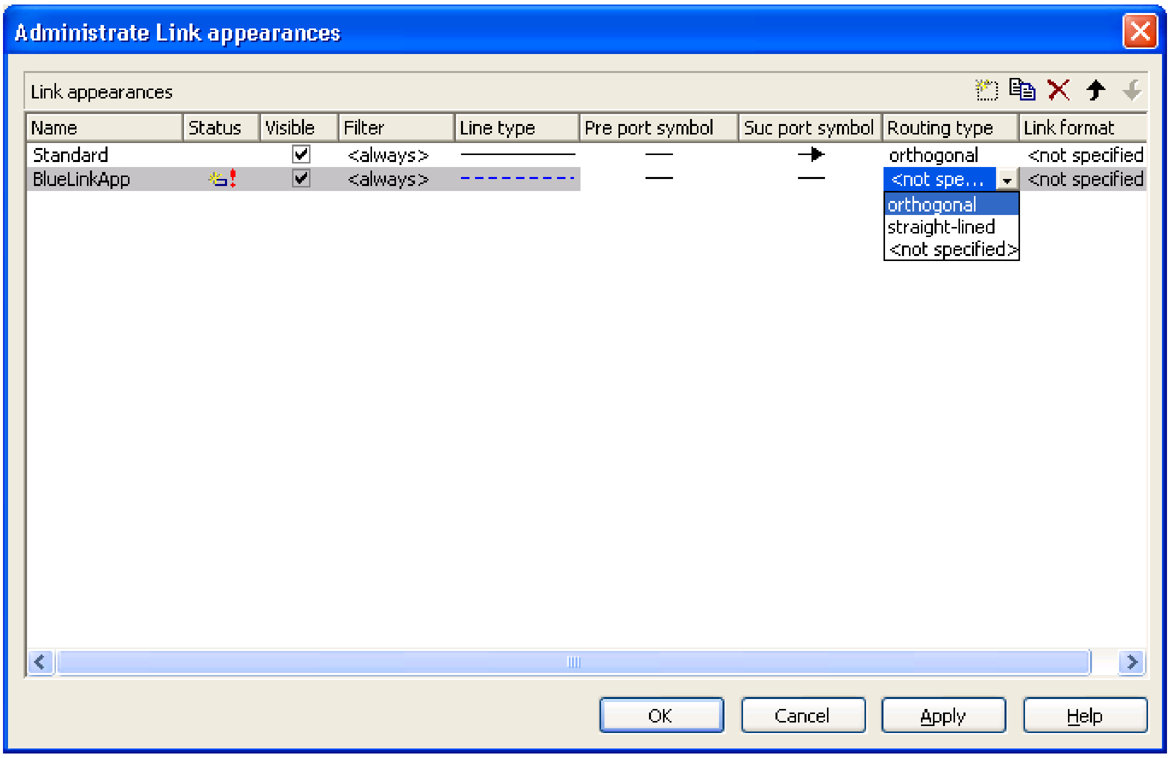
The refreshing of the legend is needed after modifications in the chart, such as adding or deleting nodes, because they are not displayed automatically. The refreshing can also be carried out by switching off and on the legend view. This concerns the loading of nodes as well. If on the property page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

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

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

3.11 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.



Name	Status	Visible	Filter	Line type	Pre port symbol	Suc port symbol	Routing type	Link format
Standard		☑	<always>	———	—	—▶	orthogonal	<not specified
BlueLinkApp	!	☑	<always>	- - - - - - -	—	—	<not spe... orthogonal straight-lined <not specified>	<not specified

The list shows the **Name** column, that displays the names of the appearances, the **Filters** column that lists the associated filters and the **Line type** column, that shows the line types defined.

> Defining a Link Appearance

Please click on the **New...** line to create a new link appearance.

> Deleting Link Appearances

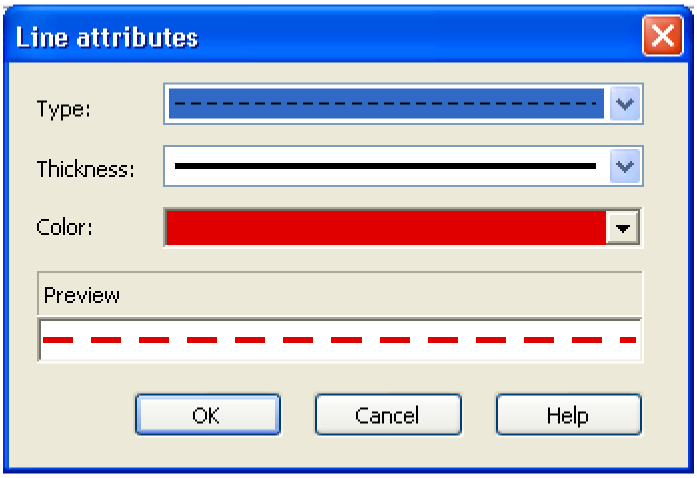
You can delete a link appearance from the **Appearances** list via the Del key.

> Defining Filters

For selecting the filter used with a link appearance, click on an entry in the **Filter** column. Select a filter from the appearing combo box, that is marked by an arrow-down button. You can edit the filter in the **Administrate Filters** dialog box by clicking on the **Edit** button. There you can generate new filters, or copy, edit and delete existing filters. Modifications on a filter are not confined to the link appearance that the filter is associated with, but are valid for all link appearances throughout your project.

> Specifying the Line Attributes of Links

When you click on the entry of the field **Line type**, an **Edit** button will occur by which you can get to the **Line Attributes** dialog. There you can set the color, type and thickness of the line.



> Further Specifications for the Link Appearances

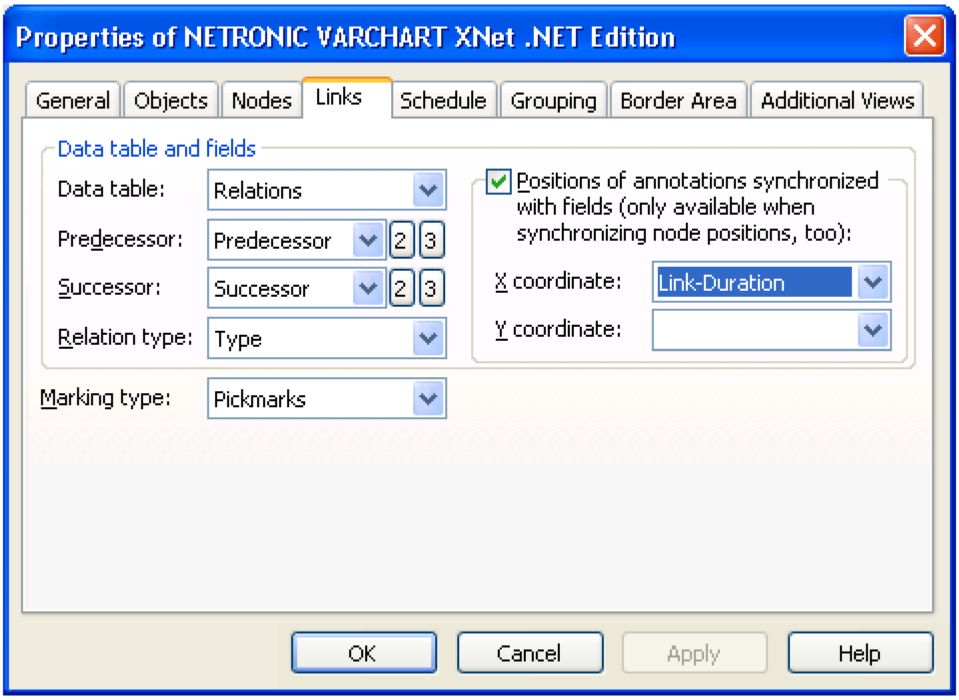
Fur further information about link appearances please see chapter 4.28 "The Administrate Link Appearances Dialog Box".

3.12 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.

> Specifying Links

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



Properties of NETRONIC VARCHART XNet .NET Edition

Data table and fields

Data table: Relations

Predecessor: Predecessor

Successor: Successor

Relation type: Type

☑ Positions of annotations synchronized with fields (only available when synchronizing node positions, too):

X coordinate: Link-Duration

Y coordinate:

Marking type: Pickmarks

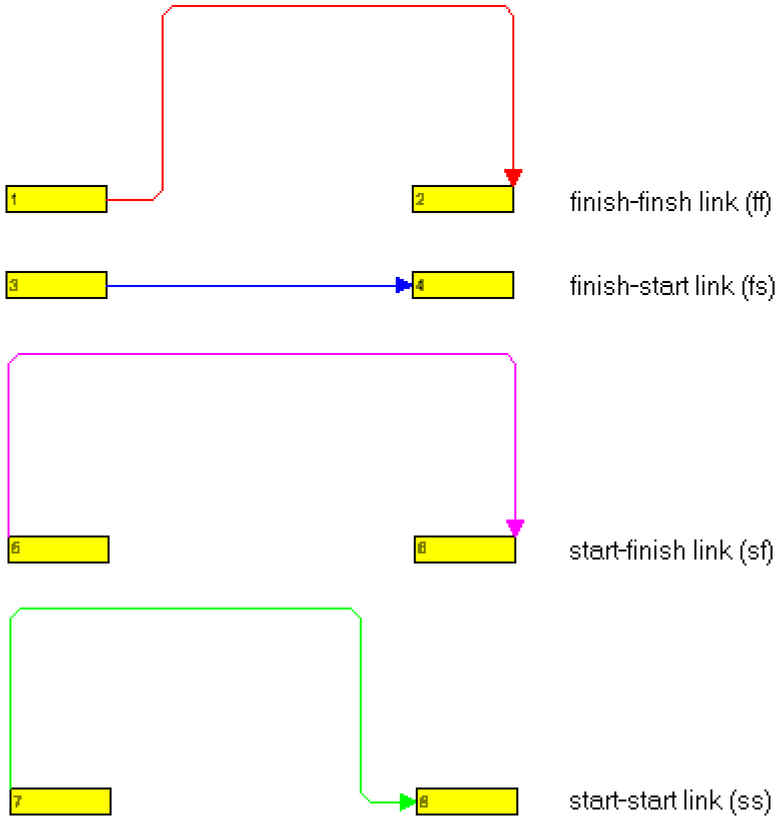
You can specify data fields in which the identifications of the predecessor/ successor nodes and the relation types are to be stored. If the identification of a predecessor or successor node consists of more than one field, the corresponding link has to match this identification. That means that according 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

Beside, you can define what a link should look like by setting the options of one or more link appearances. For each link appearance, you can select a filter, the predecessor/ successor layer, the line type and the predecessor/ successor port symbols.

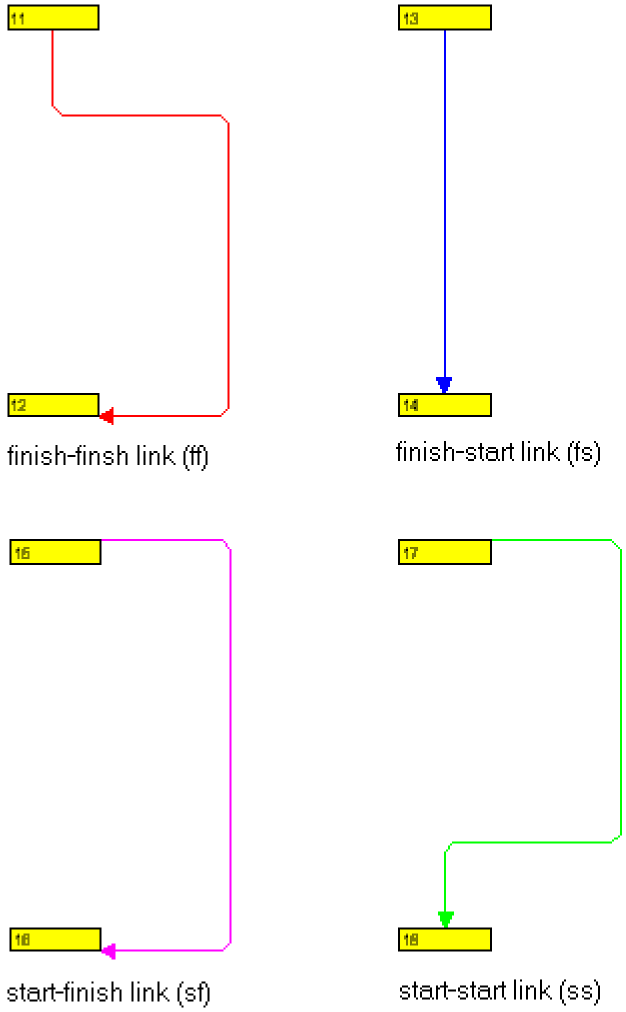
> Types of Links

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

The different types of link appearances are shown in the below pictures:



Left-to-right orientation



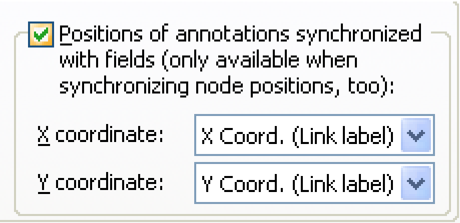
Top-to-bottom orientation

> Positions of Link Annotations

To make positions of link annotations reloadable, they need to be synchronized with corresponding data fields. For this, on the **Links** property page please tick the **Positions of annotations synchronized with data fields** check box and then select data fields that the X and Y coordinates are stored to.

- for the x coordinate: "X Coord. (link label)"
- for the y coordinate: "Y Coord. (link label)"

The appropriate data fields have to be defined before (see "Tutorial: Preparing the Interface").



☑ Positions of annotations synchronized with fields (only available when synchronizing node positions, too):

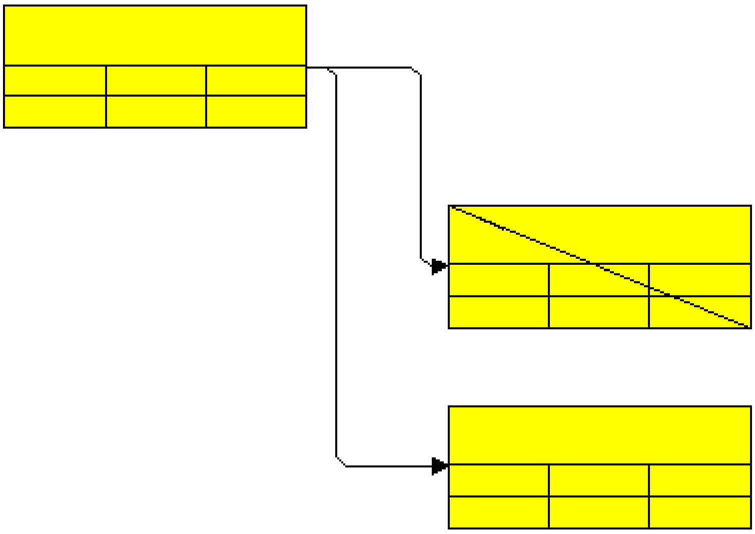
X coordinate: X Coord. (Link label)

Y coordinate: Y Coord. (Link label)

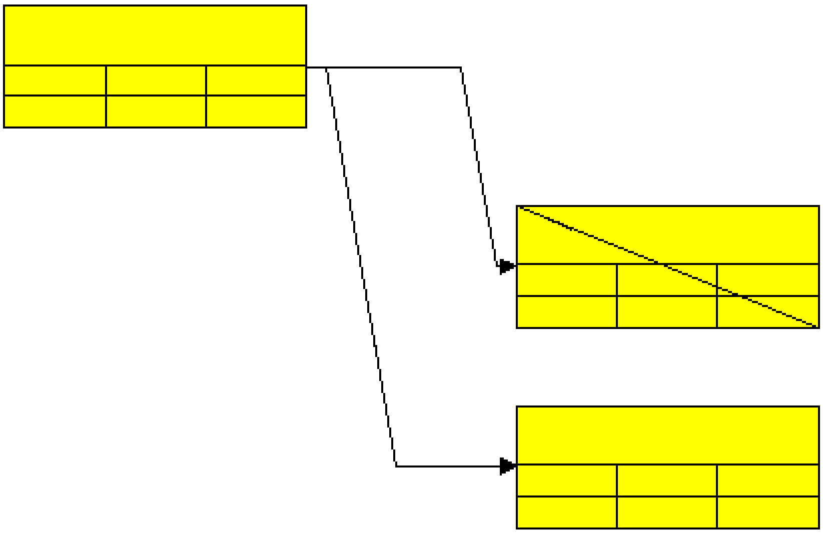
The values of the above data fields you can retrieve and, if necessary, modify via the dialog **Edit Link**. You can find an example of positions of node and link annotations in the "Nodes" paragraph of this chapter.

> Orthogonal/Oblique Link Lines

If on the **General** property page the option **Show oblique tracks on links** has been choosen, the link lines will be oblique, connecting the short horizontal line sections. Otherwise the link lines will be orthogonal. Beside, this feature can be specified with the help of the VcNet property **ObliqueTracksOnLinks**.



orthogonal link lines

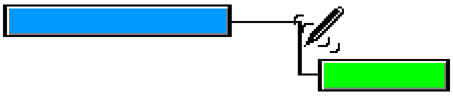


oblique link lines

> Generating Links

If on the **General** property page the option **Allow creation of nodes and links** has been chosen, the user will be able to create new links interactively by dragging the mouse from a node to another one.

If in addition the option **Edit new links** was ticked, the dialog **Edit Link** will pop up as soon as the mouse button has been released. The data of the node is displayed and can be edited.



Beside, you can generate a link via the API by the **InsertLinkRecord** method. Any link that is created will invoke the event **VcLinkCreating**.

> Marking Links

During runtime, when you are in the **Selection mode**, you can mark links by clicking on them with the left mouse button. By simultaneously pressing the Ctrl key you can mark several links.

> Editing Links

You can edit a link by clicking the right mouse button on it and then select the menu item **Edit**. You will get to the **Edit Link** dialog box, where you can modify the link data.

> Deleting Links

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

> Events

You can react to the events:

- **VcLinkCreating**
- **VcLinkCreated**
- **VcLinkDeleting**
- **VcLinkDeleted**
- **VcLinksLeftClicking**

- **VcLinksLeftDoubleClicking**
- **VcLinksMarked**
- **VcLinksMarking**
- **VcLinkModified**
- **VcLinkModifying**
- **VcLinksRightClicking**

3.13 Localization of Text Output

The **VcTextEntrySupplying** event allows to replace all items in context menus, dialogs, information boxes and error messages, 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

```
VcNet1.TextEntrySupplyingEventEnabled = True
```

Example Code C#

```
vcNet1.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 = "KW"
      Case VcTextEntryIndex.vcTXERibDay0
         e.Text = "Mo"
      Case VcTextEntryIndex.vcTXERibMon8
         e.Text = "September"
      Case VcTextEntryIndex.vcTXERibQuar3
         e.Text = "3. Quartal"
   End Select
End Sub
```

Example Code C#

```
private void vcNet1_VcTextEntrySupplying(object sender,
NETRONIC.XNet.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;
      case VcTextEntryIndex.vcTXERibQuar3:
         e.Text = "Quarter 3";
         break;
      }
   }
```

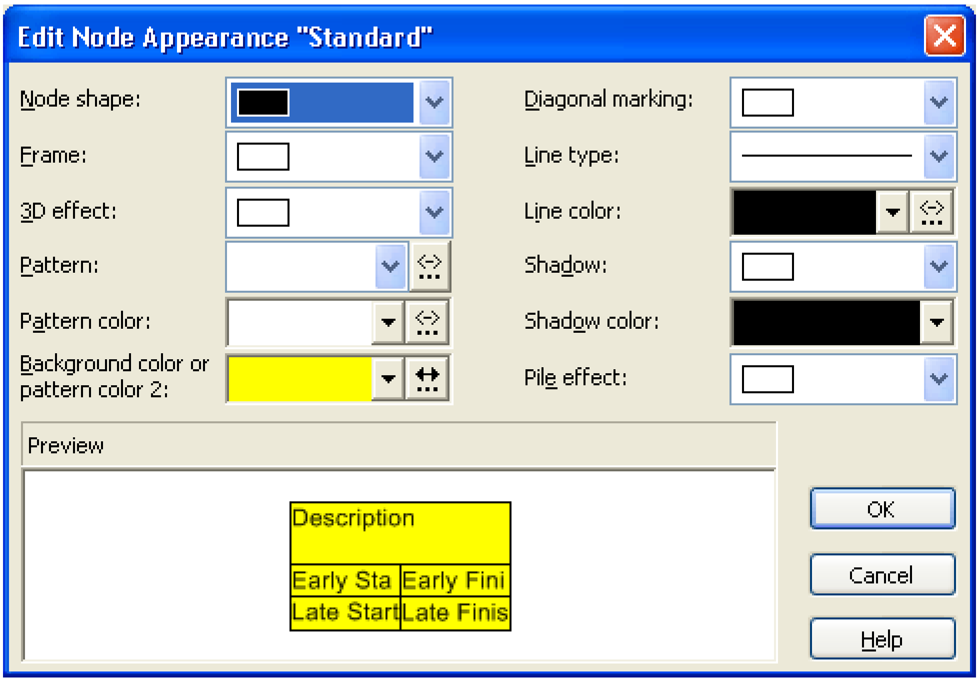
3.14 Maps

The node appearances and the node formats can be assigned to the nodes in dependence on their data. The data-controlled assignment is defined via maps.

> Node Appearance in Dependence on Node Data

For each node appearance you can assign the pattern, the pattern color, the background color or pattern color 2 and the line color data dependant via a map.

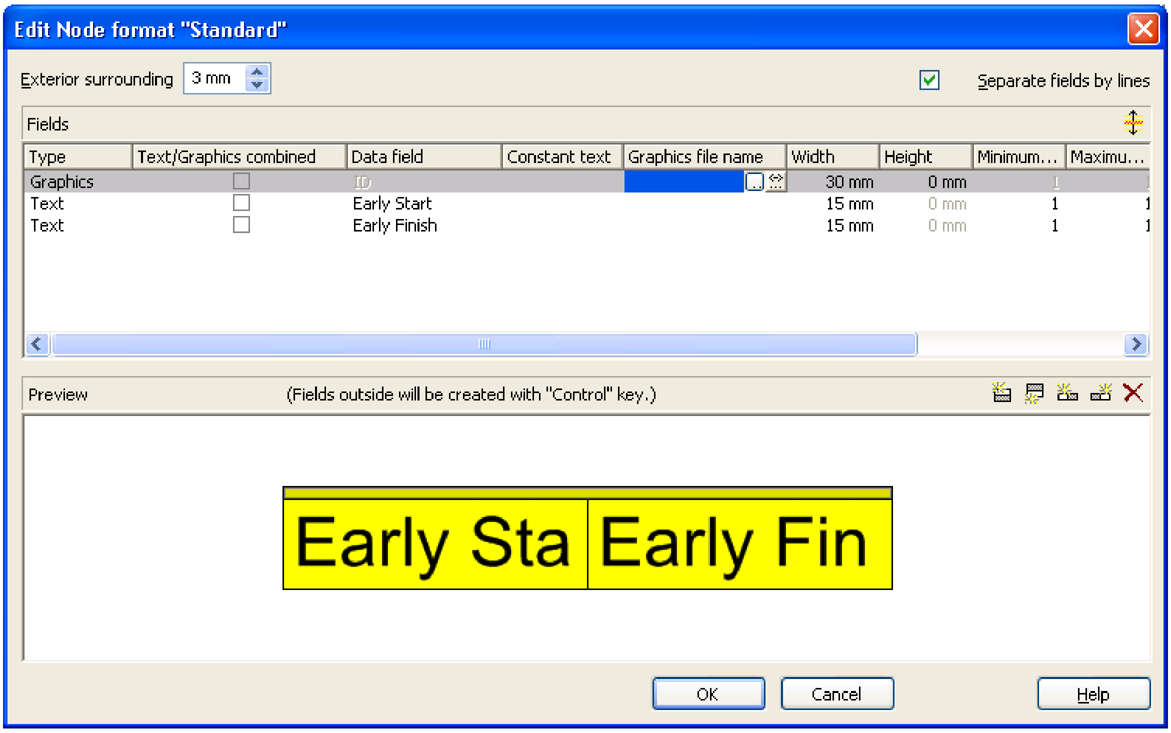
In the **Edit Node Appearance** dialog box, click on the second button besides the **Background color** field or **Line color** field respectively (↔).



Then you will reach the **Configure Mapping** dialog box.

> Graphics file for node formats in dependence on node data

For each node format the graphics file to be displayed in a format field can be specified in dependence on the node data via a map.



Edit Node format "Standard"

Exterior surrounding 3 mm ☑ Separate fields by lines

Fields

Type	Text/Graphics combined	Data field	Constant text	Graphics file name	Width	Height	Minimum...	Maximu...
Graphics	□	ID			30 mm	0 mm	1	1
Text	□	Early Start			15 mm	0 mm	1	1
Text	□	Early Finish			15 mm	0 mm	1	1

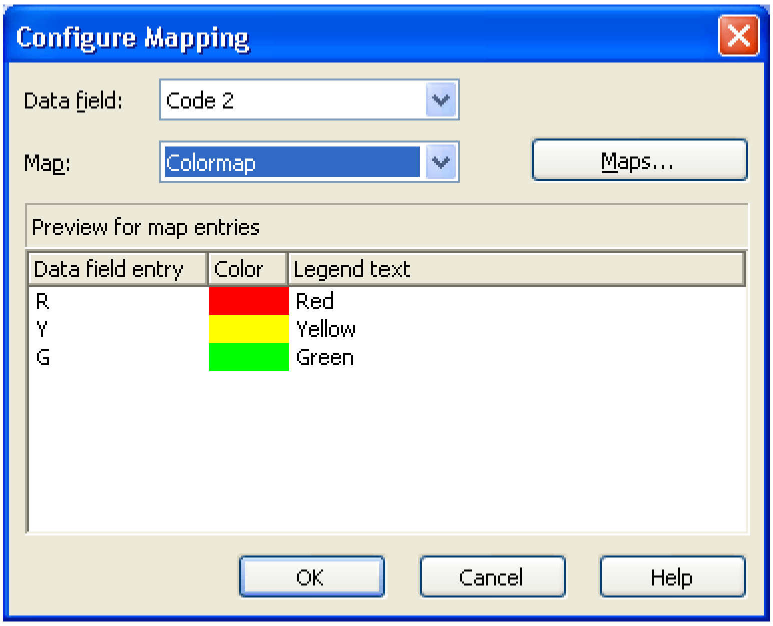
Preview (Fields outside will be created with "Control" key.)

To configure a mapping from data field entries of the type graphics to graphics files, click on the second button in the **Graphics File** field. Then the **Configure Mapping** dialog box will open.

If a map was configured, a symbol ( ) is displayed beside the file name symbol as soon as you leave the **Graphics File** field.

> Configuring Mapping

The **Configure Mapping** dialog lets you assign the background color of a node appearance or the graphics file of a node format in dependence on the node data.



Data field: Code 2

Map: Colormap

Preview for map entries

Data field entry	Color	Legend text
R	🟥	Red
Y	🟨	Yellow
G	🟩	Green

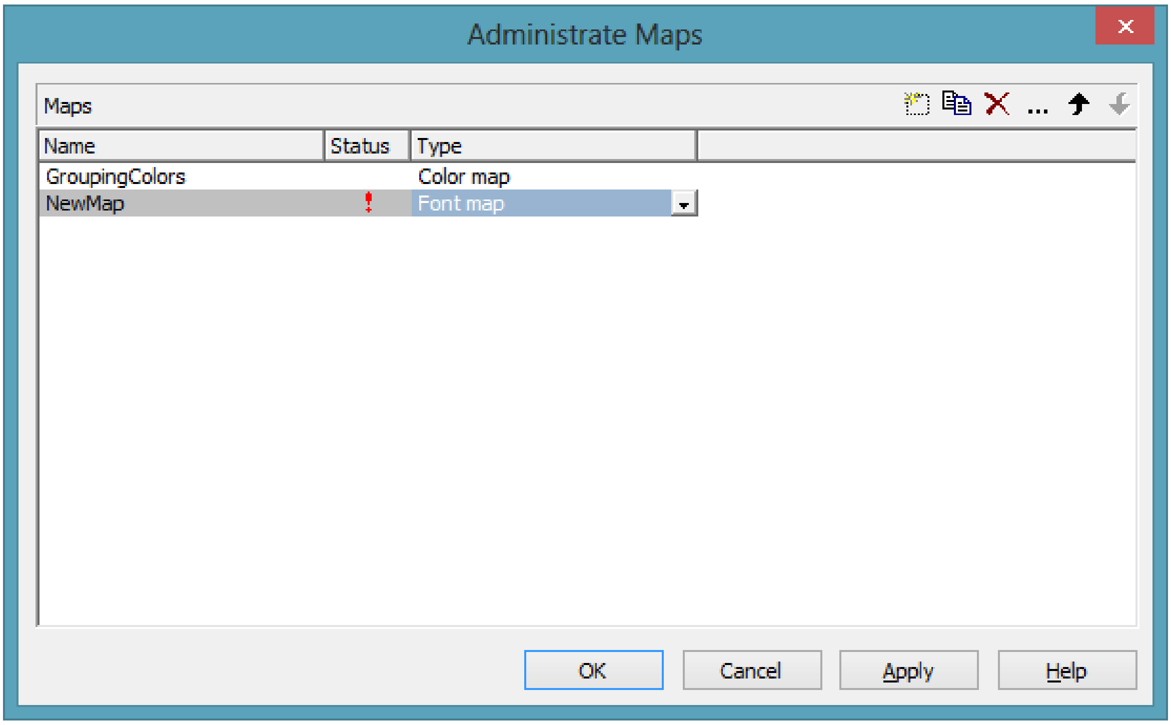
From the first combobox, select the **Data field** which a map is to be assigned. From the second combobox, select the **Map** that assigns a graphics file or a color respectively and a legend text to the data field entries.

The preview shows the mapping of the graphics file or the color respectively and of the legend text to each data field entry.

> 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.

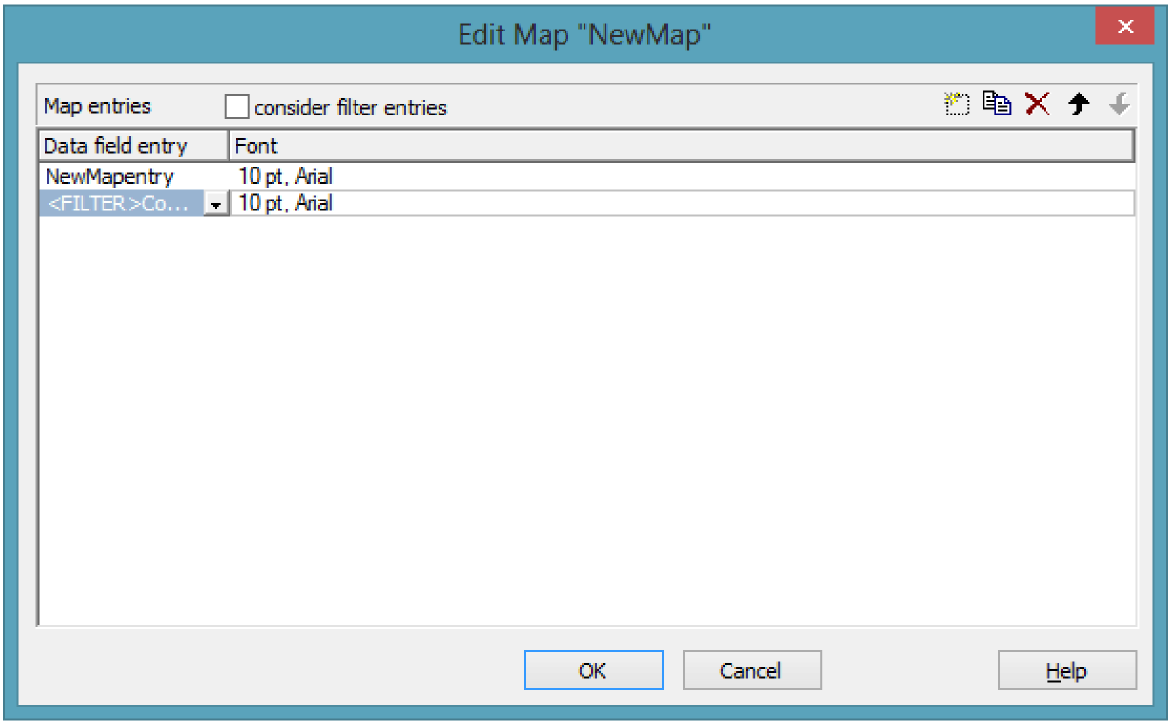
You can choose between different types of maps, according to whether colors, patterns, graphic files, fonts, lengths or numbers are to be allocated to data field contents.



Name	Status	Type
GroupingColors		Color map
NewMap	!	Font map

> 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.



Data field entry	Font
NewMapentry	10 pt, Arial
<FILTER>Co...	10 pt, Arial

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

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

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

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

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.

3.15 Node

A node is defined by a node record of the Maindata table. Nodes can be loaded via the API or generated interactively by the user.

> Generating Nodes

If on the **General** property page the option **Node and link creation allowed** has been chosen, the user will be able to create new nodes interactively by a mouse click.

If in addition the check box **Node creation with dialog** was ticked, the dialog **Edit Data** will open as soon as a node has been created via mouse click. The data of the node are displayed in the **Edit Date** dialog box and you can edit them.

Beside, you can generate a node by the API method **InsertNodeRecord**. Any node interactively created will invoke the event **VcNodeCreating** to inform the application.

> Marking Nodes

On the **Nodes** property page you can set a pattern and color to mark nodes. Just select an option from the **Marking type** combo box:

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

Note: If you select "No Mark", there will be no graphical pattern to mark a node.

Any marking/demarking of nodes will invoke the event **VcNodesMarking**. The end of an marking/demarking operation will invoke the event **VcNodesMarked**.

> Deleting Nodes

A node or several nodes can be deleted by pressing the Shift or Ctrl key and simultaneously marking them. Then press the right mouse button to pop up a context menu where you can select the menu item **Delete** or **Cut**. Marked nodes can also be deleted by the Del key.

Deleting nodes interactively will invoke the event **VcNodeDeleting**.

Beside, you can delete nodes by the VARCHART ActiveX method **DeleteNodeRecord** or by the VcNode method **DeleteNode**.

> Events

You can react to the events:

- **VcNodeCreating**
- **VcNodeCreated**
- **VcNodeDeleting**
- **VcNodeLeftClicking**
- **VcNodeLeftDoubleClicking**
- **VcNodeModified**
- **VcNodeModifiedEx**
- **VcNodeRightClicking**
- **VcNodesMarked**
- **VcNodesMarking**

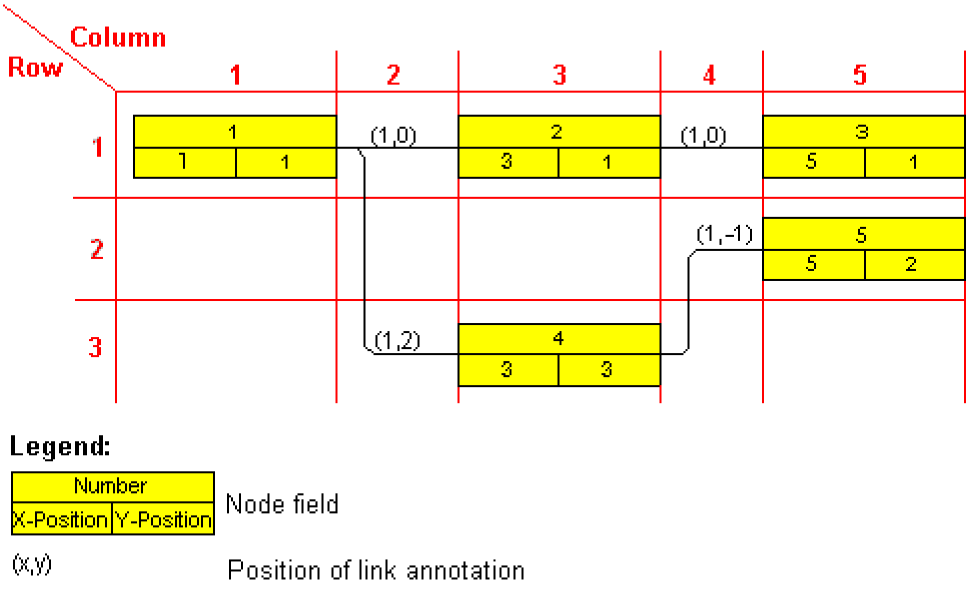
> Positions of Nodes

Positions of nodes and of link annotations are stored as coordinates in a matrix.

The X and Y coordinates of a node represent the absolute position of the node in the matrix. In contrast, the X and Y coordinates of a link annotation refer to the position of the predecessor node.

The top left postion of the matrix is defined as (X,Y) = (1,1) and is reserved for nodes. All other node coordinates are generated by continuously adding 1 to the coordinates of the top left position. Except for the top left position any position may contain a node or a link annotation.

Node coordinates, that represent absolute values, always show positive figures, whereas link annotation coordinates, that represent relative values may show negative figures. Link annotation coordinates cannot be placed in the (0,0) position.

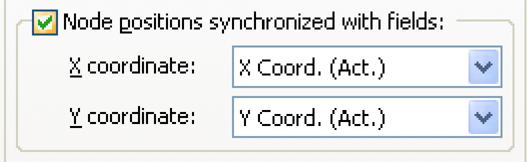


> Saving and Loading Node Positions

If you wish to restore the node positions of a diagram, you need to store them to data fields before. To synchronize the positions with their data fields, on the **Nodes** property page activate the check box **Node positions synchronized with data fields** and select the following data fields:

- for the X coordinate: "X Coord. (Act.)"
- for the Y coordinate: "Y Coord. (Act.)"

These fields need to have been defined when preparing the interface. Also see "Tutorial: Preparing the interface"



☑ Node positions synchronized with fields:

X coordinate: X Coord. (Act.)

Y coordinate: Y Coord. (Act.)

> The Rank of a Node

The rank of a node is a figure defined according to the following rules:

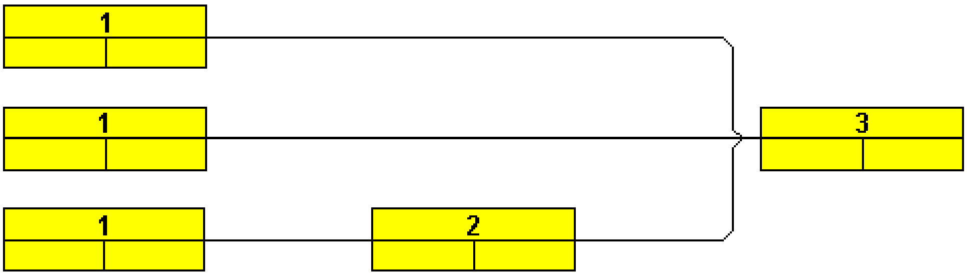
The rank of an unpreceded node equals 1. The rank of a node that has predecessors equals 1 plus the rank number of the predecessor of the top rank.

This definition avoids cyclic structures (loops) to occur in a network diagram.

Examples:

- The rank of a node, the predecessor of which is unpreceded equals 1+1=2.

- The rank of a node that has three predecessors of the ranks 1, 1 and 2 equals 1+2=3 (see sketch).



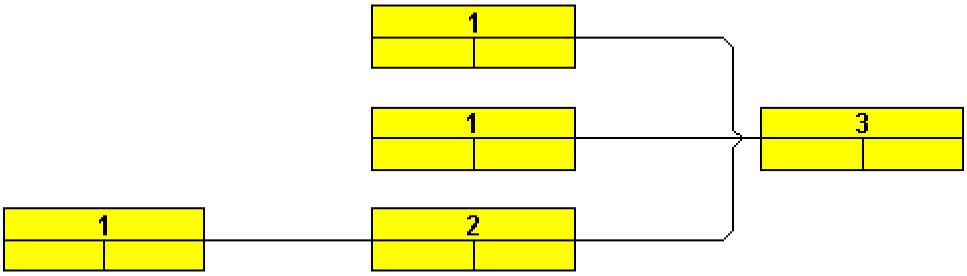
Ranks of nodes oriented from left to right

This is how ranks of nodes work:

- In a left-to-right orientation the top rank of all nodes in a node column equals the column number (link annotation columns not included).
- In a top-to-bottom orientation the top rank of all nodes in a node row equals the row number (link annotation rows not included).

Ranks are calculated by clicking on the **Arrange** item of the diagram context menu. They serve as a base to the layout algorithm to position the nodes in the overall orientation. If cyclic structures exist in the chart, VARCHART XNet will identify them by a separate algorithm and ignore them temporarily. This makes the layout look natural. The links ignored will appear as returning links.

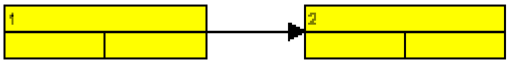
By the property **ShortenedLinks** or by ticking the **Shorten links on arrange** check box on the **General** property page the positions will be positioned far right or far below to reduce the length of links to a minimum.



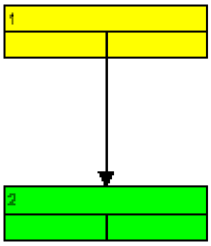
> Auxiliary Nodes

In some applications it may be useful not to keep all nodes in the same orientation. In a left-to-right orientation you can put nodes above or below their predecessors, in a top-to-bottom orientation you can place them left or right of their predecessors. The way to do this is to diminish the rank number of a node.

In a left-to-right arrangement the auxiliary node, the rank number of which was diminished by 1, is placed below or above its predecessor instead of left or right of it.

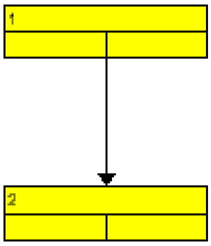


Nodes of rank 1 resp. rank 2

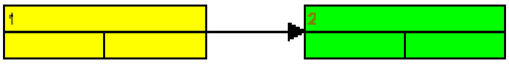


*The rank number of the second node was diminished by 1. Then the command **Arrange** was invoked.*

In a top-to-bottom arrangement the auxiliary node, the rank number of which was diminished by 1, is placed left or right of its predecessor instead of below or above it.



Nodes of rank 1 resp. rank 2



*The rank number of the second node was diminished by 1. After this, the command **Arrange** was invoked.*

To alter the rank of a node, the data field "Auxiliary node" has been introduced. The entry in the "Auxiliary node" data field will set the position of the node, allowing the values 0, 1, 2 or 3.

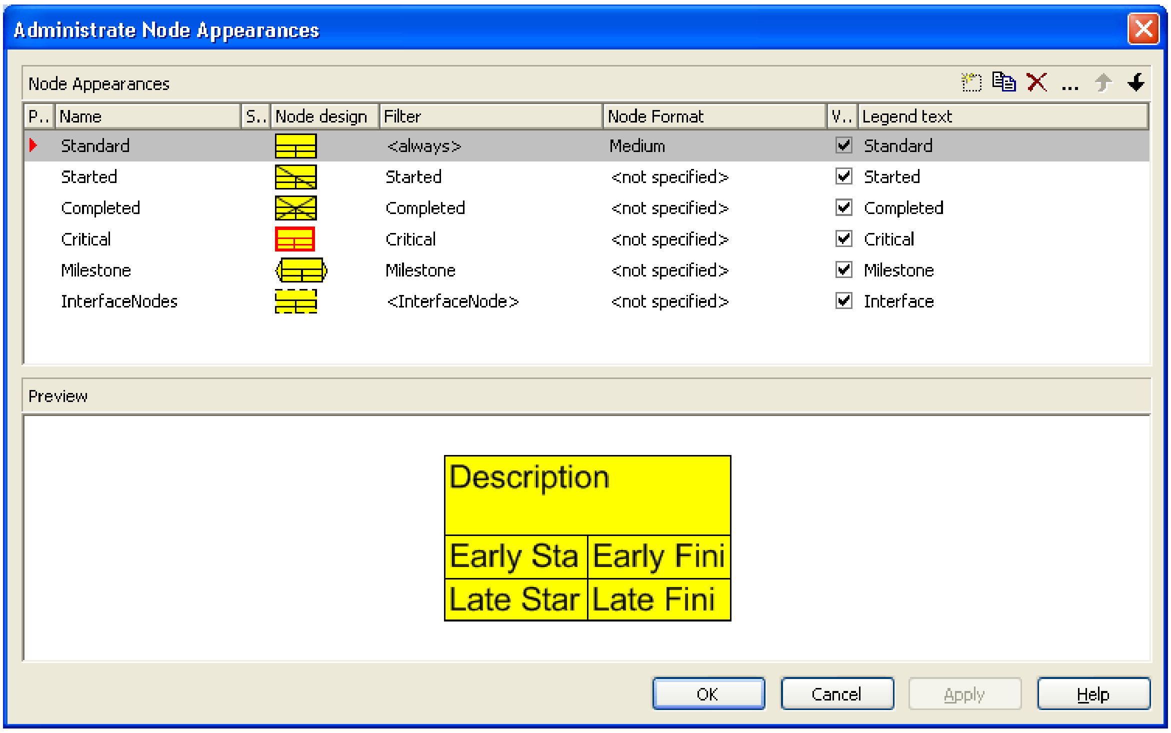
Value in the field "Auxiliary Nodes"	Top-to-bottom orientation	Left-to-right orientation
0	The rank number of the auxiliary node is not diminished.	The rank number of the auxiliary node is not diminished.
1	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears left or right of its predecessor instead of below.	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears above or below its predecessor instead of left or right of it.

Value in the field "Auxiliary Nodes"	Top-to-bottom orientation	Left-to-right orientation
2	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears to the left of its predecessor.	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears above its predecessor.
3	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears to the right of its predecessor.	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears below its predecessor.

To place auxiliary nodes in the same rank as their predecessors, please tick the check box **Nodes arranged on same rank as their predecessors in accordance to data field** on the **Nodes** property page. Select the "Auxiliary Node" data field from the combo box. You may have to define this field on the **DataDefinition** property page in case it doesn't exist. You may enter the values **0, 1, 2** or **3**. It depends on the entry of the "Auxiliary Node" field, whether or not a node is placed in the same rank as its predecessor.

3.16 Node Appearance

You can define node appearances in dependency on their data. For example, you may want nodes of Department A to show a red background, nodes of Department B a blue background etc. A defined set of graphical attributes is called an appearance. A node may have several appearances of different priorities. You can create or modify an appearance by clicking on the **Node Appearances** button on the **Objects** property page to get to the **Administrate Node Appearances** dialog. There you can edit, copy or delete node appearances or create new node appearances or modify the working off order.



P..	Name	S..	Node design	Filter	Node Format	V..	Legend text
▶	Standard		🟨	<always>	Medium	☑	Standard
	Started		🟨	Started	<not specified>	☑	Started
	Completed		🟨	Completed	<not specified>	☑	Completed
	Critical		🟥	Critical	<not specified>	☑	Critical
	Milestone		🟨	Milestone	<not specified>	☑	Milestone
	InterfaceNodes		🟨	<InterfaceNode>	<not specified>	☑	Interface

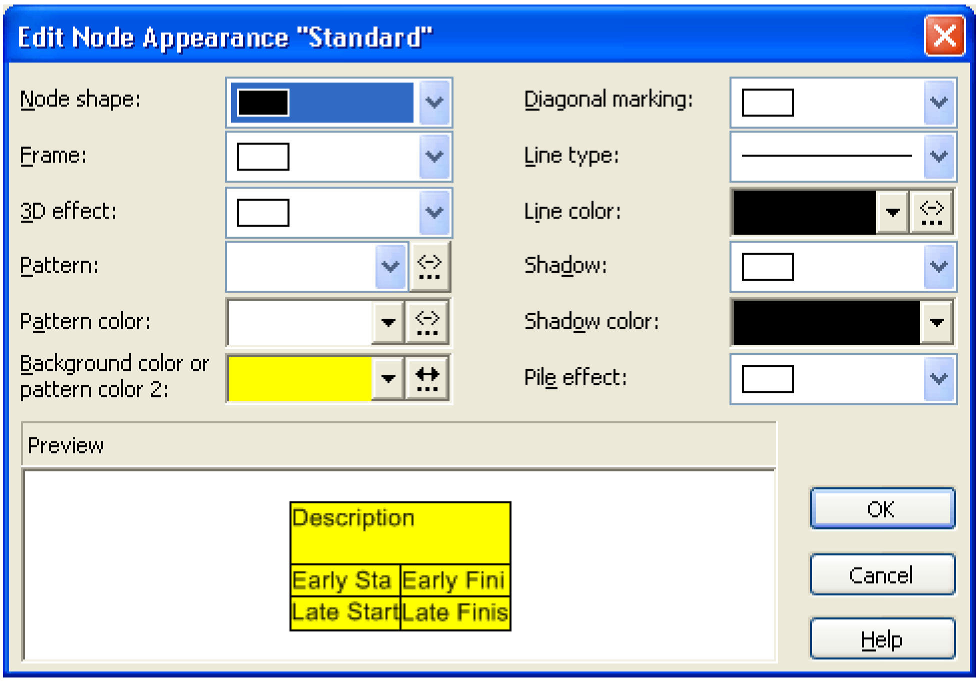
Description	
Early Sta	Early Fini
Late Star	Late Fini

A node appearance always is combined with a node format and a filter. A filter consists of conditions that are to be fulfilled by a node for the appearance to apply. For example, the appearance "Marked" is combined with the filter "Marked", that selects all marked nodes.

If a node fulfils the criteria of several node appearances, all of them will apply to the node. Each of them is of a different priority. The appearance at the bottom of the table is assigned last and will override all others. The "Standard" appearance applies to all nodes. It cannot be deleted. By default, it appears at the top.

↑ ↓ You can modify the order of working off the node appearances with the help of the arrow buttons.

To edit a node appearance, click on the **Edit node appearance** button or double-click on the **Node design** field. Then the following dialog box will appear:



Edit Node Appearance "Standard"

Node shape: ■

Frame: □

3D effect: □

Pattern:

Pattern color:

Background color or pattern color 2: 🟨

Diagonal marking: □

Line type: ——

Line color: ■

Shadow: □

Shadow color: ■

Pile effect: □

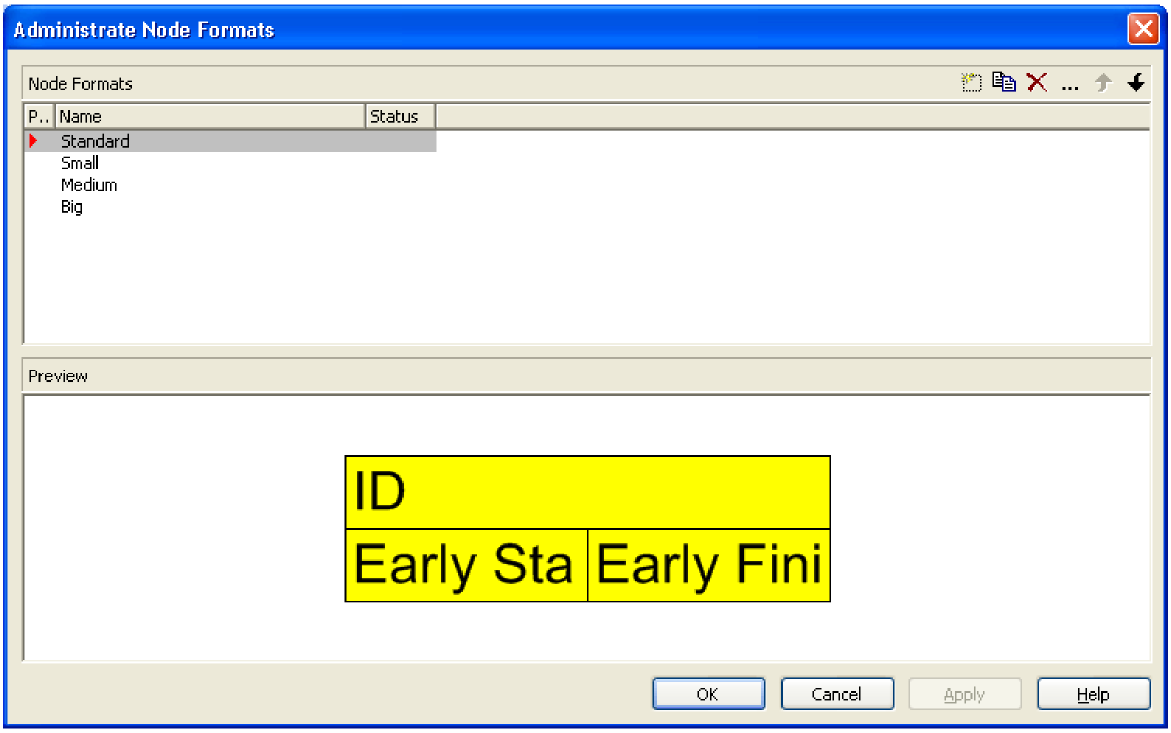
Preview

For each node appearance the background color and the line color can be assigned in dependence on the node data via a map. For details, please read the chapter "Important Concepts: Maps".

3.17 Node Format

A node appearance always is combined with a node format. The **Node format** select box in the **Administrate Node Appearances** dialog box lets you select the node format to be assigned to the node appearance.

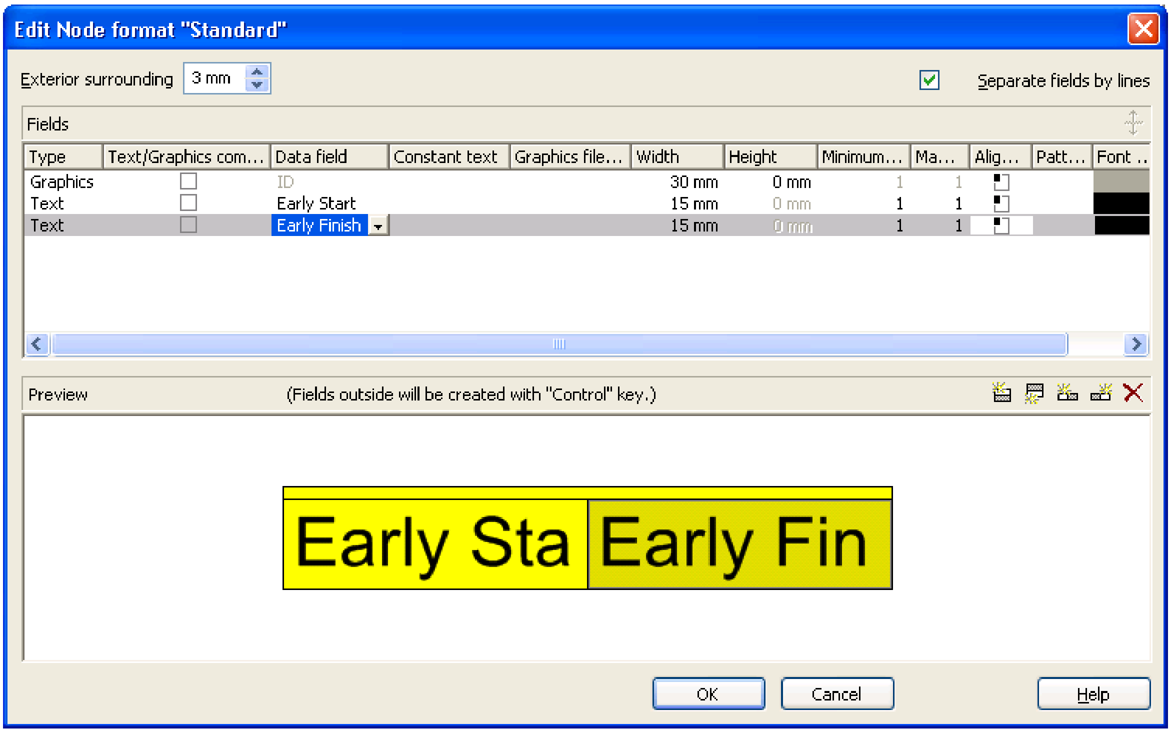
Node formats are managed in the **Administrate Node Formats** dialog, that you can get to via the the **Node formats** button in the **Objects** property page.



Administrate Node Formats

P..	Name	Status
▸	Standard	
	Small	
	Medium	
	Big	

You can edit the current node format by clicking on the **Edit node format** button that gets you to the **Edit Node Format** dialog.



Edit Node format "Standard"

Exterior surrounding 3 mm

☑ Separate fields by lines

Fields

Type	Text/Graphics com...	Data field	Constant text	Graphics file...	Width	Height	Minimum...	Ma...	Alig...	Patt...	Font ..
Graphics	□	ID			30 mm	0 mm	1	1	◰		▒
Text	□	Early Start			15 mm	0 mm	1	1	◰		■
Text	□	Early Finish			15 mm	0 mm	1	1	◰		■

Preview (Fields outside will be created with "Control" key.)

In this dialog box you can specify the following:

- whether the node fields are to be separated by lines
- the margins (distance between nodes or between a node and the margin of the chart. Unit: 1/100 mm)
- the field type: text or graphics
- for the type text: a data field whose content is to be displayed in the current field or a constant text
- for the type graphics: the name and directory of the graphics file that will be displayed in the current field
- the width and height of the marked field
- how many lines of text can be displayed in the current field
- alignment of the text/graphics of the current field
- the fill pattern and pattern colors of the current field
- the font attributes of the current field

> Date format of date fields

The date format of date fields you can set on the **General** property page.

> Displaying graphics in node fields

For each format field of the type graphics you can specify the graphics file to be displayed.

... To select a graphics file, click on the first button in the **Graphics file name** field. Then the Windows dialog box **Choose Graphics File** will open.

↔... To configure a mapping from data field entries to graphics files, click the second button. Then **Configure Mapping** dialog box will open. ↔ If a mapping has been configured, a symbol (↔...) is displayed besides the symbol file name.

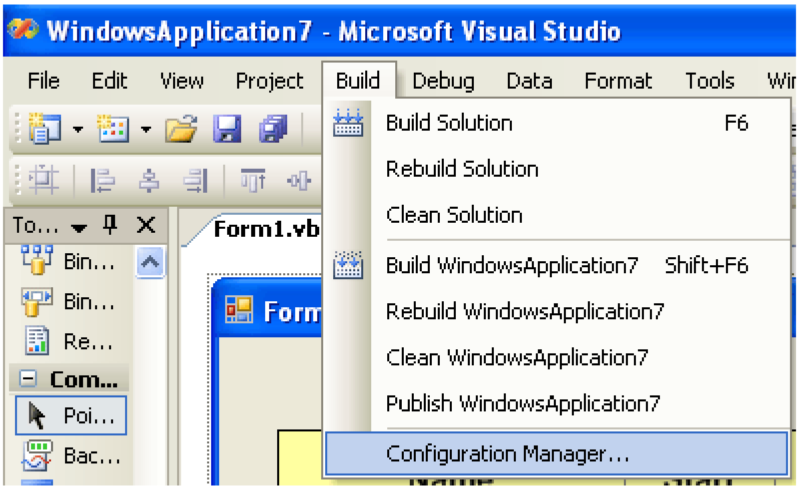
For further details please read the chapters "Property Pages and Dialog Boxes" and "Important Concepts: Maps".

3.18 Platforms x86 and x64

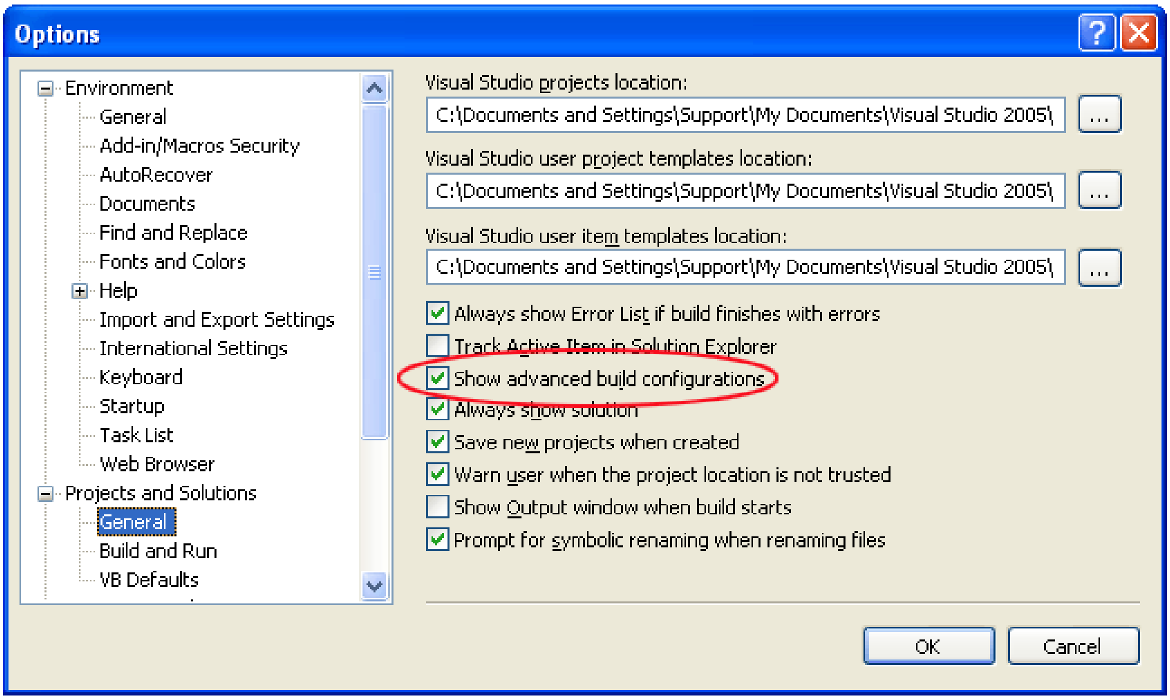
Applications written with the .NET framework are usually compiled into MSIL, a processor-independent bytecode. On starting the application, MSIL is translated into a machine code understood by the respective computer's processor and run in its full speed. Applications in MSIL can hence be run on any processor under windows as long as no components (assemblies or dlls) in pure machine code are used. They can even be run on other operating systems such as Mono with Linux as long as no operating system-dependent components are used. If an application does not fulfill the conditions for the processor-independence it should be marked accordingly. Otherwise it might be started by mistake on an unsupported processor, thus causing more or less understandable error messages when a processor- or operating system-independent component is used for the first time.

Internally VARCHART XNet is in part written in pure machine code, called **Mixed Mode** under .NET so that XNet has to be translated anew for each processor it is used with. There are versions available for x86 processors and from version 4.3 on also for x64 processors.

Applications that use VARCHART XNet are hence not processor-independent. As this is not recognized automatically by Visual Studio in the versions 2005 to 2010, the processor has to be set manually in a project or a solution. This is done in the **Configuration manager** dialog which you can open by clicking **Build/Configuration Manager**.



If this menu item is not visible you have to tickthe option **Show advanced build configurations** in the dialog **Tools/Options.../Projects and Solutions/General** first.



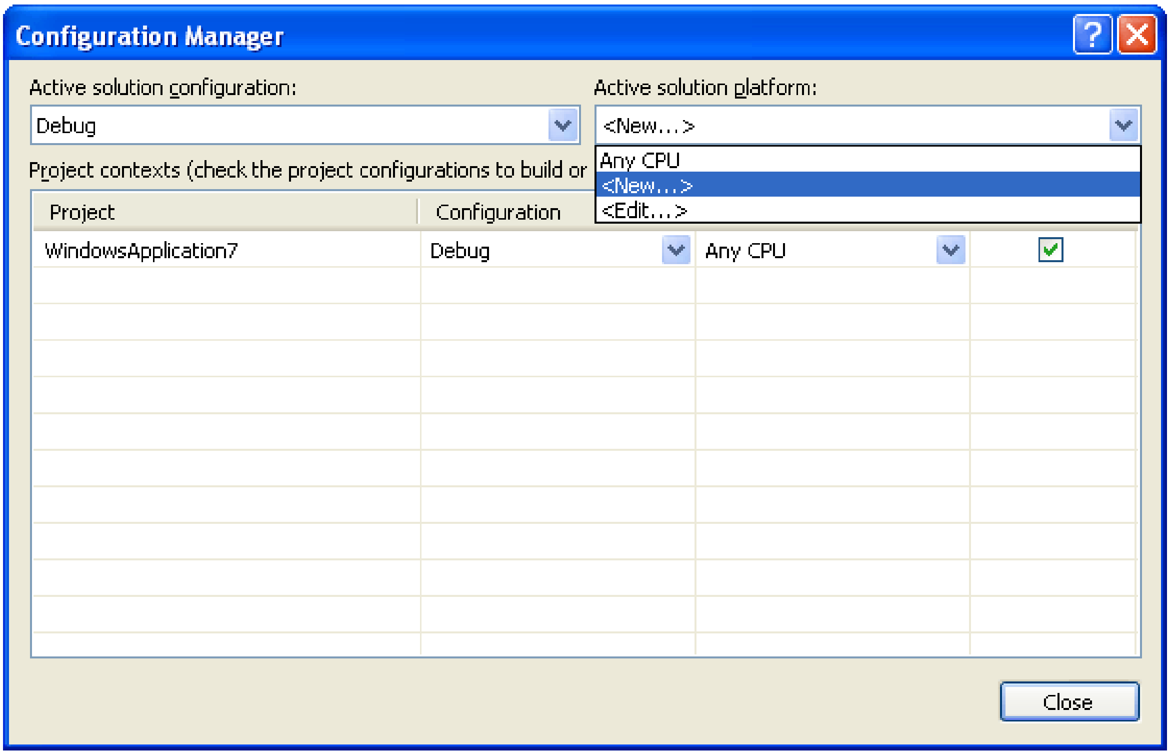
Visual Studio projects location:
C:\Documents and Settings\Support\My Documents\Visual Studio 2005\

Visual Studio user project templates location:
C:\Documents and Settings\Support\My Documents\Visual Studio 2005\

Visual Studio user item templates location:
C:\Documents and Settings\Support\My Documents\Visual Studio 2005\

☑ Always show Error List if build finishes with errors
☐ Track Active Item in Solution Explorer
☑ Show advanced build configurations
☑ Always show solution
☑ Save new projects when created
☑ Warn user when the project location is not trusted
☐ Show Output window when build starts
☑ Prompt for symbolic renaming when renaming files

In the configuration manager you can create or delete platforms. To create a new one, select **<New...>** in the **Active solution platform** dropdown list.



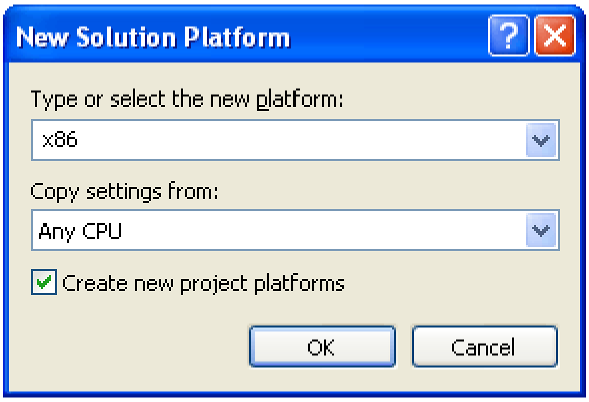
Active solution configuration: Debug

Active solution platform: <New...>

Project contexts (check the project configurations to build or

Project	Configuration		
WindowsApplication7	Debug	Any CPU	☑

In the corresponding dialog you can create the desired platforms **x86** or **x64**:



New Solution Platform

Type or select the new platform: x86

Copy settings from: Any CPU

☑ Create new project platforms

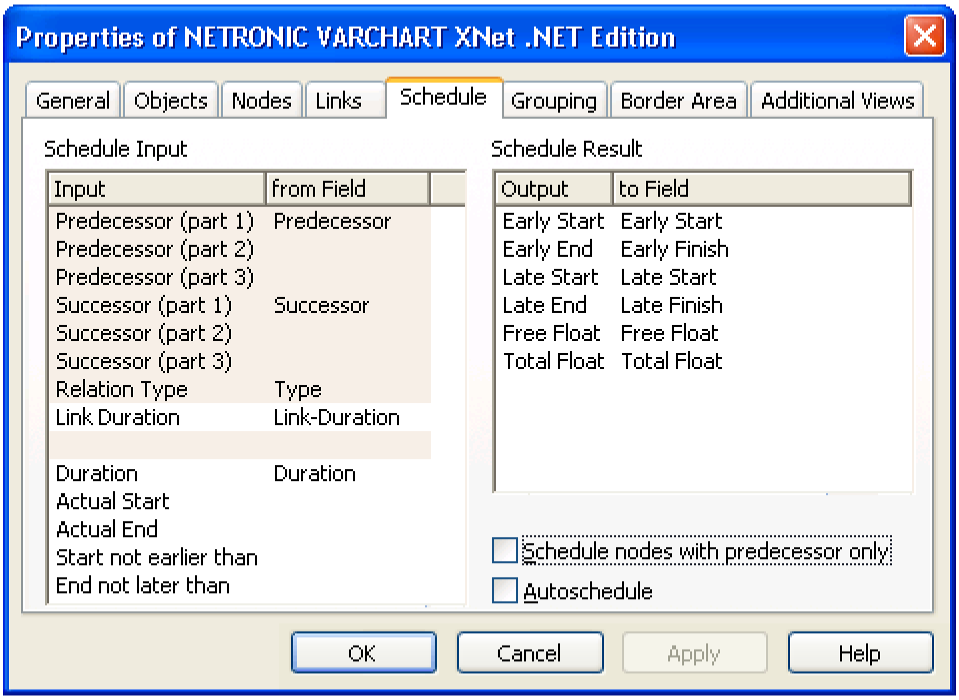
If you want to delete a platform click **<Edit...>** in the in the **platform** drop-down list and in the following dialog select the desired platform and delete it by clicking **Remove**.

To make sure that Visual Studio will always use the correct version of XNet, the following procedures, that can be found in the BuildSteps directory within the XNet installation directory (for target framework :NET 2.0 please adjust the line "set DOTNET=..." in both build events) have to be integrated into the pre-build and the post-build event. After having compiled your project once you will receive a not unexpected error message by Visual Studio. Then you have to insert a reference to the XNet.dll in the new directory C:\XNetReference (you might have to delete an existing reference to the XNet installation directory before). Finally, please compile your project once more.

3.19 Schedule

The VARCHART XNet Scheduler lets you perform simple date calculations, requiring the project start and end dates for parameters.

By the **Schedule** property page you can adapt VARCHART XNet´s date calculation settings to your interface by specifying the data fields you want to use for the input (**Schedule Input**) and output (**Schedule Result**) of the scheduler. Beside, you can set the time unit used for the calculation of duration in the corresponding data fields of nodes and links.



Schedule Input

Input	from Field
Predecessor (part 1)	Predecessor
Predecessor (part 2)	
Predecessor (part 3)	
Successor (part 1)	Successor
Successor (part 2)	
Successor (part 3)	
Relation Type	Type
Link Duration	Link-Duration
Duration	Duration
Actual Start	
Actual End	
Start not earlier than	
End not later than	

Schedule Result

Output	to Field
Early Start	Early Start
Early End	Early Finish
Late Start	Late Start
Late End	Late Finish
Free Float	Free Float
Total Float	Total Float

☐ Schedule nodes with predecessor only

☐ Autoschedule

The **Schedule Input** lets you select data fields that the data is loaded from. The scheduler uses data fields of the respective nodes and links tables.

The key data for calculating the dates are the durations of the various activities, their logical dependencies and the project start. This information is used to calculate the early/late start and end dates plus the total float and free float. The **Predecessor**, **Successor** and **Relation type** fields cannot be edited in the **Schedule Input** table. They merely show the settings that have been entered on the **Links** property page.

The output data is written to data fields of the interface. Available output options are: **Early Start, Early Finish, Late Start, Late Finish, Total Float** and **Free Float**. To each of these output options you can assign a field from the list of fields specified in the data definition.

There are several options to customize the Scheduler:

1. You can set a project start via the API, by invoking the VcNet method **ScheduleProject**:

   VcNet1.ScheduleProject "04.05.2000", 0

   The method **ScheduleProject** lets you perform a forward and a backward calculation of the project. If you pass the start date, first a forward calculation will be performed, followed by a backward calculation. If you pass the final date, first a backward calculation will be performed, followed by a forward calculation. You can pass both dates, which will add the corresponding float to the activities.

   Setting Parameters to the "ScheduleProject" Method:

Start	Finish
Date 1	0
0	Date 2
Date 1	Date 2

2. If you enter current start or end dates, the nodes will become static and cannot be moved.

3. You may enter reference dates for the conditions "Start not earlier than" and "End not later than". For these, select the corresponding data fields in the **Schedule Input** table on the **Schedule** property page. The reference date will be loaded from the fields selected. Then the earliest start of an activity will never be put before and the latest end of an activity will never be put after its reference date.

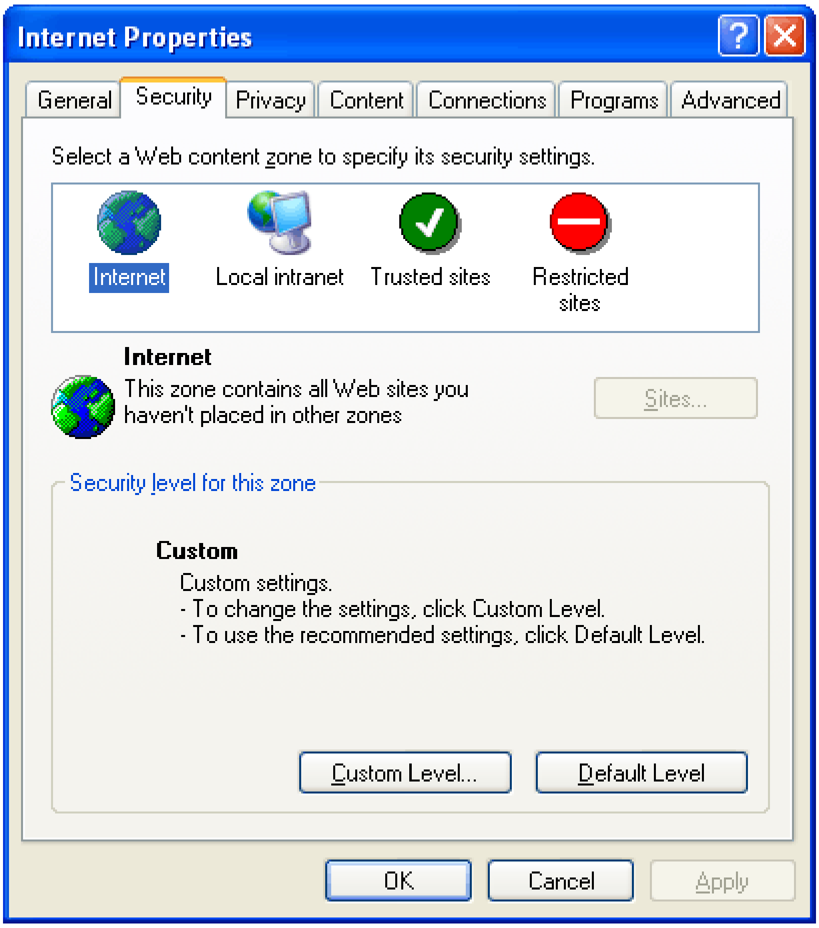
3.20 Security Guidelines for the Deployment in the Internet Explorer

In order to use the VARCHART XNet control in a HTML page in the Internet Explorer the security guidelines have to be modified.

As soon as the browser loads the control from a web server on the Internet the **Security guidelines** of the Internet_Zone become active. The default settings prevent the control from being executed. The Internet Explorer has to permit the execution of .NET components so that they become visible at all.

The guidelines can be modified in the Internet Explorer dialog **Security Settings** which you can reach by **Control Panel** > **Internet Properties** > **Security** > **Internet**.

Please select the **Zone Internet** or **Trusted Sites**.



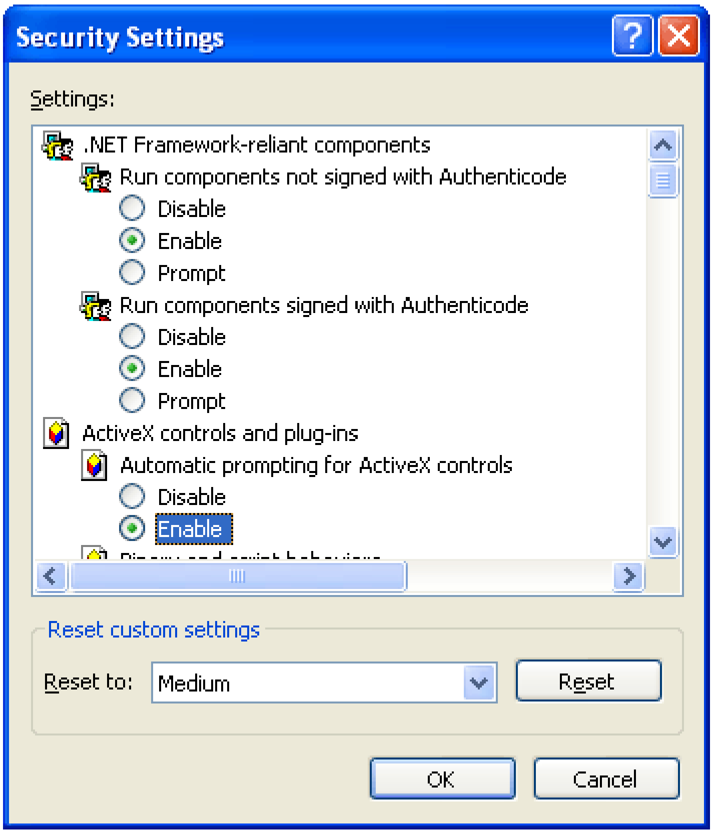
Select a Web content zone to specify its security settings.

Internet
This zone contains all Web sites you haven't placed in other zones

Security level for this zone

Custom
Custom settings.
- To change the settings, click Custom Level.
- To use the recommended settings, click Default Level.

For the zone selected, please click on **Custom Level...** and enable both, **Run Components not signed with Authenticode** and **Run Components signed with Authenticode**.



Security Settings

Settings:

.NET Framework-reliant components

Run components not signed with Authenticode

○ Disable

● Enable

○ Prompt

Run components signed with Authenticode

○ Disable

● Enable

○ Prompt

ActiveX controls and plug-ins

Automatic prompting for ActiveX controls

○ Disable

● Enable

Reset custom settings

Reset to: Medium

In addition, the runtime guidelines on the local computer need to be changed.

In the **CAS** directory of the VARCHART XGantt installation you can find two complementing batch files. The first one is **AddRights.bat**. It lets you create a permission set and a code group for NETRONIC controls. If later on you wish to deliver your application to a customer, the batch file needs to be executed on each client system before running your application. The second one is named **RemoveRights.bat** and lets you cancel permissions. Thus the VARCHART XGantt control can be executed on a HTML page in the internet Explorer using a minimum set of permissions.

3.21 Status Line Text

The **VcStatusLineTextShowing** event lets you display information in the status line on the node that was touched by the mouse.

3.22 Tooltips during Runtime

Tooltips allow to display information on the objects that the mouse is hovering over. The **VcToolTipTextSupplying** event lets you edit tooltips (None, Group, Node, LinkCollection) in order to, for example, translate them into a different language or suppress them.

To activate the event, set the VcNet property **ToolTipTextSupplyingEventEnabled** to **True**.

Alternatively, you can tick the check box **VcToolTipTextSupplying events** on the **General** property page. By reacting to the **VcToolTipTextSupplying** event you can define the text you want to have appear or whether no tooltip should be displayed at that location.

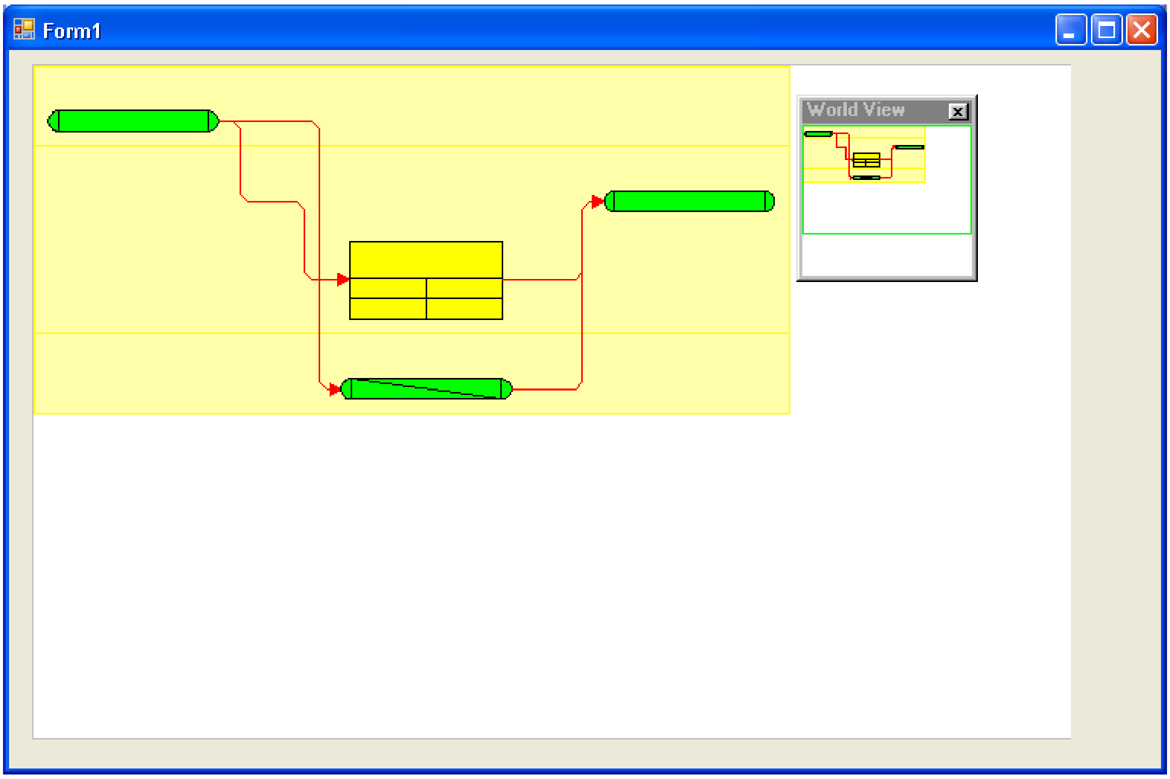
3.23 Viewer Metafile (*.vmf)

VMF is a graphics format that was especially developed for the WebViewer (a Java applet independent of platforms and browsers) by NETRONIC Software GmbH. The VMF format allows you to view, zoom or move your diagrams in a browser on the intranet/internet.

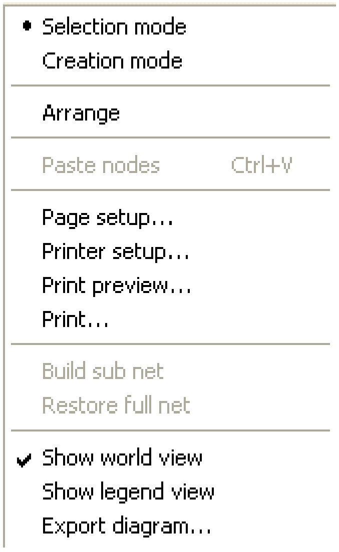
The method **ExportGraphicsToFile** of object VcNet or the default context menu for the diagram lets you store the diagram to a file.

3.24 World View

The world view is an additional window that shows the complete diagram. A frame shows the diagram section currently displayed in the main window. If you move the frame or change its size, the corresponding section in the main window will move proportionally as soon as you release the mouse button. In a similar way, you can enlarge or reduce the display in the main window by zooming the frame in the world view. Vice versa, the position or the size of the frame will be changed when you scroll or zoom the section in the main window.



At run time, you can switch on/off the world view via the item **Show world view** of the default context menu.



On the **Additional Views** property page you can specify the properties of the World View. For details please read the chapter "Property Pages and Dialog Boxes", the "Additional Views" Property Page.

Beside, you can specify the properties of the World View by the API (**VcWorldView**).

3.25 Writing PDF files

Writing PDF files is only possible if an appropriate PDF printing driver is available. The drivers that are free of charge and those that are commercially available differ in their functionality and in the quality of the created PDF files.

Due to the lack of a consistent standard for the controlling of drivers, each printing driver has to be configured individually. The target path for the output file of many PDF printing drivers for instance is preset and can only be modified by altering the Windows registry, by editing INI files or by using driver-specific function APIs or COM objects.

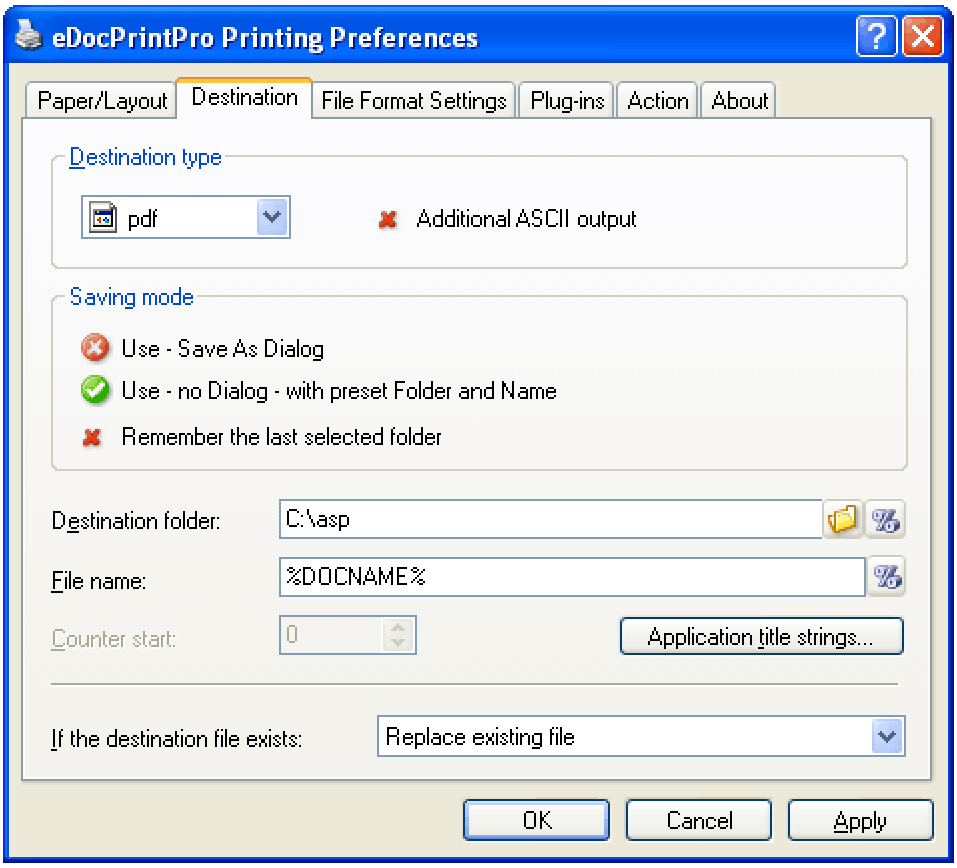
To be suitable a PDF printing driver has to fulfill the below requirements concerning controlling and print quality:

- Depending on the design of the application, it may be necessary that the driver offers the option of switching off all runtime dialogs and message boxes, in particular dialogs for setting file names and paths.
- If file names and paths shall not be set until runtime and if this is only possible by modifying entries of the Windows registry, the permissions of the user account have to be set accordingly.
- For the correct output of texts, Unicode support is needed.
- Fill patterns have to be displayed in sufficient quality. Please note that apart from bitmaps, transparencies cannot be displayed. In bitmaps however, unwanted artifacts may occur.
- The driver has to support vertical text output, otherwise the vertical annotation of date lines in VARCHART XGantt cannot be used.

The aforementioned requirements are fulfilled for instance by the printing driver included in the **Adobe Acrobat Suite** from version 6 onward [www.adobe.com] and the free driver **eDocPrintPro** [www.pdfprinter.at].

Below, please find an outline of the required steps to control the printing driver, using the example of **eDocPrintPro**:

- The dialog **Printing Preferences** can be accessed by the driver's settings in the control panel or by the driver's entry in Start/Programs or by the usual print dialog of an application. If necessary you can in that dialog select that the PDF file should be created without a dialog popping up and that the name of the target file is to be derived from the name of the document for instance. The required settings in **eDocPrintPro** then look as follows:



Destination type

pdf

✗ Additional ASCII output

Saving mode

✗ Use - Save As Dialog

✓ Use - no Dialog - with preset Folder and Name

✗ Remember the last selected folder

Destination folder: C:\asp

File name: %DOCNAME%

Counter start: 0

If the destination file exists: Replace existing file

- In the program, the VcPrinter object of VARCHART XGantt should contain the below settings:

Example Code VB.NET

```
VcNet1.Printer.PrinterName = "eDocPrintPro"
VcNet1.Printer.DocumentName = "abc.pdf"
VcNet1.PrintEx
```

Example Code C#

```
vcNet1.Printer.PrinterName = "eDocPrintPro";
vcNet1.Printer.DocumentName = "abc.pdf";
vcNet1.PrintEx;
```

Very few printing drivers require a different program code:

Example Code VB.NET

```
VcNet1.Printer.PrinterName = "Win2PDF"
VcNet1.PrintToFile "abc.pdf"
```

Example Code C#

```
vcNet1.Printer.PrinterName = "Win2PDF";
vcNet1.PrintToFile "abc.pdf";
```

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

4 Property Pages and Dialog Boxes

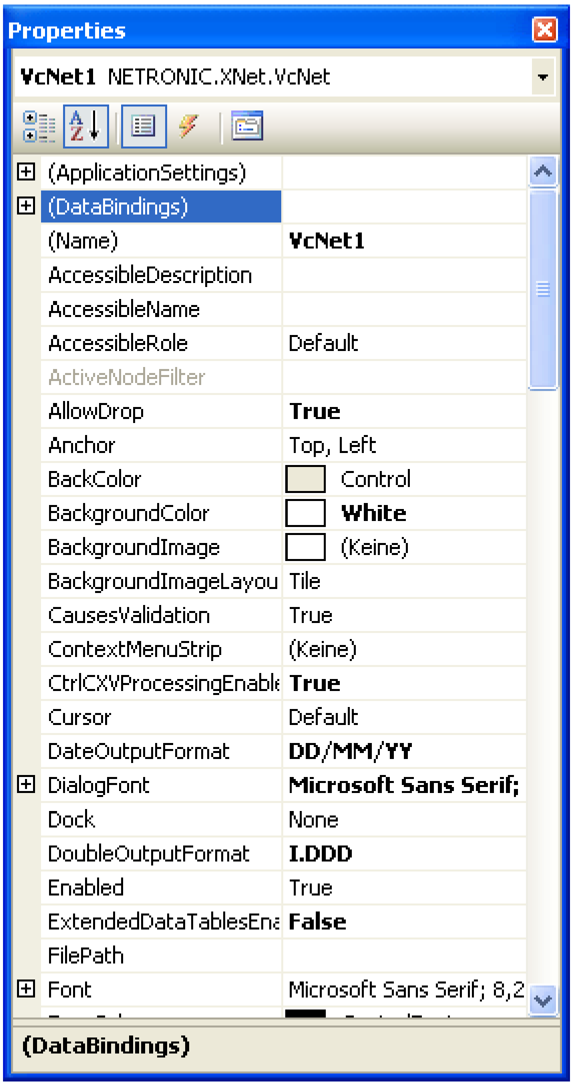
4.1 General Information

Property pages allow to configure VARCHART XNet 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 .



Properties

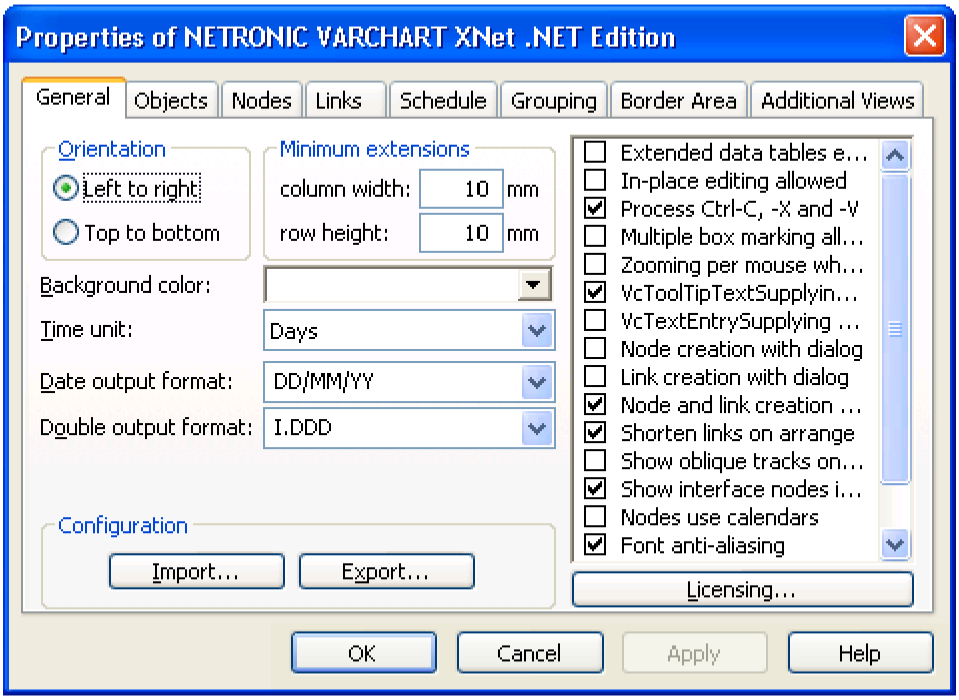
VcNet1 NETRONIC.XNet.VcNet

⊞ (ApplicationSettings)	
⊞ (DataBindings)	
(Name)	**VcNet1**
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
ActiveNodeFilter	
AllowDrop	**True**
Anchor	Top, Left
BackColor	□ Control
BackgroundColor	□ **White**
BackgroundImage	□ (Keine)
BackgroundImageLayou	Tile
CausesValidation	True
ContextMenuStrip	(Keine)
CtrlCXVProcessingEnabl	**True**
Cursor	Default
DateOutputFormat	**DD/MM/YY**
⊞ DialogFont	**Microsoft Sans Serif;**
Dock	None
DoubleOutputFormat	**I.DDD**
Enabled	True
ExtendedDataTablesEnɛ	**False**
FilePath	
⊞ Font	Microsoft Sans Serif; 8,2

(DataBindings)

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



Orientation
● Left to right
○ Top to bottom

Minimum extensions
column width: 10 mm
row height: 10 mm

Background color:
Time unit: Days
Date output format: DD/MM/YY
Double output format: I.DDD

□ Extended data tables e...
□ In-place editing allowed
☑ Process Ctrl-C, -X and -V
□ Multiple box marking all...
□ Zooming per mouse wh...
☑ VcToolTipTextSupplyin...
□ VcTextEntrySupplying ...
□ Node creation with dialog
□ Link creation with dialog
☑ Node and link creation ...
☑ Shorten links on arrange
□ Show oblique tracks on...
☑ Show interface nodes i...
□ Nodes use calendars
☑ Font anti-aliasing

On this property page you can enter the general settings of VARCHART XNet.

Orientation

These radio buttons let you arrange the nodes in a **Left to right** or **Top to bottom** orientation.

Minimum column width

Please specify the minimum column width for node columns of the diagram. Unit: mm. The width entered should correspond to the average width of a node. Setting a low column width will make links use less space in a left-to-right arrangement.

Minimum row height

Please specify the minimum row height for node rows of the diagram. Unit: mm. The height entered should correspond to the average height of a node. Setting a low row height will make links use less space in a top-to-bottom arrangement.

Background color

Please select a background color for the network diagram.

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 **VcNet.TimeUnit**.

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 **VcNet.DateOutputFormat**.

To compose the date you can use the following tokens:

D: first letter of the day of the week (not adjustable)

TD: Day of the Week (adjustable by using the event **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** 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 **VcNet.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 **VcNet.ExtendedDataTablesEnabled**.

In-place editing allowed

Tick this option if in-place editing of data fields in node fields and in boxes is to be allowed. This feature can also be set by the property **VcNet.InPlaceEditingAllowed**.

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

Process Ctrl-X, -C and -V

If you activate this check box, the key combinations Ctrl+C, Ctrl+X and Ctrl+V will be translated automatically into the clipboard commands **CopyNodesToClipboard**, **CutNodesToClipboard** and **PasteNodesFromClipboard**, respectively. You can revoke this feature by leaving the check box blank, in order to avoid interfering with menu commands in Visual Basic. This feature can also be set by the property **VcNet.CtrlCXVProcessingEnabled**.

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 **VcNet.MultipleBoxMarkingAllowed**.

Zooming by mouse wheel allowed

Tick this option if zooming by mouse wheel is to be allowed. For zooming the user has to press the Ctrl key and roll the mouse wheel.

This feature can also be set by the property **VcNet.ZoomingPerMouseWheelAllowed**.

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.

VcTextEntrySupplying events

By ticking this box you can trigger the **VcTextEntrySupplying** event. This event lets you modify the texts of context menus, dialog boxes and error messages that occur during run time, for example for translation into different languages.

This feature can also be set by the property **VcNet.TextEntrySupplyingEventEnabled**.

Node creation with dialog

This option lets you specify whether or not the **Edit Data** dialog box is to appear on interactive creation of a node by the user. If you deactivate this feature, the dialog **Edit Data** can still be invoked by a double-click on the node.

This feature can also be set by the property **VcNet.NodeCreationWithDialog**.

Link creation with dialog

This option lets you specify whether or not the **Edit Data** dialog box appears when a new link is created interactively. If this feature is deactivated, the dialog **Edit Data** can be invoked by a double-click on the link.

Node and link creation allowed

By ticking this box you allow the user to create new nodes and links interactively. Nodes he can create by pressing the right mouse button in the diagram area. Links he can create by positioning the cursor onto a node, then dragging it towards a different node while keeping the left mouse button depressed, and finally releasing the left mouse button on the second node.

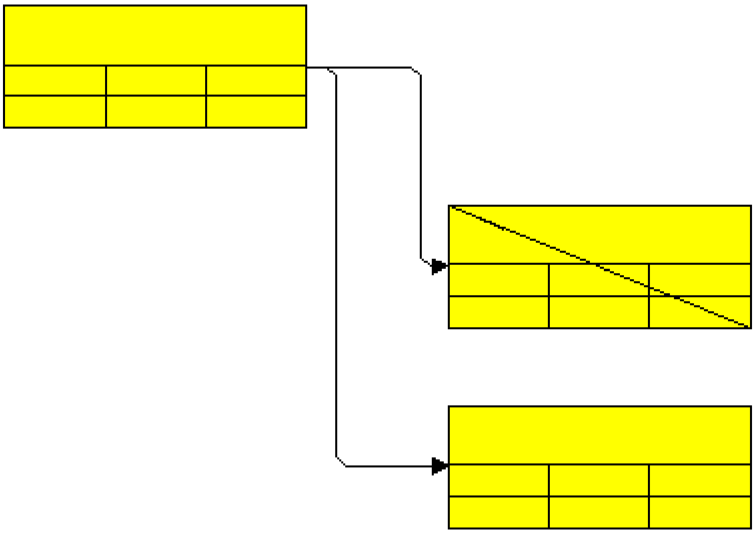
Shorten links on arrange

This property will influence the layout of a network diagram and will be considered by the method **Arrange**. If you tick this box, nodes will be placed as closely as possible near their successor nodes, thus keeping the distance between them as small as possible. If you leave the check box blank, nodes will be placed as far left or up as possible. This feature can also be set by the **ShortenedLinks** property of the vcNet class.

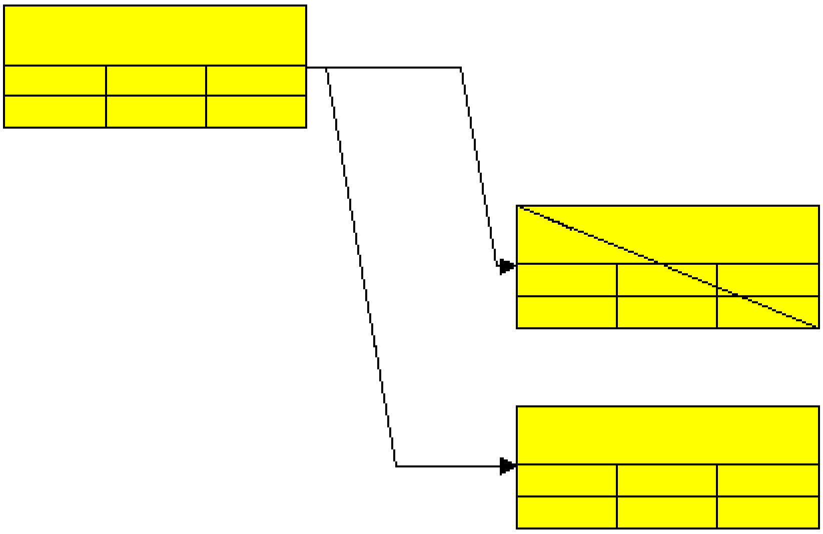
Show oblique tracks on links

If you activate this check box, the link lines will be oblique, connecting the short horizontal line sections. Otherwise the link lines will be orthogonal.

Beside, this feature can be specified with the help of the VcNet property **ObliqueTracksOnLinks**.



orthogonal link lines



oblique link lines

Show interface nodes in subnet

If you activate this check box, the interface nodes are to displayed, when a subdiagram is created. You can specify the appearance of the interface nodes in the **Specify Node Appearance** dialog box. To do so, select the special filter <InterfaceNodes>.

Nodes use calendar

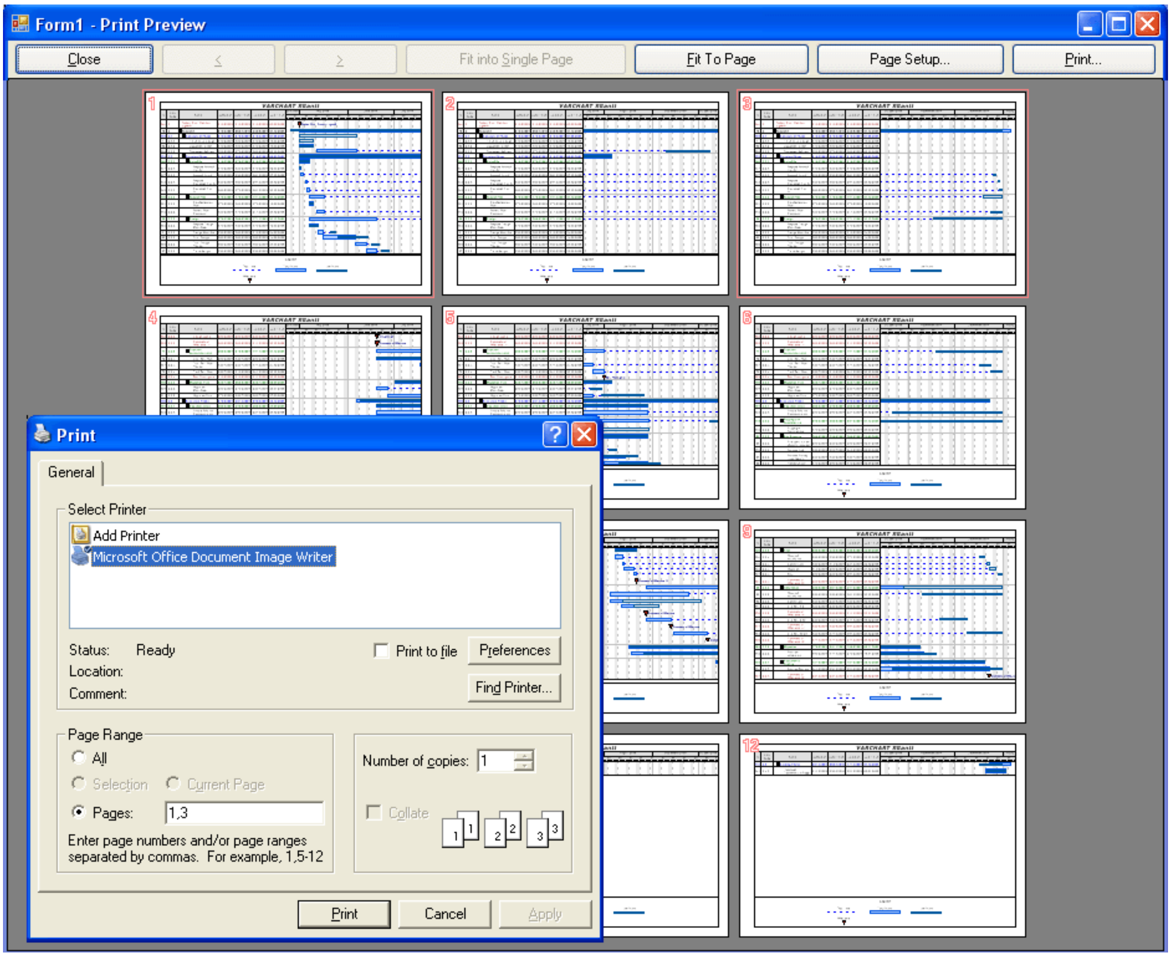
By ticking this box you can make the scheduler use the calendar for forward and backward calculation of the project. By the use of the calendar, workfree time periods will be considered in the calculation. By default, a five day calendar is used. You may define your own calendars by using the objects VcCalendar, VcWorkweek and VcWorkday and activate one of them by the call VcCalendarCollection.Active.

Use PrintDlgEx dialog

If you tick this check box, the item **Printer setup** will be missing at runtime both in the print preview and in the context menu because the corresponding dialog is now to be found in the (extended) **Print** dialog. If a new project is created, this option is ticked by default whereas in already existing projects it is ticked off for compatibility reasons.

In the print preview you can now select pages by a left click (one page) or by using CTRL + left click (more pages). The selected pages are then preset already as pages to be printed in the **Print** dialog.

If you invoke the **Print** dialog from the print preview, all pages have a page number to make the selection of pages easier.



Select Printer

- Add Printer
- Microsoft Office Document Image Writer

Status: Ready

Location:

Comment:

□ Print to file

Page Range

○ All

○ Selection ○ Current Page

● Pages: 1,3

Enter page numbers and/or page ranges separated by commas. For example, 1,5-12

Number of copies: 1

□ Collate

This feature can **not** be set by an API property.

Rounded link slants

If you tick this check box, the slants of links of the routing type **vcLRTOrthogonal** are displayed as quarter circles instead of straight lines. This feature can also be set by the **RoundedLinkSlantsEnabled** property of VcNet.

Wait cursor enabled on time-critical operations

Tick this box if you want to set us an internal wait cursor on time-critical operations.

This feature can also be set by the **VcNet.WaitCursorEnabled** property

Panning mode allowed

Tick this box to be able to move certain screen sections at runtime. The contextmenu will then show the additional item **Panning mode**.

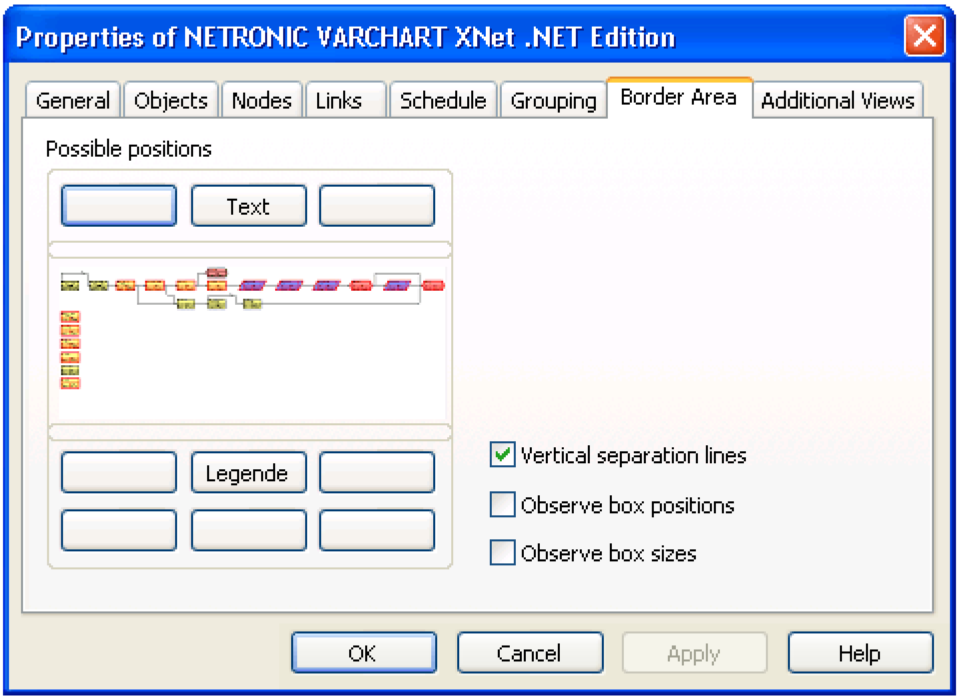
Activating the panning mode will apply to **all** view components by default. The **VcGantt.VcViewComponent** property allows to set the panning mode for certain selected components only.

This feature can also be set by the**VcNet.PanningModeAllowed** 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



Properties of NETRONIC VARCHART XNet .NET Edition

☑ Vertical separation lines

□ Observe box positions

□ Observe box sizes

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.

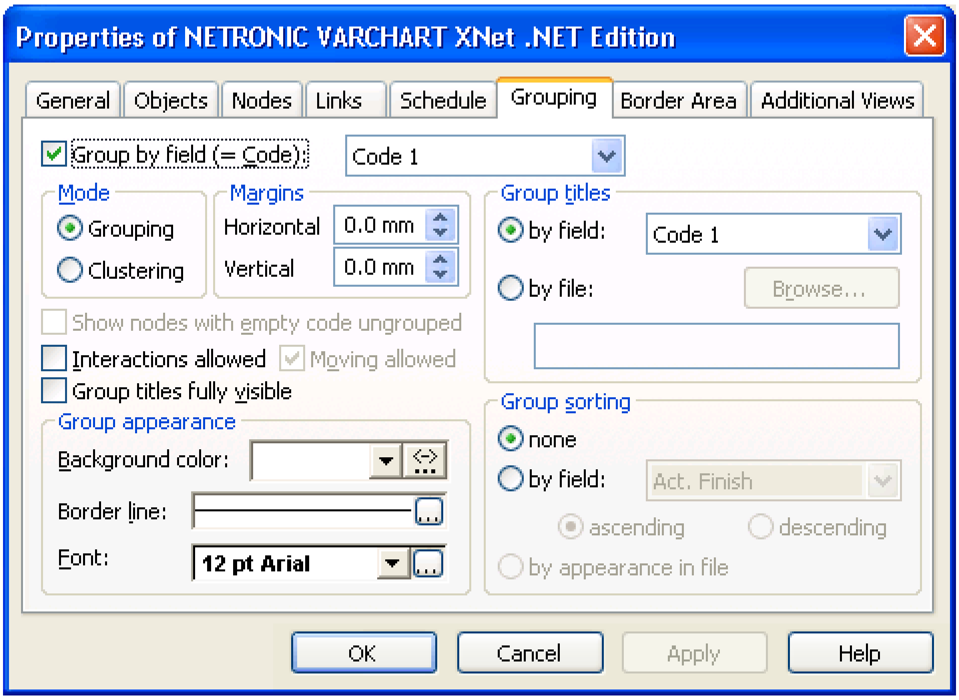
Observe box position

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 "Grouping" Property Page



☑ Group by field (= Code): Code 1

Mode
● Grouping
○ Clustering

Margins
Horizontal 0.0 mm
Vertical 0.0 mm

Group titles
● by field: Code 1
○ by file: Browse...

□ Show nodes with empty code ungrouped
□ Interactions allowed ☑ Moving allowed
□ Group titles fully visible

Group appearance
Background color:
Border line:
Font: 12 pt Arial

Group sorting
● none
○ by field: Act. Finish
● ascending ○ descending
○ by appearance in file

Group by field (= Code)

Activate this check box if you want the nodes to be grouped. Only if this check box is activated, the further options of this property page are available.

This field lets you select a field that the groups are sorted after. The field you select will be called **group code**. All nodes that show the same contents in the field selected will belong to the same group.

Mode

Select the mode:

- **Grouping:** normal visualization of groups (The width and height of each group is determined by the node positions. Each group needs the full width or height respectively of the net diagram)
- **Clustering:** The nodes are grouped very space-sparing, and the groups are placed freely in the net diagram.

Margins

Specify the width of the horizontal/vertical margins of the groups. Allowed are values between 0 and 9.9 mm.

Show nodes with empty code ungrouped

(only for mode: clustering) If this check box is activated, nodes without an entry for the group code (empty string) will not be grouped. Otherwise a special group for nodes with empty group code will be created.

Interactions allowed

If this check box is activated, the groups can be collapsed or expanded interactively by clicking on the **minus** or **plus** symbol beside the group title, respectively.

Moving allowed

(only for mode: clustering) If this check box is activated, the clustered groups can be moved interactively.

Group titles fully visible

If this box is ticked, the group titles are always visible while scrolling horizontally.

Background color

Please select a background color for the groups.

Border line

This field displays the appearance of the group border line. To edit it, please click on the **Edit** button, which will get you to the **Line attributes** dialog. There you can set the color, type and thickness of the line.

Font

This field displays the font style and color of the group title. To edit the font color, please click on the arrow button. Press the **Edit** button to get to the Windows **Font** dialog box where you can specify the font type, style and size.

Group titles by field

If you activate this radio button, group titles will be loaded from the field you select here. Although the field does not necessarily need to be the group code field, the entries of the **Group code** field and of the **Group title** field should correspond in order to give sensible group headings.

Group titles by file

If you activate this radio button, group titles will be loaded from the file you select here. Clicking on the **Browse** button will open the **Choose group titles file** dialog where you can choose a file that group titles are to be loaded from. By default, group titles are read from a file of the type *.txt. Alternatively, you can set a different file type.

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

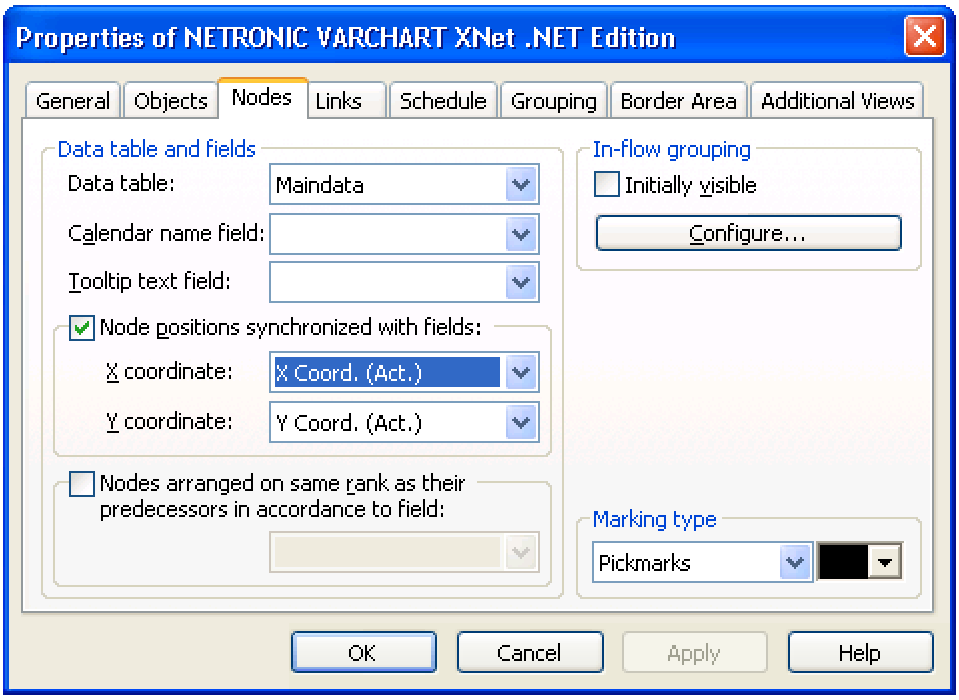
Group sorting

This section lets you specify whether the groups are to be sorted and lets you enter the settings for the group sorting. The radio buttons let you toggle between **none**, **by field** and **by appearance in file**.

If you select the **by field** option, you can select the field that the groups are sorted by. In addition, the **ascending** and **descending** options are activated, that let you choose the desired order.

If you select the **by appearance in file** option, the groups will be displayed in the sequence of their occurrence in the file.

4.5 The "Nodes" Property Page



Properties of NETRONIC VARCHART XNet .NET Edition

Data table and fields

Data table: Maindata

Calendar name field:

Tooltip text field:

☑ Node positions synchronized with fields:

X coordinate: X Coord. (Act.)

Y coordinate: Y Coord. (Act.)

☐ Nodes arranged on same rank as their predecessors in accordance to field:

In-flow grouping

☐ Initially visible

Marking type

Pickmarks ■

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, on the **General** property page the check box **Scheduler uses internal calendar** has to be activated. Beside, the calendars have to be created before loading the nodes.

This feature can also be set by the property **VcNet.NodeCalendarNameDataFieldIndex**.

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 **VcToolTipTextSupplying events** on the **General** property page or set the VcNet property **ToolTipTextSupplyingEventEnabled** = True and specify the text to be displayed in the **VcToolTipTextSupplying** event.

This feature can be also set by the property **VcNet.NodeToolTipTextDataFieldIndex**.

Node positions synchronized with data fields

Synchronizing node positions with data fields is required if node positions are to be restored after closing the project.

Please activate this check box to synchronize node positions with the data fields selected. Choose a data field, that the X and Y positions of each node position are to be loaded from and stored to.

Nodes arranged on same rank as their predecessors in accordance to data fields

The nodes' ranks are represented by their positions in the chart. You can modify the layout of the chart by positioning defined nodes on the same rank as their predecessors. To do so, please activate this check box and select a data field (e.g. the data field "auxiliary node"). The contents of the data field that you select will determine whether or not the node will be postioned on the same rank as its predecessor.

If the field doesn't exist, please create it in the **Administrage Data Tables** dialog. You may enter the values **0, 1, 2** or **3** as its contents.

Value of the data field	Top-to-bottom orientation	Left-to-right orientation
0	The rank of the auxiliary node will not be lowered.	The rank of the auxiliary node will not be lowered.
1	The rank of the auxiliary node will be lowered by 1. The auxiliary node will not be positioned beneath its predecessor, but left or right of it.	The rank of the auxiliary node will be lowered by 1. The auxiliary node will not be positioned left of its predecessor, but beneath or on top of it.
2	The rank of the auxiliary node will be lowered by 1. The auxiliary node will be positioned left of its predecessor.	The rank of the auxiliary node will be lowered by 1. The auxiliary node will be positioned above its predecessor.
3	The rank of the auxiliary node will be lowered by 1. The auxiliary node will be positioned right of its predecessor.	The rank of the auxiliary node will be lowered by 1. The auxiliary node will be positioned below its predecssor.

Note: The rank of a node is represented by a number. The rank of a node that does not have predecessors equals 1. To the rank of a node that has predecessors the rank number of the highest ranked predecessor is added.

More information you can find in the chapter "Important Concepts: Nodes".

Marking type

Specify whether node marks are used interactively and, if desired, select the type of node marking from the list:

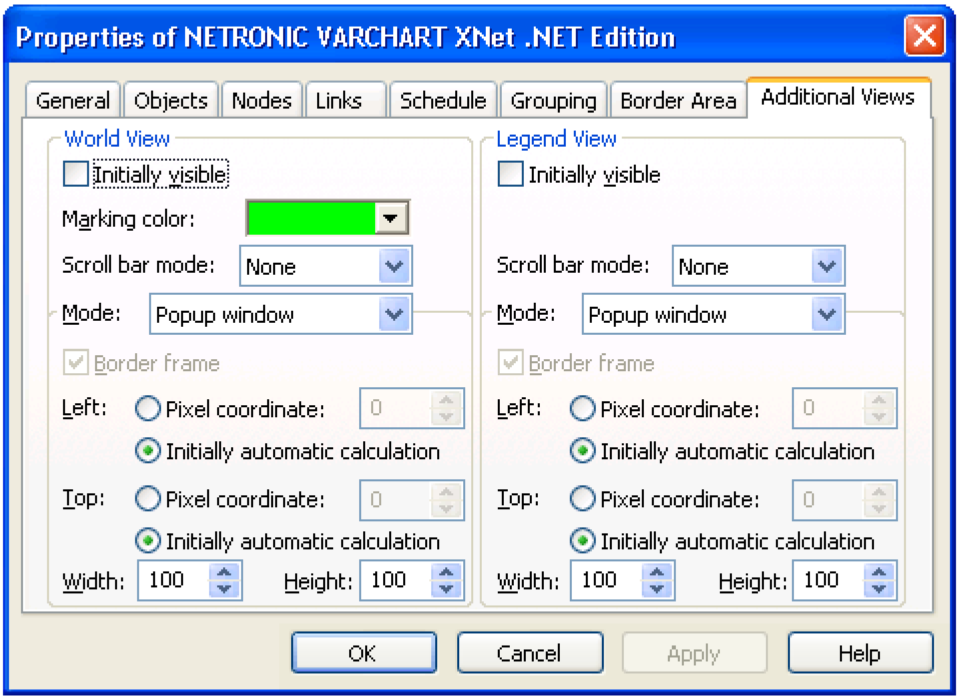
- No Mark
- Surround
- Surround inside
- Invert
- Pickmarks
- Pickmarks inside

Note: If you select **no mark**, there will be no graphical pattern to mark a node.

In-flow grouping

By the **Configure** button the **Edit In-Flow Grouping** dialog can be opened. Activate the **Initially visible** check box to activate the in-flow grouping at the start of the program.

4.6 The "Additional Views" Property Page



Properties of NETRONIC VARCHART XNet .NET Edition

World View

□ Initially visible

Marking color: 🟩

Scroll bar mode: None

Mode: Popup window

☑ Border frame

Left: ○ Pixel coordinate: 0

● Initially automatic calculation

Top: ○ Pixel coordinate: 0

● Initially automatic calculation

Width: 100 Height: 100

Legend View

□ Initially visible

Scroll bar mode: None

Mode: Popup window

☑ Border frame

Left: ○ Pixel coordinate: 0

● Initially automatic calculation

Top: ○ Pixel coordinate: 0

● Initially automatic calculation

Width: 100 Height: 100

On this property page you can set the properties of the "world view" and the legend view.

The world view is an additional small window that displays the diagram completely. A frame in it indicates the section currently displayed in the main window.

The legend view lets you display a legend.

At run time, you can switch on or off both views in the default context menu by clicking **Show world view** or **Show legend view** respectively. You can alternatively use the **Close** button of the title bar to switch off either view.

The description of the possible settings which you find below, is valid for both views, if not stated otherwise.

Initially visible

Activate this check box if the view is to be visible when the program is started.

This property can also be set by the API calls **VcWorldView.Visible** and **VcLegendView.Visible**

Marking color (only World View)

Select the line color of the rectangle that indicates in the World View the currently selected section.

This property can also be set by the API calls **VcWorldView.MarkingColor** and **VcLegendView.MarkingColor**.

Scroll bar mode

You can select a mode of displaying scrollbars. By using scrollbars, empty areas are avoided and there is more space for displaying the chart or the legend.

- **None:** The view always displays the complete chart or legend. Thus empty areas may occur if the view's proportions do not correspond to those of the chart/the legend.
- **Horizontal:** A horizontal scrollbar is displayed if required.
- **Vertical:** A vertical scrollbar is displayed if required.
- **Automatic:** A horizontal or a vertical scrollbar is displayed if required.

This property can also be set by the API calls **VcWorldView.ScrollBarMode** and **VcLegendView.ScrollBarMode**.

Mode

You can select a mode of displaying the the view:

- **Fixed at left side:** The view appears on the left side of the control window. The width can be varied, whereas the position and the height are fixed.
- **Fixed at right side:** The view appears on the right side of the control window. The width can be varied, whereas the position and the height are fixed.
- **Fixed at top side:** The view is displayed in the top section of the control window. The height can be varied, whereas the position and the width are fixed.
- **Fixed at bottom side:** The view is displayed in the bottom section of the control window. The height can be varied, whereas the position and the width are fixed.
- **Position not fixed:** The view is a subwindow of the parent window of the control. It can be positioned anywhere and has no fixed size. The parent window can be modified by the property **VcWorldView.ParentHWnd**.

- **Popup window:** The view is a popup window that has its own frame. The user can modify its position and extension, open it by using the default context menu, and close it by the **Close** button in the frame.

This property can also be set by the API calls **VcWorldView.Mode** and **VcLegendView.Mode**.

Border frame

*Not active if the mode **Popup window** has been selected.* Activate this check box if the view is to have a frame and select a color in the drop down list..

This options can also be set by the API calls **VcWorldView.Border** and **VcWorldView.Border.Color** or **VcLegendView.Border** and **VcLegend-View.Border.Color**

Left

*Only active if the mode **Position not fixed** or **Popup window** has been selected.* Select the left position of the view. There are two possibilities:

1. Specify a **Pixel coordinate** value. Note that this is a system coordinate.
2. Select the **Initially automatic calculation** option.

This property can also be set by the API calls **VcWorldView.Left** and **VcLegendView.Left**

Top

*Only active if the mode **Position not fixed** or **Popup window** has been selected.* Select the top position of the view. There are two possibilities:

1. Specify a **Pixel coordinate** value. Note that this is a system coordinate.
2. Select the **Initially automatic calculation** option.

This property can also be set by the API calls **VcWorldView.Top** and **VcLegendView.Top**

Width

*Not active if the mode **Fixed at left/right side** has been selected.* Select the horizontal extension of the view. Note that the pixel coordinate is a system coordinate.

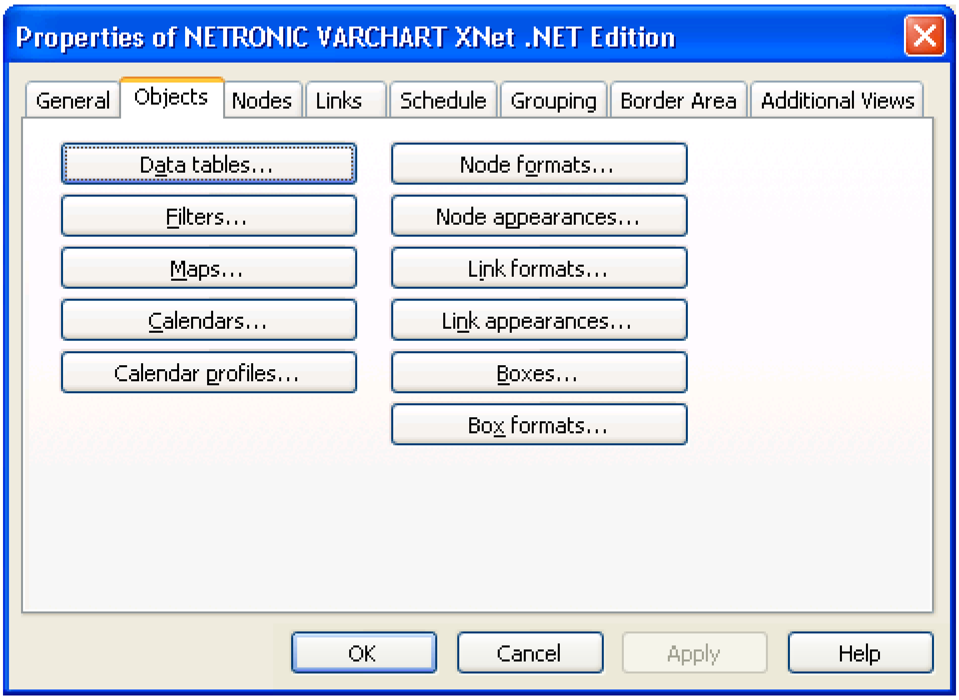
This property can also be set by the API calls **VcWorldView.Width** and **VcLegendView.Width**

Height

Not active if the mode ***Fixed at left/right side*** *has been selected.* Select the vertical extension of the view. Note that the pixel coordinate is a system coordinate.

This property can also be set by the API calls **VcWorldView.Height** and **VcLegendView.Height**

4.7 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**.

Node formats

This button lets you open the dialog **Administrate Node Formats**.

Node appearances

This button will open the dialog **Administrate Node Appearances**.

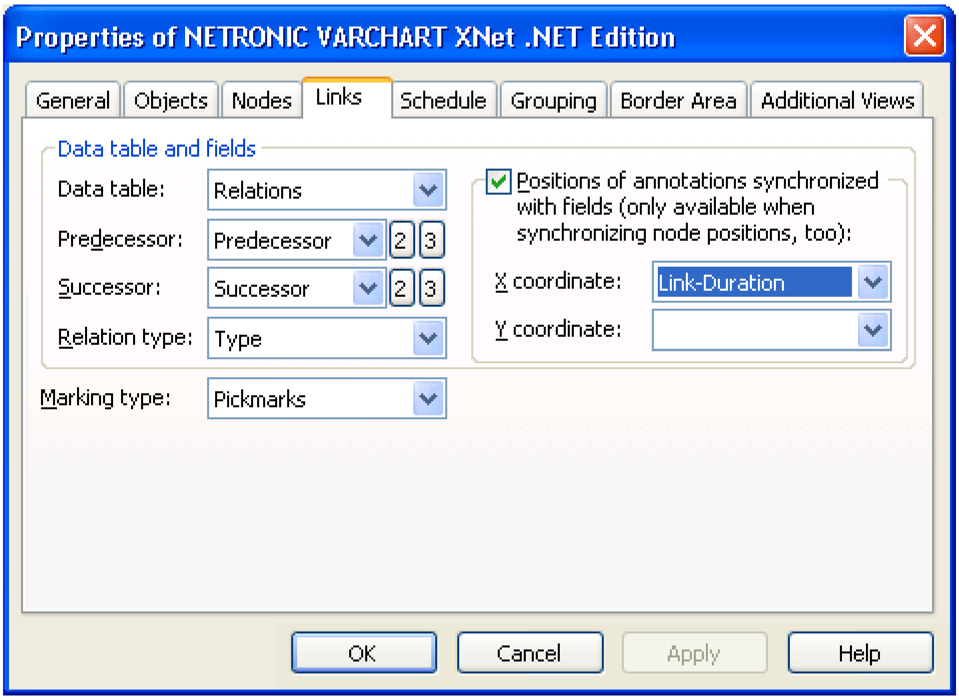
Boxes

Opens the dialog **Administrate Boxes**.

Box formats

Opens the dialog **Administrate Box Formats**.

4.8 The "Links" Property Page



Data table and fields

Data table: Relations

Predecessor: Predecessor 2 3

Successor: Successor 2 3

Relation type: Type

☑ Positions of annotations synchronized with fields (only available when synchronizing node positions, too):

X coordinate: Link-Duration

Y coordinate:

Marking type: Pickmarks

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 **VcNet.LinksDataTableName**.

Predecessor

This field lets you select a data field which contains the identification of the predecessor node of the link.

Successor

This field lets you select a data fieldwhich contains the identification of the successor node of the link.

Relation type

Please select the data field to store information on the link. The field must not contain any other information than two characters that describe the link type:

- Start-Start (SS)

- Start-Finish (SF)
- Finish-Start (FS)
- Finish-Finish (FF).

The values in brackets are valid field contents that represent the link types.

Marking type

Specify whether node marks are used interactively and, if desired, select the type of node marking from the list:

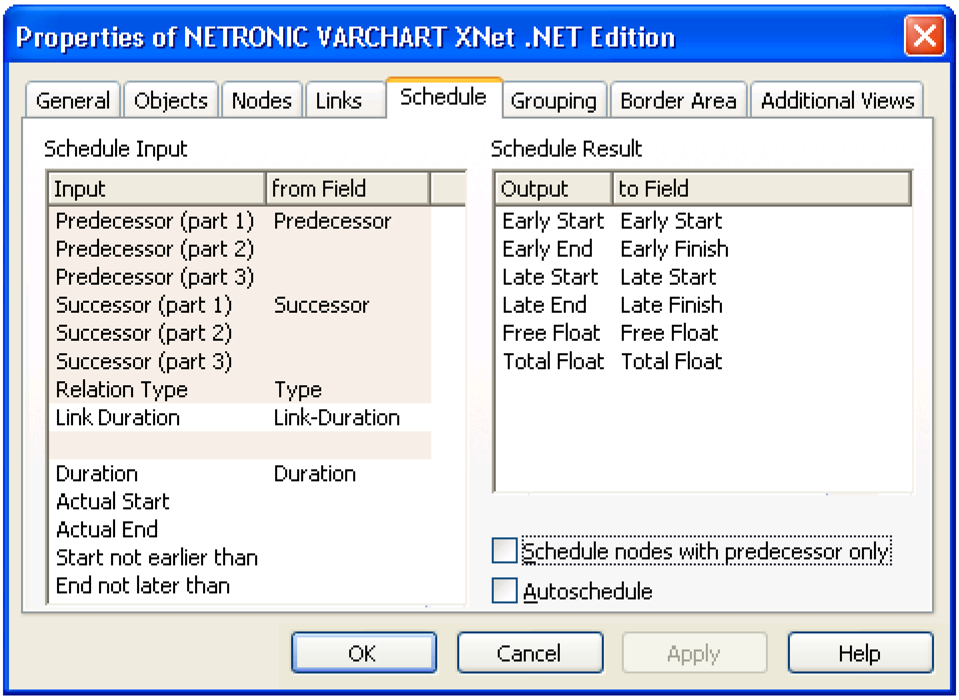
- Surround
- Invert
- No Mark
- Pickmarks

Note: If you select **No Mark**, there will be no graphical pattern to mark a node.

Positions of annotations synchronized with data fields

Ticking this box will keep annotation positions continuously stored to data fields, thus synchronizing the values in the chart with the values in the data fields. You may need these values when restoring the positions of link annotations after closing and reopening your project. Ticking this box activates the fields **X coordinate** and **Y coordinate**, where you can select a data field to store the X and Y coordinate to.

4.9 The "Schedule" Property Page



Schedule Input

Input	from Field
Predecessor (part 1)	Predecessor
Predecessor (part 2)	
Predecessor (part 3)	
Successor (part 1)	Successor
Successor (part 2)	
Successor (part 3)	
Relation Type	Type
Link Duration	Link-Duration
Duration	Duration
Actual Start	
Actual End	
Start not earlier than	
End not later than	

Schedule Result

Output	to Field
Early Start	Early Start
Early End	Early Finish
Late Start	Late Start
Late End	Late Finish
Free Float	Free Float
Total Float	Total Float

□ Schedule nodes with predecessor only

□ Autoschedule

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.

Time unit for durations

The unit selected from the combo box will be used for calculations of dates and floats in data fields of nodes and links.

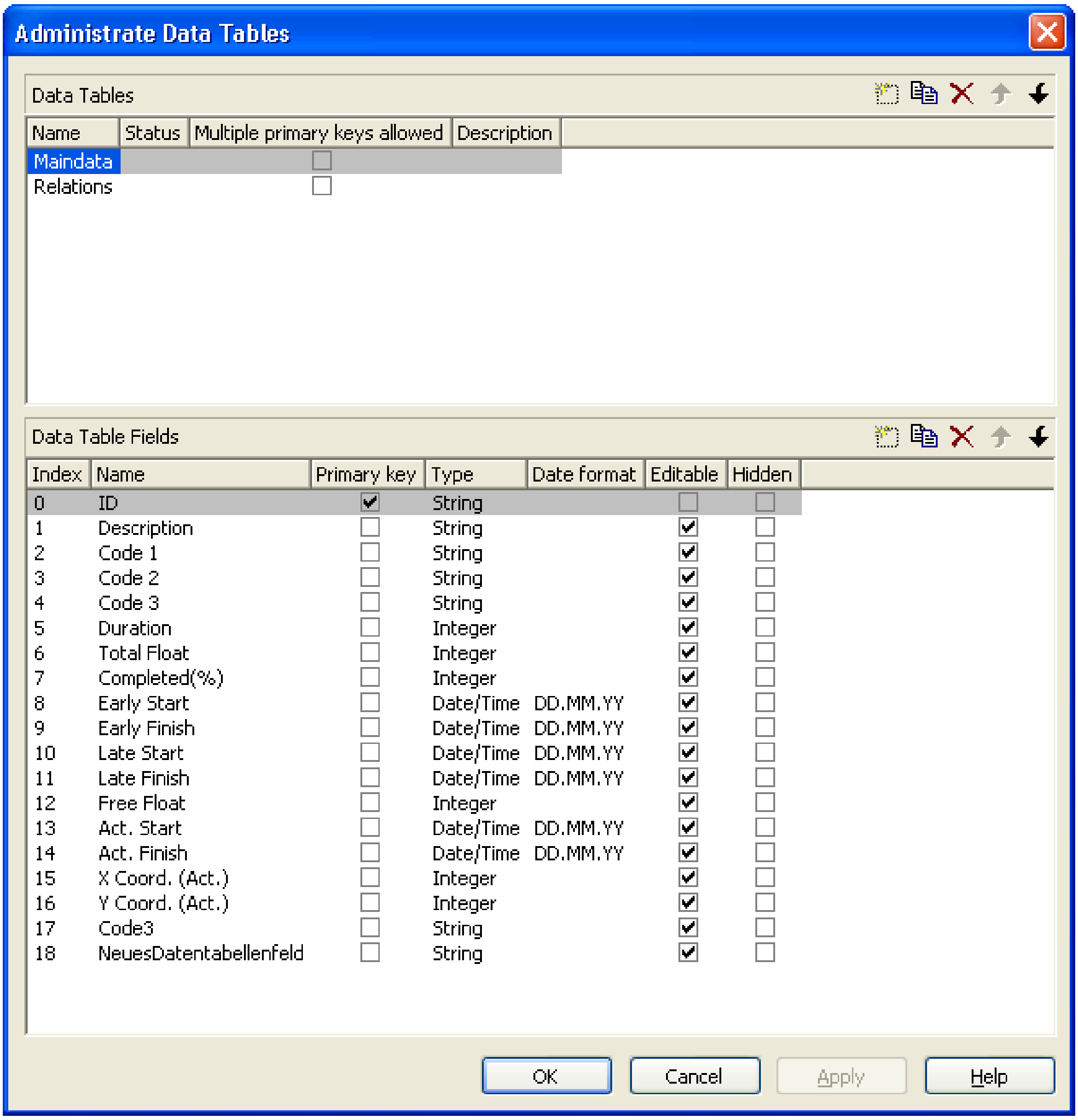
Schedule Input

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.

4.10 The "Administrate Data Tables" Dialog Box



Data Tables

Name	Status	Multiple primary keys allowed	Description
Maindata		□	
Relations		□	

Data Table Fields

Index	Name	Primary key	Type	Date format	Editable	Hidden
0	ID	☑	String		□	□
1	Description	□	String		☑	□
2	Code 1	□	String		☑	□
3	Code 2	□	String		☑	□
4	Code 3	□	String		☑	□
5	Duration	□	Integer		☑	□
6	Total Float	□	Integer		☑	□
7	Completed(%)	□	Integer		☑	□
8	Early Start	□	Date/Time	DD.MM.YY	☑	□
9	Early Finish	□	Date/Time	DD.MM.YY	☑	□
10	Late Start	□	Date/Time	DD.MM.YY	☑	□
11	Late Finish	□	Date/Time	DD.MM.YY	☑	□
12	Free Float	□	Integer		☑	□
13	Act. Start	□	Date/Time	DD.MM.YY	☑	□
14	Act. Finish	□	Date/Time	DD.MM.YY	☑	□
15	X Coord. (Act.)	□	Integer		☑	□
16	Y Coord. (Act.)	□	Integer		☑	□
17	Code3	□	String		☑	□
18	NeuesDatentabellenfeld	□	String		☑	□

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)** fields. As soon as you have checked the box **Multiple primary keys allowed** you can select up to three data fields for the primary key in the **Data table fields** section. The box **Multiple primary keys allowed** can only be unchecked if no more than one field is selected as primary key in the **Data table fields**section .
- **Description:** Here you can describe the data table.

Add / copy / delete / edit / promote / demote data table

By these buttons you can create, copy or delete data tables or move them by one position up or down in the list, respectively.

Data Table Fields

Here you can create and edit data table fields 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:

  String

  Integer

  Date/Time

  Double
- **Date format:** If the type **Date/Time** has been selected, you can specify the date format for the corresponding data field here. Choose a predefined date format or define your own date format (for example DD.MMM.YY hh:mm). You can compose the format of the following strings:

YY or **YYYY** (two-digit or four-digit figure for the year), **MM** or **MMM** (two-digit figure or three-digit character string for the month), **DD** (two-digit figure for the day), **hh** (two-digit figure for the hour), **mm** (two-digit figure for the minute), **ss** (two-digit figure for the second).

Please note that the date format set here needs to be the same as defined for your node dates.

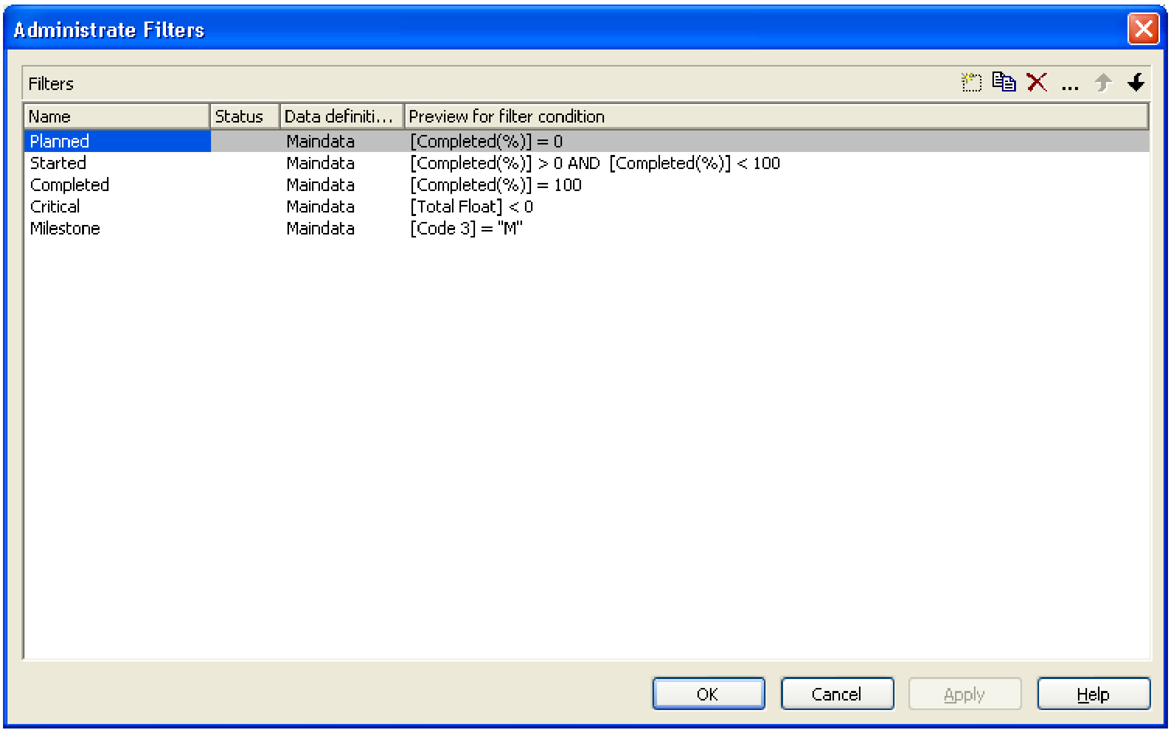
The date format set here only is relevant for entering data, but not for displaying data.

- **Editable:** Please activate this check box for all data table fields that shall be editable in the dialog **Edit Data**.
- **Hidden:** Please activate this check box for all data table fields that shall be hidden in the dialog **Edit Data**.
- **Relationship:**This field allows to define a relationship to another table. The data records of this table will be related to the data records of the other table by the field defined as the primary key. This is why only those tables are offered for selection for which a primary key was defined.

Add / copy / delete / edit / promote / demote data table field

By these buttons you can create, copy or delete data table fields or move them by one position up or down in the list, respectively.

4.11 The "Administrate Filters" Dialog Box



Name	Status	Data definiti...	Preview for filter condition
Planned		Maindata	[Completed(%)] = 0
Started		Maindata	[Completed(%)] > 0 AND [Completed(%)] < 100
Completed		Maindata	[Completed(%)] = 100
Critical		Maindata	[Total Float] < 0
Milestone		Maindata	[Code 3] = "M"

You can get to this dialog box via the **Objects** or the **Links** 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.

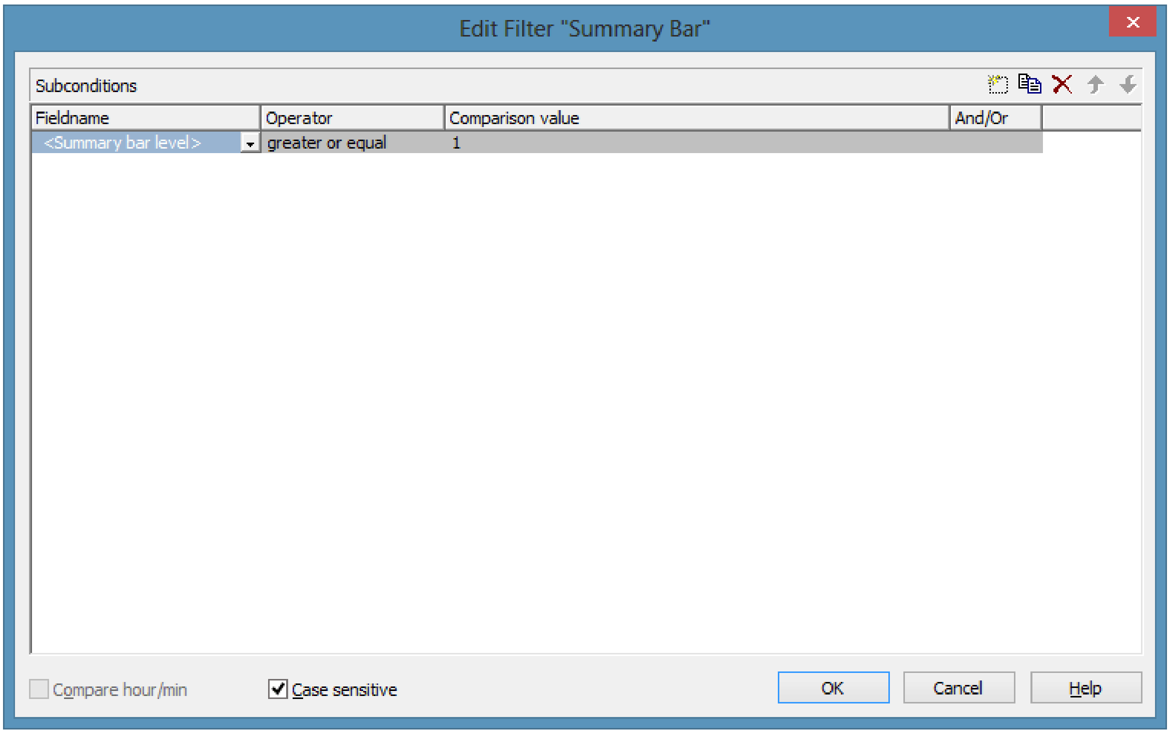
Edit filter

... 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.

4.12 The "Edit Filter" Dialog Box



Edit Filter "Summary Bar"

Subconditions

Fieldname	Operator	Comparison value	And/Or
<Summary bar level>	greater or equal	1	

☐ Compare hour/min

☑ Case sensitive

You can get to this dialog box either

- by the **Objects** property page
- or by the **Administrate Node Appearances** dialog box
- or by the **Administrate Link Appearances** dialog box, where you can activate the **Administrate Filters** dialog box and then click on the **Edit filter** button. The head line of this dialog box displays the name of the filter being edited.

Add subcondition

Inserts a new line for a subcondition above the selected line.

Copy subcondition

Copies the selected subcondition.

Delete subcondition

Deletes the selected subcondition.

Evaluate subcondition earlier/later

↑ ↓ If a filter consists of several subconditions, 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 with the comparison value.

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).

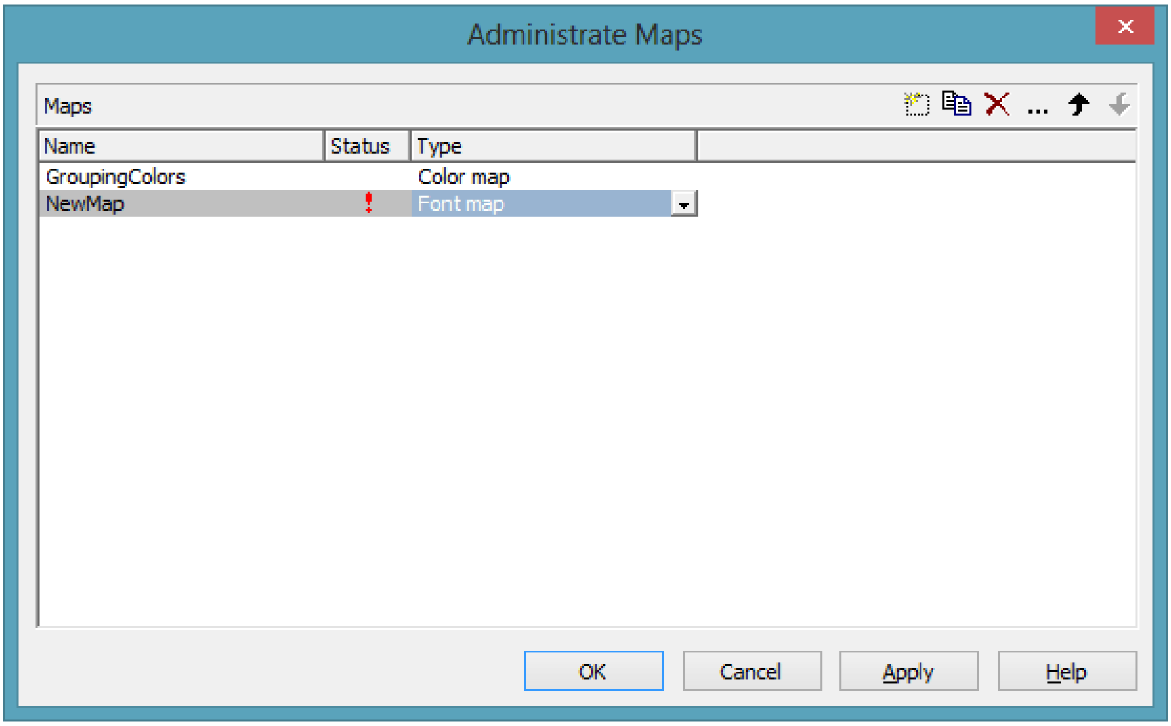
Compare hour/min

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.

4.13 The "Administrate Maps" Dialog Box



Name	Status	Type
GroupingColors		Color map
NewMap	!	Font map

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 (for further development)
- Graphics file maps

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.

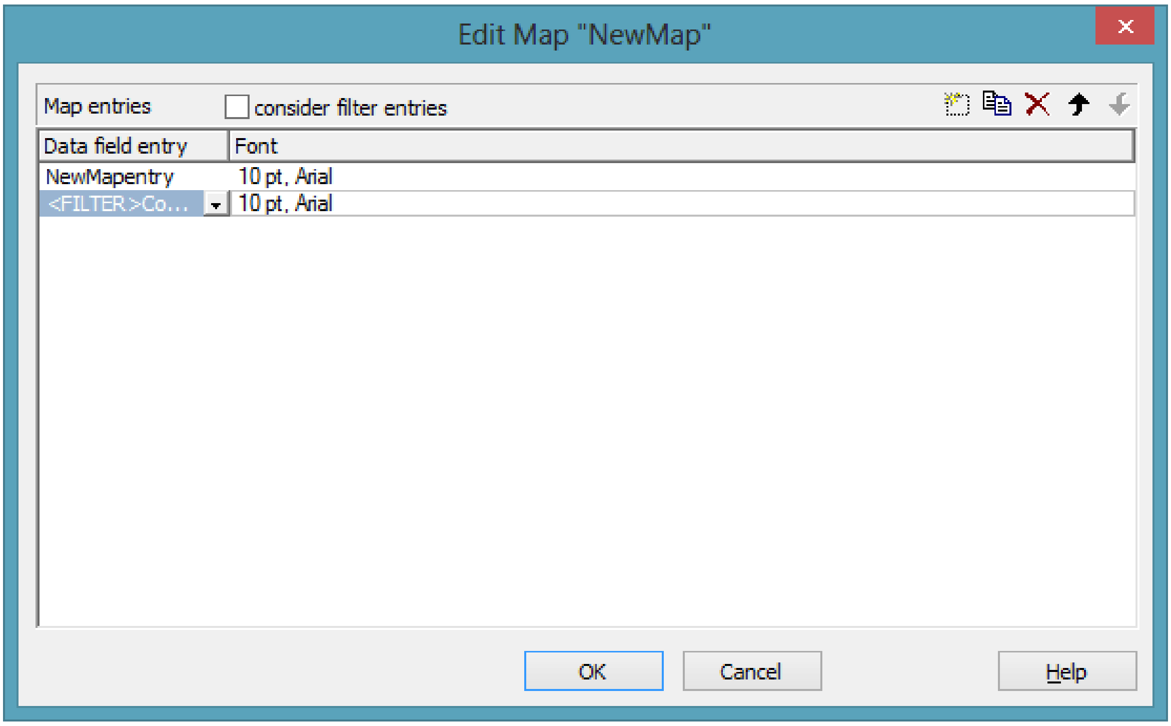
Edit map

... 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.

4.14 The "Edit Map" Dialog Box



Data field entry	Font
NewMapentry	10 pt, Arial
<FILTER>Co...	10 pt, Arial

You invoke this dialog box by clicking the **Edit map** button (…) of the **Administrate Maps** dialog box.

In a map you can set up to 150 allocations. If you wish to set more allocations, please create a new map, e. g. as a copy of an existing one.

consider filter entries

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

Data field entry

Specify the entries of the data field selected for which colors or graphics files respectively and legend texts are to be assigned.

Color/Graphics File Name

Assign colors or graphics files respectively to the data field entries. To do so, click on the corresponding field. Then a dialog box opens that lets you select a color or a graphics file respectively.

If a relative file name has been specified, at run time the file will be searched in the path set in the VARCHART ActiveX property **FilePath** first. If it won´t be found there, the file will be searched in the current directory of the application and in the installation directory of the control.

Legend text

(only for color and pattern maps) 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.

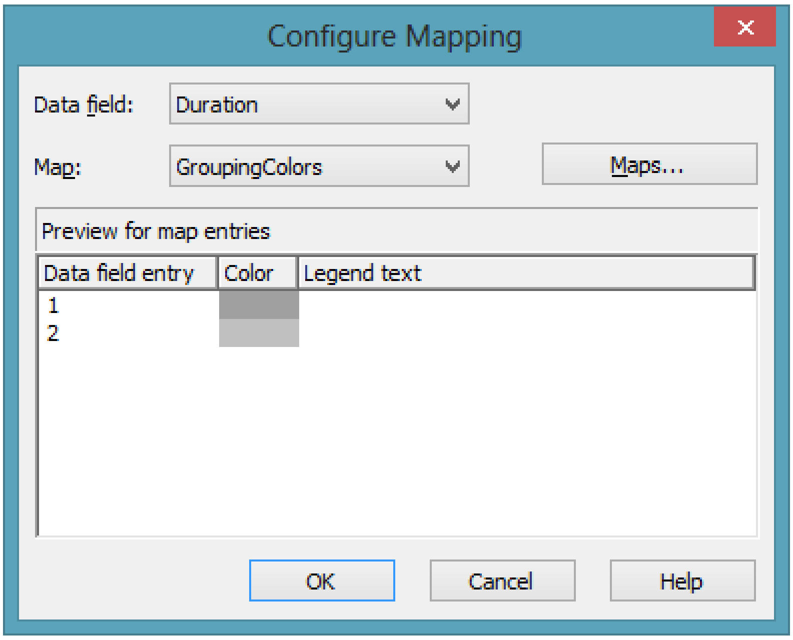
Delete map entry

✕ 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.

4.15 The "Configure Mapping" Dialog Box



Data field: Duration

Map: GroupingColors

Preview for map entries

Data field entry	Color	Legend text
1	▇	
2	▇	

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 assigns a color or a graphics file to the data field entries.

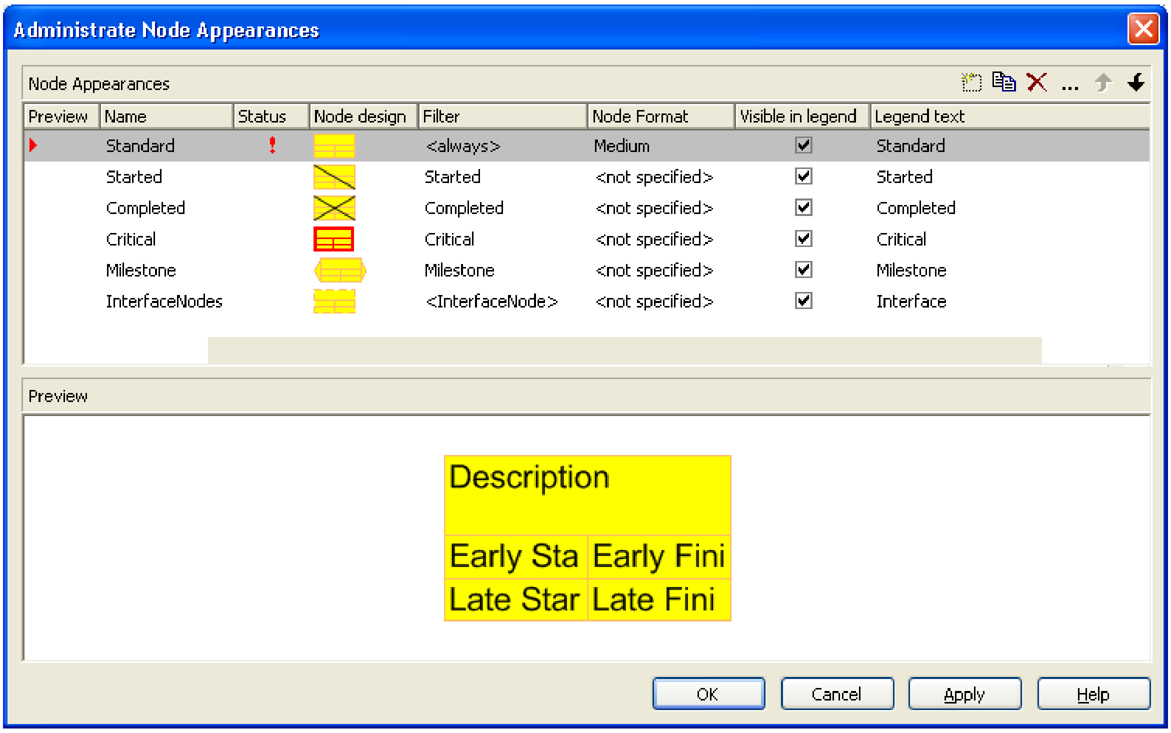
Maps

Opens the **Administrate Maps** dialog box, where you can create, edit, copy or delete maps.

Preview for map entries

The preview shows the selected map: the data field entries and the colors and legend texts or the graphics files respectively assigned to the data field entries.

4.16 The "Administrate Node Appearances" Dialog Box



Preview	Name	Status	Node design	Filter	Node Format	Visible in legend	Legend text
▶	Standard	!		<always>	Medium	☑	Standard
	Started			Started	<not specified>	☑	Started
	Completed			Completed	<not specified>	☑	Completed
	Critical			Critical	<not specified>	☑	Critical
	Milestone			Milestone	<not specified>	☑	Milestone
	InterfaceNodes			<InterfaceNode>	<not specified>	☑	Interface

Description	
Early Sta	Early Fini
Late Star	Late Fini

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

The appearance of nodes is defined by using filters to dynamically assign one or more node appearances to the nodes.

Preview

All node appearances marked by a small arrowhead in the **Preview** column are displayed and piled in the preview window in the sequence of working off.

The node appearance on which the cursor is currently positioned is marked by a green arrowhead.

Name

This column displays a list of names of the existing node appearances. The names can be edited.

Status

In this column each node appearance added (✳) and/or modified (!) after the dialog box was last opened is marked by a symbol.

Node design

Displays a representation of each node appearance. To modify a node design, i. e. the graphical attributes of a node appearance, click on the **Edit node appearance** button above the table or double-click on the **Node design** representation to get to the **Edit Node Appearance** dialog box.

Filter

The filter that is associated with a node appearance selects for nodes to wich the node appearance should be assigned.

For most node appearances you can select a filter of your choice. Only for the node appearances "Standard" and "Collapsed" the filters were pre-selected ("<always>" or "<InterfaceNode>").

To assign a filter to a node appearance, mark the **Filter** field. Two buttons will appear: a button of a select box which lists all available filters and an **Edit** button. Either select a filter for the node appearance from the select box, or click on the **Edit** button to get to the **Administrate Filters** dialog box where you can edit, copy, define or delete filters.

Node format

A node format defines the number, arrangement and format of the fields used to annotate a node in your charts. In this column, select the node format for the appropriate node appearance. To do so, mark the **Node format** field. Two buttons will appear: a button of a select box which lists all available formats and an **Edit** button. Either select a format from the select box, or click on the **Edit** button to get to the **Administrate Node Formats** dialog box where you can edit, copy, define or delete node format.

Visible in legend

Activate this check box for all node appearances that are to be visible in the legend.

Legend text

Enter a legend text for a node appearance.

Add node appearance

A new node appearance is added to the end of the list.

Copy node appearance

Copies the selected node appearance.

Delete node appearance

✕ This button lets you delete a node appearance that is not needed any more. Before it can be deleted, you need to answer a confirmation request. The node appearance "Standard" cannot be deleted.

Edit node appearance

... This button gets you to the dialog **Edit Node Appearance**.

Work off the node appearance earlier/later

If more than one node appearance is assigned to a node, the node appearances are worked off one after the other. The table lists the node appearances according to their processing order. The default node appearance is always at the top of the table as it is always applied and processed first. The node appearance processed last is located at the bottom of the table.

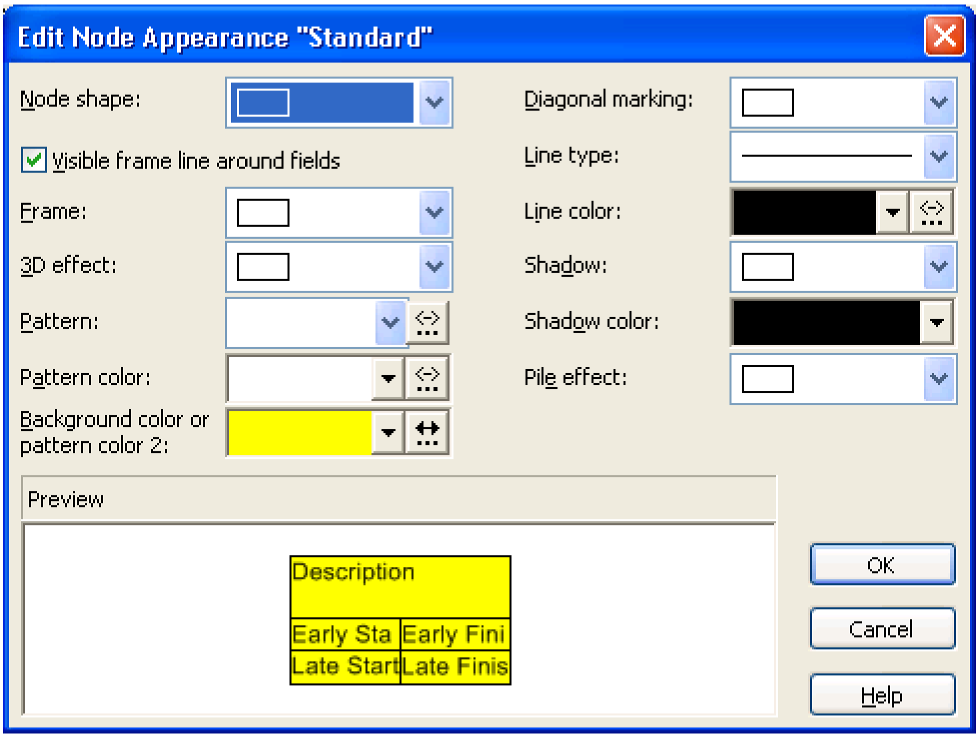
If several node appearances apply to a node, the attributes of each node appearance are replaced by the attributes of the node appearances that are processed later. Only the attributes whose value is "not specified" do not replace the attributes of their predecessors.

You can use these buttons to change the processing priority of a highlight:

↑ The selected node will be moved up one position in the table and processed correspondingly earlier.

↓ The selected node will be moved down one position in the table and processed correspondingly later.

4.17 The "Edit Node Appearance" Dialog Box



Edit Node Appearance "Standard"

Node shape:

☑ Visible frame line around fields

Frame:

3D effect:

Pattern:

Pattern color:

Background color or pattern color 2:

Diagonal marking:

Line type:

Line color:

Shadow:

Shadow color:

Pile effect:

The title line displays the name of the node appearance being edited.

If several appearances have been assigned to a node, the attributes of an appearance of low priority will be replaced by the attributes of an appearance of high priority, except for attributes that are set to "unchanged".

Node shape

This field lets you select a node shape or the entry <not specified> or <without frame>.

Visible frame line around fields

With this property you can specify whether the frame lines around fields shall be visible or not. This does not concern the outer frame line of the shape so that the effects of the property may vary depending on the frame shape. It has, for example, no effect on the type **vcRectangle**.

This feature can also be set by the property **VcNodeAppearance.FrameAroundFieldsVisible** gesetzt werden.

Frame

This field lets you specify whether the nodes are displayed with an ordinary or a double frame.

3D effect

This field lets you specify whether a three dimensional appearance is added to the nodes.

Pattern

This field lets you select a background pattern for the node appearance.

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

By the second button you can get to the **Configure Mapping** dialog box.

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

Pattern color

This field lets you select a pattern color for the node.

By the arrow button you can open the Color picker to select a line color.

By the second button you reach the **Configure Mapping** dialog box.

If a mapping was configured, the arrow on the button will appear solid.

Background color or pattern color 2

This field lets you select a background color of the node appearance.

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

By the second button you can get to the **Configure Mapping** dialog box.

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

Diagonal marking

This field lets you specify whether a diagonal marking is to be applied to the nodes and lets you select the type of diagonal marking.

Line type

This field lets you select a line type for the frame line of the node.

Line color

This field lets you select a color for the frame line of the node.

▼ By the arrow button you can open the Color picker to select a line color.

↔ By the second button you reach the **Configure Mapping** dialog box.

↔ If a mapping was configured, the arrow on the button will be displayed in solid.

Shadow

This field lets you add a shadow to the nodes.

Shadow color

Select the color for the shadow or the pile effect.

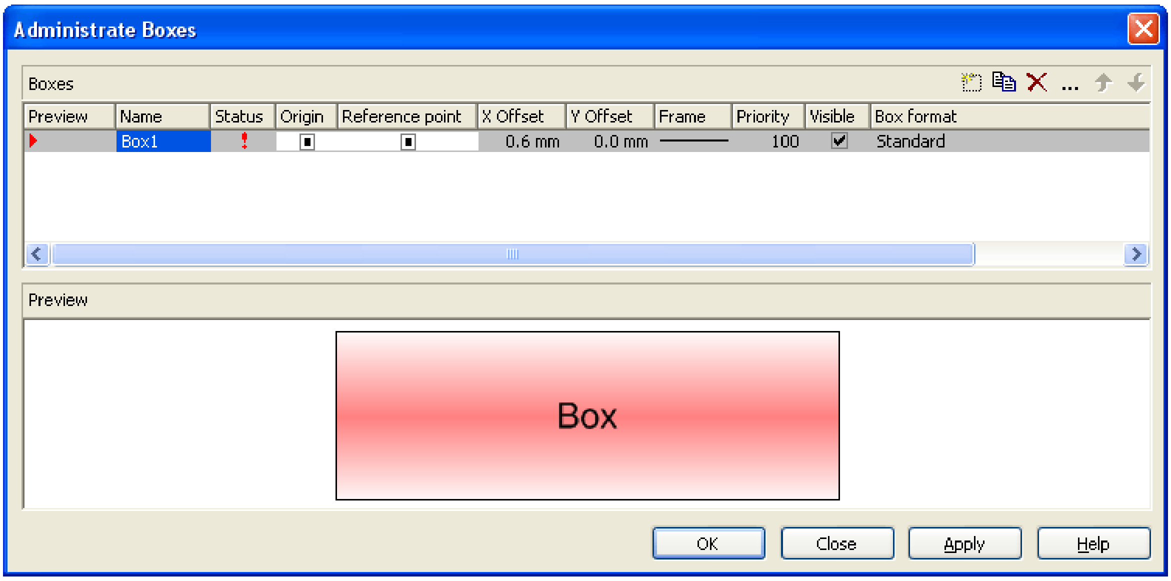
Pile effect

By this field you can set, whether or not nodes are to be displayed as a pile. A pile may consist of up to eight nodes.

Preview

By this window the current node appearance is displayed.

4.18 The "Administrate Boxes" Dialog Box



Preview	Name	Status	Origin	Reference point	X Offset	Y Offset	Frame	Priority	Visible	Box format
▶	Box1	!	▣	▣	0.6 mm	0.0 mm	───	100	☑	Standard

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

Preview

The 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.

Update behavior

Select an update behavior for this box. Leaving the setting to <not selected> means that the setting for boxes made in the **Edit Update behavior** dialog will apply

Moveable

By moving a box its offset will be modified. Activate this check box if the box is to be moveable in the diagram at run time. Deactivate the check box if you do not want the box to be moved at run time.

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.

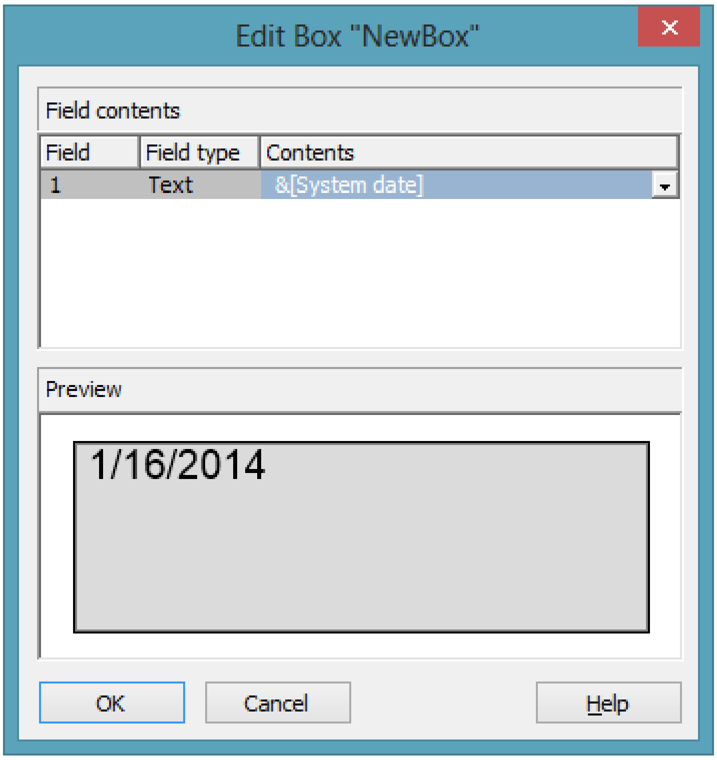
Edit box

... 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.

4.19 The "Edit Box" Dialog Box



Edit Box "NewBox"

Field contents

Field	Field type	Contents
1	Text	&[System date]

Preview

1/16/2014

You can get to this dialog by the **Objects** property page and the dialog box **Administrate Boxes** by clicking on the the **Edit box** button. This dialog box will also appear at run time when double-clicking on a box.

Field

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

Field Type

This column displays the field types (text or graphics).

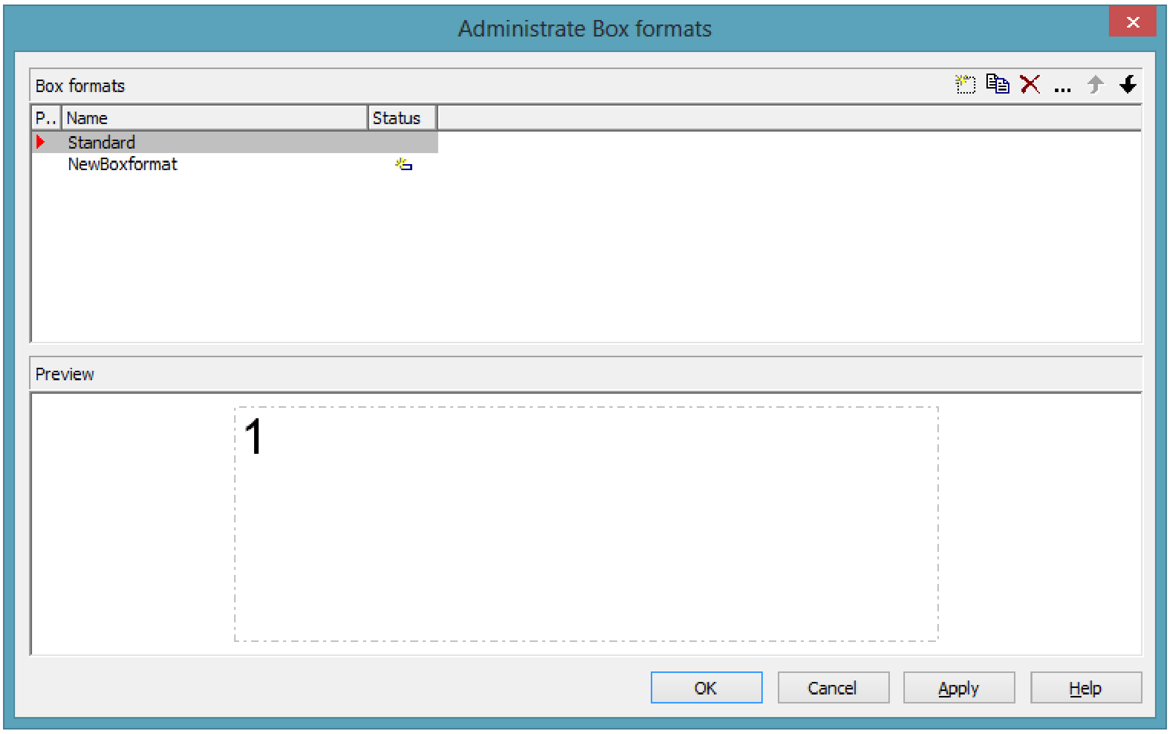
Contents

Type the contents of the field or a graphics file name here.

If a text field contains more than one line, you can use "\n" in the text string to separate two lines of the text field (Example: "Line1\nLine2"). Otherwise the lines will be separated at blanks.

Graphics formats available: WMF, JPG, BMP, GIF, PCX, PNG, TIF.

4.20 The "Administrate Box/Node Formats" Dialog Box



P..	Name	Status
▶	Standard	
	NewBoxformat	

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/node format

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

Copy box/node format

⧉ The marked format will be copied.

Delete box/node format

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

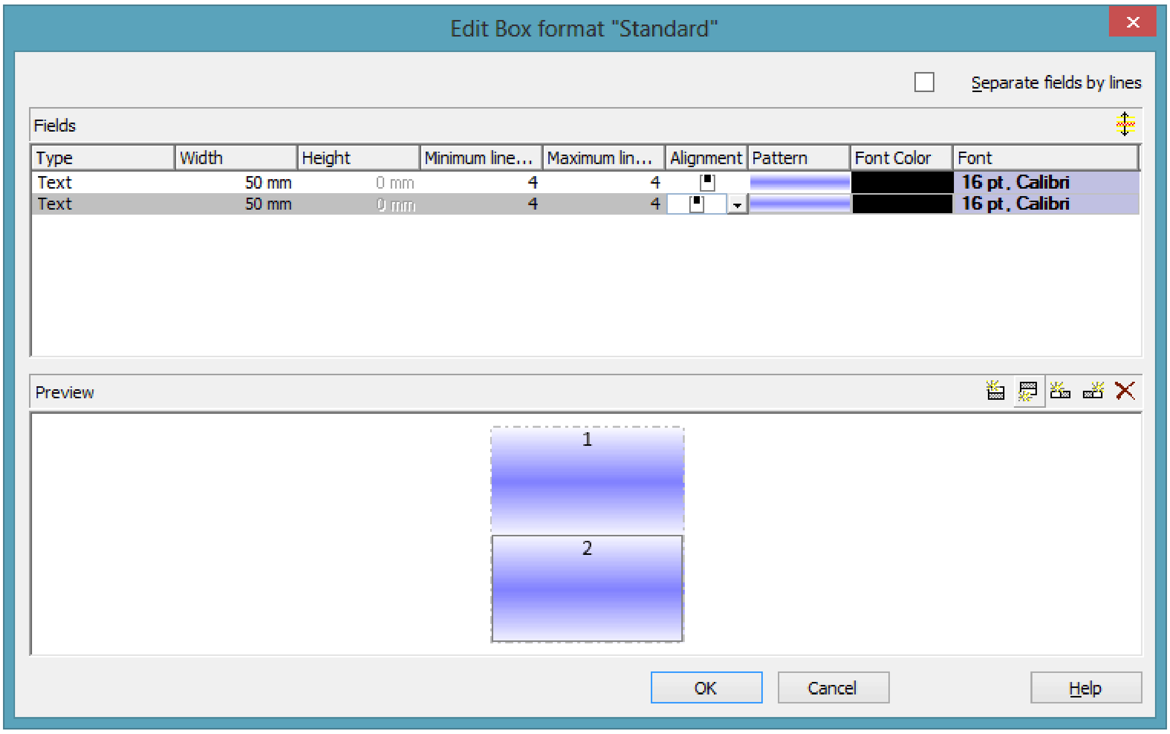
Edit box/node format

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

Promote / demote box / node format

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

4.21 The "Edit Box Format" Dialog Box



Type	Width	Height	Minimum line...	Maximum lin...	Alignment	Pattern	Font Color	Font
Text	50 mm	0 mm	4	4			■	16 pt, Calibri
Text	50 mm	0 mm	4	4			■	16 pt, Calibri

This dialog box will appear if you activate the **Administrate Box Formats** dialog box on the **Objects** property page and then click on the **Edit box format** button.

Separate fields by lines

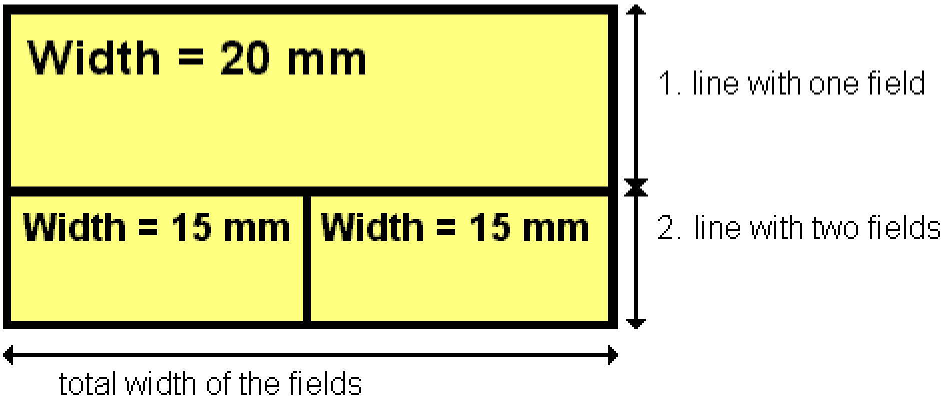
Activate this check box if the box fields are to be separated by lines.

Type

Select the field type: text or graphics.

Width

Specify the width for the selected field (in mm). The maximum width of a field is 200 mm. If the rows are split into two or more fields and the total widths of the rows vary, the total width will be equal to the width of the widest row.



Height

(only for the type graphics) Specify the minimum height for the selected field (in mm). The maximum height is 200 mm.

Minimum/Maximum line count

(only for the type text) Specify the minimum/maximum number of lines of text that can be displayed in the current field. Each field can contain a maximum of nine lines of text.

Alignment

Specify the alignment of the content of the selected field (9 possibilities).

Pattern

Select the fill pattern and color for the current field. By clicking on ... you open the **Edit pattern attributes** dialog where you can specify a pattern, a background color and, if needed, a second pattern color . You can define your own colors in addition to the ones suggested. Also, transparent colors are available.

Font Color

(only for the type text) Indicates the font color for the current field.

▾ By the **arrow** button you can open the color picker to select a font color.

Font

(only for the type text) Indicates the font style for the current field.

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

Apply selected property to all fields

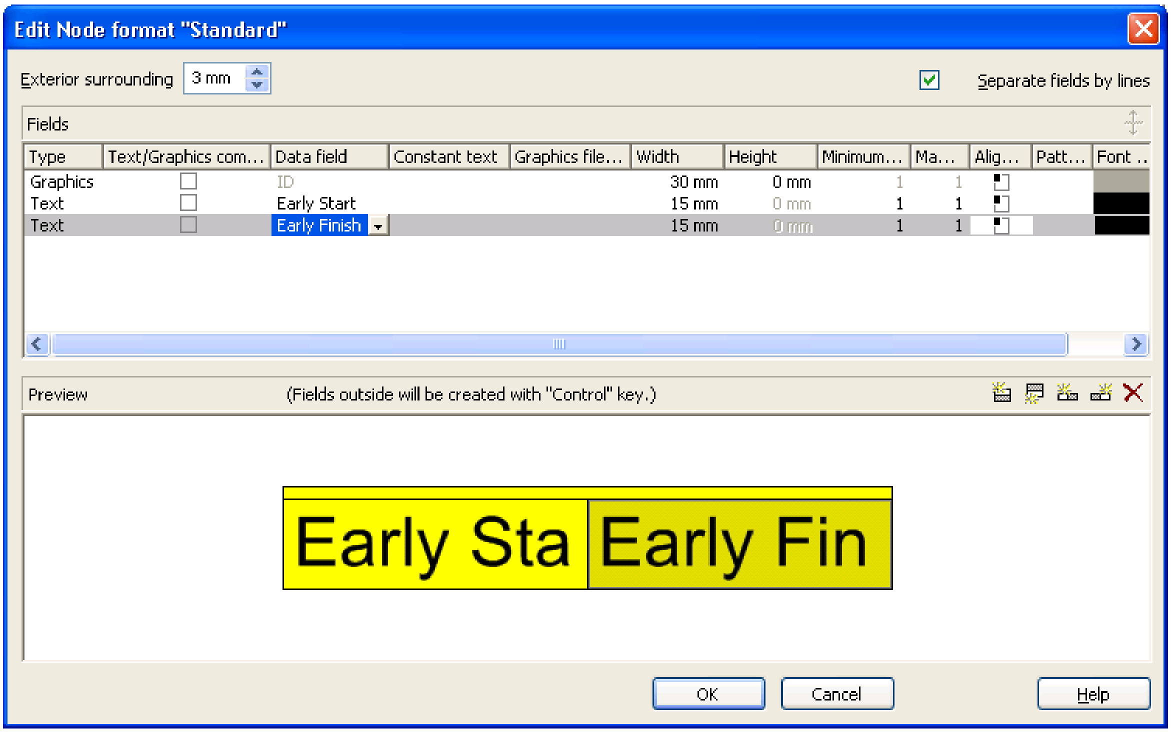
Applies the marked property to all fields.

Preview

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

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

4.22 The "Edit Node Format" Dialog Box



Edit Node format "Standard"

Exterior surrounding: 3 mm

☑ Separate fields by lines

Fields

Type	Text/Graphics com...	Data field	Constant text	Graphics file...	Width	Height	Minimum...	Ma...	Alig...	Patt...	Font ..
Graphics	□	ID			30 mm	0 mm	1	1			
Text	□	Early Start			15 mm	0 mm	1	1			■
Text	□	Early Finish			15 mm	0 mm	1	1			■

Preview (Fields outside will be created with "Control" key.)

This dialog will open after clicking on the **Edit format** button of the **Administrate Node Formats** dialog.

Exterior surrounding

By this field you can set the distance between nodes or between a node and the margin of the chart. Unit: 1/100 mm. The default is 300, i.e. 3 mm. If you choose a value smaller than this, graphical elements in the chart may overlap. You should use values below the default only if there are good reasons for it.

Separate fields by lines

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

Type

Select the field type: text or graphics.

Text/Graphic combined

If this combobox is activated, in the node field a text and a graphics can be combined as follows:

- **Type**: Text, **Text/Graphic combined**: no: only text will be displayed (as specified for **Data field** or for **Constant text**)
- **Type**: Graphics, **Text/Graphic combined**: no: only a graphics will be displayed (as specified for **Graphics file name**)
- **Type**: Text, **Text/Graphic 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/Graphic 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 whose content is to be displayed in the current field. If the content of a data field does not fit into the current field, the excess will be cropped in the diagram.

Constant Text

(only if no data field has been specified) Type a constant text to be displayed in the current field.

Graphics file name

Indicates the name and directory of the graphics file that will be displayed in the current field.

As soon as you click on a **Graphics file name** field, two buttons appear:

... Click the first button to open the Windows dialog box **Choose Graphics File**. There you can select a graphics file to be displayed in the current format field.

If a relative file name has been specified, at run time the file will be searched in the path set in the VARCHART Windows Forms property **FilePath** first. If it won´t be found there, the file will be searched in the current directory of the application and in the installation directory of VARCHART Windows Forms control.

<-> ... Click this button if you want to use a map to display graphics in node fields in dependence on the node data. Then the **Configure Mapping** dialog

box will open which lets you configure a mapping from data field entries to graphics files.

If in the **Configure Mapping** dialog box only a data field, but no map is selected, the content of the data field will be used as graphics file name. If in the data field or in the map no valid graphics file name is found, the file name specified in the **Symbol file field** will be used.

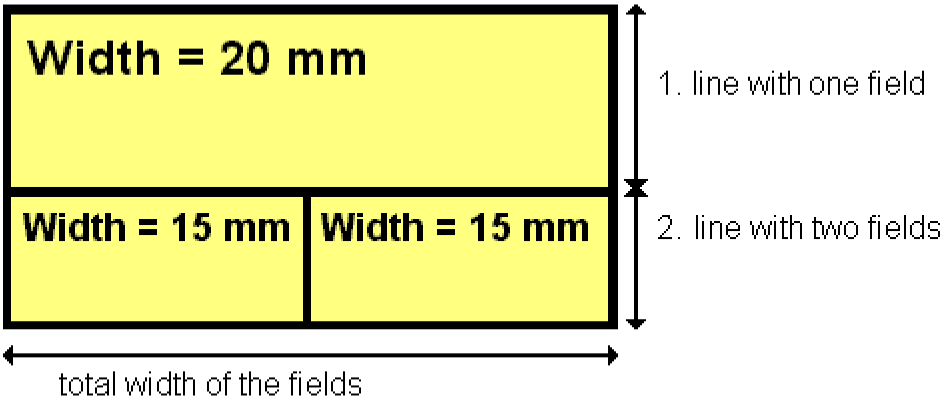
If a mapping has been configured, the arrow on the second button will be displayed in bold (↔).

↔ As soon as you leave the **Symbol File Name** field, a symbol indicates that a mapping has been configured.

When the graphics is displayed, the color of the pixel in the upper left corner will be replaced by the color of the diagram background. That means that all pixels of the graphics that have this color will be displayed transparent.

Width

Specify the width for the selected field (in mm). The maximum field width is 99 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 of node formats is 99 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 text/graphics in the selected field.

Pattern

Select the fill pattern and color for the current field. By clicking on ... you open the **Edit pattern attributes** dialog where you can specify a pattern, a background color and, if needed, a second pattern color . You can define your own colors in addition to the ones suggested. Also, transparent colors are available.

By clicking this button in the **Edit pattern attributes** you can get to the **Configure Mapping** dialog box where you can assign the respective attribute to fields in dependence of data.

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

If you do not set an attribute to a format field, the attribute of the node appearance will apply.

Font Color

(only for the type text) Specify the font color for the field. If you click on the field, two buttons will apperar:

▼ 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 on 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, a button will appear (...) that lets you open the Windows **Font** dialog box.

Apply selected property to all fields

Applies the marked property to all fields.

Preview

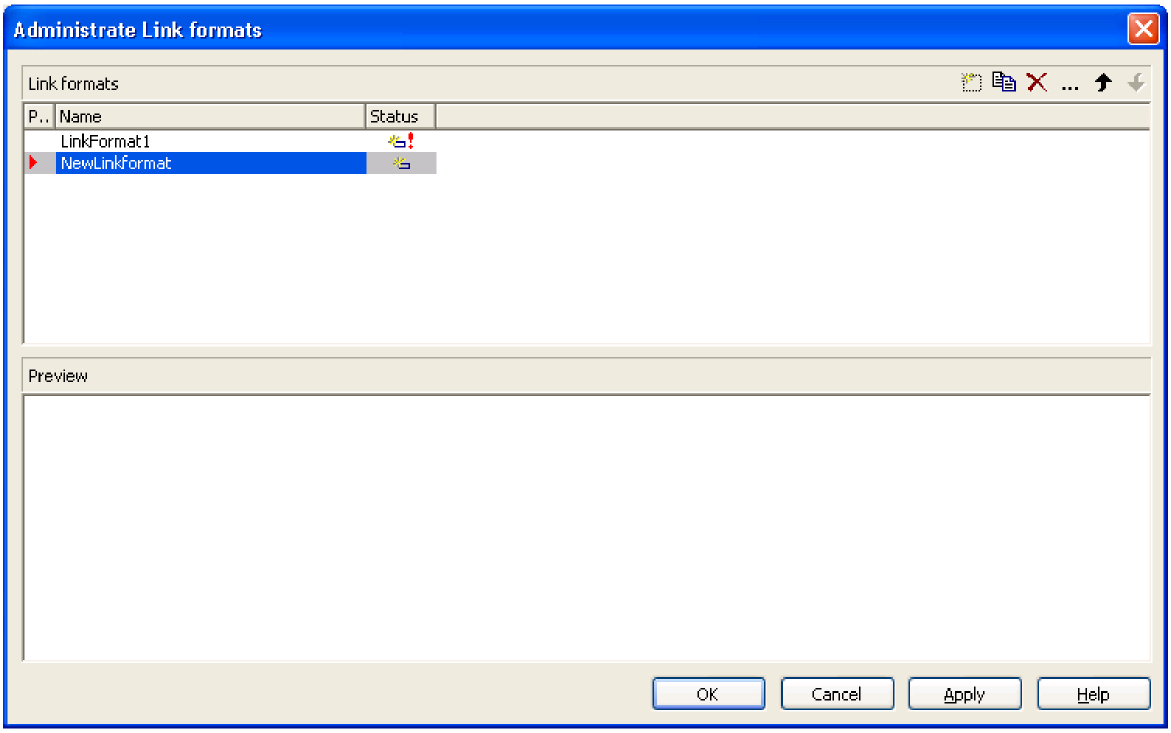
The current node format is displayed in the preview window. If you click on a field in the preview window you can modify its attributes in the **Fields** table.

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.

If you want to add new fields outside of the node, press the Ctrl button.

4.23 The "Administrate Link Formats" Dialog Box



P..	Name	Status
	LinkFormat1	✳ !
▶	NewLinkformat	✳

You can get to this dialog by clicking the **Link formats...** button on the **Objects** property page or by clicking … in the field **Link formats** in the dialog **Administrate Link Appearance**.

Preview

In this column a red triangle marks the link format which is displayed in the preview below.

Name

Lists the names of all link formats that are defined. The names can be edited.

Status

In this column each link format that has been added (✳) and/or modified (!) since the dialog box was opened is marked.

Add link format

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

Copy link format

Copies the selected line format.

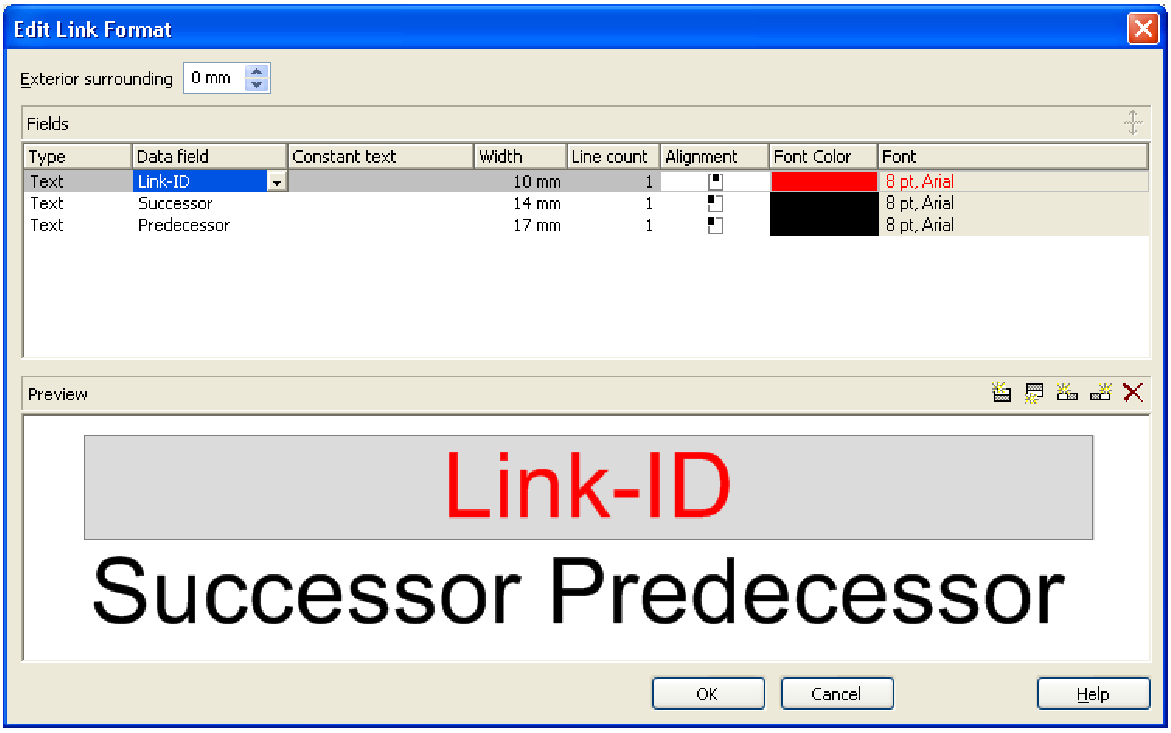
Delete link format

✕ The marked filter in the list will be deleted. You can only delete filters that are not currently used.

Promote / demote link format

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

4.24 The "Edit Link Format" Dialog Box



Exterior surrounding: 0 mm

Type	Data field	Constant text	Width	Line count	Alignment	Font Color	Font
Text	Link-ID		10 mm	1	⬒	🟥	8 pt, Arial
Text	Successor		14 mm	1	◧	⬛	8 pt, Arial
Text	Predecessor		17 mm	1	◧	⬛	8 pt, Arial

Link-ID
Successor Predecessor

This dialog will open after clicking on the **Edit format** button of the **Links** property page.

Exterior surronding

By this field you can set the distance between links and nodes. Unit: 1/100 mm. The default is 300, i.e. 3 mm. If you choose a value smaller than this, graphical elements in the chart may overlap. You should use values below the default only if there are good reasons for it.

Type

The field type is text.

Data field

Select the data field whose content is to be displayed in the current field.

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

Constant Text

(only if no data field has been specified) Type a constant text to be displayed in the current field.

Width

Specify the width for the selected field (in mm). The maximum field width is 99 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.

Line count

Specify the 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 text in the selected field.

Font Color

Specify the font color for the field. If you click on the field, a button will apperar (▾) that lets you open the Color picker to select a font color.

Font

Indicates the font style for the current field. If you click on the field, a button (...) will appear by which you can open the Windows **Font** dialog.

Apply selected property to all fields

Applies the marked property to all fields.

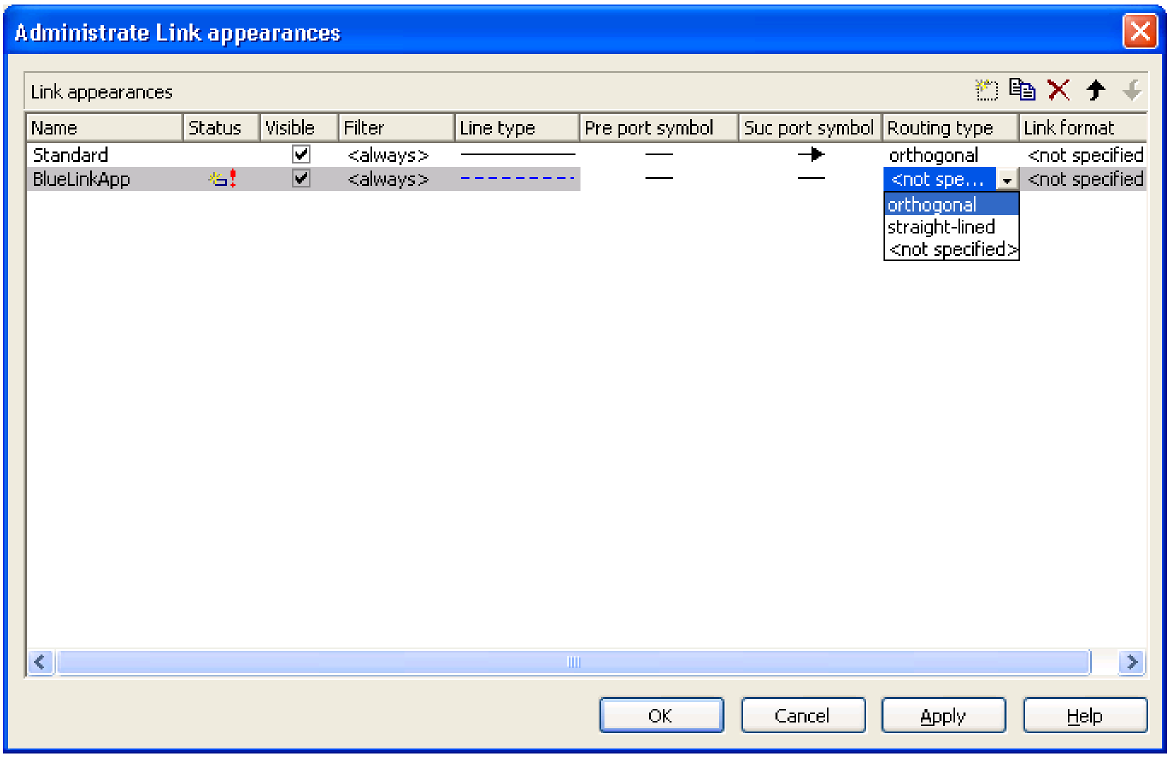
Preview

The current link format is displayed in the preview window. If you click on a field in the preview window 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.

4.25 The "Administrate Link Appearances" Dialog Box



Administrate Link appearances

Link appearances

Name	Status	Visible	Filter	Line type	Pre port symbol	Suc port symbol	Routing type	Link format
Standard		☑	<always>	———	—	→	orthogonal	<not specified
BlueLinkApp	✳ !	☑	<always>	- - - - - - - - -	—	—	<not spe...	<not specified

You can get to this dialog by clicking the **Link appearances** button on the **Objects** property page.

Name

This column displays the names of the link apperances available. The names can be edited.

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

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

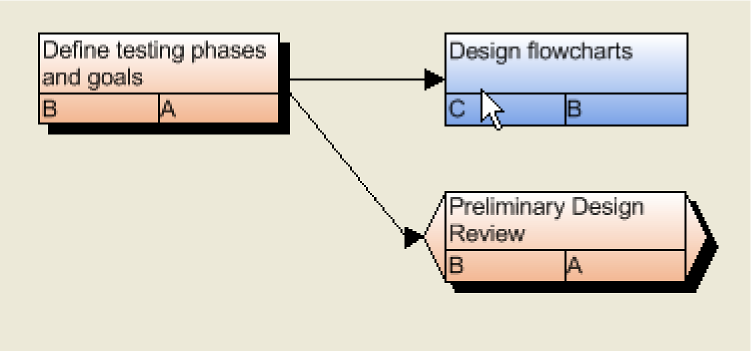
Select a port symbol for a link that visually accentuates the junction of the link and the successor node.

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

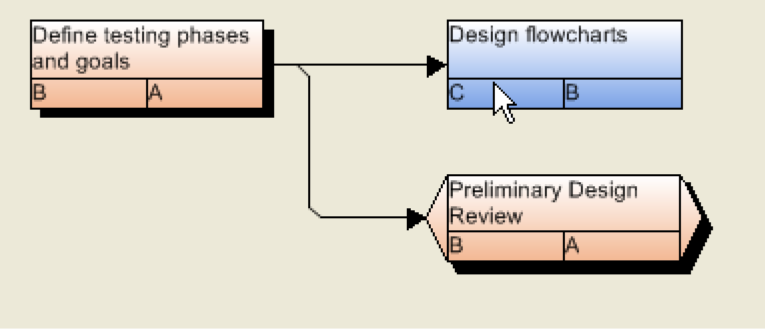
Routing type

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

Link format

Click on ▾ to select a link format or click on ... to open the dialog **Administrate link formats** where link formats can be created or edited.

This feature can also be set by the property **VcLinkAppearance.Format-Name>.**

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.

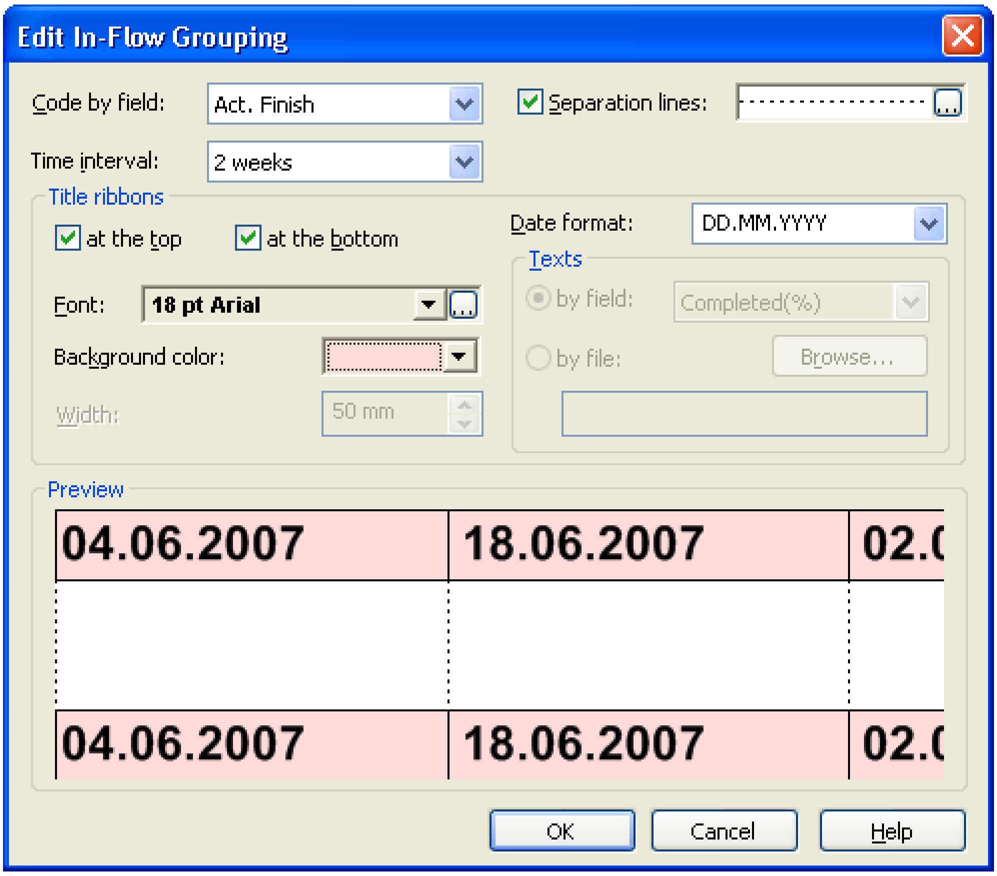
Delete link appearance

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 appearanceby one position up or down in the list.

4.26 The "Edit In-Flow Grouping" Dialog Box



Edit In-Flow Grouping

Code by field: Act. Finish

Time interval: 2 weeks

☑ Separation lines:

Title ribbons

☑ at the top ☑ at the bottom

Font: 18 pt Arial

Background color:

Width: 50 mm

Date format: DD.MM.YYYY

Texts

● by field: Completed(%)

○ by file:

Preview

04.06.2007	18.06.2007	02.(
04.06.2007	18.06.2007	02.(

You can get to this dialog via the Nodes property page. In this dialog you can define the criteria for the in-flow grouping and the layout. If the diagram has a left to right orientation, you can display an annotated ribbon at the top and/or bottom of the diagram area. For diagrams with a top to bottom orientation you can display an annotated ribbon at the left and/or right side of the diagram area.

Code by field

Select the data field that controls the in-flow grouping.

Time interval

(Only available if for ***Code by field*** *a date field is selected)* Specify the time interval that defines a time period for the ribbons (e.g. 1 second, 1 minute, 1 hour, 1 day, 2 months, 1 year).

Separation lines

Tick this box, if you want to display separating lines in the diagram. If you have chosen a top to bottom orientation, vertical separation lines will be displayed, otherwise horizontal ones. If you have selected a date field from the **Code by field** combobox, the distance of the separation lines is controlled by the value specified for the **Time Interval**. Otherwise after each value of the data field a separation line will be drawn.

By the **Edit** button you can get to the **Line Attributes** dialog box where you can specify the color, thickness and type of the lines.

Title ribbons at the top/at the bottom or at left/at right

Specify whether annotated ribbons should be displayed:

- left-to-right-orientation: **at the top** and/or **at the bottom** of the diagram
- top-to-bottom-orientation: **at left** and/or **at right** of the diagram.

Font

Indicates the style and color of the font used for the annotation of the ribbons.

... opens the Color Picker where you can select the font color.

▼ opens the Windows dialog box **Font**.

Background color

Specify the background color of the ribbons.

Width

(Only for top-to bottom orientation) Specify the width of the vertical ribbons in mm.

Date format

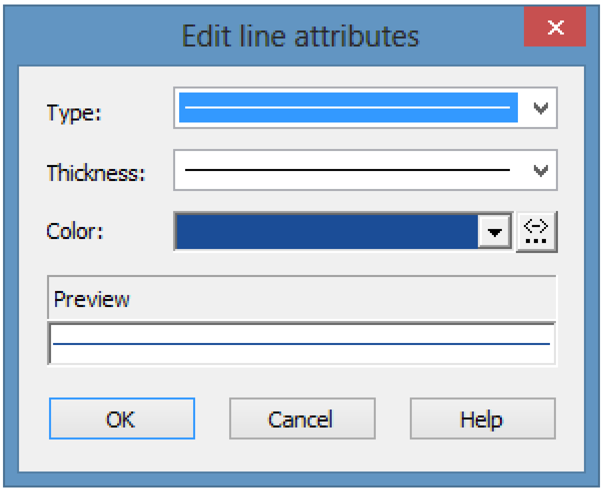
Select this option if you have selected a date field for **Code by field** and then specify the date format for the annotation of the ribbons.

Texts

- **by field:** Select this option, if the ribbon annotation shall be controlled by a data field.

- **by file:** Select this option, if the ribbon annotation shall be controlled by a file, and then specify the file name.

4.27 The "Edit Line Attributes" Dialog Box



This dialog which can in each case be invoked by clicking on ... is available for the link appearance, layers and for box frames.

Type

Select the line type (dashed, dotted etc.).

Thickness

Define the line thickness.

Color

Select the line color.

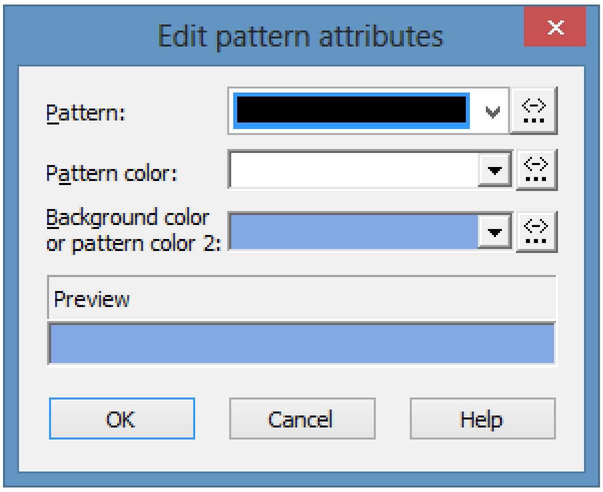
↔ This button will open the **Configure Mapping** dialog box where you can specify the line color data-dependent.

↔ After having mapped the line color, the arrow on the button will appear bold.

Preview

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

4.28 The "Edit Pattern Attributes" Dialog Box



The pattern dialog which can in each case be invoked by clicking on ... is available for filling of curves in a histogram, for calendar grids, for the group title, for intervals, for time scale sections, for box and node formats.

Pattern

Here you can select a fill pattern.

Pattern color

Select the foreground color of the fill pattern.

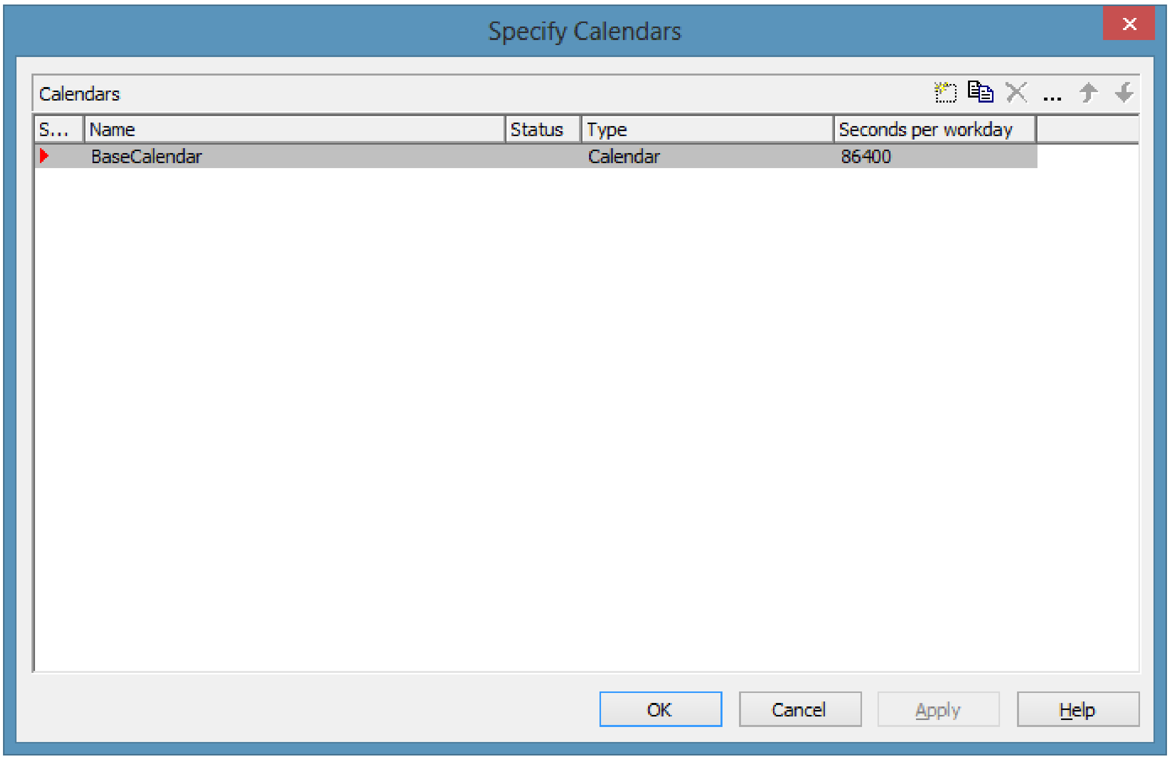
Background color or pattern color 2

Select the background color or a second pattern color.

Preview

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

4.29 The "Specify Calendars" Dialog Box



S...	Name	Status	Type	Seconds per workday
▶	BaseCalendar		Calendar	86400

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.

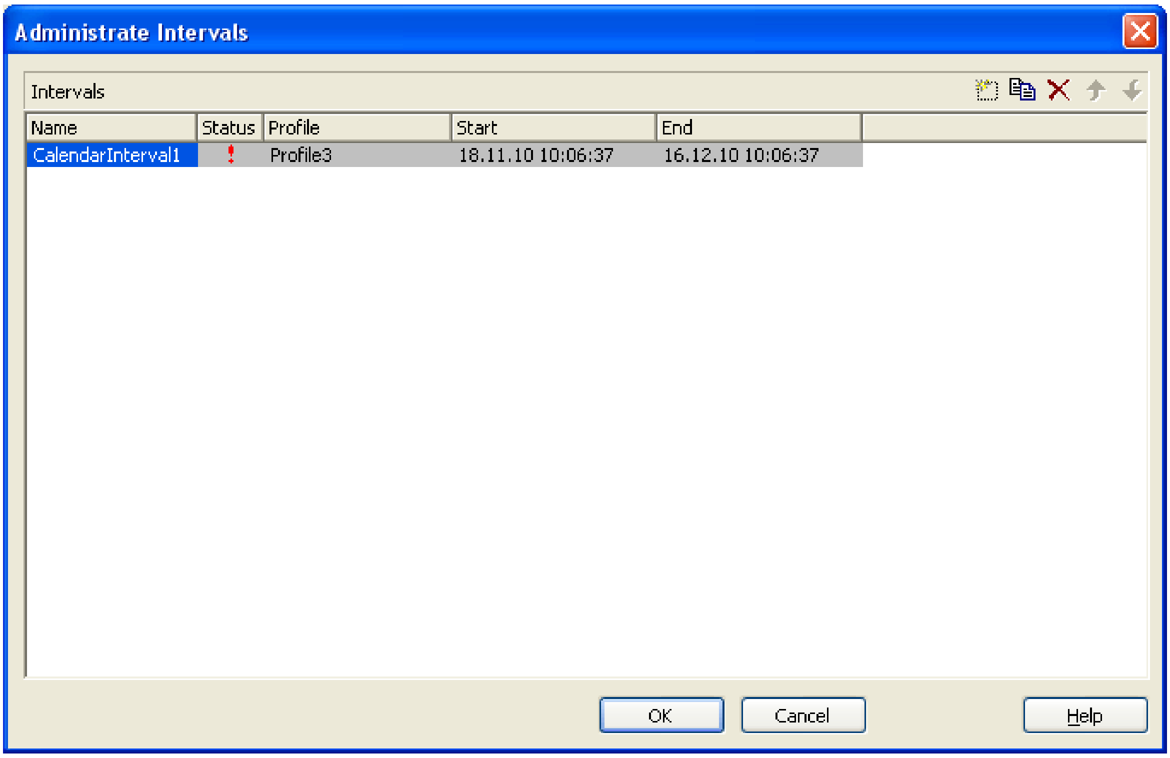
Delete calendar

✕ The marked calendar is deleted.

Edit calendar

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

4.30 The "Administrate Intervals" Dialog Box (Calendar)



Name	Status	Profile	Start	End
CalendarInterval1	!	Profile3	18.11.10 10:06:37	16.12.10 10:06:37

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.

Add interval

A new interval will be created. You can modify the marked name by double-clicking and editing it.

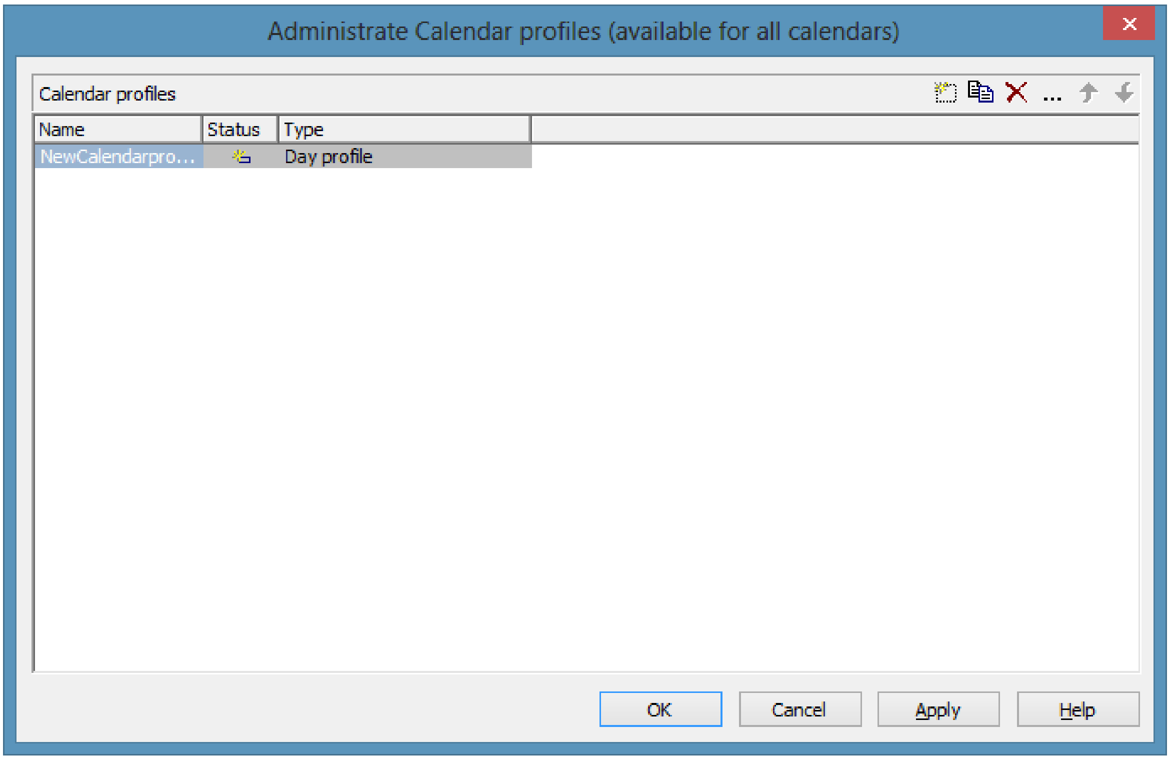
Copy interval

Click on this button to copy the marked interval.

Delete interval

Click on this button to delete the marked interval.

4.31 The "Administrate Calendar Profiles" Dialog Box


Administrate Calendar profiles (available for all calendars)
×
Calendar profiles
...
OK
Cancel
Apply
Help

Name	Status	Type
NewCalendarpro...	✱	Day profile

In this dialog you can create and modify calendar profiles.

Name

Lists the names of all calendar profiles. All names can be edited.

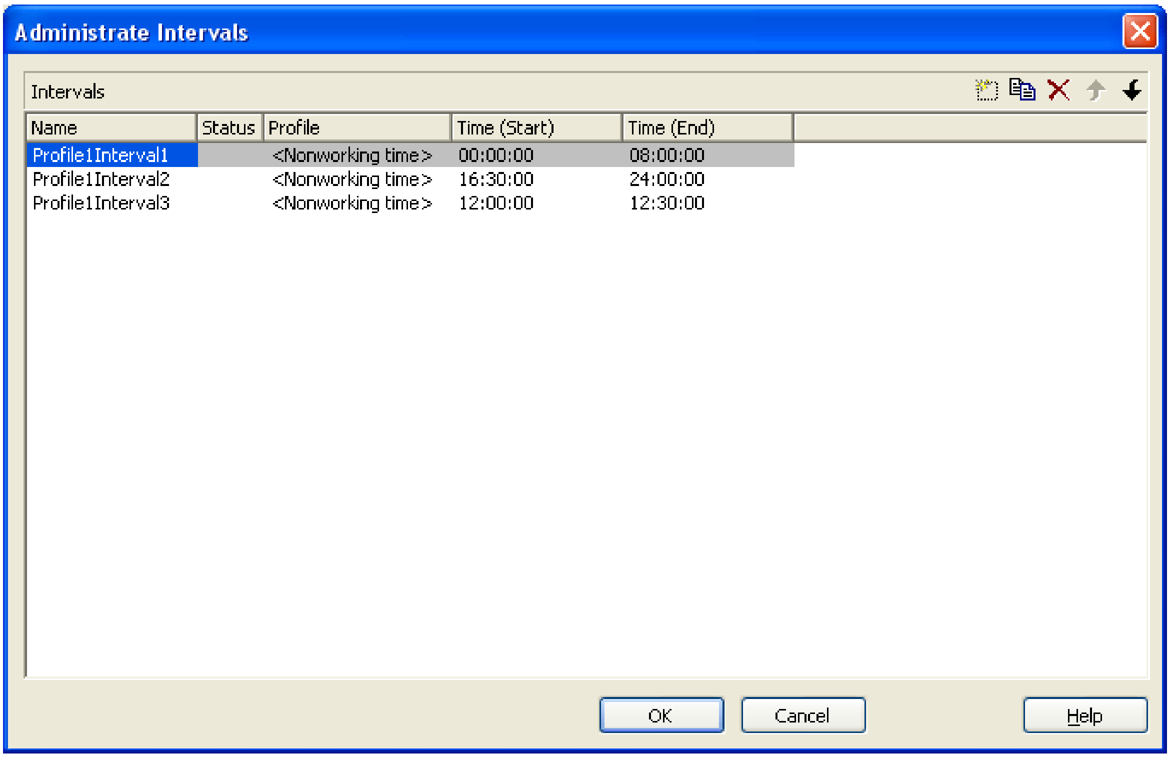
Status

In this column each calendar profile that has been added (✱) and/or modified (!) since the dialog box was opened is marked by a symbol.

Type

By clicking ▾ you can select the calendar profile type. You can choose between <Day profile>, <Week profile>, <Year profile> and <Variable profile>.

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



Name	Status	Profile	Time (Start)	Time (End)
Profile1Interval1		<Nonworking time>	00:00:00	08:00:00
Profile1Interval2		<Nonworking time>	16:30:00	24:00:00
Profile1Interval3		<Nonworking time>	12:00:00	12:30:00

You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a day profile.

Name

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

Status

In this column each interval that has been added (✱) and/or modified (!) since the dialog box was opened is marked by a symbol.

Profile

Here you can select a profile for your interval by clicking ▼.

Time Start/Time End

In this field you can set the start or end time of an interval by clicking on the arrow buttons.

Add interval

A new interval will be created. You can modify the marked name by double-clicking and editing it.

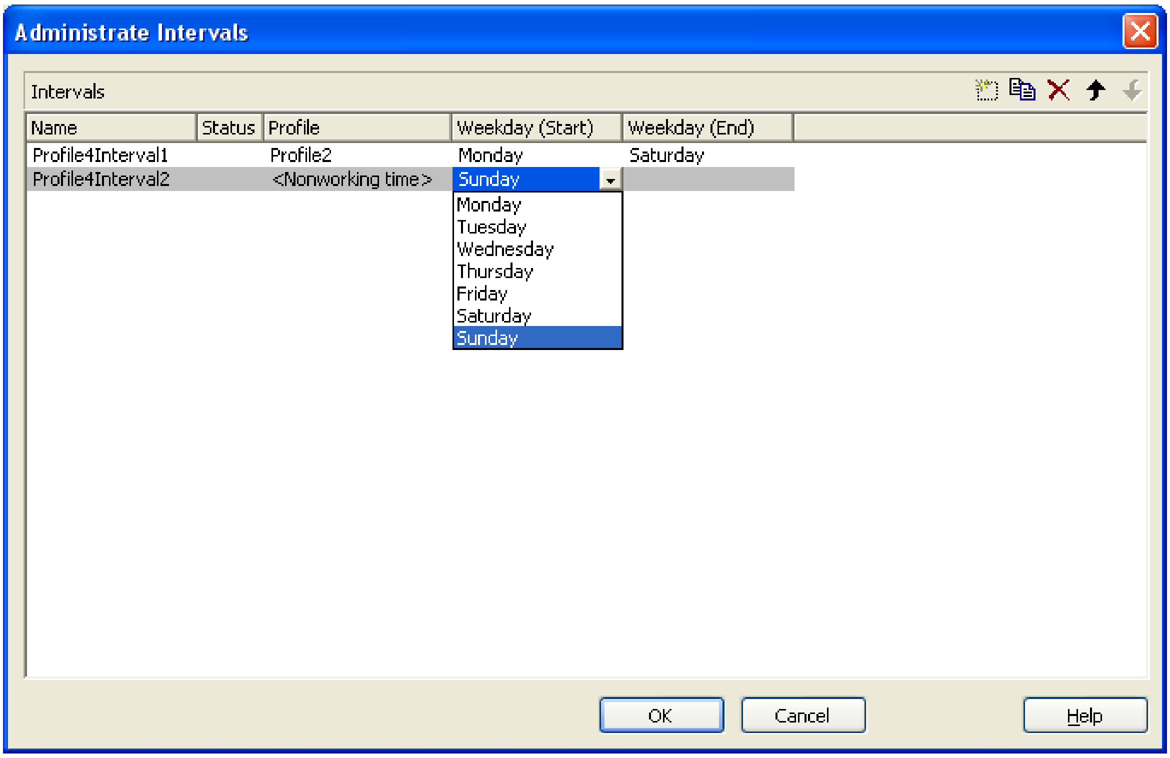
Copy interval

Click on this button to copy the marked interval.

Delete interval

Click on this button to delete the marked interval.

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



Name	Status	Profile	Weekday (Start)	Weekday (End)
Profile4Interval1		Profile2	Monday	Saturday
Profile4Interval2		<Nonworking time>	Sunday	

You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a week profile.

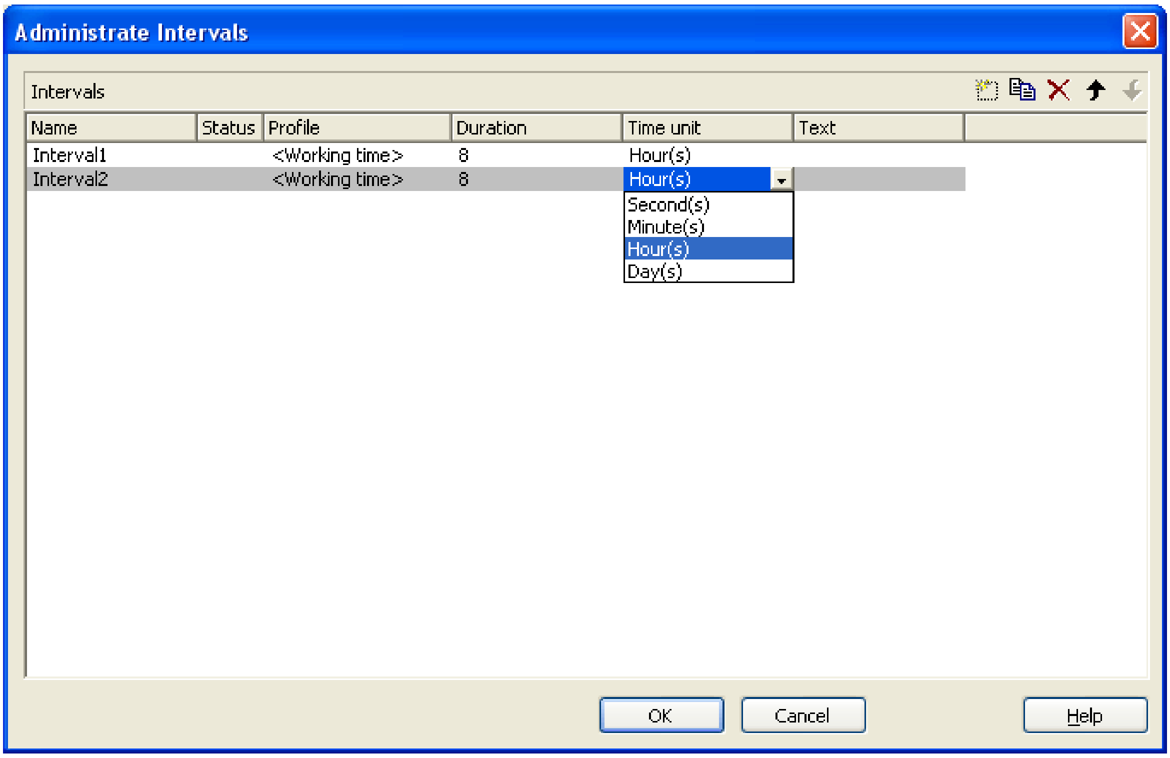
Weekday Start/Weekday End

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

Weekday Start/Weekday End

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

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



Name	Status	Profile	Duration	Time unit	Text
Interval1		<Working time>	8	Hour(s)	
Interval2		<Working time>	8	Hour(s)	

You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a variable profile.

Duration

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

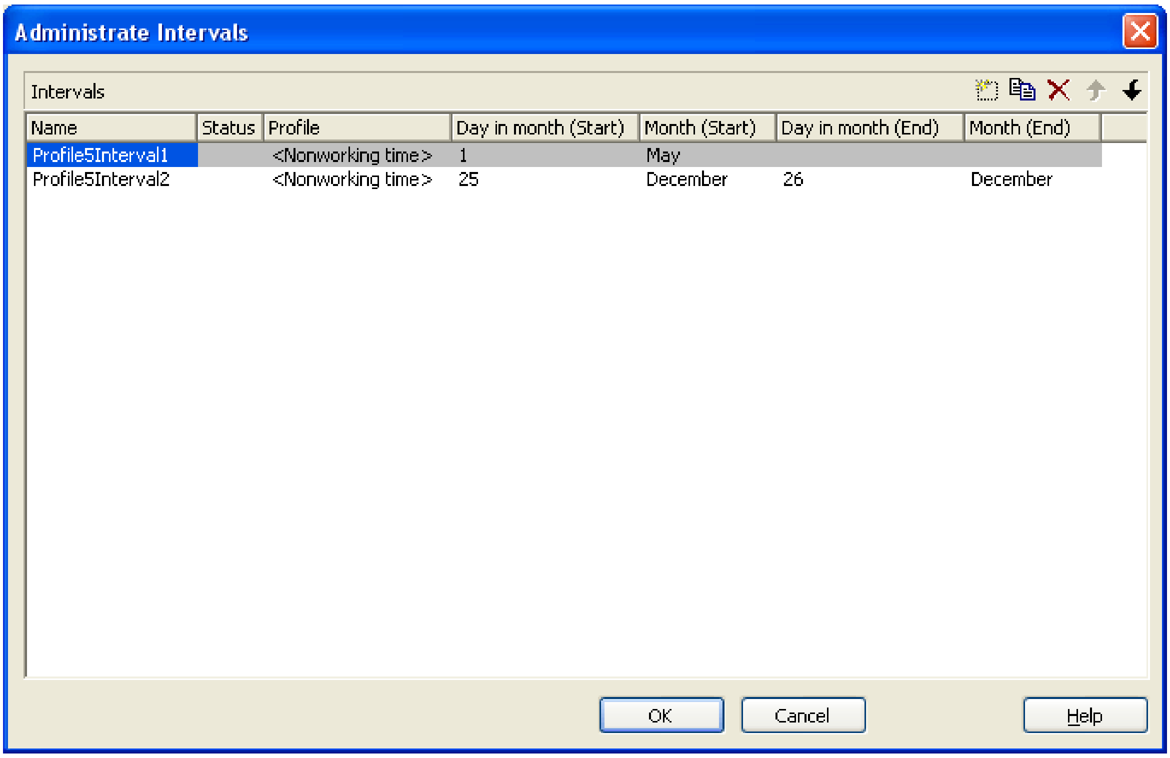
Time unit

Here you can specify the time unit of the interval. This feature can also be set by the property **VcInterval.TimeUnit**

Text

Here you can specify the text of the time ribbon This feature can also be set by the property **VcInterval.Text**

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



Name	Status	Profile	Day in month (Start)	Month (Start)	Day in month (End)	Month (End)
Profile5Interval1		<Nonworking time>	1	May		
Profile5Interval2		<Nonworking time>	25	December	26	December

You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a year profile.

Day in month (Start)/Day in month (End)

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

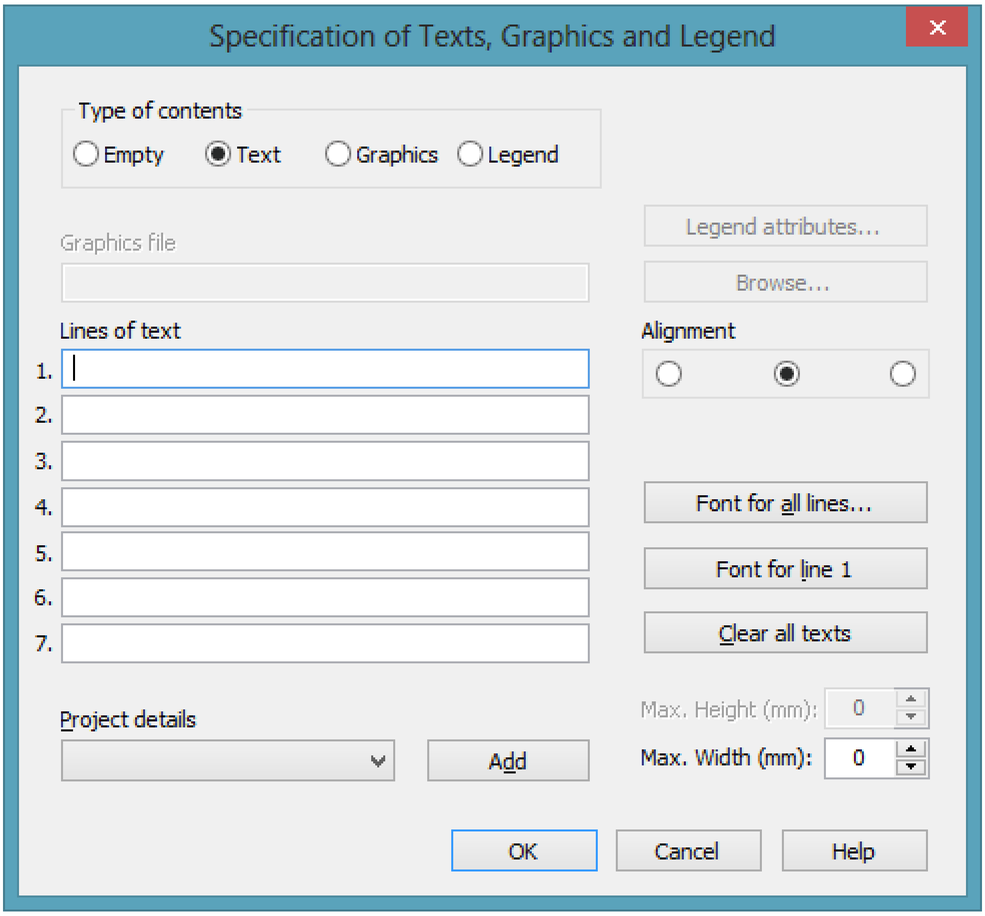
Month (Start)/Month (End)

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

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**

4.36 The "Specification of Texts, Graphics and Legend" Dialog Box



Specification of Texts, Graphics and Legend

Type of contents: ○ Empty ● Text ○ Graphics ○ Legend

Graphics file

Lines of text

1.

2.

3.

4.

5.

6.

7.

Alignment: ○ ● ○

Project details

Max. Height (mm): 0

Max. Width (mm): 0

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 by clicking on the **Add** button.

The place holders will be replaced by the required data and will continuously be kept up-to-date in the print preview and the printout.

Add

*Only activated 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.

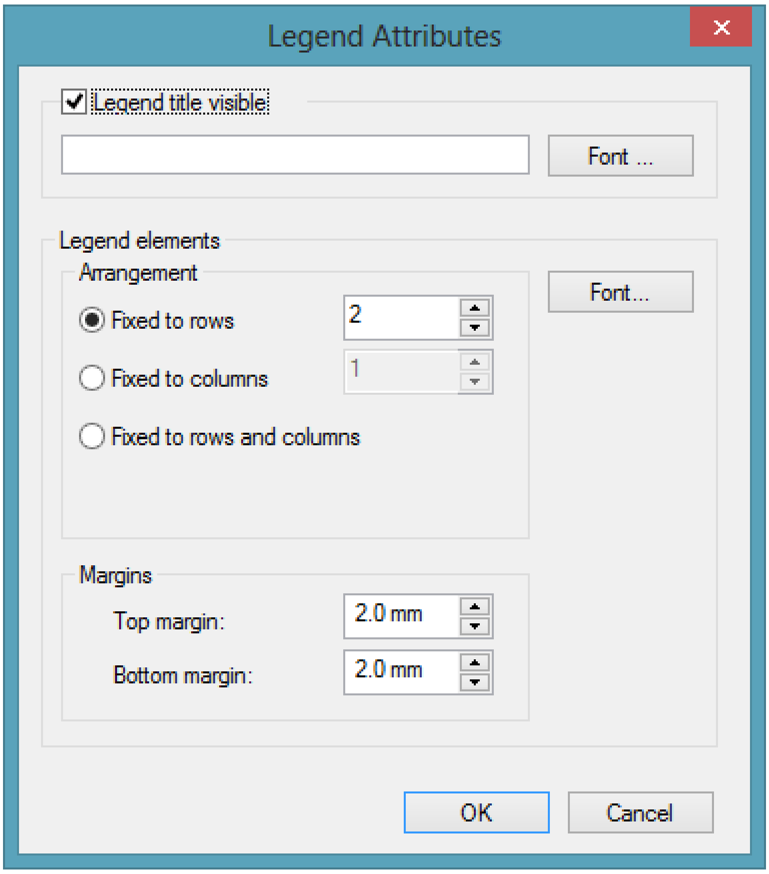
Max. Height (mm)

*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.

4.37 The "Legend Attributes Dialog Box"



☑ Legend title visible

● Fixed to rows 2

○ Fixed to columns 1

○ Fixed to rows and columns

Top margin: 2.0 mm

Bottom margin: 2.0 mm

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.

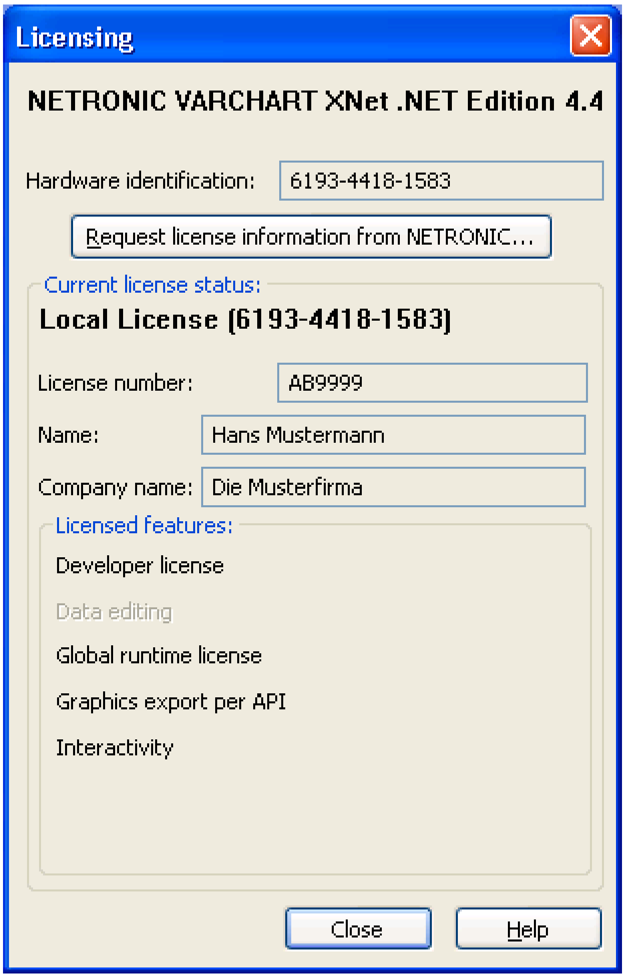
Margins

- 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.

4.38 The "Licensing" Dialog Box



Licensing

NETRONIC VARCHART XNet .NET Edition 4.4

Hardware identification: 6193-4418-1583

Current license status:

Local License (6193-4418-1583)

License number: AB9999

Name: Hans Mustermann

Company name: Die Musterfirma

Licensed features:

Developer license

Data editing

Global runtime license

Graphics export per API

Interactivity

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.

Current license status

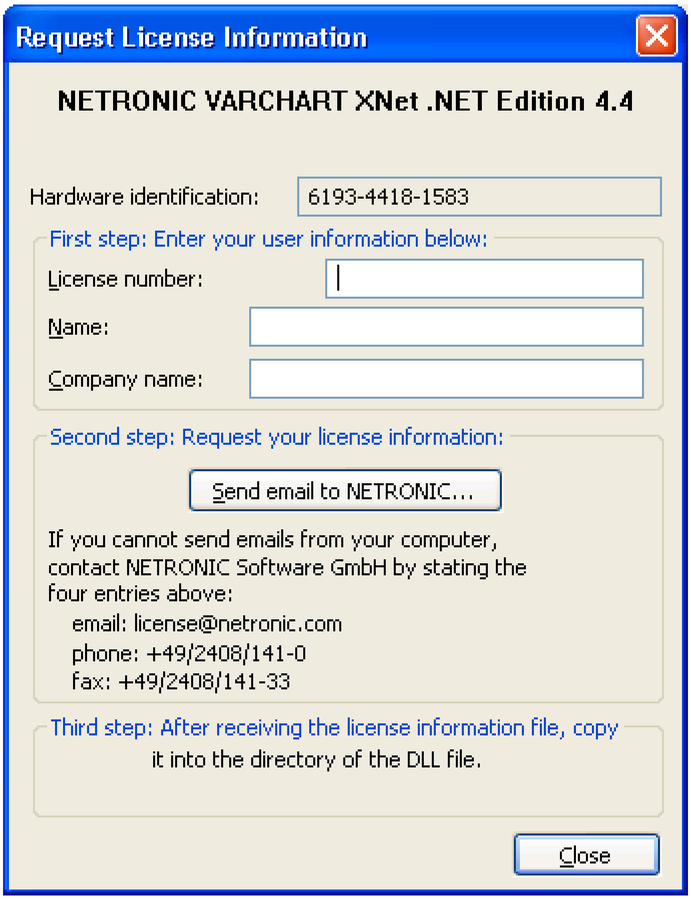
Indicates the modules that have been licenced. If the licencing procedure was successful, the licenced modules are activated.

- **Developer license**
- **Global runtime license** (VARCHART ActiveX runs in the runtime mode on each computer.)
- **Single-place runtime licenses** (The VARCHART ActiveX has to be licensed individually on each computer on which it shall run.)
- **Graphics export per API**
- **Interactivity**

Close

Quits the dialog box.

4.39 The "Request License Information" Dialog Box



Request License Information

NETRONIC VARCHART XNet .NET Edition 4.4

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.

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 (vcnet.lic) and mail it back to you.

After having received the file, please copy it to the directory in which the file vcnet.ocx is stored.

After licensing, you need to activate the new license in each of your projects. So please open a property page in each of your projects, make some change and store it. Then the new license will be activated.

5 User Interface

5.1 Overview

The below list gives an overview of possible user interactions.

- Navigating in the Diagram
- Zooming
- Generating nodes and links
- Marking, deleting or moving nodes and links
- Editing nodes and links
- Editing the legend
- Setting up pages
- Using the print preview

Context menus (right mouse key):

- Context menu for the diagram
- Context menu for nodes
- Context menu for links
- Context menu for the legend

All these interactions trigger an event so that you will be informed about it and will be able to react to it.

5.2 Navigation in the Diagram

You can use the arrow buttons to move the marking from one node to the other in the selected direction.

You can scroll in the diagram via the arrow buttons while the Ctrl key is pressed.

The following buttons can be used for navigation:

- **Ctrl** + **Pos1:** scrolling to the left upper diagram border
- **Ctrl** + **End:** scrolling to the right lower diagram corner
- **Ctrl** + **screen up/down:** scrolling to the upper/lower diagram corner
- **Ctrl** + **Num +**: zoom in
- **Ctrl** + **Num -**: zoom out
- **Ctrl** + **Num ***: scroll to the next node (scroll to node)
- **Ctrl** + **Num /**: complete view

Via **Ctrl + C**, **Ctrl + X** or **Ctrl + V** respectively you can copy, cut or insert marked nodes. Via the **Del** button you can delete marked nodes.

5.3 Zooming

The following shortcuts can be used for zooming:

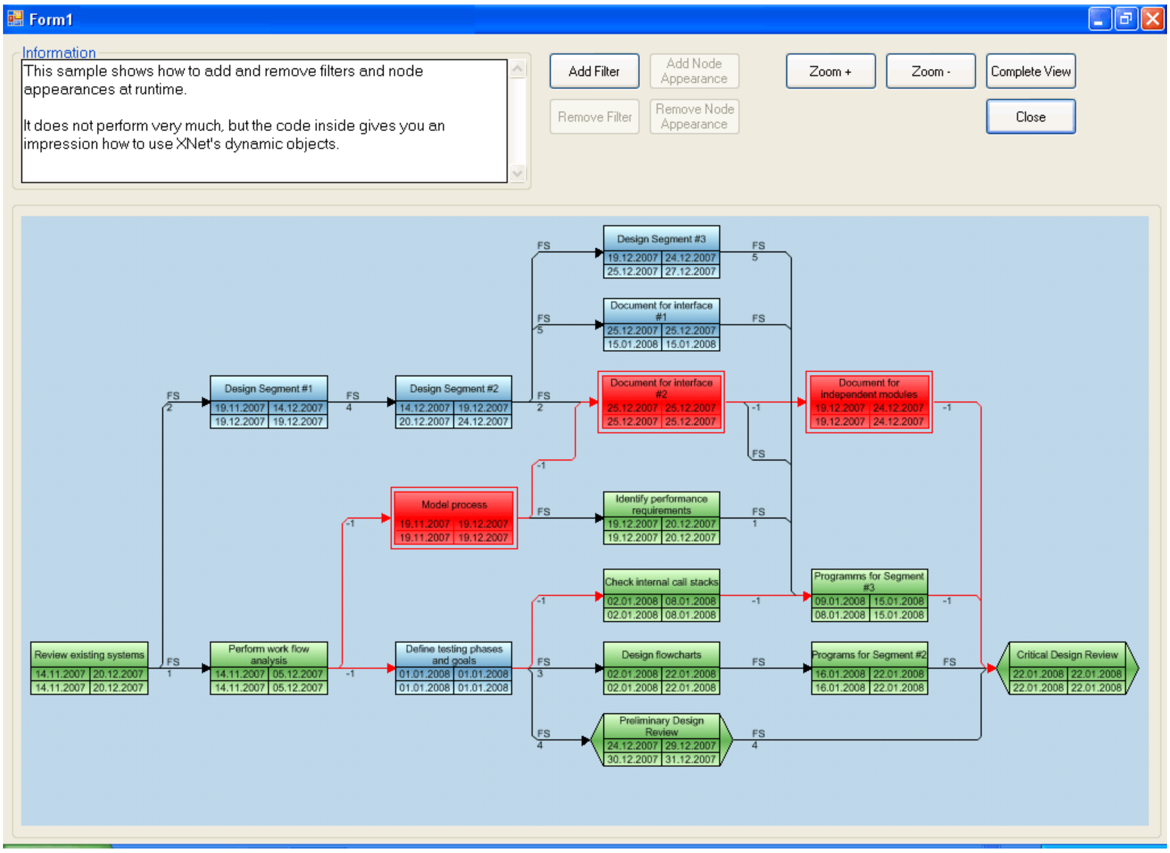
- **Ctrl** + **Num -**: zoom out
- **Ctrl** + **Num +**: zoom in

You can also use the mouse for zooming:

- Turn the mouse wheel while holding down the Ctrl key. For that purpose the usage of the mouse wheel for zooming has to be permitted. This can be done by ticking the **AllowZoomingByMouseWheel** box on the **General** property page or by setting the property **VcNet1.ZoomingPerMouseWheelAllowed** to **True**. This property is set to **False** by default.
- You can mark a section of your diagram and display it full screen. Use the left mouse key to draw a frame around the section to be zoomed, hold the left mouse key down and press the right mouse key. Use the scrollbars to shift the section and to view other parts of the diagram that are magnified to the same scale.

The API method **ShowAlwaysCompleteView** lets you display your diagram always completely. In this mode, the zoom factor will adapt automatically to any value smaller than 100%. The maximum zoom factor will never exceed 100%, so nodes will never appear larger than their original size.

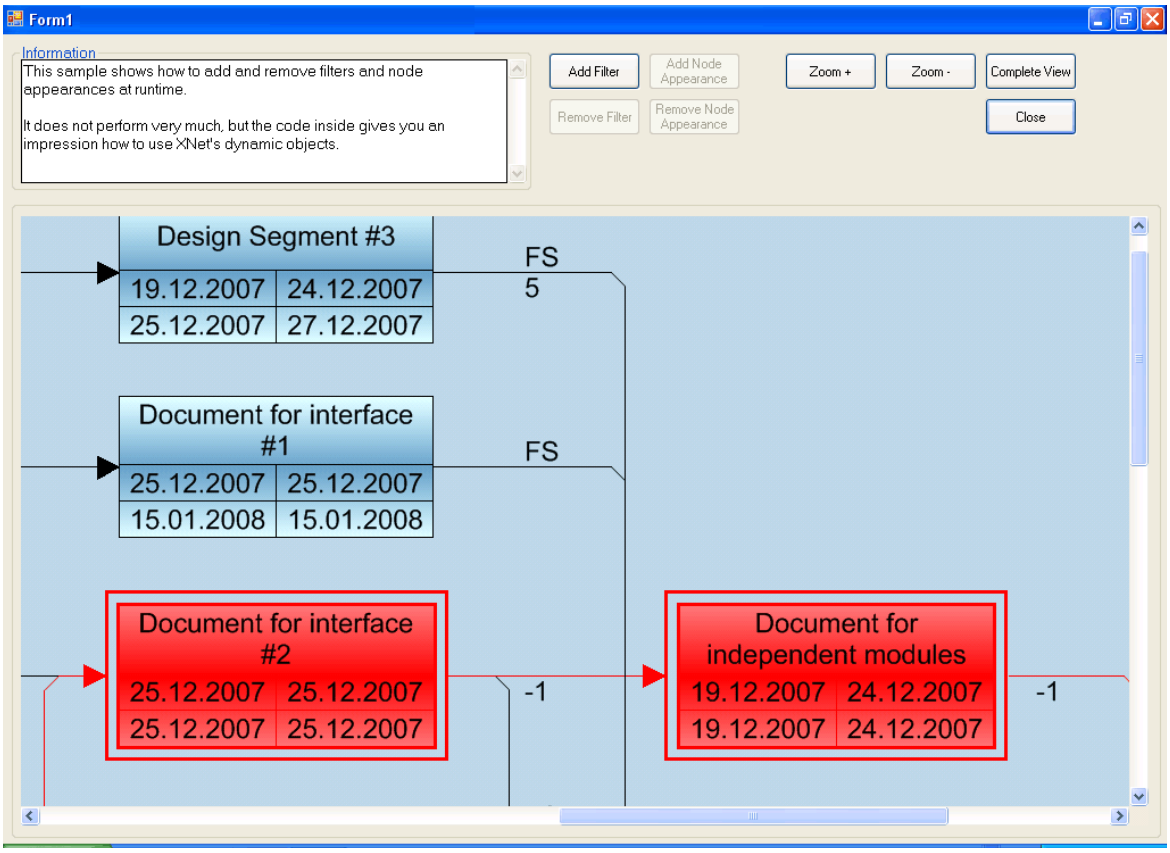
For further information about zoom settings for the print output please see chapter 5.21 "Setting up pages".



This sample shows how to add and remove filters and node appearances at runtime.

It does not perform very much, but the code inside gives you an impression how to use XNet's dynamic objects.

Before zooming



This sample shows how to add and remove filters and node appearances at runtime.

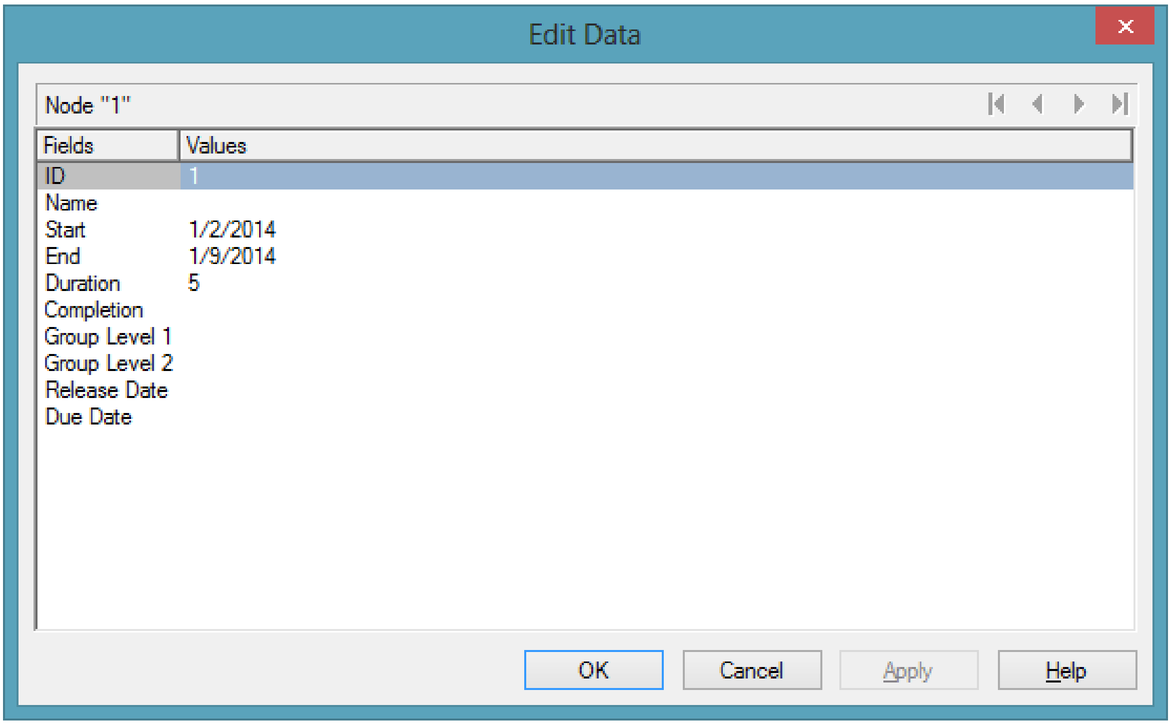
It does not perform very much, but the code inside gives you an impression how to use XNet's dynamic objects.

After zooming

5.4 Edit Node Data

In the dialog "Edit data" you can edit all node data. You open this dialog by either clicking on the **Edit** item of the corresponding context menu or by double-clicking on the node.

To edit several nodes, you mark the desired nodes and then click the **Edit** item of the context menu of one of the marked nodes to pop up the **Edit Data** dialog. Now you can edit the data of the marked nodes one after another



Node "1"

Fields	Values
ID	1
Name	
Start	1/2/2014
End	1/9/2014
Duration	5
Completion	
Group Level 1	
Group Level 2	
Release Date	
Due Date	

By double-clicking on a node, the event **VcNodeLeftDoubleClicking** is triggered.

Modifiying a node interactively, e.g. by the **Edit Data** dialog, triggers the event **VcNodeModifying**. By the **modificationType** parameter you get further information of the kind of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

The "Edit data dialog"

The name of the node as well as the number of the current node out of the total number of nodes marked is indicated.

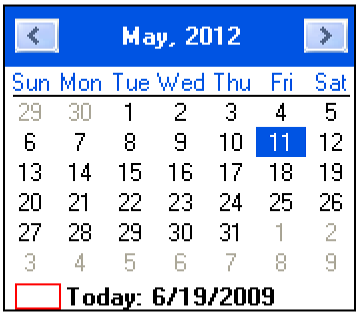
The table displays the data and values of the current node and lets you edit them. With the help of the arrow buttons above the table, you can navigate between the nodes. To store the current node data, click the **Apply** button.

Fields

This column displays the data fields that define the marked node. The data fields available are the ones defined by the data definition in the **Administrate data tables** dialog. Only data fields that are **not** defined as **hidden** are displayed.

Values

This column lets you edit the values of the nodes marked, but only if they were defined to be **Editable> in the Administrate Data Tables** dialog. If you edit a data field of the **Date/Time** type, a **Date** dialog will appear that you can select a date from.



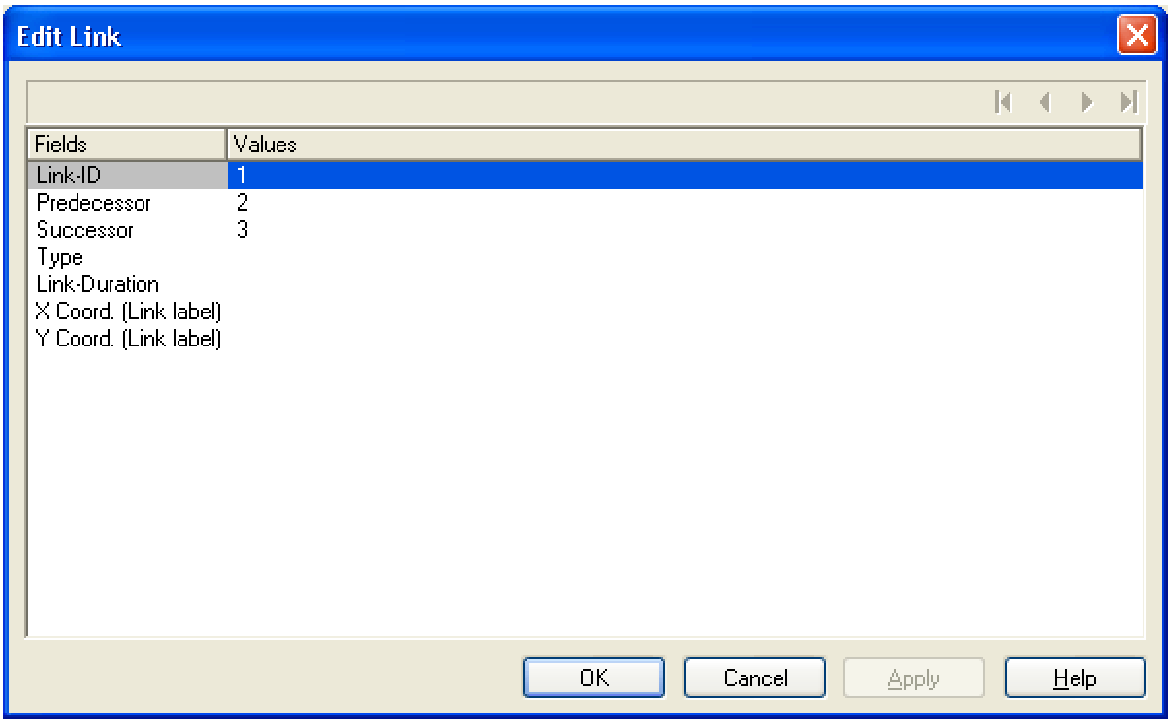
May, 2012

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	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	31	1	2
3	4	5	6	7	8	9

Today: 6/19/2009

The **Date Output Format** is defined on the **General** property page. When editing a field of the type **Integer** you can modify the value by a spin control that offers the desired values by up and down arrows.

5.5 Edit Links



Fields	Values
Link-ID	1
Predecessor	2
Successor	3
Type	
Link-Duration	
X Coord. (Link label)	
Y Coord. (Link label)	

This dialog you can reach via a double-click on a marked link (event **VcLinksLeftDoubleClicking**). It lets you edit the data of the link marked. The ID of the link is indicated. The table displays the data and values of the current link and lets you edit them.

Data fields

This column displays the data fields that define the marked link. The data fields available are the ones defined by the data definition. Only data fields that are not defined as **hidden** are displayed.

Values

This column lets you edit the values of the objects marked, if they haven't been defined to be **Read only** on the **DataDefinition** property page.

5.6 Navigation via Keyboard

You can use the arrow buttons to move the marking from one node to the other in the selected direction.

You can scroll in the diagram via the arrow buttons while the Ctrl key is pressed.

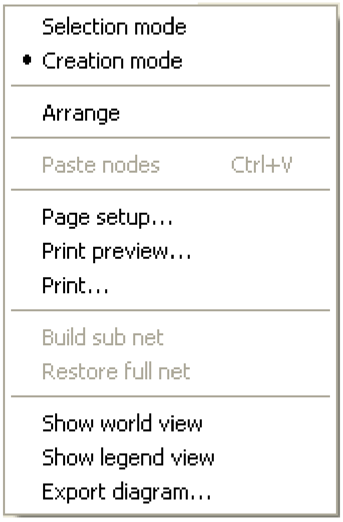
The following buttons can be used for navigation:

- **Ctrl** + **Pos1:** scrolling to the left upper diagram border
- **Ctrl** + **End:** scrolling to the right lower diagram corner
- **Ctrl** + **screen up/down:** scrolling to the upper/lower diagram corner
- **Ctrl** + **Num +**: zoom in
- **Ctrl** + **Num -**: zoom out
- **Ctrl** + **Num ***: scroll to the next node (scroll to node)
- **Ctrl** + **Num /**: complete view

Via **Ctrl + C**, **Ctrl + X** or **Ctrl + V** respectively you can copy, cut or insert marked nodes. Via the **Del** button you can delete marked nodes.

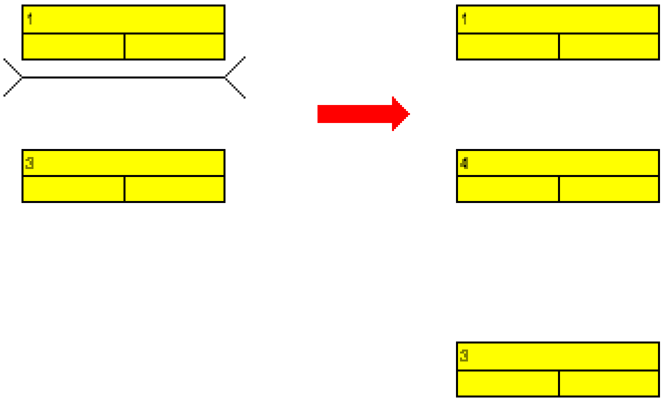
5.7 Creating Nodes and Links

There are two modes that you can toggle between in VARCHART XNet: The **Selection mode** and the **Creation mode**. Nodes and links can be generated in Creation mode only. To change modes, press the right mouse key on an empty area in the diagram and select the appropriate menu item from the context menu popping up.

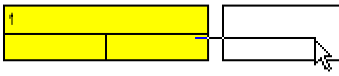


In Creation mode the cursor will transform into a rectangular frame. You can create a node by clicking on the left mouse key in an empty area of the diagram.

If you place the mouse between two nodes that are close together, the cursor will adopt a bone shape, i.e. a line with an inverted arrow tip at each of its ends. If you click by the left mouse key while the bone cursor is showing, the two nodes will shift apart and a new node will be inserted in the gap.



Links are generated by dragging the mouse from a node to a different one while keeping the left mouse key depressed. During the dragging operation, the cursor will transform into an arrow that draws a line.



As soon as you release the mouse key, the link will occur. If you drag the mouse between a node and an empty place, both a node and a link will be generated.

5.8 Marking, Deleting or Moving Nodes and Links

You can mark a node or a link by clicking on it via the left mouse key. Several nodes you can mark by pressing the Shift or Ctrl key. When pressing the Shift key, the links will be marked in addition. You can then for example delete all marked nodes via the Del key or by clicking on the **Delete** item of the context menu.

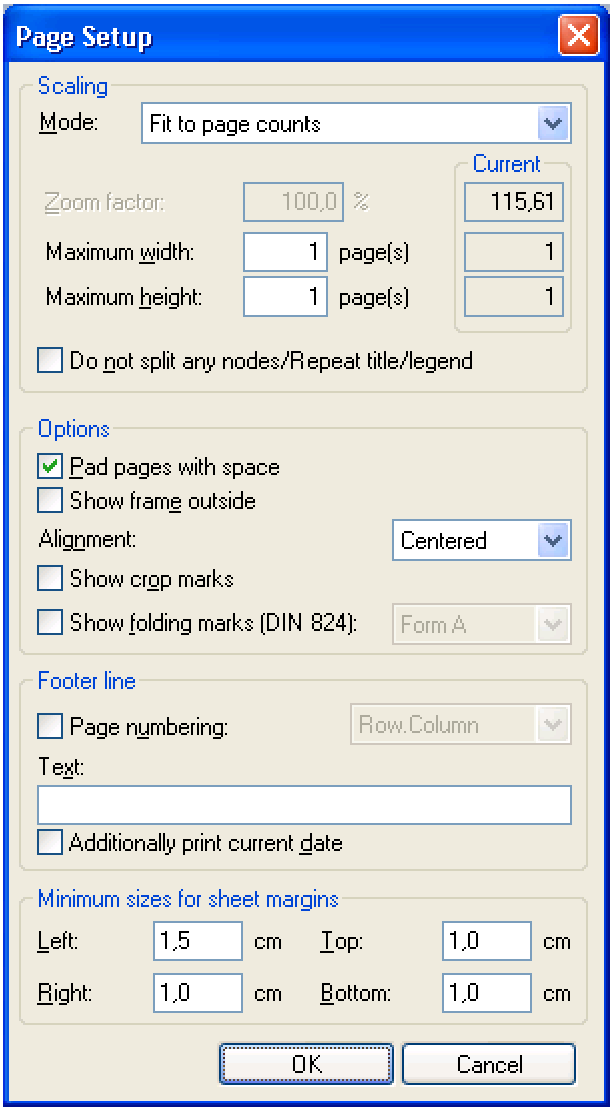
Beside, you can mark several nodes by dragging a framing rectangle around the nodes via the left mouse key.

If in selection mode you place the cursor on a node and press the left mouse key, you can move the node as long as you keep the mouse key depressed. The links joining will follow automatically.

If in selection mode you place the cursor on a link and press the left mouse key, the cursor will turn into a small rectangle and four arrows. You can move the link selected as long as you keep the mouse key depressed.

5.9 Setting up Pages

All settings concerning the page layout can be done in the corresponding dialog which can be opened either by selecting the **Page setup** item of the diagram contextmenu or by clicking the corresponding button in the **Print preview**.



Page Setup

Scaling

Mode: Fit to page counts

Current

Zoom factor: 100,0 % — 115,61

Maximum width: 1 page(s) — 1

Maximum height: 1 page(s) — 1

□ Do not split any nodes/Repeat title/legend

Options

☑ Pad pages with space

□ Show frame outside

Alignment: Centered

□ Show crop marks

□ Show folding marks (DIN 824): Form A

Footer line

□ Page numbering: Row.Column

Text:

□ Additionally print current date

Minimum sizes for sheet margins

Left: 1,5 cm Top: 1,0 cm

Right: 1,0 cm Bottom: 1,0 cm

Mode

By selecting a scaling mode from the drop down list and setting the corresponding values **Zoom factor** and **Maximum width/height** you specify a zoom factor for your output. After having clicked the **Apply** button, the values which result from your settings are shown under **Current**.

Zoom factor

100% is equivalent to the original size; a smaller value correspondingly reduces the size of the diagram, a greater value increases it.

Fit to page counts

By selecting this option you can specify the maximum number of pages, both heightwise and widthwise, into which the diagram may be split for the output. If necessary, one of the two values may be ignored in order to print the diagram as large as possible while preventing it from being distorted.

Do not split any nodes/Repeat title/legend

By ticking this check box nodes of a diagram that was partitioned into pages will not be split. If a title and legend exist, they will be added to each page.

Pad pages with space

This option lets you specify whether enough space is to be left between the diagram and the boxes of the title and legend area so that the boxes are always printed in full width and are fixed to the margin. If the option is not selected, there will be no space left between the diagram and the boxes and their width may vary on the different pages depending on the diagram.

Frame outside

Only activated if the ***Do not split any nodes/Repeat title*** *check box was ticked*. If you tick this box, each page will be given a frame, otherwise a frame will be drawn around the whole diagram.

Alignment

Select one of the possible alignments for the diagram from the list.

Show crop marks

If you tick this check box, crop marks will be printed on the edges of the diagram that help gluing together the single pages to get a complete chart.

Show folding marks (DIN 824)

Specify folding marks to fold your drawing according to DIN standard 824 (current version from 1981) for the folding of constructional drawings. The following formats are available:

- **Form A:** includes a filing margin on the left side so that the drawing can be punched and filed away
- **Form B:** slightly smaller so that a flexi filing fastener can be applied and together with the fastener the drawing corresponds to the width of DIN A4.
- **Form C:** the folded drawing is not to be punched but to be put in a sheet protector

The available folding marks can be displayed for every format, whereas the DIN 824 only mentions the formats DIN A0 to A3 explicitly.

Page numbers

If you tick this check box, a page number will be displayed in the bottom left-hand corner of each page. The following possibilities are available:

- **Row.Column**: Useful for charts stretching across more than one pages both heigthwise and widthwise. The vertical position of the page is displayed before the dot, the horizontal position after it.
- **Column.Row**: Useful for charts stretching across more than one pages both heigthwise and widthwise. The horizontal position of the page is displayed before the dot, the vertical position after it.
- **Page/Count**: The current page number is displayed before the slash and after it the total number of pages: 1/6, 2/6 etc.

Text

Please tick this check box to set a text into the bottom left-hand corner of each page. If there is a page number, the additional text will be placed right of it.

For numbering the pages you may enter in **Additional text** the following place holders which will be replaced with the appropriate contents on the printout:

{PAGE}	= consecutive numbering of pages
{NUMPAGES}	= total number of pages
{ROW}	= line position of the section in the complete chart

{COLUMN} = column position of the section in the complete chart

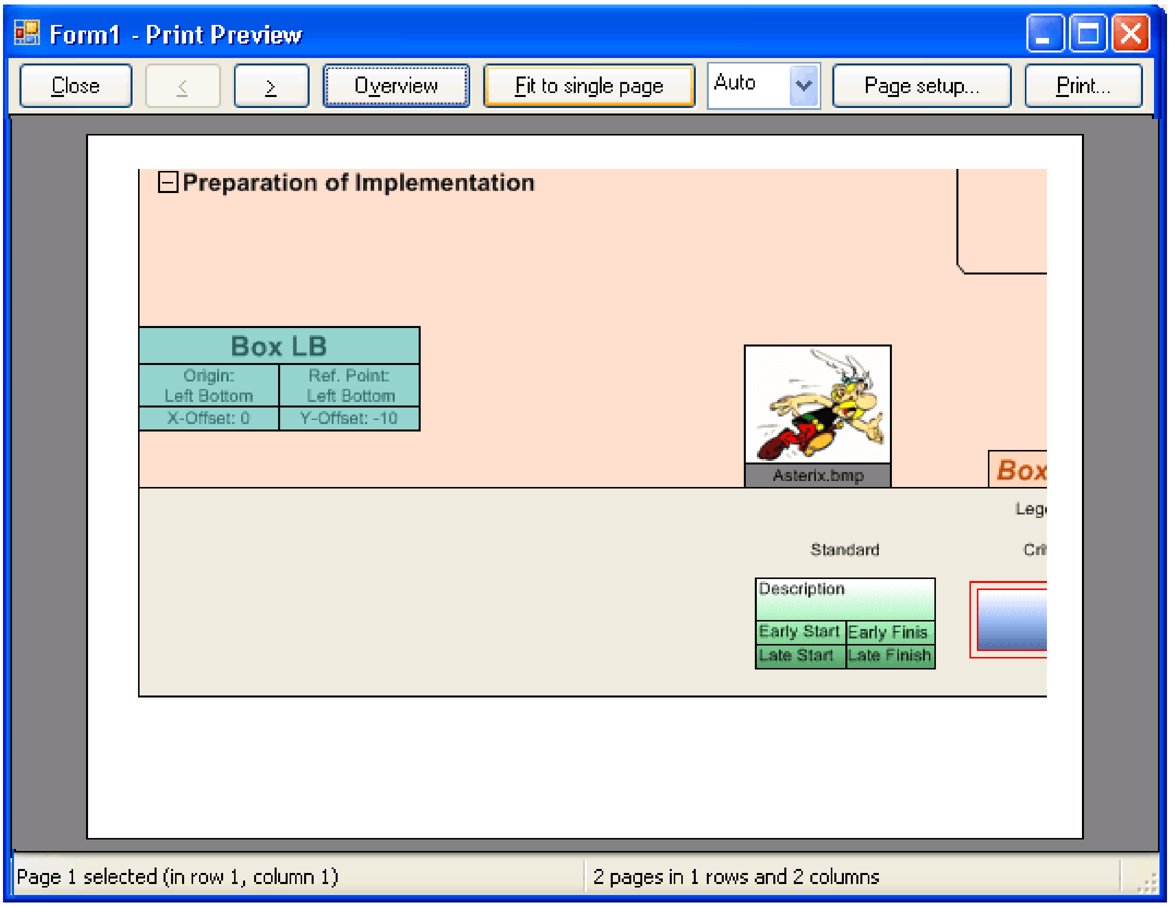
Additionally print current date

If you tick this check box, the date of printing will be printed in the bottom left corner. If there is a page number or an additional text, the print date will be placed right of them.

Sheet margins

The fields **Top, Botttom, Left** and **Right** let you set the margin between the diagram and the edge of the paper sheet (unit: cm). Minimum margins existing for technical reasons cannot be overridden by the values entered here. Printers that by default print minimum margins will add the values entered here to the default minimum margins, thus resulting in broader margins than visible in these settings.

5.10 Print Preview



Page 1 selected (in row 1, column 1)

2 pages in 1 rows and 2 columns

Before printing, you can view the diagram in the print preview where it will be displayed as defined by the settings of the **Page Setup** dialog and as it will be printed.

You can view single pages or an overview of all pages or you can zoom and print a certain section of your diagram interactively.

The status bar shows the total number of pages and their horizontal and vertical spreading. In the **Single Page** mode, also the number of the current page is shown.

Close

By clicking on this button, you will leave the page preview and return to your diagram.

<

Only activated when the ***Single*** *button has been pressed*. If the diagram consists of more than one page, you can click this button to view the previous page. You traverse the pages horizontally starting from the bottom right and finishing at the top left page.

>

*Only activated when the **Single** button has been pressed*. If the diagram consists of more than one page, you can press this button to view the next page. You traverse the pages horizontally starting from the top left and finishing at the bottom right page.

Show Single Page/Overview

If the diagram consists of more than one page you can either view the pages one by one or in the overview. The overview shows all pages, their size depending on the total number of pages. The **Single Page** mode inititally shows the first page in full size, the buttons > and < allowing to browse through the pages. By double-clicking a page you can easily switch between the two modes **Single Page** and **Overview**.

If you want to zoom a certain section of your diagram, switch to the **Single Page** mode and with the mouse draw a rectangle around the desired section while holding down the left mouse button. As soon as you release the button, the selected section will be enlarged and can be printed by clicking the Print area... button that appears in place of the **Print** button. Please note that the zooming factor will not influence the scaling factor set in the **Page Setup** dialog.

Fit To Single Page

This button lets you scale down a multiple-page diagram to one page. The **Fit To Single Page** mode also allows to zoom a certain section as described under **Show Single Page/Overview**

Zoom factor

You can modify the size of the diagram by selecting a zoom factor from the list or by defining an individual one. This is only possible in the "Show Single Page" mode. To modify the zoom factor you can also use the scroll-wheel while holding down the <CTRL> key. The zoom factor it will not modify the size of the output. Depending on the selected zoom factor, vertical and/or horizontal scroll bars will be displayed. Alternatively, you can use the mouse wheel to scroll vertically, holding down <Shift> to scroll horizontally.

The zoom factor **Auto** is the pre-set default and will always enlarge or downsize the sheet to the full size of the screen.

Page Setup

When clicking on this button, you will get to the dialog **Page Setup** to modify page settings.

Printer Setup

*Only visible if the check box **Use PrintDlgEx dialog** on the **General** property page has not been ticked*.

When clicking on this button, you will get to the Windows dialog **Printer Setup**, where you can modify printer settings.

Print/Print Area

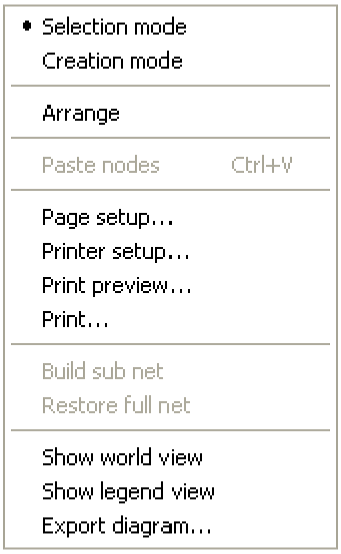
Click on this button to reach the Windows **Print** dialog box to start the print procedure.

If you have zoomed a section in the page preview, the button's label will change to **Print Area** and when you click it, the **Selection** radio button in the Windows **Print** dialog box will already be selected. If you click on **OK** the section displayed on the screen will be printed.

Please note that the zooming factor will not influence the scaling factor set in the **Page Setup** dialog.

5.11 The Context Menu of the Diagram

A right mouse click in an empty area of the diagram will open the below context menu:



Selection Mode

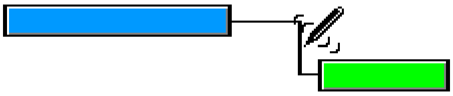
The selection mode is the default mode.

Creation Mode

This mode can be switched on only after on the **General** property page the option **Nodes and link creation allowed** has been ticked.

The pointer will turn into a node phantom of rectangular shape. In this mode, a click on the mouse will generate a new node. If on the **General** property page the **Node creation with dialog** box was activated, the **Edit Data** dialog box will open automatically as soon as you release the mouse botton. It lets you edit all data of the node.

If in the creation mode you drag the mouse from one node to another, a link will be created between them. If on the **General** property page the **Link creation with dialog** box was activated, the **Edit Data** dialog box will open automatically as soon as you release the mouse botton. You can edit all data of the link.



The creation mode can be activated by two different ways:

1. by the default context menu popping up on a double-click of the right mouse button in an empty spot of the diagram area
2. by setting the VcNet property **InteractionMode** to **vcCreateNodesAndLinks**.

Arrange

This menu item will arrange nodes and links moved by the user to result in an optimum layout.

Paste nodes

This menu item is available only after cutting or copying nodes. It lets you paste nodes at the position of the pointer.

Page Setup

The dialog **Page Setup** appears.

Printer Setup

*Only selectable if the check box **Use PrintDlgEx dialog** on the **General** property page has not been ticked*.

This menu item gets you to the Windows dialog **Printer Setup**.

Print Preview

The dialog box **Page Preview** appears.

Print

The Windows dialog **Print** appears.

Build sub net

(only active if nodes are marked) Select this item to display a subnet of the marked nodes.

Restore Full Net

*(only active if the option **Build Subnet** has been selected before)* Select this item to restore the full net.

Show world view

This menu item lets you switch on/off the world view. The world view is an additional window that shows the complete diagram. A frame marks the diagram section currently displayed in the main window. If you move this frame with the mouse, the according diagram section is displayed in the main window.

The world view also can be displayed oder hidden by the property **VcWorldView.Visible**.

Show legend view

This menu item lets you switch on or off the legend view. The legend will appear in a separate window.

The legend view also can be displayed oder hidden by the property **VcLegendView.Visible**.

Export Diagram

When you select this menu item, you will get to the Windows dialog box **Save as**, that lets you save the diagram as a graphics file.

This dialog box also can be invoked by the VcNet method **ShowExport-GraphicsDialog**.

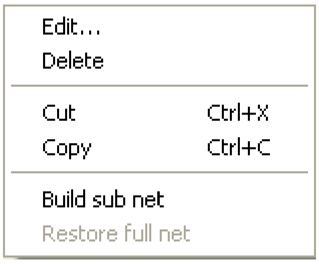
When exporting, the size of the exported diagram will be calculated this way:

- PNG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input.
- GIF, TIFF, BMP, JPEG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. In addition, an internal limit of 50 MBs of memory size is required for the uncompressed source bit map in the memory; so larger diagrams may have a smaller resolution than expected.
- WMF: A fixed resolution is assumed where the longer side uses co-ordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.
- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm.

For further details on the different formats please read the chapter "Important Concepts: Graphics Formats".

5.12 The Context Menu of Nodes

A right mouse click on one or several marked nodes will open the below menu:



Edit

Opens the **Edit Data** dialog box.

Delete

The marked nodes will be deleted.

Cut nodes

The marked nodes are cut from the diagram.

Copy nodes

The marked nodes are copied.

Build sub net

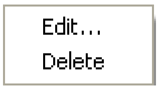
A subnet of the marked nodes will be displayed.

Restore full net

*(only active if the option **Build sub net** has been selected before)* The full net will be restored.

5.13 The Context Menu of Links

A right mouse click on a link will open the below menu:



Edit

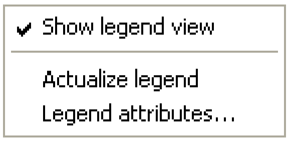
This menu item will pop up the dialog **Edit Link** where you can edit the data of the selected link.

Delete

To delete the marked link click on the **Delete** menu item.

5.14 Context Menu of the Legend

A right mouse click on the legend will open the below menu:



Show legend view

This menu item lets you switch on or off the legend view.

Actualize legend

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

Legend attributes

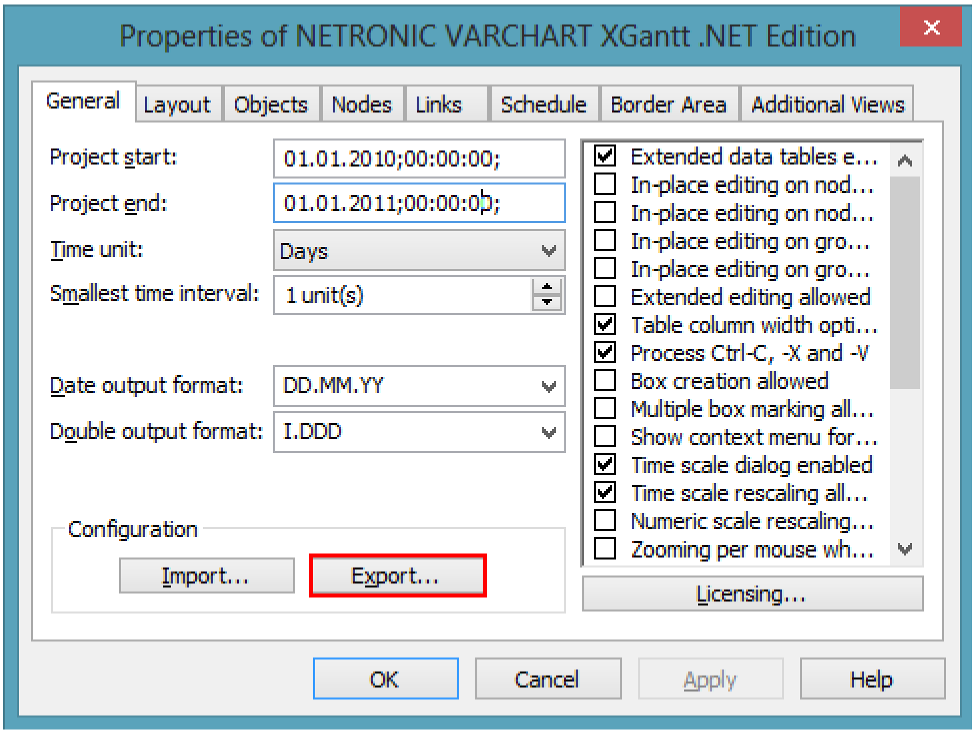
With this item you open the corresponding dialog where you can specify the settings concerning legend title, legend elements and margins. For further information about this dialog please see chapter 4.44 "The Legend Attributes Dialog Box".

6 Frequently Asked Questions

6.1 How to to Upgrade from VARCHART XGantt .NET 4.4 to VARCHART XGantt .NET 5.0?

6.2 How to Upgrade from one Build of VARCHART XGantt .NET to a new one (within the same version)?

1. Before installing VARCHART XGantt.NET 5.0, please open the form designer of Visual Studio with the form using XGantt 4.4 and save the current configuration of XGantt by clicking the **Export** button on the **General** property page:



Project start: 01.01.2010;00:00:00;

Project end: 01.01.2011;00:00:00;

Time unit: Days

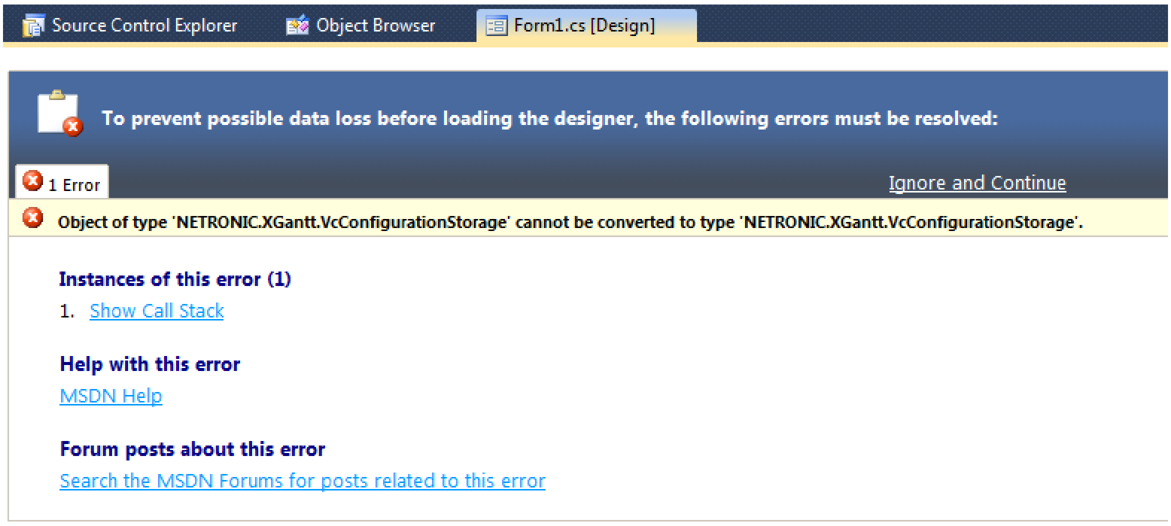
Smallest time interval: 1 unit(s)

Date output format: DD.MM.YY

Double output format: I.DDD

☑ Extended data tables e...
☐ In-place editing on nod...
☐ In-place editing on nod...
☐ In-place editing on gro...
☐ In-place editing on gro...
☐ Extended editing allowed
☑ Table column width opti...
☑ Process Ctrl-C, -X and -V
☐ Box creation allowed
☐ Multiple box marking all...
☐ Show context menu for...
☑ Time scale dialog enabled
☑ Time scale rescaling all...
☐ Numeric scale rescaling...
☐ Zooming per mouse wh...

2. First, close the form and then end Visual Studio.
3. Install the new build of VARCHART XGantt .NET in the same folder as the old build.
4. Open the form designer with the form containing XGantt. The following error message will appear:



To prevent possible data loss before loading the designer, the following errors must be resolved:

1 Error

Ignore and Continue

Object of type 'NETRONIC.XGantt.VcConfigurationStorage' cannot be converted to type 'NETRONIC.XGantt.VcConfigurationStorage'.

Instances of this error (1)

1. Show Call Stack

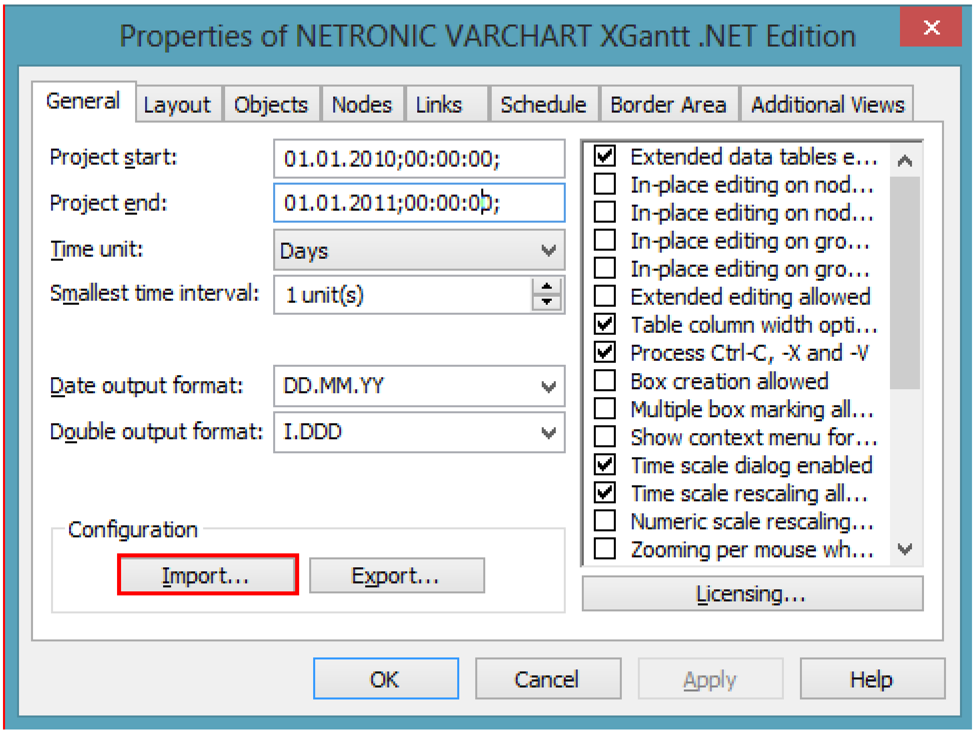
Help with this error

MSDN Help

Forum posts about this error

Search the MSDN Forums for posts related to this error

5. Click **Ignore and Continue**.The form in the form designer will be displayed correctly again but the XGantt will be set back to ist default configuration.
6. Now import the configuration you have saved before by clicking the **Import** button on the **General** property page.



Properties of NETRONIC VARCHART XGantt .NET Edition

Project start: 01.01.2010;00:00:00;

Project end: 01.01.2011;00:00:00;

Time unit: Days

Smallest time interval: 1 unit(s)

Date output format: DD.MM.YY

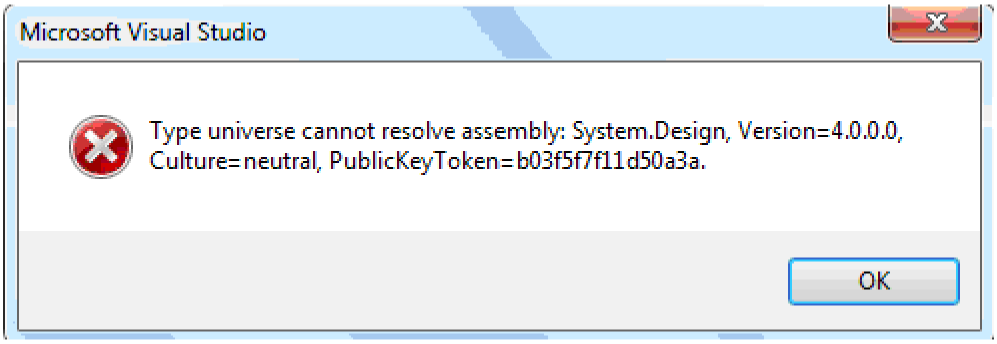
Double output format: I.DDD

☑ Extended data tables e…
☐ In-place editing on nod…
☐ In-place editing on nod…
☐ In-place editing on gro…
☐ In-place editing on gro…
☐ Extended editing allowed
☑ Table column width opti…
☑ Process Ctrl-C, -X and -V
☐ Box creation allowed
☐ Multiple box marking all…
☐ Show context menu for…
☑ Time scale dialog enabled
☑ Time scale rescaling all…
☐ Numeric scale rescaling…
☐ Zooming per mouse wh…

7.

VARCHART XGantt now uses your individual configuration again.

6.3 Why does an error message occur, when I create a new project in Visual Studio 2010 and try to drag the control onto the form?


Microsoft Visual Studio

Type universe cannot resolve assembly: System.Design, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b03f5f7f11d50a3a.

This error message occurs because in Visual Studio 2010 the **.NET Framework 4 Client Profile** is set as default but the NETRONIC VARCHART requires the target framework **.NET Framework 4** since the former lacks the System.Design.dll, which is required by the property pages at design-time. Hence you have to change the target framework from **.NET Framework Client Profile** to **.NET Framework 4** in the **Application Settings** (C#) or **Advanced Compiler Settings** (VB) **before** you drag the control onto the form.

6.4 How can I Activate the License File?

1. Please close your programming environment.
2. Copy the license file NETRONIC.XNet.VcNet.lic to the installation directory of VARCHART XNet.NET.
3. Please re-start your programming environment and re-build your project again.

6.5 Why can I not Create Nodes Interactively at Times?

If during runtime you cannot create nodes via the mouse, please activate the check box **Node and link creation allowed** on the **General** property page.

Check if the VARCHART VcNet property **NodeAndLinkCreationAllowed** has not been set to **False**.

6.6 Why can I not Create Links Interactively at Times?

If during runtime you cannot create links interactively, the causes may be of different kind:

1. Please verify if on the property page **General** the check box **Node and link creation allowed** was activated. After ticking it, you should be able to create links interactively.
2. If you still cannot recognize any links on the screen, take a look at the settings of the links. The links may be invisible. Please open the **Links** property page and verify the line type of each link appearance. If the line color is identical with the background color of the chart, select a different line color.
3. Please verify the criteria set in the filter. Filter criteria defined the wrong way may lead to invisible links.
4. If the definition of the link appearance makes sense, and there are still no links in the chart, please verify whether the data fields (**Predecessor**, **Successor**, **Relation type**) have been defined properly.

6.7 How can I Disable the Interactive Creation of Nodes and Links?

There are several ways to revoke interactive creating of nodes and links:

1. You can deactivate the check box **Node and link creation allowed** on the **General** property page.
2. You can set the return status of the event **VcNodeCreating** to **vcRetStatFalse** to enable deleting of interactively generated nodes.
3. You can add the following code:

Example Code

```
Sub Form_Load
   VcNet1.AllowNewNodesAndLinks = False
End Sub
```

6.8 How can I Disable the Default Context Menus?

You can disable a predefined context menu to occur by setting the returnStatus to **vcRetStatNoPopup**.

Example Code VB.NET

```
'switching off the context menu of diagram
Private Sub VcNet1_VcDiagramRightClicking(ByVal x As Long, ByVal y As
Long, _
                              returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

'switching off the context menu of links
Private Sub VcNet1_VcLinksRightClicking(ByVal linkCltn As _
                              VcNetLib.VcLinkCollection, ByVal x As
Long, _
                              ByVal y As Long, returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

'switching off the context menu of nodes
Private Sub VcNet1_VcNodeRightClicking(ByVal node As VcNetLib.VcNode, _
                              ByVal location As _
                              VcNetLib.VcLocation, _
                              ByVal x As Long, _
                              ByVal y As Long, _
                              returnStatus As Variant)
   returnStatus = vcRetStatNoPopup
End Sub
```

6.9 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 im improve the overall performance considerably.

Example Code

```
VcNet1.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 node
   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 node
   End If

VcNet1.SuspendUpdate (False)
```

If you modify table formats in large projects, you also should use the **SuspendUpdate** method.

> Graphics

Another reason for a low performance may be graphics in table, node or box fields that are too large or that have to many pixels.

6.10 Error Messages

> Error messages at runtime caused by the developer

To be completed.

> Error messages at runtime caused by the end user or by the developer

Error Reason	Message
Cycles detected in the method **ScheduleProject**	Project has cycled links!

6.11 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.

6.12 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.

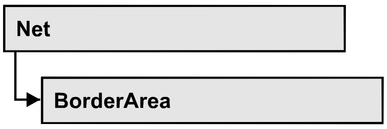
7 API Reference

7.1 Object Types

- VcBorderArea
- VcBorderBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcCalendar
- VcCalendarCollection
- VcCalendarProfile
- VcCalendarProfileCollection
- VcDataRecord
- VcDataRecordCollection
- VcDataTable
- VcDataTableCollection
- VcDataTableField
- VcDataTableFieldCollection
- VcFilter
- VcFilterCollection
- VcFilterSubCondition
- VcGroup
- VcGroupCollection
- VcInterval
- VcIntervalCollection
- VcLegendView
- VcLink
- VcLinkAppearance
- VcLinkAppearanceCollection
- VcLinkCollection
- VcLinkFormat
- VcLinkFormatCollection
- VcLinkFormatField
- VcMap

- VcMapCollection
- VcMapEntry
- VcNet
- VcNode
- VcNodeAppearance
- VcNodeAppearanceCollection
- VcNodeCollection
- VcNodeFormat
- VcNodeFormatCollection
- VcNodeFormatField
- VcPrinter
- VcRect
- VcScheduler
- VcWorldView

7.2 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
Parameter:		
boxPosition	VcBorderBoxPosition	Box position
	Possible Values:	
	.vcBBXPBottomBottomCentered 8	second line in the bottom area, centered
	.vcBBXPBottomBottomLeft 7	second line in the bottom area, left
	.vcBBXPBottomBottomRight 9	second line in the bottom area, right
	.vcBBXPBottomTopCentered 5	first line in the bottom area, centered
	.vcBBXPBottomTopLeft 4	first line in the bottom area, left
	.vcBBXPBottomTopRight 6	first line in the bottom area, right
	.vcBBXPLegend 51	legend
	.vcBBXPTopCentered 2	top centered
	.vcBBXPTopLeft 1	top left
	.vcBBXPTopRight 3	top right
Return value	VcBorderBox	Box of the title and legend area

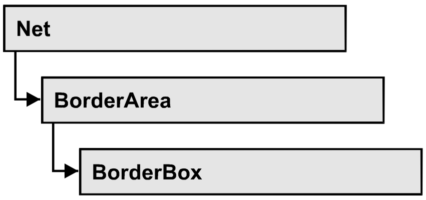
Example Code VB.NET

```
Dim boardArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

boardArea = VcNet1.BorderArea
bBoxBBL = boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

Example Code C#

```
VcBorderArea boardArea = vcNet1.BorderArea;

VcBorderBox bBoxBBL =
boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
bBoxBBL.LegendTitle = "Explanation";
```

7.3 VcBorderBox



An object of the type **VcBorderBox** designates one of the boxes in the title or legend area of the graphics.

Properties

- Alignment
- GraphicsFileName
- LegendElementsArrangement
- LegendElementsBottomMargin
- LegendElementsMaximumColumnCount
- LegendElementsMaximumRowCount
- LegendElementsTopMargin
- LegendFont
- LegendTitle
- LegendTitleFont
- LegendTitleVisible
- Text
- TextFont
- Type

Properties

Alignment

Property of VcBorderBox

This property lets you set or retrieve the alignment of this BorderBox object.

	Data Type	Explanation
Property value	VcBorderBoxAlignment	Alignment of the border box
	Possible Values: .vcBBXACentered -1 .vcBBXALeft -3	 Center Left

	.vcBBXARight -2	Right

GraphicsFileName

Property of VcBorderBox

This property lets you set or retrieve the name of the graphics file used in the VcBorderBox object. *Available formats:*

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

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	Data Type	Explanation
Property value	System.String	Name of the graphics file

Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomTopRight)
borderBox.Type = VcBorderBoxType.vcBBXTGraphics
borderBox.GraphicsFileName = "C:\Asterix.jpg"
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.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
Property value	VcLegendElementsArrangement	Type of arrangement of the legend elements
	Possible Values:	
	.vcLEAFixedToColumns 0	The legend elements are merely aligned along columns.
	.vcLEAFixedToRows 1	The legend elements are merely aligned along rows.
	.vcLEAFixedToRowsAndColumns 2	The legend elements are aligned along rows and columns.

LegendElementsBottomMargin

Property of VcBorderBox

This property lets you set or retrieve the width between the legend elements and the bottom of the border box (unit: mm).

	Data Type	Explanation
Property value	System.Int16	Width of bottom margin

LegendElementsMaximumColumnCount

Property of VcBorderBox

This property lets you set or retrieve the number of columns to which the elements in the legend should disperse.

	Data Type	Explanation
Property value	System.Int16	Number of columns

LegendElementsMaximumRowCount

Property of VcBorderBox

This property lets you set or retrieve the number of rows to which the elements in the legend should disperse.

	Data Type	Explanation
Property value	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	System.DrawingFont	Font attributes of the legend

Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendFont.Name)
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.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 = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitle = "Explanation"
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.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	System.DrawingFont	Font attributes of the legend title

Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendTitleFont.Name)
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.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 = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitleVisible = False
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.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
Parameter: rowIndex	System.Int16	Row index {0...6}
Property value	System.String	Text in text boxes

Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTText
borderBox.Text(index) = "Department A"
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.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
Parameter: rowIndex	System.Int16	Row index {0...6}
Property value	System.DrawingFont	Font attributes of the text

Example Code VB.NET

```
Dim borderArea As VcBorderArea
Dim bBoxTL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)

bBoxTL.TextFont(i).Bold = False
bBoxTL.TextFont(i).Italic = False
bBoxTL.TextFont(i).Name = "Symbol"
```

Example Code C#

```
// Text for Title
VcBorderBox borderBox =
VcNet1.BorderArea.BorderBox(VcBorderBoxPosition.vcBBXPTopCentered);
borderBox.Type = VcBorderBoxType.vcBBXTText;

Font titleFont1 = new Font("Arial", 20, FontStyle.Bold);

borderBox.set_Text(1, "Time Scheduler");
borderBox.set_TextFont(1, titleFont1);
```

Type

Property of VcBorderBox

This property lets you set or retrieve the type of the BorderBox object.

	Data Type	Explanation
Property value	VcBorderBoxType	Box type
	Possible Values:	
	.vcBBXTGraphics 3	graphics
	.vcBBXTLegend 4	legend
	.vcBBXTNothing 0	nothing
	.vcBBXTText 1	text
	.vcBBXTTextWithGraphics 2	text and graphics

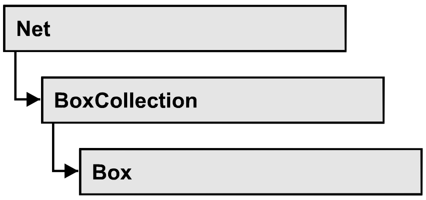
Example Code VB.NET

```
Dim bBoxBBL As VcBorderBox

bBoxBBL = boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomLeft)
bBoxBBL.Type = VcBorderBoxType.vcBBXTGraphics
```

Example Code C#

```
VcBorderArea boardArea = vcNet1.BorderArea;

VcBorderBox bBoxBBL =
boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
bBoxBBL.Type = VcBorderBoxType.vcBBXTGraphics;
```

7.4 VcBox



An object of the type **VcBox** designates a box to display texts or graphics.

Properties

- FieldText
- FormatName
- LineColor
- LineThickness
- LineType
- Marked
- Moveable
- Name
- Origin
- Priority
- ReferencePoint
- UpdateBehaviorName
- Visible

Methods

- GetActualExtent
- GetTopLeftPixel
- GetXYOffset
- IdentifyFormatField
- 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.

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
Parameter: ⇨ fieldIndex	System.Int16	Field index
Property value	System.String	Field content

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.FieldText(0) = "User: "
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Property value	VcBoxFormat	BoxFormat object or **Nothing**

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.FormatName = "Standard"
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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 = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.LineColor = System.Drawing.Color.Blue
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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.

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 **Default value:** As defined in the dialog

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.LineThickness = 2
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineThickness = 2;
```

LineType

Property of VcBox

This property lets you set or retrieve the type of the border line of the box.

	Data Type	Explanation
Property value	VcLineType	Line type **Default value:** vcSolid
	Possible Values: .vcDashed 4 .vcDashed 4 .vcDashedDotted 5	 Line dashed Line dashed Line dashed-dotted

.vcDashedDotted 5	Line dashed-dotted
.vcDotted 3	Line dotted
.vcDotted 3	Line dotted
.vcLineType0 100	Line Type 0
.vcLineType1 101	Line Type 1
.vcLineType10 110	Line Type 10
.vcLineType11 111	Line Type 11
.vcLineType12 112	Line Type 12
.vcLineType13 113	Line Type 13
.vcLineType14 114	Line Type 14
.vcLineType15 115	Line Type 15
.vcLineType16 116	Line Type 16
.vcLineType17 117	Line Type 17
.vcLineType18 118	Line Type 18
.vcLineType2 102	Line Type 2
.vcLineType3 103	Line Type 3
.vcLineType4 104	Line Type 4
.vcLineType5 105	Line Type 5
.vcLineType6 106	Line Type 6
.vcLineType7 107	Line Type 7
.vcLineType8 108	Line Type 8
.vcLineType9 109	Line Type 9
.vcNone 1	No line type assigned
.vcNone 1	No line type
.vcNotSet -1	No line type assigned
.vcSolid 2	Line solid
.vcSolid 2	Line solid

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.LineType = VcLineType.vcDotted
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Property value	System.Boolean	**True**: box marked; **false**: box unmarked

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Marked = True
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Marked = true;
```

Moveable

Property of VcBox

This property lets you set or retrieve whether the box can be moved interactively.

	Data Type	Explanation
Property value	System.Boolean	Movable (True)/ not Movable (False) **Default value:** True

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Moveable = False
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Moveable = false;
```

Name

Property of VcBox

This property lets you set or retrieve the name of a box. You can also specify the name in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	System.String	Box name

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim boxName As String

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Name)
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Property value	VcBoxOrigin	Origin of the box

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Origin = VcBoxOrigin.vcBOTopCenter
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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 = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Priority = 3
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Property value	VcBoxReferencePoint	Reference point of the box

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight;
```

UpdateBehaviorName

Property of VcBox

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	System.String	Name of the UpdateBehavior

Visible

Property of VcBox

This property lets you set or retrieve whether a box is visible. You also can specify this property in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	System.Boolean	Box visible/invisible **Default value:** True

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Visible = False
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Visible = false;
```

Methods

GetActualExtent

Method of VcBox

This method lets you retrieve the actual extent of the box (unit: 1/100 mm).

By regarding these values when setting the XY offset, you can modify the reference point of the anchoring line without changing the position of the box.

	Data Type	Explanation
Parameter:		
⇦ width	System.Int32	width of the box
⇦ height	System.Int32	height of the box
Return value	System.Boolean	Extent of the box is returned/not returned

GetTopLeftPixel

Method of VcBox

This method lets you convert to pixel and display the saved XY offset for the top left corner.

The x value can be further used with the method **VcGantt.GetDate** for instance to get a date.

	Data Type	Explanation
Parameter:		
⇦ x	System.Int32	X value of the offset
⇦ y	System.Int32	Y value of the offset
Return value	System.Boolean	Offset is returned/not returned

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
Parameter:		
⇦ xOffset	System.Int32	X value of the offset
⇦ yOffset	System.Int32	Y value of the offset
Return value	System.Boolean	Offset is returned/not returned

IdentifyFormatField

Method of VcBox

This method lets you retrieve the index of the format field at the specified position. If there is a field at the position specified, **True** will be returned, if there isn't, the method will deliver **False**.

	Data Type	Explanation
Parameter:		
⇨ x	System.Int32	X coordinate of the position
⇨ y	System.Int32	Y coordinate of the position
⇦ format	VcBoxFormat	Identified format
⇦ formatFieldIndex	System.Int16	Index of the format field
Return value	System.Boolean	A format field exists/does not exist at the position specified

SetXYOffset

Method of VcBox

This method lets you specify the distance between origin and reference point in x and y direction (unit: 1/100 mm).

You also can specify the offset in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Parameter:		
⇨ xOffset	System.Int32	X value of the offset
⇨ yOffset	System.Int32	Y value of the offset
Return value	System.Boolean	Offset is set (True)/not set (False)

Example Code VB.NET

```
Dim offSet As Boolean
offSet = VcNet1.BoxCollection.FirstBox.SetXYOffset(100, 100)
```

Example Code C#

```
bool offSet = vcNet1.BoxCollection.FirstBox().SetXYOffset(100, 100);
```

SetXYOffsetByTopLeftPixel

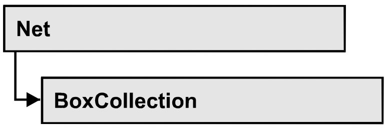
Method of VcBox

This method lets you internally convert the specified pixel value of the top left corner to an XY offset and then save the offset.

This enables you for instance to place a box at an XY coordinate from an event.

	Data Type	Explanation
Parameter:		
⇨ x	System.Int32	X value of the offset
⇨ y	System.Int32	Y value of the offset
Return value	System.Boolean	Offset is set (True) / not set (False)

7.5 VcBoxCollection



The VcBoxCollection object contains all boxes available. You can access all objects in an iterative loop by **For Each box In BoxCollection** or by the methods **First...** and **Next...**. You can access a single box by the method **BoxByName**. 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 = VcNet1.BoxCollection
numberOfBoxes = boxCltn.Count
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Parameter: ⇨ boxName	System.String	Box name
Return value	VcBox	New box object

Example Code VB.NET

```
newBox = VcNet1.BoxCollection.Add("box1")
```

Example Code C#

```
newBox = vcNet1.BoxCollection.Add("box1");
```

AddBySpecification

Method of VcBoxCollection

This method lets you create a box by using by a box specification. This way you can keep a box persistent. This way of creating allows box objects to become persistent. The specification of a box can be saved and re-loaded (see VcBox property **Specification**). In a subsequent the box can be created can be created again from the specification and is identified by its name. To make the new box visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter: ⇨ specification	System.String	Box specification
Return value	VcBox	New box object

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
boxCltn.AddBySpecification(textSpecification)
boxCltn.Update()
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Parameter: ⇨ index	System.Int16	Index of the box
Return value	VcBox	Box object returned

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByIndex(0)
box.LineThickness = 2
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Parameter:		
⇨ boxName	System.String	Box name
Return value	VcBox	Box

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByName("BoxOne")
box.LineThickness = 3
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Parameter:		
⇨ boxName	System.String	Name of the box to be copied
⇨ newBoxName	System.String	Name of the new box
Return value	VcBox	Box object

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
boxCltn.Copy("BoxOne", "NewBox")
boxCltn.Update()
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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 = VcNet1.BoxCollection
box = boxCltn.FirstBox
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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

Example Code VB.NET

```
Dim box As VcBox

For Each box In VcNet1.BoxCollection
   ListBox1.Items.Add(box.FormatName)
Next
```

Example Code C#

```
foreach (VcBox box in vcNet1.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 = VcNet1.BoxCollection
box = boxCltn.FirstBox

While Not box Is Nothing
   ListBox1.Items.Add(box.Name)
   box = boxCltn.NextBox
End While
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Parameter: ⇨ boxName	System.String	Box name
Return value	System.Boolean	Box deleted (True)/not deleted (False)

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.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
Return value	System.Boolean	Update successful (True)/ not successful (False)

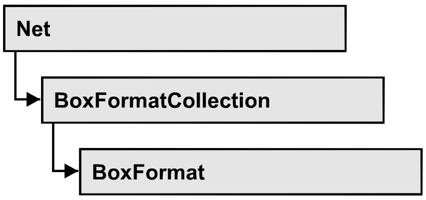
Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
boxCltn.Remove(box.Name);
boxCltn.Update();
```

7.6 VcBoxFormat



An object of the type **VcBoxFormat** defines the formats of boxes. With **For Each 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 = VcNet1.BoxFormatCollection.FormatByIndex(0)
boxFormat.FieldsSeparatedByLines = True
```

Example Code C#

```
VcBoxFormat boxFormat = vcNet1.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
Parameter:		
index	System.Int16 0 ... .FormatFieldCount-1	Index of the box format field
Property value	VcBoxFormatField	Nox format field

Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FirstFormat
formatField = boxFormat.FormatField(0)
MsgBox(formatField.FormatName)
```

Example Code C#

```
VcBoxFormat boxFormat = vcNet1.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
Property value	System.Int16	Number of fields of the box format

Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FirstFormat
MsgBox(boxFormat.FormatFieldCount)
```

Example Code C#

```
VcBoxFormat boxFormat = vcNet1.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 VcNet1.BoxFormatCollection
   ListBox1.Items.Add(boxFormat.Name)
Next
```

Example Code C#

```
foreach (VcBoxFormat boxFormat in vcNet1.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

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormat = boxFormatCltn.FirstBoxFormat
MsgBox(boxFormat.Specification)
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat boxFormat = boxFormatCltn.FirstBoxFormat();
MessageBox.Show(boxFormat.Specification);
```

Methods

CopyFormatField

Method of VcBoxFormat

This method allows to copy a box format field. The new VcBoxFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	VcFormatFieldInnerPosition	Position of the new box format field
	Possible Values:	
	.vcInnerAbove 1	above
	.vcInnerBelow 3	below
	.vcInnerLeftOf 0	left of
	.vcInnerRightOf 4	right of
⇨ refIndex	System.Int16	Index of the reference box format field
Return value	VcBoxFormatField	Box format field object

Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FormatByIndex(2)
boxFormat.CopyFormatField(VcFormatFieldInnerPosition.vcInnerRightOf, 0)
```

Example Code C#

```
VcBoxFormat boxFormat = vcNet1.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
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FirstFormat
For Each formatField In boxFormat
   ListBox1.Items.Add(formatField.FormatName)
Next
```

Example Code C#

```
VcBoxFormat boxFormat = vcNet1.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
Parameter:		
⇨ index	System.Int16	Index of the box format field to be deleted

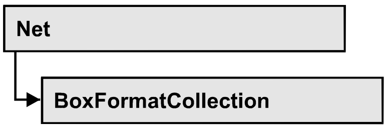
Example Code VB.NET

```
Dim boxFormat As VcBoxFormat
Dim i As Integer

boxFormat = VcNet1.BoxFormatCollection.FirstFormat

For i = 0 To boxFormat.FormatFieldCount - 1
   boxFormat.RemoveFormatField(i)
Next
```

Example Code C#

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FirstFormat();
for (short i=0; i<boxFormat.FormatFieldCount-1; i++)
   boxFormat.RemoveFormatField(i);
```

7.7 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 box formats in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the box formats in the corresponding way.

Properties

- 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 = VcNet1.BoxFormatCollection
numberOfBoxformats = boxFormatCltn.Count
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Parameter: ⇨ formatName	System.String	Box format name
Return value	VcBoxFormat	New box format object

Example Code VB.NET

```
Dim newBoxFormat = VcNet1.BoxFormatCollection.Add("boxFormat1")
```

Example Code C#

```
newBoxFormat = vcNet1.BoxFormatCollection.Add("boxFormat1");
```

AddBySpecification

Method of VcBoxFormatCollection

This method lets you create a box format by using a box format specification. This way of creating allows box format objects to become persistent. The specification of a box format can be saved and re-loaded (see VcBoxFormat property **Specification**). In a subsequent session the box format can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ formatSpecification	System.String	Box format specification
Return value	VcBoxFormat	New box format object

Copy

Method of VcBoxFormatCollection

By this method you can copy a box format. If the box format that is to be copied exists, and if the name for the new box format does not yet exist, the new box format object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	System.String	Name of the box format to be copied
⇨ newFormatName	System.String	Name of the new box format
Return value	VcBoxFormat	Box format object

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat")
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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).

	Data Type	Explanation
Return value	VcBoxFormat	First box format

Example Code VB.NET

```
Dim format As VcBoxFormat

format = VcNet1.BoxFormatCollection.FirstFormat
```

Example Code C#

```
VcBoxFormat format = vcNet1.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
Parameter:		
⇨ index	System.Int16	Index of the box format
Return value	VcBoxFormat	Box format object returned

Example Code VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcNet1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByIndex(2)
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Parameter:		
⇨ formatName	System.String	Name of the box format
Return value	VcBoxFormat	Box format

Example Code VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcNet1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByName("Standard")
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
   ListBox1.Items.Add(boxFormat.Name)
Next
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Return value	VcBoxFormat	Subsequent box format

Example Code VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcNet1.BoxFormatCollection
formatBox = formatBoxCltn.FirstFormat

While Not formatBox Is Nothing
   ListBox1.Items.Add(formatBox.Name)
   formatBox = formatBoxCltn.NextFormat
End While
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Parameter: ⇨ FormatName	System.String	Box format name
Return value	System.Boolean	Box format deleted (True)/not deleted (False)

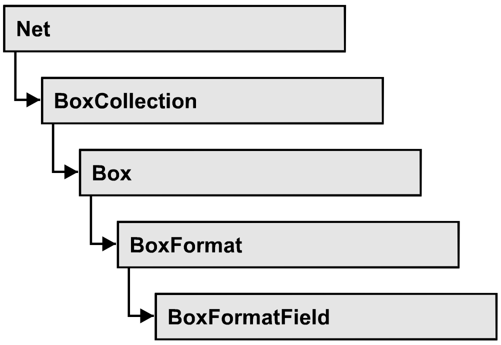
Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormat = boxFormatCltn.FormatByIndex(1)
boxFormatCltn.Remove(boxFormat.Name)
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat boxFormat = boxFormatCltn.FormatByIndex(1);
boxFormatCltn.Remove(boxFormat.Name);
```

7.8 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
- FormatName
- GraphicsHeight
- Index
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColor
- PatternColorAsARGB
- PatternEx
- TextFont
- TextFontColor
- Type

Properties

Alignment

Property of VcBoxFormatField

This property lets you set or retrieve the alignment of the content of the box format field.

	Data Type	Explanation
Property value	VcFormatFieldAlignment	Alignment of the field content
	Possible Values:	
	.vcFFABottom 28	Bottom
	.vcFFABottomLeft 27	Bottom left
	.vcFFABottomRight 29	Bottom right
	.vcFFACenter 25	Center
	.vcFFALeft 24	Left
	.vcFFARight 26	Right
	.vcFFATop 22	Top
	.vcFFATopLeft 21	Top left
	.vcFFATopRight 23	Top right

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter;
```

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 = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.FormatName)
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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 0 ... 99	Height (in mm) of the graphics 0...200

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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 = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.Index)
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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 0 ... 9	Maximum number of lines

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTText
boxFormatField.MaximumTextLineCount = 5
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTText;
boxFormatField.MaximumTextLineCount = 5;
```

MinimumTextLineCount

Property of VcBoxFormatField

This property lets you set or retrieve the minimum number of lines in the box format field, if it is of the type **vcFFTText**. If there is more text than can be taken by the lines, the format field will be enlarged dynamically up to the maximum number of lines. When assigning a value by this property, please also remember to set the **MaximumTextLineCount** value anew, since otherwise the minimum value might overwrite the maximum value.

	Data Type	Explanation
Property value	System.Int16 0 ... 9	Minimum number of lines 0...20

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTText
boxFormatField.MinimumTextLineCount = 3
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTText;
boxFormatField.MinimumTextLineCount = 3;
```

MinimumWidth

Property of VcBoxFormatField

This property lets you set or retrieve the minimum width of the box field in mm. The field width may be enlarged, if above or below the field fields exist that have greater minimum widths.

	Data Type	Explanation
Property value	System.Int16 0 ... 9	Minimum width of the box format field 0...200

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.MinimumWidth = 100
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.MinimumWidth = 100;
```

PatternBackgroundColor

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

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.BackgroundColor = Color.Red
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.BackgroundColor = Color.Red;
```

PatternColorAsARGB

Read Only Property of VcBoxFormatField

This property lets you set or retrieve the pattern color of the box format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** -1

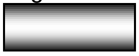
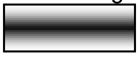
Example Code VB.NET

```
boxFormatField.PatternColor = RGB(0, 255, 0)
```

PatternEx

Property of VcBoxFormatField

This property lets you set or retrieve the pattern of the field background of the box format field.

	Data Type	Explanation
Property value	VcFieldFillPattern	Pattern type
	Possible Values:	
	.vcAeroGlassPattern 44	Vertical color gradient in the color of the fill pattern Engine Cabin Rig & Sail
	.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

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Pattern = VcFillPatternSingleColored.vcSingleColoredNoPatter
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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	System.DrawingFont	Font type of the box format

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.TextFont.FontFamily.ToString())
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Property value	System.Drawing.Color	Font color of the box format **Default value:** Color.Black

Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFontColor = Color.Red
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.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
Property value	VcFormatFieldType	Type of the box format field
	Possible Values: .vcFFTGraphics 64 .vcFFTText 36	 Graphics Text

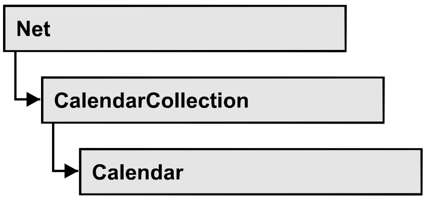
Example Code VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

Example Code C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics;
boxFormatField.GraphicsHeight = 150;
```

7.9 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. A calendar is useful for scheduling, e.g. to count the work days between two set dates.

Properties

- CalendarProfileCollection
- IntervalCollection
- Name
- SecondsPerWorkday
- Specification

Methods

- AddDuration
- CalcDuration
- Clear
- GetEndOfPreviousWorktime
- GetNextIntervalBorder
- GetPreviousIntervalBorder
- GetStartOfInterval
- GetStartOfNextWorktime
- IsWorktime
- Update

Properties

CalendarProfileCollection

Read Only Property of VcCalendar

This property gives access to the CalenderProfileCollection object that contains all calendar profiles available in this VcCalendar object.

	Data Type	Explanation
Property value	VcCalendarProfileCollection	CalendarProfileCollection object

IntervalCollection

Read Only Property of VcCalendar

This property gives access to the IntervalCollection object that contains all intervals available.

	Data Type	Explanation
Property value	VcIntervalCollection	IntervalCollection object

Name

Read Only Property of VcCalendar

This property lets you retrieve the name of a calendar.

	Data Type	Explanation
Property value	System.String	Name of the calendar

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim calendarName As String

calendar = VcNet1.CalendarCollection.FirstCalendar
calendarName = calendar.Name
```

Example Code C#

```
VcCalendar calendar = vcNet1.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

Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.FirstCalendar
MsgBox(calendar.Specification)
```

Example Code C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();
MessageBox.Show(calendar.Specification);
```

Methods

AddDuration

Method of VcCalendar

This method lets you assign a duration (work time) to a date of the calendar, considering the settings of the calendar. If e.g. you have defined workfree weekends to your calendar, a duration of three days added to a Friday will result in the Wednesday following.

	Data Type	Explanation
Parameter:		
⇨ date	System.DateTime	Date the duration is to be inserted at
⇨ duration	System.Int32	Number of time units (e.g.days)
Return value	System.DateTime	Date the duration was inserted at

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim newDate As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
newDate = calendar.AddDuration("16.06.2017", 3)
```

Example Code C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime newDate = calendar.AddDuration(Convert.ToDateTime("16.06.2017"), 3);
```

CalcDuration

Method of VcCalendar

This method lets you retrieve the number of work time elements (e.g. work days) available between two defined dates. The unit (e.g. days) of the value returned is the one defined in the **Time Unit** field on the **General** property page.

	Data Type	Explanation
Parameter:		
⇨ fromDate	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
Return value	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 = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
duration = calendar.CalcDuration("01.01.2014", "31.12.2014")
```

Example Code C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
int duration = calendar.CalcDuration(Convert.ToDateTime("01.01.2014"),
Convert.ToDateTime("31.12.2014"));
```

Clear

Method of VcCalendar

Removes the profiles and intervals formerly defined in this VcCalendar object, thus completely clearing it (=> 100% working time). The changes will only be displayed after an update.

	Data Type	Explanation

GetEndOfPreviousWorktime

Method of VcCalendar

This method lets you retrieve the end of the work time that precedes the reference date. The reference date has to belong to a non-working period.

	Data Type	Explanation
Parameter:		
⇨ date	System.DateTime	Date that the previous work time refers to
Return value	System.DateTime	Final date of the previous work time

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim endOfWork As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
endOfWork = calendar.GetEndOfPreviousWorktime("18.06.2014")
```

Example Code C#

```
VcCalendar calendar = vcNet1.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
Parameter: ⇨ date	System.DateTime	Date that the subsequent interval border refers to
Return value	System.DateTime	Start date of the subsequent interval border

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim nextIntervalBorder As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
nextIntervalBorder = calendar.GetNextIntervalBorder("18.06.2014")
```

Example Code C#

```
VcCalendar calendar = vcNet1.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
Parameter: ⇨ date	System.DateTime	Date that of the preceding interval border refers to
Return value	System.DateTime	End date of the interval border preceding

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim previousIntervalBorder As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
previousIntervalBorder = calendar.GetPreviousIntervalBorder("18.06.2014")
```

Example Code C#

```
VcCalendar calendar = vcNet1.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
Parameter: ⇨ date	System.DateTime	Reference date of the interval, that the start date is to be retrieved of
Return value	System.DateTime	Start date of the interval

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim startOfInterval As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
startOfInterval = calendar.GetStartOfInterval("18.06.2014")
```

Example Code C#

```
VcCalendar calendar = vcNet1.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
Parameter: ⇨ date	System.DateTime	Reference date, of which the start date of the subsequent work time is to be retrieved
Return value	System.DateTime	Start date of the subsequent work time

Example Code VB.NET

```
Dim calendar As VcCalendar
Dim startOfNextWorktime As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
startOfNextWorktime = calendar.GetStartOfNextWorktime("18.06.2017")
```

Example Code C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime startOfNextWorktime =
calendar.GetStartOfNextWorktime(Convert.ToDateTime("18.06.2017"));
```

IsWorktime

Method of VcCalendar

This method lets you retrieve whether or not the date passed is in a work time.

	Data Type	Explanation
Parameter:		
⇨ date	System.DateTime	Date to be checked for being a work time
Return value	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 = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
isWorktime = calendar.IsWorktime ("18.06.2014")
```

Example Code C#

```
VcCalendar calendar = vcNet1.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
Return value	Void	

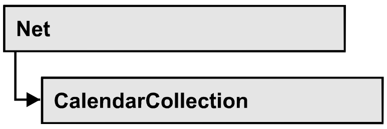
Example Code VB.NET

```
Dim calendar As VcCalendar

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
calendar.Update()
```

Example Code C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
calendar.Update();
```

7.10 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 = VcNet1.WorkdayCollection.CreateWorkday("Work day")
workday.AddNonWorkInterval("00:00:00", "00:00:00")
workday.AddWorkInterval("08:00:00", "16:30:00")

freeday = VcNet1.WorkdayCollection.CreateWorkday("Workfree day")
freeday.AddNonWorkInterval("00:00:00", "00:00:00")

calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.CreateCalendar("New calendar")

workweek = VcNet1.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 = vcNet1.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 = vcNet1.WorkdayCollection.CreateWorkday("Workfree day");
freeday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));

VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
VcCalendar calendar = calendarCltn.CreateCalendar("New calendar");

VcWorkweek workweek = vcNet1.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 = VcNet1.CalendarCollection
numberOfCalendar = calendarCltn.Count
```

Example Code C#

```
VcCalendarCollection calendarCltn = vcNet1.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
Parameter: ⇨ calendarName	System.String	Calendar name
Return value	VcCalendar	New calendar object

AddBySpecification

Method of VcCalendarCollection

This method lets you create a calendar by using a calendar specification. This way of creating allows calendar objects to become persistent. The specification of a calendar can be saved and re-loaded (see VcCalendar property **Specification**). In a subsequent the calendar can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ Specification	System.String	Calendar specification
Return value	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
Parameter: ⇨ index	System.Int16	Index of the calendar
Return value	VcCalendar	Calendar object returned

CalendarByName

Method of VcCalendarCollection

By this method you can retrieve a calendar by its name. If a calendar of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ calendarName	System.String	Name of the calendar
Return value	VcCalendar	Calendar

Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection

calendarCltn = VcNet1.CalendarCollection
calendarCltn.Active = calendarCltn.CalendarByName("Calendar_1")
```

Example Code C#

```
VcCalendarCollection calendarCltn = vcNet1.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.

	Data Type	Explanation
Parameter:		
⇨ calendarName	System.String	Name of the calendar to be copied
⇨ newCalendarName	System.String	Name of the calendar
Return value	VcCalendar	Calendar object

FirstCalendar

Method of VcCalendarCollection

This method can be used to access the initial value, i.e. the first calendar of a calendar collection, 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
Return value	VcCalendar	First calendar

Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.FirstCalendar
```

Example Code C#

```
VcCalendarCollection calendarCltn = vcNet1.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
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim calendar As VcCalendar

For Each calendar In VcNet1.CalendarCollection
   MsgBox(calendar.Name)
Next
```

Example Code C#

```
foreach (VcCalendar calendar in vcNet1.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
Return value	VcCalendar	Succeeding calendar

Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar
calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.FirstCalendar

While Not calendar Is Nothing
   ListBox1.Items.Add(calendar.Name)
   calendar = calendarCltn.NextCalendar
End While
```

Example Code C#

```
VcCalendarCollection calendarCltn = vcNet1.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.

	Data Type	Explanation
Return value	System.Boolean	Calendar deleted (True)/not deleted (False)

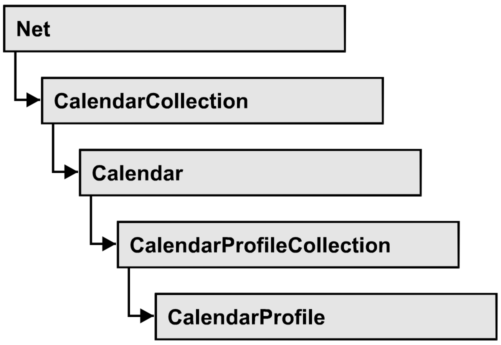
Update

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)

7.11 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

Example Code VB.NET

```
Dim calendarProfileCltn As VcCalendarProfileCollection
Dim calendarProfile As VcCalendarProfile

calendarProfileCltn = VcNet1.CalendarProfileCollection
calendarProfile = calendarProfileCltn.FirstCalendarProfile
MsgBox(calendarProfile.Specification)
```

Example Code C#

```
VcCalendarProfileCollection calendarProfileCltn =
vcNet1.CalendarProfileCollection;
VcCalendarProfile calendar = calendarProfileCltn.FirstCalendarProfile();
MessageBox.Show(calendarProfile.Specification);
```

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	Type of the calendar profile

Methods

PutInOrderAfter

Method of VcCalendarProfile

This method lets you set the calendar profile behind the calendar profile specified by name, within the CalendarProfileCollection. If you set the name to "", the calendar profile will be put in the first position. The order of the calendar profiles within the collection determines the order by which they apply to the calendars.

	Data Type	Explanation
Parameter:		
refNameParam	String	Name of the calendar profile behind which the current calendar profile is to be put.

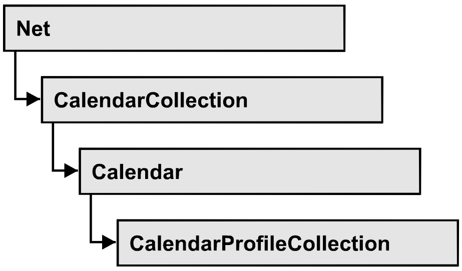
Example Code VB.NET

```
Dim calProfCltn As VcCalendarProfileCollection
Dim calProf1 As VcCalendarProfile
Dim calProf2 As VcCalendarProfile

calProfCltn = VcGantt1.CalendarProfileCollection()
calProf1 = calProfCltn.Add("calProf1")
calProf2 = calProfCltn.Add("calProf2")
calProf1.PutInOrderAfter("calProf2")
calProfCltn.Update()
```

Example Code C#

```
VcCalendar ProfileCollection calProfCltn = vcGantt1.Calendar ProfileCollection;
VcCalendar Profile calProf1 = calProfCltn.Add("calProf1");
VcCalendar Profile calProf2 = calProfCltn.Add("calProf2");
calProf1.PutInOrderAfter("calProf2");
calProfCltn.Update();
```

7.12 VcCalendarProfileCollection



An object of the type VcCalendarProfileCollection automatically contains all available calendar profiles. You can access all objects in an iterative loop by **For Each calendarProfile In CalendarProfileCollection** or by the methods **First...** and **Next...**. You can access a single calendar profile using the methods **CalendarProfileByName** and **CalendarProfileByIndex**. The number of calendar profiles in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the calendar profiles in the corresponding way.

Properties

- Count

Methods

- Add
- AddBySpecification
- CalendarProfileByIndex
- CalendarProfileByName
- Copy
- FirstCalendarProfile
- NextCalendarProfile
- Remove
- SelectCalendarProfiles
- Update
- Update

Properties

Count

Read Only Property of VcCalendarProfileCollection

This property lets you retrieve the number of calendar profiles in the calendar profile collection.

	Data Type	Explanation
Property value	System.Int32	Number of CalendarProfile objects

Methods

Add

Method of VcCalendarProfileCollection

By this method you can create a calendar profile as a member of the CalendarProfileCollection. If the name has not been used before, the new filter object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ profileName	System.String	Calendar profile name
Return value	VcCalendarProfile	New calendar profile object

AddBySpecification

Method of VcCalendarProfileCollection

This method lets you create a calendar profile by using a calendar profile specification. This way of creating allows calendar profile objects to become persistent. The specification of a calendar profile can be saved and re-loaded (see VcCalendarProfile property **Specification**). In a subsequent the calendar profile can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ Specification	System.String	Calendar profile specification
Return value	VcCalendarProfile	New calendarprofile object

CalendarProfileByIndex

Method of VcCalendarProfileCollection

This method lets you access a calendar profile by its index. If no calendar profile of the specified index does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	System.Int16	Index of the calendar profile
Return value	VcCalendarProfile	Calendar profile object returned

CalendarProfileByName

Method of VcCalendarProfileCollection

By this method you can retrieve a calendar profile by its name. If no calendar profile of the specified name does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ profileName	System.String	Name of the calendar profile object
Return value	VcCalendarProfile	Calendar profile object returned

Copy

Method of VcCalendarProfileCollection

By this method you can copy a calendar profile. If the calendar profile that is to be copied exists, and if the name for the new calendar profile does not yet exist, the new calendar profile object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ profileName	System.String	Name of the calendar profile to be copied
⇨ newProfileName	System.String	Name of the new calendar profile
Return value	VcCalendarProfile	Calendar profile object

FirstCalendarProfile

Method of VcCalendarProfileCollection

This method can be used to access the initial value, i.e. the first calendar profile of a calendar profile collection, and then to continue in a forward iteration loop by the method **NextCalendarProfile** for the calendar profiles following. If there is no calendar profile in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendarProfile	First calendar profile object

NextCalendarProfile

Method of VcCalendarProfileCollection

This method can be used in a forward iteration loop to retrieve subsequent calendar profiles from a calendar profile collection after initializing the loop by the method **FirstCalendarProfile**. If there is no calendar profile left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendarProfile	Subsequent calendar profile object

Remove

Method of VcCalendarProfileCollection

This method lets you delete a calendar profile. If the calendar profile is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter: ⇨ profileName	System.String	Calendar profile name
Return value	System.Boolean	Calendar profile deleted (True)/not deleted (False)

SelectCalendarProfiles

Method of VcCalendarProfileCollection

This method lets you specify the calendar profiles that the calendar profile collection is to contain.

	Data Type	Explanation
Parameter: ⇨ selectionType	CalendarProfileTypeEnum	Type of calendar profile to be selected
Return value	System.Int32	Number of calendar profiles selected

Example Code VB.NET

```
Dim calendarProfileCltn As VcCalendarProfileCollection

Set calendarProfileCltn = VcNet1.CalendarProfileCollection
calendarProfileCltn.SelectCalendarProfile (vcSelected)
```

Update

Method of VcCalendarProfileCollection

This method lets you update a calendar profile collection after having modified it.

	Data Type	Explanation
Return value	System.Boolean	update successful (True)/ not successful (False)

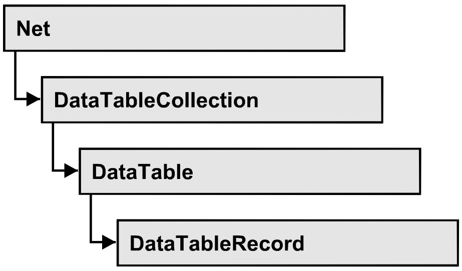
Update

Method of VcCalendarProfileCollection

This method lets you update a calendar profile collection after having modified it.

	Data Type	Explanation
Return value	System.Boolean	update successful (True)/ not successful (False)

7.13 VcDataRecord



A data record is the logical base of an object in a Net 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
- IdentifyObject
- RelatedDataRecord

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.

	Data Type	Explanation
Property value	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 = VcNet1.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 = vcNet1.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
Parameter:		
⇨ index	System.Int16	Index of data field

Property value	System.Object	Content of the data field

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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.

	Data Type	Explanation
Property value	System.String	Name of the associated table

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

MsgBox(dataRecord.DataTableName)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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 = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)
MsgBox(dataRecord.ID)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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 = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.Delete()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

dataRecord.Delete();
```

IdentifyObject

Method of VcDataRecord

This method lets you identify the object having been established via this VcDataRecord object.

The return value will be **true** if a data-based object could be identified, i.e. if a data-based object could be created for the graphic from the record.

	Data Type	Explanation
Parameter:		
⇨ establishedObject Param	System.Object	Identified object
establishedObjectTypeParam	VcObjectType	Object type
	Possible Values:	
	.vcObjTypeBox 15	object type **box**
	.vcObjTypeGroup 7	object type **group**
	.vcObjTypeLinkCollection 3	object type **link collection**
	.vcObjTypeNode 2	object type **node**
	.vcObjTypeNone 0	no object
Return value	System.Boolean	data-based object has been/has not been established

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
Parameter:		
⇨ index	System.Int16	Index of data field
Return value	VcDataRecord	Related data record

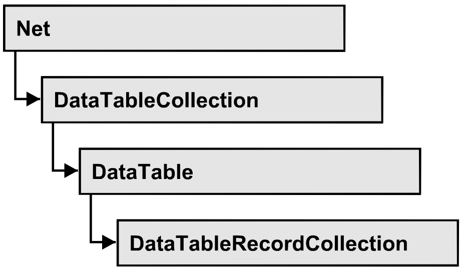
Example Code VB.NET

```
   Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
      Dim dataTable As VcDataTable
      Dim dataRecordCltn As VcDataRecordCollection
      Dim firstDataRecord As VcDataRecord
      Dim secondDataRecord As VcDataRecord

      dataTable = VcNet1.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 vcNet1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   VcDataTable dataTable = vcNet1.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());
   }
```

7.14 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
- NextDataRecord
- Remove
- Update

Properties

Count

Read Only Property of VcDataRecordCollection

This property lets you retrieve the number of data records in the DataRecordCollection object.

	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 = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
MsgBox("Number of DataRecords: " & dataRecordCltn.Count)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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. After adding the data record, the method **VcNet.EndLoading** needs to be invoked to make the modification take effect.

	Data Type	Explanation
Parameter: ⇨ dataRecordContent	VcObject	Content of the data record (as an array or a string)
Return value	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 = VcNet1.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)
VcNet1.EndLoading()

' 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 = vcNet1.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);
VcNet1.EndLoading();

// 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
Parameter:		
⇨ dataRecordID	System.String	ID of the data record
Return value	VcDataRecord	Data record object

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(0)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Return value	VcDataRecord	First data record

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.FirstDataRecord
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Return value	VcObject	Enumerator object

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcNet1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.Update()
   dataRecord = dataRecordCltn.NextDataRecord
End While

VcNet1.SuspendUpdate(False)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcNet1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
   dataRecord.set_DataField(4, "10");
   dataRecord.Update();
   dataRecordCltn.NextDataRecord();
}

vcNet1.SuspendUpdate(false);
```

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 = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcNet1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.Update()
   dataRecord = dataRecordCltn.NextDataRecord
End While

VcNet1.SuspendUpdate(False)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcNet1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
   dataRecord.set_DataField(4, "10");
   dataRecord.Update();
   dataRecordCltn.NextDataRecord();
}

vcNet1.SuspendUpdate(false);
```

Remove

Method of VcDataRecordCollection

This method lets you delete a data record. The method returns **true** after having deleted a data record and **false** when no data record was deleted. The content of the data record is used to identify the object by its identification.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	VcObject	Content of the data record (as an array or a string)
Return value	System.Boolean	true

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Remove("1;1Activity; Y;Z;18.01.14;;5")
VcNet1.EndLoading()

' equivalent
' dataRecord = dataRecordCltn.DataRecordByID(1)
' dataRecord.Delete()
' dataRecord.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

dataRecCltn .Remove("1;1Activity Y;Z;18.01.14;;5");
VcNet1.EndLoading();

// 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()**. After updating the data record, the method **VcNet.EndLoading** needs to be invoked to make the modification take effect.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	VcObject	Content of the data record (as an array or a string)
Return value	System.Boolean	Update successful (true) / not successful (false)

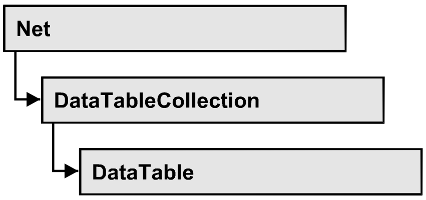
Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Update("1;1.8.2017;;8")
VcNet1.EndLoading()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Update("1;1.8.2017;;8");
VcNet1.EndLoading();
```

7.15 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 = VcNet1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataRecordCollection.Count)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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 have been terminated before data records can be filled in the table.

	Data Type	Explanation
Property value	VcDataTableFieldCollection	DataTableFieldCollection object

Example Code VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
MsgBox(dataTable.DataTableFieldCollection.Count)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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 **Default value:** Empty string

Example Code VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataTable.Description = "This table contains data for nodes"
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Property value	System.Boolean	Use of composite primary keys allowed (true)/not allowed (false) **Default value:** 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 **DataTable-Collection** you can retrieve a reference to the data table object.

	Data Type	Explanation
Property value	System.String	Name of the data table **Default value:** Empty string

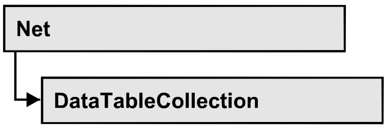
Example Code VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
MsgBox(dataTable.Name)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByIndex(0);
MessageBox.Show(dataTable.Name);
```

7.16 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 XNet 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 = VcNet1.DataTableCollection
MsgBox(dataTableCltn.Count.ToString())
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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. 90 data tables can be created at maximum.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	System.String	Name of the new data table
Return value	VcDataTable	Data table generated

Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTableCltn.Update()
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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** was set to **true**, data tables can be copied. If the data table could be copied, a new object of the type **VcDataTable** will be returned; otherwise **Nothing** in Visual Basic or **0** in other languages. The table names are case sensitive.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	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)
Return value	VcDataTable	Data table object generated

Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Copy("Resources", "NewResources")
dataTableCltn.Update()
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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
Parameter:		
⇨ index	System.Int16	Index of the data table
Return value	VcDataTable	Data table object returned

Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.DataTableByIndex(2)
MsgBox(dataTable.Name)
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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
Parameter:		
⇨ dataTableName	System.String	Name of the data table
Return value	VcDataTable	Data table object returned

Example Code VB.NET

```
Dim dataTablecltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTablecltn = VcNet1.DataTableCollection
dataTable = dataTablecltn.DataTableByName("Resources")
MsgBox(dataTable.Description)
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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
Return value	VcDataTable	First data table

Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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 = VcNet1.DataTableCollection
For Each dataTable In dataTableCltn
   ListBox1.Items.Add(dataTable.Name)
Next
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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 = VcNet1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
For i = 1 To dataTableCltn.Count
   ListBox1.Items.Add(dataTable.Name)
   dataTable = dataTableCltn.NextDataTable
Next
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.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
Return value	System.Boolean	Update successful (true) / not successful (false)

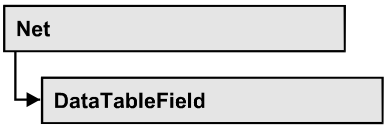
Example Code VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTable.DataTableFieldCollection.Add("Id")
dataTableCltn.Update()
```

Example Code C#

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.Add("Resources");
dataTable.DataTableFieldCollection.Add("Id");
dataTableCltn.Update();
```

7.17 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 = VcNet1.DataTableCollection.FirstDataTable
MsgBox(dataTable.DataTableFieldCollection.FirstDataTableField.DataTableName)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Property value	System.String	Date format {DMYhms:;./}

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
'DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY"
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
//DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY";
vcNet1.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
Property value	System.Boolean	Field editable (True) / not editable (False) **Default value:** True

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Editable = False
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Editable = false;
VcNet1.DataTableCollection.Update();
```

Hidden

Property of VcDataTableField

This property lets you set or retrieve whether the data field should be hidden at run time in the dialogs **EditNode** and **EditLink**.

	Data Type	Explanation
Property value	System.Boolean	Field hidden (True) / not hidden (False) **Default value:** False

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Hidden = True
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Hidden = true;
vcNet1.DataTableCollection.Update();
```

Index

Read Only Property of VcDataTableField

This property lets you retrieve the index of the data table field in the associated data table.

	Data Type	Explanation
Property value	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
Property value	System.String	Name of the field **Default value:** Empty string

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.Add("Start")
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Start");
vcNet1.DataTableCollection.Update();
```

PrimaryKey

Property of VcDataTableField

This property lets you set or retrieve whether this field contains the primary key, which is used for the unique identification of a data record. In a data table, only one of the fields that were defined can be the primary key. Within the same table, assigning the primary key function to a field automatically cancels the previous assignment. A primary key is required in a table if records of a different table are to depend on the records of the former one.

	Data Type	Explanation
Property value	System.Boolean	The field serves (True) / does not serve (False) as a primary key. **Default value:** False

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
Dim isPrimaryKey As Boolean

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Id")
dataTableField.PrimaryKey = True
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Id");
dataTableField.PrimaryKey = true;
vcNet1.DataTableCollection.Update();
```

RelationshipFieldIndex

Property of VcDataTableField

This property lets you combine a data field and its data description. For this, please set the index of the data record field to which the settings of this data table field shall refer.

	Data Type	Explanation
Property value	System.Int32	Index of the record field to which the data definition of the data table field refers. **Default value:** -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 = VcNet1.DataTableCollection.Add("Task")
dataTaskFieldId = dataTableTask.DataTableFieldCollection.Add("Id")
dataTaskFieldId.PrimaryKey = True
dataTaskFieldName = dataTableTask.DataTableFieldCollection.Add("Name")
dataTaskFieldName.Type = VcDataTableFieldType.vcDataTableFieldStringType

'Create table Operation
dataTableOperation = VcNet1.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 =
VcNet1.DetectFieldIndex("Task", "Id")
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
//Create table Task
VcDataTable dataTableTask = vcNet1.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 = vcNet1.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 =
vcNet1.DetectFieldIndex("Task","Id");
vcNet1.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
Property value	VcDataTableFieldType	Data type of the field, can contain 512 characters maximum **Default value:** vcDataTableFieldIntegerType

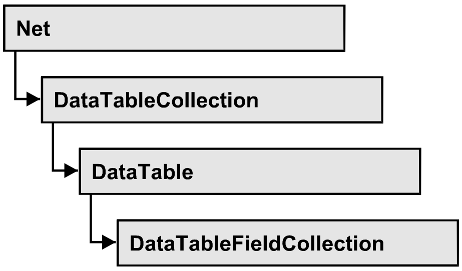
Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
vcNet1.DataTableCollection.Update();
```

7.18 VcDataTableFieldCollection



An object of the type VcDataTableFieldCollection automatically contains all data fields of a data table. The property **Count** retrieves the number of fields present in the collection; the Enumerator object and the methods **FirstDataField** and **NextDataField** allow to access data fields by iteration while by **DataFieldByname** and **DataFieldByIndex** single data fields can be accessed. **Add** and **Copy** represent basic administering methods.

Properties

- 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 = VcNet1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataTableFieldCollection.Count.ToString())
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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. 9,999 fields can be created at maximum.

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	System.String	Name of the data table field to be generated
Return value	VcDataTableField	Data table field generated

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Add("Priority")
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Priority");
vcNet1.DataTableCollection.Update();
```

Copy

Method of VcDataTableFieldCollection

This method lets you copy a data table field. The field is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	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)
Return value	VcDataTableField	Data table field generated

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Copy("Name", "NewName")
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Copy("Name", "NewName");
vcNet1.DataTableCollection.Update();
```

DataTableFieldByIndex

Method of VcDataTableFieldCollection

This method lets you access a data table field by its index. If a data field does not exist at the index specified, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	System.Int16	Index of the data table field
Return value	VcDataTableField	Data table field returned

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByIndex(1)
MsgBox(dataTableField.Name)
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Parameter: ⇨ dataTableFieldName	System.String	Name of the data table field
Return value	VcDataTableField	Data table field returned

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Name")
dataTableField.Editable = False
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Name");
dataTableField.Editable = false;
vcNet1.DataTableCollection.Update();
```

FirstDataTableField

Method of VcDataTableFieldCollection

This method can be used to access the initial value, i.e. the first data table field of a data table field collection, and to continue in a forward iteration loop by the method **NextDataTableField** for the fields following. If there is no field in the data table field collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTableField	First data table field

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.FirstDataTableField()
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Return value	VcObject	Enumerator object

Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
For Each dataTableField In dataTable.DataTableFieldCollection
   ListBox1.Items.Add(dataTableField.Name)
Next
```

Example Code C#

```
VcDataTable dataTable = vcNet1.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
Return value	VcDataTable	Succeeding data table field

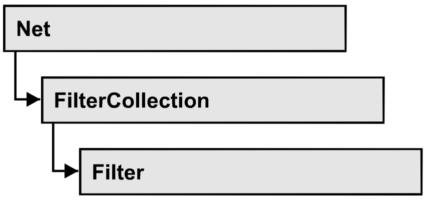
Example Code VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableFieldCltn As VcDataTableFieldCollection
Dim dataTableField As VcDataTableField
Dim i As Integer

dataTable = VcNet1.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 = vcNet1.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();
   }
```

7.19 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
- Evaluate
- 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	Type of data definition table
	Possible Values:	
	.vcMainData 0	Definition of node data
	.vcMaindata 0	table type **vcMaindata** (for nodes)
	.vcRelations 1	Definition of link data
	.vcRelations 1	table type **vcRelations** (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 = VcNet1.FilterCollection

For Each filter In filterCltn
   ListBox1.Items.Add(filter.Name)
Next
```

Example Code C#

```
VcFilterCollection filterCltn = vcNet1.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

Example Code VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filterCltn.FirstFilter
MsgBox(filter.Specification)
```

Example Code C#

```
VcFilterCollection boxCltn = vcNet1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();
MessageBox.Show(filter.Specification);
```

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
Parameter: ⇨ index	System.Int16	Index of the filter subcondition {0 ... VcFilter.SubConditionCount-1}
Property value	VcFilterSubCondition	Filter subcondition object

SubConditionCount

Read Only Property of VcFilter

This property lets you enquire the number of filter subconditions.

	Data Type	Explanation
Property value	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
Parameter:		
⇨ 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)}
Return value	VcFilterSubCondition	Filter subcondition object

CopySubCondition

Method of VcFilter

This method lets you copy a filter subcondition by its index. The new filter subcondition will be inserted into the collection at the position specified by the index. It will be returned as a VcFilterSubCondition object.

	Data Type	Explanation
Parameter:		
⇨ fromIndex	System.Int16	Index of the filter subcondition to be copied
⇨ 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)}
Return value	VcFilterSubCondition	Filter subcondition object

Evaluate

Method of VcFilter

This methods lets you check whether the specified filter applies for a certain data record or not. You should only pass objects that are internally linked with data records of the data tables. Those are **VcNode, VcLink, VcGroup,**

VcDataRecord. If an object is passed that is not listed, an exception will be triggered.

	Data Type	Explanation
Parameter: ⇨ dataObjectParam	Variant	Data record object
Return value	Boolean	Filter applies for data record (True)/does not apply (False)

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
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim filter As VcFilter
Dim filterCond As VcFilterSubCondition

filter = VcNet1.FilterCollection.FirstFilter

For Each filterCond In filter
   Debug.Write(filterCond.Index)
Next
```

Example Code C#

```
VcFilter filter = vcNet1.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 subconditons will remain valid.

	Data Type	Explanation
Return value	System.Boolean	Filter subconditions correct (True)/ not correct (False)

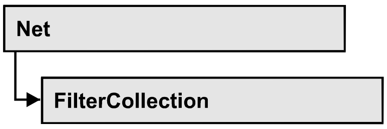
RemoveSubCondition

Method of VcFilter

This method lets you delete a filter subcondition by its index.

	Data Type	Explanation
Parameter:		
⇨ index	System.Int16	Index of the filter subcondition to be removed

7.20 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 = VcNet1.FilterCollection
numberOfFilters = filterCltn.Count
```

Example Code C#

```
VcFilterCollection filterCltn = vcNet1.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

```
VcNet1.ActiveNodeFilter = VcNet1.FilterCollection.MarkedNodesFilter
```

Example Code C#

```
vcNet1.ActiveNodeFilter = vcNet1.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
Parameter: ⇨ newName	System.String	Filter name
Return value	VcFilter	New filter object

Example Code VB.NET

```
newFilter = VcNet1.FilterCollection.Add("foo")
```

Example Code C#

```
newFilter = vcNet1.FilterCollection.Add("foo");
```

AddBySpecification

Method of VcFilterCollection

This method lets you create a filter by using filter specification. This way of creating allows filter objects to become persistent. The specification of a filter can be saved and re-loaded (see VcFilter property **Specification**). In a subsequent the filter can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ filterSpecification	System.String	Filter specification
Return value	VcFilter	New filter object

Copy

Method of VcFilterCollection

By this method you can copy a filter. If the filter that is to be copied exists, and if the name for the new filter does not yet exist, the new filter object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	System.String	Name of the filter to be copied
⇨ newName	System.String	Name of the new filter
Return value	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
Parameter: ⇨ index	System.Int16	Index of the filter
Return value	VcFilter	Filter object returned

FilterByName

Method of VcFilterCollection

By this method you can retrieve a filter by its name. If a filter of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ filterName	System.String	Filter name
Return value	VcFilter	Filter

Example Code VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filterCltn.FilterByName("Department A")
```

Example Code C#

```
VcFilterCollection filterCltn = vcNet1.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 = VcNet1.FilterCollection
filter = filtercltn.FirstFilter
```

Example Code C#

```
VcFilterCollection filterCltn = vcNet1.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 = VcNet1.FilterCollection.FirstFilter

For Each filterCond In filter
   Debug.Write(filterCond.FilterName)
Next
```

Example Code C#

```
VcFilter filter = vcNet1.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
Return value	VcFilter	Next filter

Example Code VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filtercltn.FirstFilter

While Not filter Is Nothing
   ListBox1.Items.Add(filter.Name)
   filter = filterCltn.NextFilter
End While
```

Example Code C#

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();

while (filter != null)
   {
   ListBox.Items.Add(filter.Name);
   filter = filterCltn.NextFilter();
   }
```

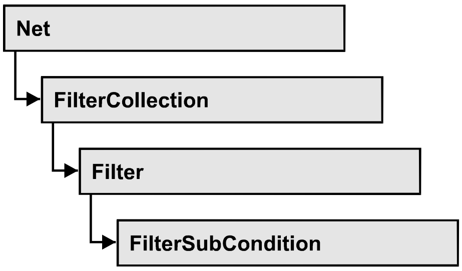
Remove

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
Parameter: ⇨ name	System.String	Filter name
Return value	System.Boolean	Filter deleted (True)/not deleted (False)

7.21 VcFilterSubCondition



An object of the type VcFilterSubCondition contains a single filter subcondition. It does not have a name, but only an index that specifies its position in the filter.

In the **Edit Filter** dialog each line corresponds to a subcondition. The properties specified at design time in that dialog can be modified via the API at runtime.

Properties

- ComparisonValueAsString
- ConnectionOperator
- DataFieldIndex
- FilterName
- Index
- Operator

Methods

- IsValid

Properties

ComparisonValueAsString

Property of VcFilterSubCondition

This property lets you 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\""

- Date: included by # signs. Example: "#18/06/2015;12:34;56;#". 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	Operator for the connection holding the below subcondition
	Possible Values:	
	.vcAnd 1	And operator
	.vcInvalidConnOp 0	invalid operator
	.vcOr 2	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 types of the comparison value and of the operator.

Special value: -1: no data field (invalid)

	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.

	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
	Possible Values:	
	.vcDateEarlier 27	Date earlier than
	.vcDateEarlierOrEqual 28	Date earlier than or equal
	.vcDateEqual 25	Date equal
	.vcDateIn 31	Date in
	.vcDateLater 29	Date later than

.vcDateLaterOrEqual 30	Date later than or equal
.vcDateNotEqual 26	Date not equal
.vcDateNotIn 32	Date not in
.vcIntEqual 9	integer equal
.vcIntGreater 13	integer greater
.vcIntGreaterOrEqual 14	integer greater or equal
.vcIntIn 15	integer in
.vcIntLess 11	integer smaller than
.vcIntLessOrEqual 12	integer smaller than or equal
.vcIntNotEqual 10	integer not equal
.vcIntNotIn 16	integer not in
.vcInvalidOp 0	invalid operator
.vcStringBeginsWith 3	string begins with
.vcStringContains 5	string contains
.vcStringEqual 1	string equal
.vcStringIn 7	string contains
.vcStringNotBeginsWith 4	string does not begin with
.vcStringNotContains 6	string does not contain
.vcStringNotEqual 2	string is not equal
.vcStringNotIn 8	string is not in

Methods

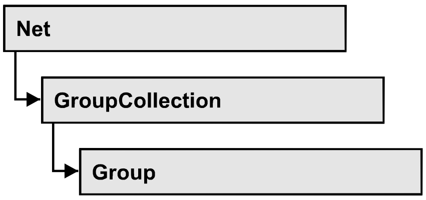
IsValid

Method of VcFilterSubCondition

This property checks whether the filter subcondition is correct.

	Data Type	Explanation
Return value	System.Boolean	Filter subcondition correct (True)/ not correct (False)

7.22 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

- BackgroundColor
- LineColor
- LineThickness
- LineType
- Name
- NodeCollection
- Title
- TitleLineCount
- X
- Y

Methods

- SetXY

Properties

BackgroundColor

Property of VcGroup

This property lets you assign/retrieve a background color to a group. The default color is white.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})

Example Code VB.NET

```
Dim groupCltn  As VcGroupCollection
Dim group As VcGroup

groupCltn  = VcNet1.GroupCollection
group = groupCltn.FirstGroup

group.BackColor = RGB(128, 128, 128)
```

Example Code C#

```
VcGroupCollection groupCltn  = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup;
group.BackColor = RGB(128, 128, 128);
```

LineColor

Read Only Property of VcGroup

This property lets you set or retrieve the line color of the group's border line. The line color can also be set in the **Administrate Intervals** dialog. This feature can also be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})

LineThickness

Read Only Property of VcGroup

This property lets you set or retrieve the line thickness of the border line of the group.

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.

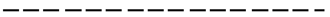
If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	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 **Default value:** As defined in the dialog

LineType

Read Only Property of VcGroup

This property lets you set or retrieve the (border) line type of a group. This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	VcLineType	Line type **Default value:** vcSolid
	Possible Values:	
	.vcDashed 4	Line dashed
	.vcDashed 4	Line dashed
	.vcDashedDotted 5	Line dashed-dotted
	.vcDashedDotted 5	Line dashed-dotted
	.vcDotted 3	Line dotted
	.vcDotted 3	Line dotted
	.vcLineType0 100	Line Type 0
	.vcLineType1 101	Line Type 1
	.vcLineType10 110	Line Type 10
	.vcLineType11 111	Line Type 11
	.vcLineType12 112	Line Type 12
	.vcLineType13 113	Line Type 13
	.vcLineType14 114	Line Type 14
	.vcLineType15 115	Line Type 15
	.vcLineType16 116	Line Type 16
	.vcLineType17 117	Line Type 17 

.vcLineType18 118	Line Type 18 —···· —···· —···· —···· —···· —···· —··
.vcLineType2 102	Line Type 2 ······································
.vcLineType3 103	Line Type 3 --------------------------------------
.vcLineType4 104	Line Type 4 -
.vcLineType5 105	Line Type 5 — — — — — — — — — — — —
.vcLineType6 106	Line Type 6 — — — — — — — —
.vcLineType7 107	Line Type 7 - - - - - - - - - - - - - - - - - - -
.vcLineType8 108	Line Type 8 – – – – – – – – – – – – – – –
.vcLineType9 109	Line Type 9 —---—---—---—---—---—---—
.vcNone 1	No line type assigned
.vcNone 1	No line type
.vcNotSet -1	No line type assigned
.vcSolid 2	Line solid
.vcSolid 2	Line solid

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 = VcNet1.GroupCollection
group = groupCltn.FirstGroup
groupName = group.Name
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
string groupName = group.Name;
```

NodeCollection

Read Only Property of VcGroup

This property lets you access all nodes of a group.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
nodeCltn = group.NodeCollection
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
VcNodeCollection nodeCltn = group.NodeCollection;
```

Title

Property of VcGroup

This property allows you to set or retrieve the group title. The group title will be displayed in the top row of the group. If you do not set this property, nor define a valid file name by VcNet.GroupDescriptionName, nor define a filed by VcNet.GroupTitleField, the name of the group will simply be displayed in the top row.

	Data Type	Explanation
Property value	System.String	Title of the Group

Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroupMsgBox(group.Title)
```

Example Code C#

```
VcGroupCollection groupCltn = Dummyobject2.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
MessageBox.Show(group.Title);
```

TitleLineCount

Property of VcGroup

This property allows you to set or retrieve for the current group the number of lines of the title text.

	Data Type	Explanation
Property value	System.Int16 1 ... 5	Number of lines of the title text **Default value:** 1

Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
group.TitleLineCount = 5
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
group.TitleLineCount = 5;
```

X

Read Only Property of VcGroup

This property lets you require the current x coordinate of the group.

	Data Type	Explanation
Property value	System.Int32	X coordinate

Y

Read Only Property of VcGroup

This property lets you require the current y coordinate of the group.

	Data Type	Explanation
Property value	System.Int32	Y coordinate

Methods

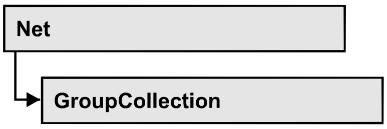
SetXY

Method of VcGroup

This method lets you set the position of the group. This method only can be used for the grouping mode clustering (GroupMode = vcGMClustering), and only if the group is collapsed or if this method is called in the **OnGroupCreate** event.

	Data Type	Explanation
Parameter:		
⇨ x	System.Int32	X coordinate in band numbers
⇨ y	System.Int32	Y coordinate in band numbers
Return value	System.Boolean	Values set successfully (True)/not set successfully (False)

7.23 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

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 = VcNet1.GroupCollection
numberOfGroups = groupCltn.Count
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.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 = VcNet1.GroupCollection
group = groupCltn.FirstGroup
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.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. Beforehand, you have to select the group by the method **SelectGroups**. If a group of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ Rückgabewert	VcGroup	Group
⇨ groupName	System.String	Name of group
Return value	VcGroup	Group

Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.GroupByName("Group A")
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.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
Return value	VcGroup	Subsequent group

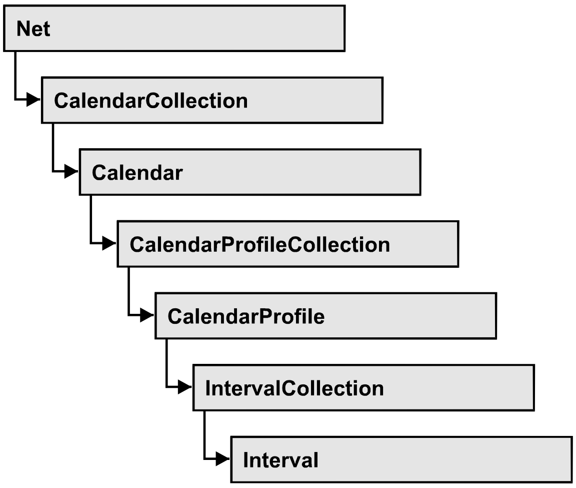
Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
While Not group Is Nothing
   ListBox1.Items.Add(group.Name)
   group = groupCltn.NextGroup
End While
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
while (group != null)
   {
   listBox1.Items.Add(group.Name);
   group = groupCltn.NextGroup();
   }
```

7.24 VcInterval



An object of the type **VcInterval** offers the possibility of defining time intervals that are interpreted as working or non-working time. The distinction between the two characteristics is made by the special settings **<WORK>** and **<NONWORK>** of the property **CalendarProfileName**. An interval may refer to other already defined calendar profiles by its property **CalendarProfileName**.

According to the current interval type (**vcCalendarInterval**, **vcDayProfileInterval**, **vcWeekProfileInterval**, **vcYearProfileInterval** oder **vcShiftProfileInterval**) which is not set explicitely but derives from the context of use, only certain properties of the object take effect.

The following table lists the interval types and their corresponding properties:

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

A **CalendarInterval** designates a non-recurring time span within a precisely defined period. Example: 5/5/2010 11:30 to 9/15/2010 5:00.

A **YearProfileInterval** allows to define a yearly recurring day or time span. Example: 5/1 or 12/24 to 12/26.

A **WeekProfileInterval** applies to single or several days in succession of a week. Example: Saturday or Monday to Friday.

A **DayProfileInterval** specifiies certain time spans during a day. Example: 8:00 to 5.00

A **ShiftProfile** designates a time span within the specified unit **vcDay, vcHours, vcMinute** or v**cSeconds** without refererring to a date. Example: 4 hours.

Properties

- CalendarProfileName
- DayInEndMonth
- DayInStartMonth
- EndDateTime
- EndMonth
- EndTime
- EndWeekday
- Name
- Specification
- StartDateTime
- StartMonth
- StartTime
- StartWeekday
- Type

Methods

- PutInOrderAfter

Properties

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

EndDateTime

Property of VcInterval

This property returns or sets the end date and time of this interval object (for profiles of the type **vcCalendar** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	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	End month of interval
	Possible Values:	
	.vcApril 4	**April**
	.vcAugust 8	**August**
	.vcDecember 12	**December**
	.vcFebruary 2	**February**
	.vcJanuary 1	**Januar**
	.vcJuly 7	**July**
	.vcJune 6	**une**
	.vcMarch 3	**March**
	.vcMay 5	**May**
	.vcNovember 11	**November**
	.vcOktober 10	**October**
	.vcSeptember 9	**September**

EndTime

Property of VcInterval

This property returns or sets the end time of this interval object (for profiles of the type **vcDayProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	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 **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	VcWeekday	Last weekday of interval
	Possible Values:	
	.vcFriday 5	Week day **Friday**
	.vcMonday 1	Week day **Monday**
	.vcSaturday 6	Week day **Saturday**
	.vcSunday 7	Week day **Sunday**
	.vcThursday 4	Week day **Thursday**
	.vcTuesday 2	Week day **Tuesday**
	.vcWednesday 3	Week day **Wednesday**

Name

Read Only Property of VcInterval

This property lets you retrieve the name of an interval. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	System.String	Name of the interval

Specification

Read Only Property of VcInterval

This property lets you retrieve the specification of an interval. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create an interval by the method **VcInterval-Collection.AddBySpecification**.

	Data Type	Explanation
Property value	System.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	Start month of interval
	Possible Values:	
	.vcApril 4	**April**
	.vcAugust 8	**August**
	.vcDecember 12	**December**
	.vcFebruary 2	**February**
	.vcJanuary 1	**Januar**
	.vcJuly 7	**July**
	.vcJune 6	**une**
	.vcMarch 3	**March**
	.vcMay 5	**May**
	.vcNovember 11	**November**
	.vcOktober 10	**October**
	.vcSeptember 9	**September**

StartTime

Property of VcInterval

This property returns or sets the start time of this interval object (for profiles of the type **vcDayProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	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	Start weekday of interval
	Possible Values:	
	.vcFriday 5	Week day **Friday**
	.vcMonday 1	Week day **Monday**
	.vcSaturday 6	Week day **Saturday**
	.vcSunday 7	Week day **Sunday**
	.vcThursday 4	Week day **Thursday**
	.vcTuesday 2	Week day **Tuesday**
	.vcWednesday 3	Week day **Wednesday**

Type

Read Only Property of VcInterval

This property lets you enquire the type of the interval. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	VcIntervalType	Type of the interval

Methods

PutInOrderAfter

Method of VcInterval

This method lets you set the interval behind an interval specified by name, within the IntervalCollection. If you set the name to "", the interval will be put in the first position. The order of the intervals within the collection determines the order by which they apply to the calendars.

	Data Type	Explanation
Parameter: refName	System.String	Name of the interval behind which the current interval is to be put.
Return value	Void	

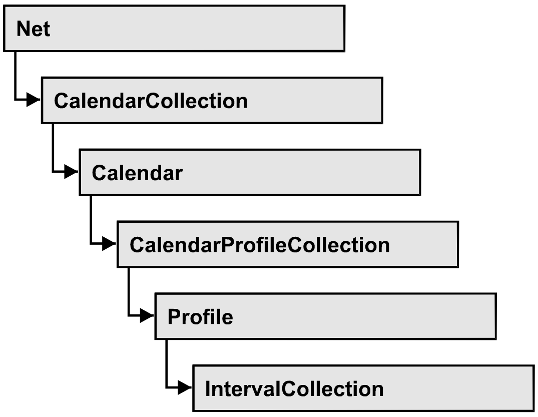
Example Code VB.NET

```
Dim intvlCltn As VcIntervalCollection
Dim intvl1 As VcInterval
Dim intvl2 As VcInterval

intvlCltn = VcGantt1.IntervalCollection()
intvl1 = intvlCltn.Add("intvl1")
intvl2 = intvlCltn.Add("intvl2")
intvl1.PutInOrderAfter("intvl2")
intvlCltn.Update()
```

Example Code C#

```
VcIntervalCollection intvlCltn = vcGantt1.IntervalCollection;
VcInterval intvl1 = intvlCltn.Add("intvl1");
VcInterval intvl2 = intvlCltn.Add("intvl2");
intvl1.PutInOrderAfter("intvl2");
intvlCltn.Update();
```

7.25 VcIntervalCollection



The VcIntervalCollection object contains all intervals available. You can access all objects in an iterative loop by **For Each Interval In BoxFormatCollection** or by the methods **First...** and **Next....** You can access a single interval by the methods **IntervalByName** and **ntervalByIndex**. The number of intervals in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the intervals in the corresponding way.

Properties

- 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
Parameter:		
⇨ intervalName	System.String	Interval name
Return value	VcInterval	New interval object

AddBySpecification

Method of VcIntervalCollection

This method lets you create an interval by using an interval specification. This way of creating allows interval objects to become persistent. The specification of an interval can be saved and re-loaded (see VcInterval property **Specification**). In a subsequent session the interval can be created again from the specification including its former name.

	Data Type	Explanation
Parameter:		
⇨ Specification	System.String	Interval specification
Return value	VcInterval	New Interval object

Copy

Method of VcIntervalCollection

By this method you can copy an interval. If the interval that is to be copied exists, and if the name for the new interval does not yet exist, the new interval object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ intervalName	System.String	Name of the interval to be copied
⇨ newIntervalName	System.String	Name of the new interval
Return value	VcInterval	interval object

FirstInterval

Method of VcIntervalCollection

This method can be used to access the initial value, i.e. the first interval of an interval collection, and then to continue in a forward iteration loop by the method **NextInterval** for the intervals following. If there is no interval in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcInterval	First interval object

IntervalByIndex

Method of VcIntervalCollection

This method lets you access an interval by its index. If no interval of the specified index does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ Index	System.Int16	Index of the interval
Return value	VcInterval	Interval object returned

IntervalByName

Method of VcIntervalCollection

By this method you can retrieve an interval by its name. If no interval of the specified name does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ intervalName	System.String	Name of the interval object
Return value	VcInterval	interval object returned

NextInterval

Method of VcIntervalCollection

This method can be used in a forward iteration loop to retrieve subsequent intervals from an interval collection after initializing the loop by the method **FirstInterval**. If there is no interval left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcInterval	Subsequent interval object

Remove

Method of VcIntervalCollection

This method lets you delete an interval. If the interval is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ intervalName	System.String	interval name
Return value	System.Boolean	interval deleted (True)/not deleted (False)

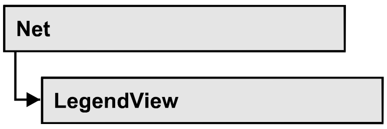
Update

Method of VcIntervalCollection

This method lets you update an interval collection after having modified it.

	Data Type	Explanation
Return value	System.Boolean	update successful (True)/ not successful (False)

7.26 VcLegendView



An object of the type **VcLegendView** designates the legend view window.

Properties

- Border
- Height
- HeightActualValue
- Left
- LeftActualValue
- ScrollBarMode
- Top
- TopActualValue
- Visible
- Width
- WidthActualValue
- WindowMode

Methods

- Update

Properties

Border

Property of VcLegendView

This property lets you set or retrieve whether the world view has a frame (not in **vcPopupWindow** mode). he color of the frame is **Color.Black**. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Boolean	Legend view with a border line (True)/without border line (False) **Default value:** True

Example Code VB.NET

```
VcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed
VcNet1.LegendView.Border = True
```

Example Code C#

```
vcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed;
vcNet1.LegendView.Border = true;
```

Height

Property of VcLegendView

This property lets you retrieve the vertical extension of the legend view. It can also be set in the modes **vcFixedAtTop** and **vcFixedAtBottom**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Height of the legend view **Default value:** 100

Example Code VB.NET

```
VcNet1.LegendView.Height = 100
```

Example Code C#

```
vcNet1.LegendView.Height = 100;
```

HeightActualValue

Read Only Property of VcLegendView

This property lets you retrieve the vertical extension of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

	Data Type	Explanation
Property value	System.Int32	Actual height of the legend view {0, ...}

Example Code VB.NET

```
VcNet1.LegendView.Height = 300
```

Example Code C#

```
vcNet1.LegendView.Height = 100;
```

Left

Property of VcLegendView

This property lets you retrieve the left position of the world view. It can also be set in the modes **vcNotFixed** and **vcPopupWindow**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Left position of the legend view **Default value:** 0

Example Code VB.NET

```
VcNet1.LegendView.Left = 200
```

Example Code C#

```
vcNet1.LegendView.Left = 200;
```

LeftActualValue

Read Only Property of VcLegendView

This property lets you retrieve the left position of the legend view which actually ist displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

	Data Type	Explanation
Property value	System.Int32	Actual left position of the legend view {0, ...}

Example Code VB.NET

```
VcNet1.LegendView.LeftActualValue = 150
```

Example Code C#

```
vcNet1.LegendView.LeftActualValue = 150;
```

ScrollBarMode

Property of VcLegendView

This property lets you set or retrieve the scroll bar mode of the world view. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	VcLegendViewScrollBarMode	Scrollbarmode **Default value:** NoScrollBar
	Possible Values:	
	.vcAutomaticScrollBar 3	Display of a horizontal or vertical scrollbar if required.
	.vcHorizontalScrollBar 1	Display of a horizontal scrollbar if required.
	.vcNoScrollBar 0	The chart is always displayed completely without scrollbars.
	.vcVerticalScrollBar 2	Display of a vertical scrollbar if required.

Example Code VB.NET

```
VcNet1.LegendView.ScrollBarMode = vcAutomaticScrollbar
```

Example Code C#

```
vcNet1.LegendView.ScrollBarMode = vcAutomaticScrollBar;
```

Top

Property of VcLegendView

This property lets you retrieve the top position of the legend view. It can also be set in the modes **vcNotFixed** and **vcPopupWindow**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Top position of the legend view **Default value:** 0

Example Code VB.NET

```
VcNet1.LegendView.Top = 20
```

Example Code C#

```
vcNet1.LegendView.Top = 20;
```

TopActualValue

Read Only Property of VcLegendView

This property lets you enquire the top position of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

	Data Type	Explanation
Property value	System.Int32	Actual top position of the legend view {0, ...}

Example Code VB.NET

```
VcNet1.LegendView.TopActualValue = 40
```

Example Code C#

```
vcNet1.LegendView.TopActualValue = 40;
```

Visible

Property of VcLegendView

This property lets you enquire/set whether the legend view is visible or not. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Boolean	Legend view visible (True)/not visible (False) **Default value:** False

Example Code VB.NET

```
VcNet1.LegendView.Visible = True
```

Example Code C#

```
vcNet1.LegendView.Visible = true;
```

Width

Property of VcLegendView

This property lets you retrieve the horizontal extent of the world view. It can also be set in the modes **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Horizontal extension of the legend view **Default value:** 100

Example Code VB.NET

```
VcNet1.LegendView.Width = 200
```

Example Code C#

```
vcNet1.LegendView.Width = 200;
```

WidthActualValue

Read Only Property of VcLegendView

This property lets you retrieve the horizontal extent of the world view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or width is preset.

	Data Type	Explanation
Property value	System.Int32	Actual horizontal extension of the legend view {0, ...}

Example Code VB.NET

```
VcNet1.LegendView.WidthActualValue = 600
```

Example Code C#

```
vcNet1.LegendView.WidthActualValue = 600;
```

WindowMode

Property of VcLegendView

This property lets you set or retrieve the legend view mode. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	VcLegendViewWindowMode	Mode of the legend view **Default value:** vcPopupWindow
	Possible Values:	

.vcFixedAtBottom 4	The Legend view is displayed on the bottom of the VARCHART .NET control window. Then the height can be specified, whereas the position and the width are fixed.
.vcFixedAtLeft 1	The Legend view is displayed on the left side of the VARCHART .NET control window. Then the width can be specified, whereas the position and the height are fixed.
.vcFixedAtRight 2	The Legend view is displayed on the right side of the VARCHART .NET control window. Then the width can be specified, whereas the position and the height are fixed.
.vcFixedAtTop 3	The Legend view is displayed on the top of the VARCHART .NET control window. Then the height can be specified, whereas the position and the width are fixed.
.vcNotFixed 5	The Legend view is a child window of the current parent window of the VARCHART .NET control. It can be positioned at any position with any extension. The parent window can be modified via the property **VcLegendView.ParentHWnd**.
.vcPopupWindow 6	The Legend view is a popup window with its own frame. The user can modify its position and extension, open it via the default context menu, and close it via the **Close** button in the frame.

Example Code VB.NET

```
VcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed
```

Example Code C#

```
vcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed;
```

Methods

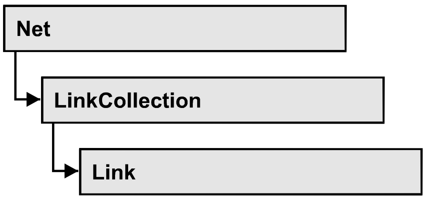
Update

Method of VcLegendView

This method lets you update the legend.

	Data Type	Explanation

7.27 VcLink



A VcLink object represents the logical and graphical link between two nodes. What a node looks like is defined by those LinkAppearance objects the filters of which match the link data. You can generate links either interactively or by the **InsertLinkRecord** method of the **VcNet** object.

Properties

- AllData
- DataField
- ID
- Marked
- 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 = VcNet1.LinkCollection
link = linkCltn.FirstLink
allDataOfLink = link.AllData
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.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
Parameter:		
⇨ index	System.Int16	Index of the data field
Property value	System.Object	Content of data field

Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim message As String

linkCltn = VcNet1.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 = vcNet1.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

Marked

Read Only Property of VcLink

This property lets you set/retrieve whether this link is marked.

	Data Type	Explanation
Property value	Boolean	Link is marked (True)/not marked (False)

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 = VcNet1.LinkCollection
link = linkCltn.FirstLink
node = link.PredecessorNode
nodeName = node.DataField(1)
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.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 = VcNet1.LinkCollection
link = linkCltn.FirstLink
node = link.SuccessorNode
nodeName = node.DataField(1)
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.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
Return value	System.Boolean	Link was/was not successfully deleted

Example Code VB.NET

```
Private Sub VcNet1_VcLinksRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksRightClicking
  Dim message As String
  message = "Delete link: " + e.LinkCollection.FirstLink.AllData

  If MsgBox(message, MsgBoxStyle.OKCancel, "Delete link") = MsgBoxResult.OK Then
      e.LinkCollection.FirstLink.Delete()
  End If
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

Example Code C#

```
private void vcNet1_VcLinksRightClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
   {
   string message = "Delete link: " + e.LinkCollection.FirstLink().AllData;
   DialogResult retVal = MessageBox.Show(message, "Deleting link",
MessageBoxButtons.OKCancel);
   if (retVal ==  DialogResult.OK)
      e.LinkCollection.FirstLink().Delete();
   else
      e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

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
Parameter: ⇨ index	System.Int16	Index of data field that holds the key
Return value	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
Return value	System.Boolean	Link was/was not successfully updated

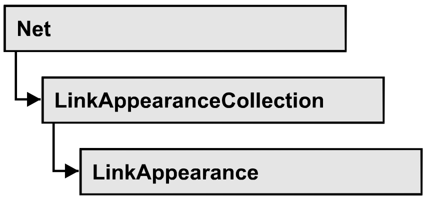
Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
link.DataField(2) = 10
link.Update()
```

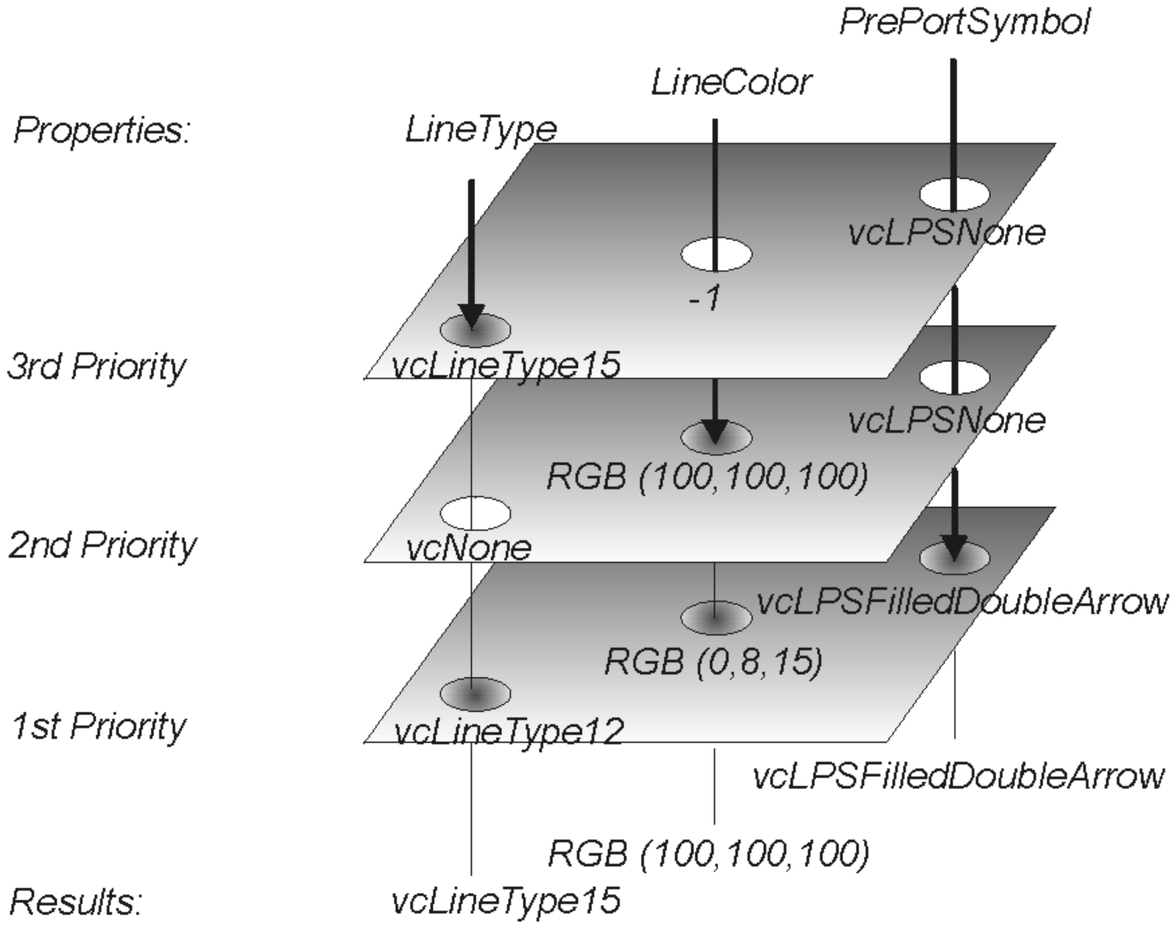
Example Code C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();

link.set_DataField(2, 10);
link.Update();
```

7.28 VcLinkAppearance



A VcLinkAppearance object defines the appearance of a link, if the link data comply with the conditions defined by the filters assigned. Different link appearances can be set on the **Link** property page in the **Appearances** table. The sketch below shows the influence of the LinkAppearances and their priorities on the appearances of links. LinkAppearances matching the links are displayed in descending order of priority. A property that has not been set to a LinkAppearance object will give way to a property of a LinkAppearance object that is next in the descending hierarchy.



Properties

- FilterName
- FormatName

- LineColor
- LineThickness
- LineType
- Name
- PredecessorPortSymbol
- RoutingType
- Specification
- 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

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim filterOfLinkApp As String

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
filterOfLinkApp = linkAppearance.FilterName
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
string filterOfLinkApp = linkAppearance.FilterName;
```

FormatName

Property of VcLinkAppearance

This property lets you set or retrieve a format of the linkAppearance object. When set to **Nothing**, the property will give way to the property of a linkAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **Nothing** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	System.String	Name of a LinkFormat object or empty string

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
MsgBox(nodeAppearance.FormatName)
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
MessageBox.Show(nodeAppearance.FormatName);
```

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 ({0...255},{0...255},{0...255})	RGB color values

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineColor = Color.Blue
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.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
Property value	System.Int32	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined on property page

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
linkAppearance.LineThickness = 4
```

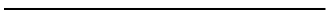
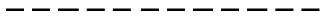
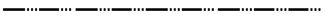
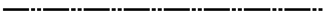
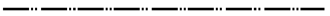
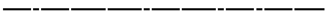
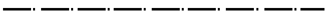
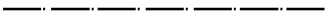
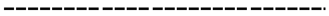
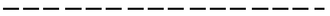
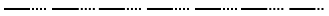
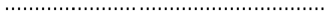
Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Standard");
linkAppearance.LineThickness = 4;
```

LineType

Property of VcLinkAppearance

This property lets you set or retrieve the line type of a LinkAppearance object. This property can be also be set in the **Line attributes** dialog, that you can get to by the **Link** property page.

	Data Type	Explanation
Property value	VcLineType	Line type **Default value:** vcSolid
	Possible Values: .vcDashed 4	Line dashed
	.vcDashed 4	Line dashed
	.vcDashedDotted 5	Line dashed-dotted
	.vcDashedDotted 5	Line dashed-dotted
	.vcDotted 3	Line dotted
	.vcDotted 3	Line dotted
	.vcLineType0 100	Line Type 0 
	.vcLineType1 101	Line Type 1 
	.vcLineType10 110	Line Type 10 
	.vcLineType11 111	Line Type 11 
	.vcLineType12 112	Line Type 12 
	.vcLineType13 113	Line Type 13 
	.vcLineType14 114	Line Type 14 
	.vcLineType15 115	Line Type 15 
	.vcLineType16 116	Line Type 16 
	.vcLineType17 117	Line Type 17 
	.vcLineType18 118	Line Type 18 
	.vcLineType2 102	Line Type 2 

	.vcLineType3 103	Line Type 3 ---
	.vcLineType4 104	Line Type 4 ---------------------------------
	.vcLineType5 105	Line Type 5 — — — — — — — — — —
	.vcLineType6 106	Line Type 6 — — — — — — — —
	.vcLineType7 107	Line Type 7 - - - - - - - - - - - - - - - - - -
	.vcLineType8 108	Line Type 8 – – – – – – – – – – – – – –
	.vcLineType9 109	Line Type 9 —---—---—---—---—---—---—
	.vcNone 1	No line type assigned
	.vcNone 1	No line type
	.vcNotSet -1	No line type assigned
	.vcSolid 2	Line solid
	.vcSolid 2	Line solid

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineType = 5
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.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 of the link appearance

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
nameLinkApp = linkAppearance.Name
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
string nameLinkApp = linkAppearance.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 **Default value:** vcLPSNone

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSDoubleSemiCircle
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.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
Property value	VcRoutingType	Routing type **Default value:** vcLRTOrthogonal

Specification

Read Only Property of VcLinkAppearance

This property lets you retrieve the specification of a link appearance. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a link appearance by the method **VcLinkAppearanceCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the link appearance

Example Code C#

```
VcBoxCollection boxCltn =vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

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 **Default value:** vcLSSNone

Example Code VB.NET

```
VcLinkAppearanceCollection linkAppearanceCltn = VcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.SuccessorPortSymbol =
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.SuccessorPortSymbol =
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

Visible

Property of VcLinkAppearance

This property lets you set or retrieve whether the link is to be visible or not, taking no effect, however, on the phantom lines for links while dragging.

This property can also be set on the **Links** property page, but here also applying to the phantom lines.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active **Default value:** True

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.Visible = False
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.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 appearence will be put in the first position. The order of the link appearances within the collection determines the order by which they apply to the links.

	Data Type	Explanation
Parameter:		
refLinkAppearanceName	System.String	Name of the link appearance behind which the current link appearance is to be put.

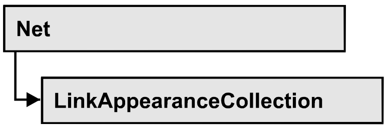
Example Code VB.NET

```
Dim linkAppCltn As VcLinkAppearanceCollection
Dim linkApp1 As VcLinkAppearance
Dim linkApp2 As VcLinkAppearance

linkAppCltn = VcGantt1.LinkAppearanceCollection()
linkApp1 = linkAppCltn.Add("linkApp1")
linkApp2 = linkAppCltn.Add("linkApp2")
linkApp1.PutInOrderAfter("linkApp2")
linkAppCltn.Update()
```

Example Code C#

```
VcLinkAppearanceCollection linkAppCltn = vcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkApp1 = linkAppCltn.Add("linkApp1");
VcLinkAppearance linkApp2 = linkAppCltn.Add("linkApp2");
linkApp1.PutInOrderAfter("linkApp2");
linkAppCltn.Update();
```

7.29 VcLinkAppearanceCollection



An object of the type VcLinkAppearanceCollection automatically contains all available link appearances. You can access all objects in an iterative loop by **For Each linkAppearance In LinkAppearanceCollection** or by the methods **First...** and **Next...**. You can access a single line format by the methods **LinkAppearanceByName** and **LinkAppearandeByIndex**. The number of link appearances in the collection object can be retrieved by the property **Count**.

Properties

- 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 = VcNet1.LinkAppearanceCollection
numberOfLinkAppearance = linkAppearanceCltn.Count
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
int numberOfLinkAppearance = linkAppearanceCltn.Count;
```

Methods

Add

Method of VcLinkAppearanceCollection

By this method you can create a new linke 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
Parameter: ⇨ newName	System.String	Link appearance name
Return value	VcLinkAppearance	New link appearance object

Example Code VB.NET

```
newLinkAppearance = VcNet1.LinkAppearanceCollection.Add("linkapp1")
```

Example Code C#

```
newLinkAppearance = vcNet1.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
Parameter:		
⇨ linkAppearanceSpecification	System.String	Link appearance specification
Return value	VcLinkAppearance	New link appearance object

Copy

Method of VcLinkAppearanceCollection

By this method you can copy a link appearance. When the link appearance has come into existence and if the name for the new link appearance did not yet exist, the new link appearance object will be returned. Otherwise "Nothing" (Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	System.String	Name of the link appearance to be copied
⇨ newName	System.String	Name of the new link appearance
Return value	VcLinkAppearance	Link appearance object

FirstLinkAppearance

Method of VcLinkAppearanceCollection

This method can be used to access the initial value, i.e. the first link appearance of a link appearance collection and then to continue in a forward iteration loop by the method **NextLinkAppearance** for the link appearances following. If there is no link appearance in the link appearance collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLinkAppearance	First linkAppearance object

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.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
Return value	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
Parameter: ⇨ index	System.Int16	Index of the link appearance object
Return value	VcLinkAppearance	LinkAppearance object returned

LinkAppearanceByName

Method of VcLinkAppearanceCollection

This method retrieves a link appearance object by its name.

	Data Type	Explanation
Parameter:		
⇨ linkAppearanceName	System.String	Name of the link appearance object
Return value	VcLinkAppearance	LinkAppearance object

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.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
Return value	VcLinkAppearance	Succeeding linkAppearance object

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.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 = vcNet1.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
Parameter: ⇨ name	System.String	Name of the link appearance
Return value	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
Return value	System.Boolean	Link appearance collection was/was not successfully updated

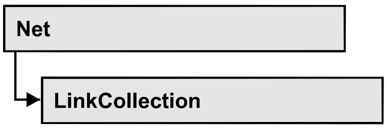
Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0)
linkAppearanceCltn.Remove(linkAppearance.Name)
linkAppearanceCltn.Update()
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0);
linkAppearanceCltn.Remove(linkAppearance.Name);
linkAppearanceCltn.Update();
```

7.30 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 = VcNet1.LinkCollection
numberOfLinks = linkCltn.Count
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.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 = VcNet1.LinkCollection
link = linkCltn.FirstLink
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.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
Return value	VcLink	Succeeding link

Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
While Not link Is Nothing
   ListBox1.Items.Add(link.AllData)
   link = linkCltn.NextLink
End While
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.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
Parameter:		
⇨ selectionType	VcSelectionType	Links to be selected
	Possible Values:	
	.vcAll 0	All objects in the diagram will be selected
	.vcAllLinksCausingCycles 7	If this selection type is chosen, the link collection will contain all links that cause the existence of cycles. If these links are deleted, cycles will cede to exist in this chart.
	.vcAllLinksInCycles 6	If this selection type is chosen, the link collection will contain all links that participate in forming cycles. Cycles are chains of nodes and links of which the beginning and end join.
	.vcAllVisible 1	All visible objects will be selected
	.vcMarked 2	All marked objects will be selected
Return value	System.Int32	Number of links selected

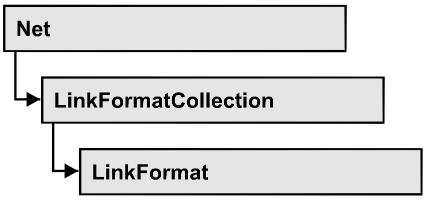
Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
linkCltn = VcNet1.LinkCollection
linkCltn.SelectGroups (vcAllMarked)
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
linkCltn.SelectGroups (vcAllMarked);
```

7.31 VcLinkFormat



An object of the type VcLinkFormat defines the contents and the format of links. At run time, link formats are administered and edited in the **Administrate Node Formats** dialog box that you can get to by the **Links** property page.

Properties

- FormatField
- FormatFieldCount
- Name
- Specification

Methods

- CopyFormatField
- GetEnumerator
- RemoveFormatField

Properties

FormatField

Read Only Property of VcLinkFormat

This property gives access to a VcLinkFormatField object by its index. The index has to be in the range from 0 to FormatFieldCount-1.

	Data Type	**Explanation**
Parameter:		
index	System.Int16	Index of the link format field
Property value	VcNodeFormatField	Link format field

FormatFieldCount

Read Only Property of VcLinkFormat

This property allows to determine the number of fields in a link format.

	Data Type	Explanation
Property value	System.Int16	Number of fields of the link format

Example Code VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.FormatFieldCount)
```

Example Code C#

```
      VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.FormatFieldCount.ToString());
```

Name

Property of VcLinkFormat

This property lets you set or retrieve the name of the link format.

	Data Type	Explanation
Property value	System.String	Name of the link format

Example Code VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.Name)
```

Example Code C#

```
VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.Name);
```

Specification

Read Only Property of VcLinkFormat

This property lets you retrieve the specification of a link 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 link format by the method **VcLinkFormatCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the link format

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

Methods

CopyFormatField

Method of VcLinkFormat

This method allows to copy a node format field. The new VcLinkFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	VcFormatFieldPosition	Position of the new link format field
	Possible Values: .vcAbove 1 .vcBelow 3 .vcLeftOf 0 .vcOutsideAbove 9 .vcOutsideBelow 11 .vcOutsideLeftOf 8 .vcOutsideRightOf 12 .vcRightOf 4	 above below left of outside, above outside, below outside, left of outside, right of right of
⇨ refIndex	System.Int16	Index of the reference link format field
Return value	VcLinkFormatField	Link format field object generated

GetEnumerator

Method of VcLinkFormat

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 node format fields included.

	Data Type	Explanation
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim format As VcNodeFormat
For Each format In VcNet1.NodeFormatCollection
    Debug.Write(format.Name)
Next
```

Example Code C#

```
foreach (VcNodeFormat format in vcNet1.NodeFormatCollection)
     Console.Write(format.Name);
```

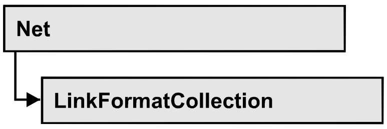
RemoveFormatField

Method of VcLinkFormat

This method lets you remove a node format field by its index. After that, the program will set all link format field indexes newly in order to number them consecutively.

	Data Type	Explanation
Parameter:		
⇨ index	System.Int16	Index of the link format field to be deleted

7.32 VcLinkFormatCollection



An object of the type VcLinkFormatCollection automatically contains all link formats available to a link. You can access all objects in an iterative loop by **For Each format InLink FormatCollection** or by the methods **First...** and **Next...**. You can retrieve a single link format by the method **FormatByName**. The property **Count** will return the number of link formats contained in the collection. By using you can retrieve all link formats.

Properties

- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- GetEnumerator
- NextFormat
- Remove

Properties

Count

Read Only Property of VcLinkFormatCollection

This property lets you retrieve the number of link formats in the link format collection.

	Data Type	Explanation
Property value	System.Int32	Number of link formats

Example Code VB.NET

```
Dim formatCltn As VcLinkFormatCollection
Dim numberOfFormats As Integer

formatCltn = VcNet1.LinkFormatCollection
numberOfFormats = formatCltn.Count
```

Example Code C#

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
int numberOfFormats = formatCltn.Count;
```

Methods

Add

Method of VcLinkFormatCollection

By this method you can create a link format as a member of the LinkFormatCollection. If the name has not been used before, the new VcLinkFormat object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

The link format has the following properties by default:

- only one field
- WidthOfExteriorSurrounding: 3 mm

The field has the following properties:

- Type: vcFFTText
- TextDataFieldIndex: IDMinimumWidth specified on the **General** property page: 3000
- Alignment: vcFFACenter
- BackColor: -1 (transparent)
- TextFontColor: RGB(0,0,0) (black)
- TextFont: Arial, 10, normal
- LeftMargin, RightMargin, TopMargin, BottomMargin: 0,3 mm

- MinimumTextLineCount, MaximumTextLineCount: 1

	Data Type	Explanation
Parameter:		
⇨ newName	System.String	Name of the link format
Return value	VcLinkFormat	Link format object

Example Code VB.NET

```
newLinkFormat = VcNet1.LinkFormatCollection.Add("linkformat1")
```

Example Code C#

```
newLinkFormat = vcNet1.LinkFormatCollection.Add("linkformat1");
```

AddBySpecification

Method of VcLinkFormatCollection

This method lets you create a link format by using link format specification. This way of creating allows link format objects to become persistent. The specification of a link format can be saved and re-loaded (see VcNodeFormat property **Specification**). In a subsequent session the link format can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ formatSpecification	System.String	Link format specification
Return value	VcLinkFormat	New link format object

Copy

Method of VcLinkFormatCollection

By this method you can copy a link format. If the link format that is to be copied exists, and if the name for the new link format does not yet exist, the new link format object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	System.String	Name of the link format to be copied
⇨ newName	System.String	Name of the new link format

Return value	VcLinkFormat	Link format object

FirstFormat

Method of VcLinkFormatCollection

This method can be used to access the initial value, i.e. the first link format of a link format collection and then to continue in a forward iteration loop by the method **NextFormat** for the formats following. If there is no link format in the link format collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLinkFormat	First link format

Example Code VB.NET

```
Dim format As VcLinkFormat

format = VcNet1.LinkFormatCollection.FirstFormat
```

Example Code C#

```
VcLinkFormat format = vcNet1.LinkFormatCollection.FirstFormat;
```

FormatByIndex

Method of VcLinkFormatCollection

This method lets you access a format by its index. If a format 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 link format
Return value	VcLinkFormat	Link format object returned

Example Code VB.NET

```
Dim formatCltn As VcLinkFormatCollection

formatCltn = VcNet1.LinkFormatCollection
format = formatCltn.FormatByIndex(0)
format.WidthOfExteriorSurrounding = 2
```

Example Code C#

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
VcLinkFormat format = formatCltn.FormatByIndex(0);
format.WidthOfExteriorSurrounding = 2;
```

FormatByName

Method of VcLinkFormatCollection

By this method you can retrieve a link format by its name. If a link format of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ formatName	System.String	Name of the link format
Return value	VcLinkFormat	Link format

Example Code VB.NET

```
Dim formatCltn As VcLinkFormatCollection
Dim format As VcLinkFormat

formatCltn = VcNet1.LinkFormatCollection
format = formatCltn.FormatByName("Standard")
```

Example Code C#

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
VcLinkFormat format = formatCltn.FormatByName("Standard");
```

GetEnumerator

Method of VcLinkFormatCollection

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 formats included.

	Data Type	Explanation
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim format As VcLinkFormat

For Each format In VcNet1.LinkFormatCollection
   Debug.Write( format.Name)
Next
```

Example Code C#

```
foreach (VcLinkFormat format In vcNet1.LinkFormatCollection)
   Console.Write(format.Name);
```

NextFormat

Method of VcLinkFormatCollection

This method can be used in a forward iteration loop to retrieve subsequent link formats from a link 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	VcLinkFormat	Subsequent link format

Example Code VB.NET

```
Dim formatCltn As VcLinkFormatCollection
Dim format As VcLinkFormat

formatCltn = VcNet1.LinkFormatCollection
format = formatCltn.Firstformat

While Not format Is Nothing
     ListBox1.Items.Add(format.Name)
      format = formatCltn.NextFormat
End While
```

Example Code C#

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
VcLinkFormat format = formatCltn.FirstFormat;

while (format != null)
   {
   listBox1.Items.Add(format.Name);
   format = formatCltn.NextFormat();
   }
```

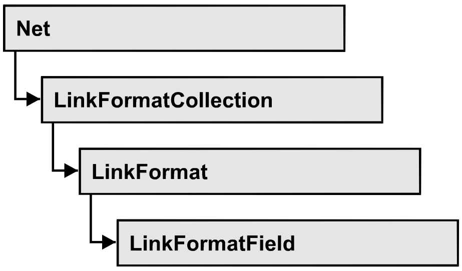
Remove

Method of VcLinkFormatCollection

This method lets you delete a link format. If the link format is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter: ⇨ name	System.String	Link format name
Return value	System.Boolean	Link format deleted (True)/not deleted (False)

7.33 VcLinkFormatField



An object of the type **VcLinkFormat** represents a field of a VcLinkFormat object. A link format field does not have a name as many other objects do, but it has an index that defines its position in the link format.

Properties

- Alignment
- ConstantText
- FormatName
- Index
- MinimumWidth
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextLineCount

Properties

Alignment

Read Only Property of VcLinkFormatField

This property lets you set or retrieve the alignment of the content of the link format field.

	Data Type	Explanation
Property value	VcFormatFieldAlignment	Alignment of the field content
	Possible Values:	
	.vcFFABottom 28	Bottom
	.vcFFABottomLeft 27	Bottom left
	.vcFFABottomRight 29	Bottom right
	.vcFFACenter 25	Center
	.vcFFALeft 24	Left
	.vcFFARight 26	Right

	.vcFFATop 22	Top
	.vcFFATopLeft 21	Top left
	.vcFFATopRight 23	Top right

ConstantText

Read Only Property of VcLinkFormatField

This property allows the link format field to display a constant text, if the link 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 VcLinkFormatField

This property lets you retrieve the name of the link format to which this link format field belongs.

	Data Type	Explanation
Property value	System.String	Name of the line format object

Index

Read Only Property of VcLinkFormatField

This property lets you enquire the index of the link format field in the corresponding link format.

	Data Type	Explanation
Property value	System.Int16	Index of the table format field

MinimumWidth

Read Only Property of VcLinkFormatField

This property lets you set or retrieve the minimum width of the link 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.Integer	Minimum width of the layer format field in mm 0 ... 99

TextDataFieldIndex

Read Only Property of VcLinkFormatField

*only for the type **vcFFTText***: This property lets you set or retrieve the index of the data field, the content of which is to be displayed in the link format field. If its value equals -1, the content of the property **ConstantText** will be returned.

	Data Type	Explanation
Property value	System.Int32	Index of the data field

TextFont

Read Only Property of VcLinkFormatField

This property lets you set or retrieve the font color of the link 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 in dependence of the data.

	Data Type	Explanation
Property value	System.DrawingFont	Font type of the table format

TextFontColor

Read Only Property of VcLinkFormatField

This property lets you set or retrieve the font color of the link 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.DrawingColor	Font color of the table format **Default value:** -1

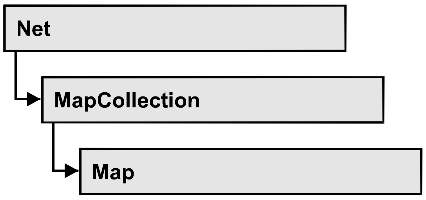
TextLineCount

Read Only Property of VcLinkFormatField

This property lets you set or retrieve the number of lines, if the size of the annotation field allows for more than one line.

	Data Type	**Explanation**
Property value	System.Integer	Number of lines

7.34 VcMap



Maps define certain properties of nodes by data field entries, for example their background color which is based on the data of the node record.

In a map you can specify 150 map entries at maximum. By the call **For Each mapEntry In Map** you can retrieve all data field entries in an iterative loop.

Properties

- ConsiderFilterEntries
- Count
- Name
- Specification
- Type

Methods

- CreateEntry
- DeleteEntry
- FirstMapEntry
- GetMapEntry
- 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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
numberOfEntries = map.Count
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
int numberOfEntries = map.Count;
```

Name

Read Only Property of VcMap

This property lets you retrieve the name of a map.

	Data Type	Explanation
Property value	System.String	Name of the map

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapName As String

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
mapName = map.Name
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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

Example Code VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

Type

Property of VcMap

This property lets you enquire/set the map type.

	Data Type	Explanation
Property value	VcMapType	Map type
	Possible Values:	
	.vcAnyMap 0	**any** (used only for selecting)
	.vcColorMap 1	**Colors**
	.vcFontMap 8	**Fonts**
	.vcGraphicsFileMap 7	**Graphics file**
	.vcMillimeterMap 9	**Millimeters**
	.vcNumberMap 10	**Numbers**
	.vcPatternMap 3	**Patterns**
	.vcTextMap 6	**Text**

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
map.Type = VcMapType.vcPatternMap
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.CreateEntry
mapCltn.Update
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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
Parameter:		
⇨ mapEntry	VcMapEntry	Map entry
Return value	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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
map.DeleteEntry(mapEntry)
mapCltn.Update
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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
Return value	VcMapEntry	First map entry

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)

map = mapCltn.FirstMap
mapEntry = map.FirstMapEntry
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry();
```

GetMapEntry

Method of VcMap

This method returns the corresponding map entry for the given data field value.

	Data Type	Explanation
Return value	System.String	Map entry according to field value

NextMapEntry

Method of VcMap

This method can be used in a forward iteration loop to retrieve subsequent entries (rows) from a map object after initializing the loop by the method **FirstMapEntry**. If there is no map entry left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMapEntry	Succeeding map entry

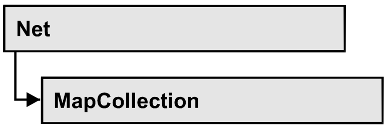
Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.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 = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry()
while (mapEntry != null)
   {
   listBox1.Items.Add(mapEntry.LegendText);
   mapEntry= map.NextMapEntry();
   }
```

7.35 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.

	Data Type	Explanation
Property value	System.Int32	Number of maps

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim numberOfMaps As Integer

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
numberOfMaps = mapCltn.Count
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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
Parameter: ⇨ mapName	System.String	Map name
Return value	VcMap	New map object

Example Code VB.NET

```
newMap = VcNet1.MapCollection.Add("Map1")
```

Example Code C#

```
VcMap newMap = vcNet1.MapCollection.Add("Map1");
```

AddBySpecification

Method of VcMapCollection

This method lets you create a map by using a map specification. This way of creating allows map objects to become persistent. The specification of a map can be saved and re-loaded (see VcMap property **Specification**). In a subsequent session the map can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ specification	System.String	Map specification
Return value	VcMap	New map object

Copy

Method of VcMapCollection

By this method you can copy a map. If the map that is to be copied exists, and if the name for the new map does not yet exist, the new map object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ mapName	System.String	Name of the map to be copied
⇨ newMapName	System.String	Name of the new map
Return value	VcMap	Map object

FirstMap

Method of VcMapCollection

This method can be used to access the initial value, i.e. the first map of a map collection and then to continue in a forward iteration loop by the method **NextMap** for the maps following. If there is no map in the MapCollection, a **none** object will be returned (**Nothing** in Visual Basic). Beforehand, you have to specify a set of maps by the method **SelectMaps**.

	Data Type	Explanation
Return value	VcMap	First map

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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
Parameter: ⇨ mapName	System.String	Name of the map
Return value	VcMap	Map

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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
Return value	VcMap	Succeeding map

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.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 = vcNet1.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
Parameter: ⇨ mapName	System.String	Map name
Return value	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
Parameter: ⇨ selectionType	VcMapType	Map type to be selected
	Possible Values:	
	.vcAnyMap 0	**any** (used only for selecting)
	.vcColorMap 1	**Colors**
	.vcFontMap 8	**Fonts**
	.vcGraphicsFileMap 7	**Graphics file**
	.vcMillimeterMap 9	**Millimeters**
	.vcNumberMap 10	**Numbers**
	.vcPatternMap 3	**Patterns**
	.vcTextMap 6	**Text**
Return value	System.Int32	Number of maps selected

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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
Return value	System.Boolean	Update successful (True)/ not successful (False)

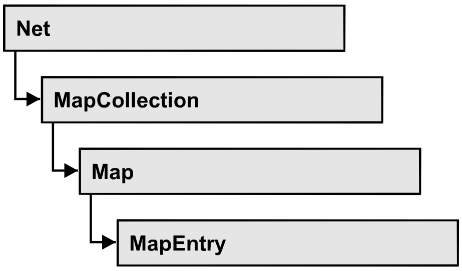
Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.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 = vcNet1.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();
```

7.36 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
- 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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcColorMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
colorOfMapEntry = mapEntry.Color
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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	Font body

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontBodyOfMapEntry As VcFontBody

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontBodyOfMapEntry = VcFontBody.vcBold
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontNameOfMapEntry = "Arial"
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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 he 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 = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontSizeOfMapEntry = 14
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.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:*

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)

- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, probably with EMF included

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	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 = VcNet1.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 = vcNet1.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";
```

Number

Property of VcMapEntry

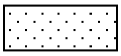
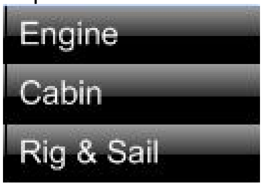
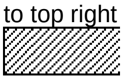
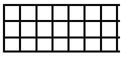
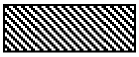
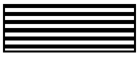
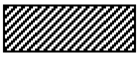
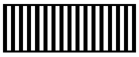
For numeric maps: This property lets you set or retrieve the numeric value of a map entry.

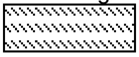
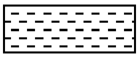
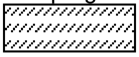
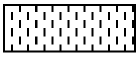
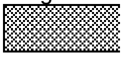
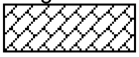
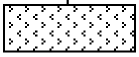
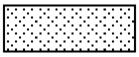
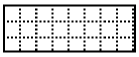
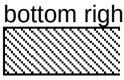
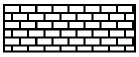
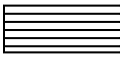
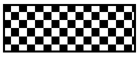
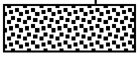
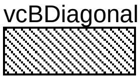
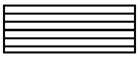
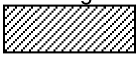
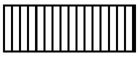
	Data Type	Explanation
Property value	System.Int32	Numeric value

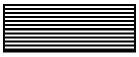
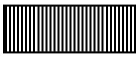
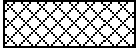
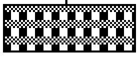
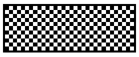
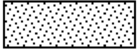
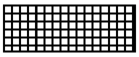
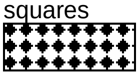
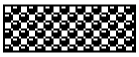
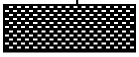
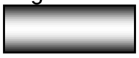
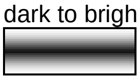
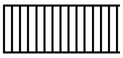
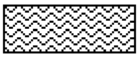
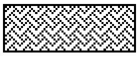
Pattern

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	Pattern type
	Possible Values: .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 

.vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right 
.vcDashedHorizontalPattern 2026	Dashed horizontal lines 
.vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right 
.vcDashedVerticalPattern 2027	Dashed vertical lines 
.vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
.vcDiagonalBrickPattern 2032	Diagonal brick pattern 
.vcDivotPattern 2036	Divot pattern 
.vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
.vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 
.vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right 
.vcHorizontalBrickPattern 2033	Horizontal brick pattern 
.vcHorizontalGradientPattern 52	Horizontal color gradient
.vcHorizontalPattern 3	Horizontal lines 
.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern 
.vcLargeConfettiPattern 2029	Confetti pattern, large 
.vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern 
.vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern 
.vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern 
.vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern 

.vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75% closer than vcHorizontalPattern 
.vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern 
.vcNoPattern 1276	No fill pattern
.vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large 
.vcPlaidPattern 2035	Plaid pattern 
.vcShinglePattern 2039	Diagonal shingle pattern
.vcSmallCheckerBoardPattern 2043	Checkerboard pattern 
.vcSmallConfettiPattern 2028	Confetti pattern 
.vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern 
.vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares 
.vcSpherePattern 2041	Checkerboard of spheres 
.vcTrellisPattern 2040	Trellis pattern 
.vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright
.vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
.vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
.vcVerticalGradientPattern 62	Vertical color gradient
.vcVerticalPattern 2	Vertical lines 
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark
.vcWavePattern 2031	Horizontal 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 vcFDiagonalPattern ▧
.vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern ▨
.vcZigZagPattern 2030	Horizontal zig-zag lines ▒

Example Code 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.vcBDiagonalPattern
```

Example Code C#

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcPatternMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
VcFillPattern pattern = VcFillPattern.vcBDiagonalPattern;
```

7.37 VcNet

Net

An object of the type VcNet is the VARCHART XNet control. You use events to control interactions with the VcNet object. It can be customized by a number of properties and methods to meet your demands.

Properties

- ActiveNodeFilter
- BorderArea
- BoxCollection
- BoxFormatCollection
- CalendarCollection
- CalendarProfileCollection
- CtrlCXVProcessingEnabled
- DataTableCollection
- DateOutputFormat
- DiagramBackgroundColor
- DialogFont
- DoubleOutputFormat
- Enabled
- ExtendedDataTablesEnabled
- FilePath
- FilterCollection
- FontAntiAliasingEnabled
- GroupCollection
- GroupHorizontalMargin
- GroupingActivated
- GroupingDataFieldIndex
- GroupingTitlesFullyVisible
- GroupingType
- GroupInteractionsAllowed
- GroupSortingDataFieldIndex
- GroupSortMode
- GroupTitleDataFieldIndex
- GroupTitlesFileName
- GroupVerticalMargin
- InbuiltMouseCursorWhileDraggingEnabled
- InFlowGroupingActivated

- InFlowGroupingDataFieldIndex
- InFlowGroupSeparationLineColor
- InFlowGroupSeparationLineType
- InFlowGroupTimeInterval
- InFlowGroupTitleDataFieldIndex
- InFlowGroupTitlesBackgroundColor
- InFlowGroupTitlesFileName
- InFlowGroupTitlesFont
- InFlowGroupTitlesVisibleAtBottomOrRight
- InFlowGroupTitlesVisibleAtTopOrLeft
- InFlowGroupTitleTimeFormat
- InFlowGroupVerticalCaptionWidth
- InPlaceEditingAllowed
- InteractionMode
- InterfaceNodesShown
- LegendView
- LinkAnnotationColumnNumberDataFieldIndex
- LinkAnnotationRowNumberDataFieldIndex
- LinkAppearanceCollection
- LinkCollection
- LinkCreationWithDialog
- LinkFormatCollection
- LinkPredecessorDataFieldIndex
- LinksDataTableName
- LinkSuccessorDataFieldIndex
- LinkTypeDataFieldIndex
- MapCollection
- MinimumColumnWidth
- MinimumRowHeight
- MouseProcessingEnabled
- MovingCollapsedClustersAllowed
- NodeAndLinkCreationAllowed
- NodeAppearanceCollection
- NodeCalendarNameDataFieldIndex
- NodeChangeRankToPredecessorRankDataFieldIndex
- NodeCollection
- NodeColumnNumberDataFieldIndex
- NodeCreationWithDialog
- NodeFormatCollection
- NodeRowNumberDataFieldIndex

- NodesDataTableName
- NodesUseCalendars
- NodeToolTipTextDataFieldIndex
- ObliqueTracksOnLinks
- Orientation
- PhantomDrawingWhileDraggingEnabled
- Printer
- RoundedLinkSlantsEnabled
- Scheduler
- ShortenedLinks
- StraightLinkDrawing
- TextEntrySupplyingEventEnabled
- TimeUnit
- ToolTipChangeDuration
- ToolTipDuration
- ToolTipPointerDuration
- ToolTipShowAfterClick
- ToolTipTextSupplyingEventEnabled
- UngroupedNodesAllowed
- ViewXCoordinate
- ViewYCoordinate
- WaitCursorEnabled
- WorldView
- ZoomFactor
- ZoomingPerMouseWheelAllowed

Methods

- Arrange
- Clear
- CompleteViewMode
- CopyNodesIntoClipboard
- CutNodesIntoClipboard
- DeleteLinkRecord
- DeleteNodeRecord
- DetectDataTableFieldName
- DetectDataTableName
- DetectFieldIndex
- DumpConfiguration
- EndLoading
- ExportGraphicsToFileEx

- GetAValueFromARGB
- GetBValueFromARGB
- GetGValueFromARGB
- GetLinkByID
- GetLinkByNodeIDs
- GetNodeByID
- GetRValueFromARGB
- IdentifyFormatField
- IdentifyObjectAt
- ImportConfiguration
- InsertLinkRecord
- InsertNodeRecord
- Load
- MakeARGB
- PasteNodesFromClipboard
- PixelsToRaster
- PrintEx
- PrintToFile
- Reset
- SaveAsEx
- ScheduleProject
- ScrollToNode
- SetImageResource
- ShowAboutDialog
- ShowExportGraphicsDialog
- ShowLinkEditDialog
- ShowNodeEditDialog
- ShowPageSetupDialog
- ShowPrintDialog
- ShowPrinterSetupDialog
- ShowPrintPreviewDialog
- SuspendUpdate
- UpdateLinkRecord
- UpdateNodeRecord
- Zoom
- ZoomOnMarkedNodes

Events

- VcBoxLeftClicking
- VcBoxLeftDoubleClicking

- VcBoxModified
- VcBoxModifying
- VcBoxRightClicking
- VcDataModified
- VcDataRecordCreated
- VcDataRecordCreating
- VcDataRecordDeleted
- VcDataRecordDeleting
- VcDataRecordModified
- VcDataRecordModifying
- VcDataRecordNotFound
- VcDiagramLeftClicking
- VcDiagramLeftDoubleClicking
- VcDiagramRightClicking
- VcDragCompleting
- VcDragStarting
- VcErrorOccurring
- VcFieldSelecting
- VcGiveFeedbackOnNodeCreating
- VcGroupCreated
- VcGroupDeleting
- VcGroupLeftClicking
- VcGroupLeftDoubleClicking
- VcGroupModified
- VcGroupModifying
- VcGroupRightClicking
- VcHelpRequested
- VcInPlaceEditorShowing
- VcLegendViewClosed
- VcLinkCreated
- VcLinkCreating
- VcLinkDeleted
- VcLinkDeleting
- VcLinkModified
- VcLinkModifying
- VcLinksLeftClicking
- VcLinksLeftDoubleClicking
- VcLinksMarked
- VcLinksMarking
- VcLinksRightClicking

- VcMouseDoubleClicking
- VcMouseDown
- VcMouseMove
- VcMouseUp
- VcNodeCreated
- VcNodeCreating
- VcNodeDeleted
- VcNodeDeleting
- VcNodeLeftClicking
- VcNodeLeftDoubleClicking
- VcNodeModifiedEx
- VcNodeModifying
- VcNodeRightClicking
- VcNodesMarked
- VcNodesMarking
- VcStatusLineTextShowing
- VcTextEntrySupplying
- VcToolTipTextSupplying
- VcWorldViewClosed
- VcZoomFactorModified

Properties

ActiveNodeFilter

Property of VcNet

This property lets you set or retrieve a filter that selects the nodes to be displayed.

	Data Type	Explanation
Property value	VcFilter	Filter object **Default value:** Nothing

Example Code VB.NET

```
VcNet1.ActiveNodeFilter = VcNet1.FilterCollection.FilterByName("Milestone")
```

Example Code C#

```
vcNet1.ActiveNodeFilter = vcNet1.FilterCollection.FilterByName("Milestone");
```

BorderArea

Read Only Property of VcNet

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 = VcNet1.BorderArea
```

Example Code C#

```
VcBorderArea borderArea = vcNet1.BorderArea;
```

BoxCollection

Read Only Property of VcNet

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 = VcNet1.BoxCollection
```

Example Code C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
```

BoxFormatCollection

Read Only Property of VcNet

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

CalendarCollection

Read Only Property of VcNet

This property lets you access the calendar collection object, i.e. to the calendars used.

	Data Type	Explanation
Property value	VcCalendarCollection	CalendarCollection object

Example Code VB.NET

```
Dim calendarCltn As VcCalendarCollection
```

```
calendarCltn = VcNet1.CalendarCollection
```

Example Code C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
```

CalendarProfileCollection

Read Only Property of VcNet

This property gives access to the CalenderProfileCollection object that contains all calendar profiles available.

	Data Type	Explanation

CtrlCXVProcessingEnabled

Property of VcNet

This property automatically translates the key combinations <Ctrl>+<C>, <Ctrl>+<X> and <Ctrl>+<V> into the clipboard commands **CopyNodesToClipboard**, **CutNodesToClipboard** and **PasteNodesFromClipboard**, respectively. You can suppress this feature by setting the property to **False**, in order to avoid conflicts with menu commands in Visual Basic. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Key combinations will/will not be translated into clipboard commands **Default value:** True

Example Code VB.NET

```
VcNet1.CtrlCXVProcessing = False
```

Example Code C#

```
vcNet1.CtrlCXVProcessing = false;
```

DataTableCollection

Property of VcNet

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 = VcNet1.DataTableCollection
For Each dataTable In dataTableCltn
   ListBox1.Items.Add(dataTable.Name)
Next
```

Example Code C#

```
VcDataTableCollection dataTablecltn = vcNet1.DataTableCollection;
foreach(VcDataTable dataTable in dataTablecltn)
   listBox1.Items.Add(dataTable.Name);
```

DateOutputFormat

Property of VcNet

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 format

Example Code VB.NET

```
VcNet1.DateOutputFormat = "DD.MM.YY"
```

Example Code C#

```
vcNet1.DateOutputFormat = "DD.MM.YY";
```

DiagramBackgroundColor

Property of VcNet

This property lets you assign/retrieve a background color to your network diagram. The default color is white.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})

Example Code VB.NET

```
VcNet1.DiagramBackgroundColor = RGB(255, 204, 204)
```

Example Code C#

```
vcNet1.DiagramBackgroundColor = RGB(255, 204, 204);
```

DialogFont

Property of VcNet

This property specifies/retrieves the font name and size in the dialogs of the VARCHART XNet control that appear at run time. The object expected is a font object of your programming environment, e.g. in Visual Basic an object of the class **StdFont**.

	Data Type	Explanation
Property value	System.String	Font name

Example Code VB.NET

```
Dim newFont As Font
newFont = Newfont ("Verdana", 14)
VcNet1.DialogFont = newFont
```

Example Code C#

```
Font newFont = newFont ("Verdana", 14);
vcNet1.DialogFont = newFont;
```

DoubleOutputFormat

Property of VcNet

This property lets you set or retrieve the output format of numbers as a double value in a network diagram. The format is presented by the below characters:

- Text
- I
- D

plus the separators **comma** and **period**. **Text** represents a character string; **I** represents the figures in front of the decimal separator and **D** represents the figures after the decimal separator. The overall sequence is **Text I D Text**, where a comma and a period can be inserted in the places desired. As an example be the number -284901,3458. By the format **I,DDDD ppm** it will be output as **-284901,3458 ppm**. By the format **$I,III.DD** it will be output as **$-284,901.35**.

	Data Type	Explanation
Property value	System.String	Character string which describes the double format, for example "$I,III.DD".

Example Code VB.NET

```
VcNet1.DoubleOutputFormat = "I,DDDD ppm"
```

Example Code C#

```
vcNet1.DoubleOutputFormat = "$I,III.DD";
```

Enabled

Read Only Property of VcNet

This property lets you disable the VARCHART XNet control so that it will not react to mouse or keyboard commands.

	Data Type	Explanation

Example Code VB.NET

```
VcNet1.Enabled = False
```

Example Code C#

```
vcNet1.Enabled = false;
```

ExtendedDataTablesEnabled

Property of VcNet

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.

	Data Type	Explanation
Property value	System.Boolean	**true**: only two data tables (Maindata and Relations) **false**: up to 99 data tables **Default value:** false

Example Code VB.NET

```
VcNet1.ExtendedDataTablesEnabled = True
```

Example Code C#

```
vcNet1.ExtendedDataTablesEnabled = true;
```

FilePath

Property of VcNet

This property lets you set the file path so that graphics files and group title 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 Windows Forms control.

This property should be set when the application is started during the initializing procedure of the VARCHART Windows Forms 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)
VcNet1.FilePath = exeDir + "\Bitmaps"
```

Example Code C#

```
string exeName = Environment.GetCommandLineArgs()[0];
vcNet1.FilePath = System.IO.Path.GetDirectoryName(exeName)+ @"\..\Bitmaps";
```

FilterCollection

Read Only Property of VcNet

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 = VcNet1.FilterCollection
```

Example Code C#

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
```

FontAntiAliasingEnabled

Read Only Property of VcNet

This property lets you set or retrieve whether fonts can be anti-aliased with GDI+. If the legibility of certain fonts - in particular non- latin ones - changes for the worse, the property should be set to **False**.

The anti-aliasing with GDI+ has yet another effect: regardless of the selected zoom factor, texts texts keep their relative dimension so that the number of characters that fits in a node field will always be the same. If the option is switched off the settings of the operating system are applied instead (the settings can be found in the **Control Panel**, dialog box **Display**, Tab **Appearance: Effects**). Thus, if the option **Smooth edges** is switched on in the **Control Panel**, the texts might still be anti-aliased, notwithstanding the settings of the **General** property page. In this case, at some zoom levels more text could be visible than at others, since the native edge smoothing does not guarantee that the same relative dimension is always kept.

This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Characters will / will not be anti-aliased **Default value:** true

GroupCollection

Read Only Property of VcNet

If a grouping is specified, this property lets you access the GroupCollection object which contains all available groups.

	Data Type	Explanation
Property value	VcGroupCollection	GroupCollection object

Example Code VB.NET

```
Dim groupCltn As VcGroupCollection
groupCltn = VcNet1.GroupCollection
```

Example Code C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
```

GroupHorizontalMargin

Property of VcNet

This property lets you specify the left and right margin of groups.

This property can also be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Single 0 ... 9.9 mm	Width (in mm) of the left/right group margins **Default value:** 0

Example Code VB.NET

```
VcNet1.GroupHorizontalMargin = 1.1
```

Example Code C#

```
VcNet1.GroupHorizontalMargin = 1.1;
```

GroupingActivated

Property of VcNet

This property lets you switch grouping of nodes on or off. This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Boolean	Grouping active (True) / not active (False)

Example Code VB.NET

```
VcNet1.Grouping = True
```

Example Code C#

```
vcNet1.Grouping = true;
```

GroupingDataFieldIndex

Property of VcNet

This property lets you set or retrieve the field of the data definition table to hold the criterion for grouping. By default, the groups created will be sorted in alphabetical or numerical order (also see GroupNodeSortingDataField-Index). This property also can be set on 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
Parameter: ⇨ groupingLevel	System.Int16	Grouping level (starting by 0)
Property value	System.Int16	Index of data definition field

Example Code VB.NET

```
Dim definitionTable As VcDataDefinitionTable
definitionTable =
VcNet1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
VcNet1.GroupingDataFieldIndex(0) =
definitionTable.DataDefinitionFieldByName("Code 1").ID
VcNet1.GroupNodes(True)

VcNet1.GroupField = dataDefinitionField.ID
```

Example Code C#

```
VcDataDefinitionTable definitionTable =
vcNet1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
vcNet1.set_GroupingDataFieldIndex(0, "Code 1");
vcNet1.GroupNodes(true);
```

GroupingTitlesFullyVisible

Read Only Property of VcNet

This property lets you set or retrieve whether titles of groups remain visible even when the chart section displayed is scrolled.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	System.Boolean	Grouping titles visible (True)/ invisible (False)
Property value	System.Boolean	Visible (True)/ not visible (False) **Default value:** False

GroupingType

Property of VcNet

This property specifies the visualization mode of groups:

- **Grouping:** normal visualization of groups (The width and height of each group is determined by the node positions. Each group needs the full width or height respectively of the net diagram)

- **Clustering:** The nodes are grouped very space-sparing, and the groups are placed freely in the net diagram.

You should not modify this property any more as soon as groups are visible in the diagram.

This property also can be set on the **Grouping** property page.

For further information please read the chapter "Important Concepts: Grouping".

	Data Type	Explanation
Property value	VcGroupMode	Mode of visualization **Default value:** 0

Example Code VB.NET

```
VcNet1.GroupMode = vcGMClustering
```

Example Code C#

```
vcNet1.GroupMode = vcGMClustering;
```

GroupInteractionsAllowed

Property of VcNet

This property specifies whether groups can be collapsed or expanded interactively (by the Plus or Minus sign beside the group title).

The interactive collapsing or expanding triggers the **VcGroupModifying** event.

You should not modify this property any more as soon as groups are visible in the diagram.

This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active (True) / not active (False) **Default value:** True

Example Code VB.NET

```
VcNet1.GroupInteractionsAllowed = False
```

Example Code C#

```
vcNet1.GroupInteractionsAllowed = false;
```

GroupSortingDataFieldIndex

Property of VcNet

This property lets you specify which field of the data definition table is to be used for sorting the groups. The default sorting of groups is the alphabetical order by the **GroupField**. By using **GroupNodeSortingDataFieldIndex**, you can specify any sorting order. This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Int16	Index of data definition field

Example Code VB.NET

```
VcNet1.GroupSortingDataFieldIndex(0) = 12
VcNet1.GroupSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcNet1.SortGroups()

VcNet1.GroupNodeSortingDataFieldIndex = dataDefinitionField.ID
```

Example Code C#

```
vcNet1.set_GroupSortingDataFieldIndex(0, 12);
vcNet1.set_GroupSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcNet1.SortGroups();
```

GroupSortMode

Property of VcNet

This property lets you set or retrieve the sorting order of groups. **vcAscending** is the default, that sorts the groups according to their group field in ascending order. This property can also be set on the **Grouping** property page.

	Data Type	Explanation
Property value	VcGroupSortMode	Sort Mode of Groups **Default value:** vcAscending

Example Code VB.NET

```
VcNet1.GroupSortMode = vcDescending
```

Example Code C#

```
vcNet1.GroupSortMode = vcDescending;
```

GroupTitleDataFieldIndex

Property of VcNet

This property allows you to set or retrieve the data definition field index of a node record that the group title is to be taken from. A group title serves as a group heading and is displayed in the top row of a group. It is recommended to use the same group title for all members of a group to avoid random titles.

If a name was defined by the property **GroupDescriptionName**, then the one in the file will be used.

This property can also be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Int32	Index of data definition field

Example Code VB.NET

```
VcNet1.GroupTitleDataFieldIndex = VcNet1.DetectFieldIndex("Maindata", "Code 2")
```

Example Code C#

```
vcNet1.GroupTitleDataFieldIndex = vcNet1.DetectFieldIndex("Maindata", "Code 2");
```

GroupTitlesFileName

Property of VcNet

This property lets you set or retrieve the name of a file, that contains the assignments of group titles to groups. The file is a simple text file and contains a single assignment per line. In a line, the group name is followed by a blank and then by the title. A group name therefore must not contain a blank. Empty group names are designated by "". The default group name is "".

If a relative file name was specified, at run time the file will be searched in the path set in the VARCHART ActiveX property **FilePath** first. If it won´t be found there, the file will be searched in the current directory of the application and in the installation directory of the VARCHART ActiveX control.

This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.String	Name

Example Code VB.NET

```
VcNet1.GroupDescriptionName = "C:\varchart\xnet\samples\net.des"
```

Example Code C#

```
vcNet1.GroupDescriptionName = "C:\varchart\xnet\samples\net.des";
```

GroupVerticalMargin

Property of VcNet

This property lets you specify the upper/bottom margins of groups.

This property can also be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Single 0 ... 9.9 mm	Height (in mm) of the upper/bottom group margins **Default value:** 0

Example Code VB.NET

```
VcNet1.GroupVerticalMargin = 0.9
```

Example Code C#

```
VcNet1.GroupVerticalMargin = 0.9;
```

InbuiltMouseCursorWhileDraggingEnabled

Read Only Property of VcNet

This property lets you disable the mouse cursor in the target control during an OLE drag operation. OLE Drag & Drop allows to set the cursor in the source control by the event **OLEGiveFeedback**. If you do this, two competing cursors will exist in the target control, that may appear to flicker. You can avoid the flickering by disabling the target cursor by this property.

Beside, if the cursor is enabled and the property **OLEDropManual** is set, objects cannot be dropped outside the joining ports of a node. If you disable the cursor, you can drop objects outside the joining ports.

You also can set this property on the **Nodes** property page.

	Data Type	Explanation
Property value	System.Boolean	Cursor does/does not occur in the target control **Default value:** True

Example Code VB.NET

```
VcNet1.OLEDragWithOwnMouseCursor = False
```

Example Code C#

```
vcNet1.OLEDragWithOwnMouseCursor = false;
```

InFlowGroupingActivated

Property of VcNet

This property lets you activate/deactivate the in-flow grouping. If it is activated, the layout calculation for the network diagram automatically will be started. (also see the VcNet method **Arrange**). This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	System.Boolean	In-flow grouping activated (True)/ deactivated (False) **Default value:** False

Example Code VB.NET

```
VcNet1.InFlowGroupingEnabled = True
```

Example Code C#

```
vcNet1.InFlowGroupingEnabled = true;
```

InFlowGroupingDataFieldIndex

Property of VcNet

This property lets you set or retrieve the data field which determines the in-flow groups. This property also can be set by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Parameter:		
⇨ Rückgabewert	System.Int16	Data field which determines the in-flow groups
Property value	System.Int.16	Data field which determines the in-flow groups **Default value:** -1

InFlowGroupSeparationLineColor

Property of VcNet

This property lets you set or retrieve the color of the separation lines of in-flow groups. This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})

Example Code VB.NET

```
VcNet1.InFlowGroupSeparationLineColor = RGB(255, 204, 204)
```

Example Code C#

```
vcNet1.InFlowGroupSeparationLineColor = RGB(255, 204, 204);
```

InFlowGroupSeparationLineType

Property of VcNet

This property lets you set or retrieve the type of the separation lines of in-flow groups. This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	VcLineType	Type of separation lines of in-flow groups
	Possible Values:	
	.vcDashed 4	Line dashed
	.vcDashed 4	Line dashed
	.vcDashedDotted 5	Line dashed-dotted
	.vcDashedDotted 5	Line dashed-dotted
	.vcDotted 3	Line dotted
	.vcDotted 3	Line dotted
	.vcLineType0 100	Line Type 0
	.vcLineType1 101	Line Type 1
	.vcLineType10 110	Line Type 10
	.vcLineType11 111	Line Type 11
	.vcLineType12 112	Line Type 12
	.vcLineType13 113	Line Type 13
	.vcLineType14 114	Line Type 14
	.vcLineType15 115	Line Type 15
	.vcLineType16 116	Line Type 16
	.vcLineType17 117	Line Type 17
	.vcLineType18 118	Line Type 18
	.vcLineType2 102	Line Type 2
	.vcLineType3 103	Line Type 3

.vcLineType4 104	Line Type 4 -
.vcLineType5 105	Line Type 5 — — — — — — — — — — — —
.vcLineType6 106	Line Type 6 — — — — — — — —
.vcLineType7 107	Line Type 7 - - - - - - - - - - - - - - - - - -
.vcLineType8 108	Line Type 8 – – – – – – – – – – – – – –
.vcLineType9 109	Line Type 9 —---—---—---—---—---—
.vcNone 1	No line type assigned
.vcNone 1	No line type
.vcNotSet -1	No line type assigned
.vcSolid 2	Line solid
.vcSolid 2	Line solid

InFlowGroupTimeInterval

Property of VcNet

This property lets specify/require the interval that defines the time period of the in-flow grouping (e.g. 1 second, 1 minute, 1 hour, 1 day, 2 months, 1 year). This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	VcTimeOrientationInterval	In-flow grouping interval
	Possible Values:	
	.vcDay 5	Day
	.vcFifteenMinutes 1848	15 minutes
	.vcFifteenSeconds 1845	15 seconds
	.vcFourHours 1853	4 hours
	.vcHalfYear 1242	half year
	.vcHour 6	hour
	.vcMinute 7	minute
	.vcMonth 3	month
	.vcQuarter 2	quarter (3 month)
	.vcSecond 8	second
	.vcSixHours 1854	6 hours
	.vcThirtyMinutes 1849	30 minutes
	.vcThirtySeconds 1846	30 seconds
	.vcThreeHours 1852	3 hours
	.vcTwelveHours 1855	12 hours
	.vcTwoHours 1851	2 hours
	.vcTwoWeeks 1238	two weeks
	.vcTwoYears 1245	two years
	.vcWeek 4	week
	.vcYear 1	year

Example Code VB.NET

```
VcNet1.InFlowGroupTimeInterval = vcDay
```

Example Code C#

```
vcNet1.InFlowGroupTimeInterval = vcDay;
```

InFlowGroupTitleDataFieldIndex

Property of VcNet

This property lets you set or retrieve the data field which is taken for in-flow group titles. This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Int32	Data field which is taken for in-flow group title ribbons

Example Code VB.NET

```
VcNet1.InFlowGroupTitleField = 1
```

Example Code C#

```
VcNet1.InFlowGroupTitleField = 1;
```

InFlowGroupTitlesBackgroundColor

Property of VcNet

This property lets you set or retrieve the background color of titles of in-flow groups. This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Drawing.Color	Background color of title ribbons of in-flow groups

InFlowGroupTitlesFileName

Property of VcNet

This property lets you set or retrieve the name of the file which contains the group title texts for the in-flow grouping. The file is a simple text file and contains a single assignment per line. In a line, the group name is followed by a blank and then by the title. A group name therefore must not contain a blank. Empty group names are designated by "". The default group name is "".

If a relative file name was specified, at run time the file will be searched in the path set in the property **FilePath** first. If it won´t be found there, the file will be searched in the current directory of the application and in the installation directory of the VARCHART control.

This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.String	File name which contains the title texts for in-flow groups

InFlowGroupTitlesFont

Property of VcNet

This property lets you set or retrieve the font attributes of the titles of in-flow groups. This property also can be set in the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	StdFont	Font of title ribbons of in-flow groups

InFlowGroupTitlesVisibleAtBottomOrRight

Property of VcNet

This property lets you set or retrieve whether titles of in-flow groups are visible at the bottom or right side of the graphics. This property also can be set by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	Visible (True)/ not visible (False)

InFlowGroupTitlesVisibleAtTopOrLeft

Property of VcNet

This property lets you set or retrieve whether titles of the in-flow groups are visible at the top or left side of the graphics. This property also can be set by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	Visible (True)/ not visible (False)

InFlowGroupTitleTimeFormat

Property of VcNet

This property lets you set or retrieve the date/time output format for in-flow grouping by date field. This property also can be set by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Boolean	Date/time output format for in-flow grouping by date field **Default value:** DD.MM.YYYY

InFlowGroupVerticalCaptionWidth

Property of VcNet

This property lets you set or retrieve the width of vertical title ribbons for in-flow grouping. This property also can be set by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	System.Int16	Width of title ribbons for in-flow grouping **Default value:** 50

Example Code VB.NET

```
VcNet1.InFlowGroupVerticalCaptionWidth = 30
```

Example Code C#

```
VcNet1.InFlowGroupVerticalCaptionWidth = 30;
```

InPlaceEditingAllowed

Property of VcNet

This property lets you set or retrieve whether at run time the in-place editing of data fields in the table, in boxes an in layers is possible. You also can set this property on the **General** property page.

Note: If certain data fields are not to be editable, the **Editable** check box in the **Administrate Data Tables** dialog must not be ticked.

	Data Type	Explanation
Property value	System.Boolean	In-place editing in node fields permitted (True) / not permitted (False) **Default value:** True

Example Code VB.NET

```
VcGantt1.InPlaceEditingAllowed = True
```

Example Code C#

```
vcGantt1.InPlaceEditingAllowed = true;
```

InteractionMode

Property of VcNet

This property activates/retrieves one of the available modes of interaction.

	Data Type	Explanation
Property value	VcInteractionMode	Interaction mode **Default value:** vcPointer
	Possible Values:	
	.vcCreateLink 4	Link creating mode
	.vcCreateNodesAndLinks 1	Nodes and links creating mode
	.vcPointer 0	Select mode

Example Code VB.NET

```
VcNet1.InteractionMode = vcCreateNodesAndNodes
```

Example Code C#

```
vcNet1.InteractionMode = vcCreateNodesAndNodes;
```

InterfaceNodesShown

Property of VcNet

This property lets you specify whether the interface nodes are to be displayed (True) or not (False), when a subdiagram is created. You can specify the appearance of the interface nodes in the **Specify Node Appearance** dialog box. To do so, select the special filter <InterfaceNodes>. This property also can be specified by the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active **Default value:** True

Example Code VB.NET

```
VcNet1.InterfaceNodesShown = False
```

Example Code C#

```
vcNet1.InterfaceNodesShown = false;
```

LegendView

Read Only Property of VcNet

This property gives access to the LegendView object that lets you define the legend view.

	Data Type	Explanation
Property value	VcLegendView	LegendView object

Example Code VB.NET

```
Dim legendview As VcLegendView
legendview = VcNet1.LegendView
legendview.Visible = True
```

Example Code C#

```
VcLegendView legendview = vcNet1.LegendView;
legendview.Visible = true;
```

LinkAnnotationColumnNumberDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which holds the column number of a link annotation. Setting this property is only possible if data was not loaded yet.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the column number of a link annotation

LinkAnnotationRowNumberDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which holds the row number of a link annotation. Setting this property is only possible if data was not loaded yet.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the column number of a link annotation

Example Code VB.NET

```
     Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load

        VcNet1.NodeRowNumberDataFieldIndex =
VcNet1.DetectFieldIndex("NodeDataTable", "SortNumber")

        'Load data
        LoadData()

        VcNet1.UpdateRowNumberFields()
        VcNet1.SaveAsEx("C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding)
    End Sub
```

Example Code C#

```
private void Form1_Load(object sender, System.EventArgs e)
  {
      vcNet1.NodeRowNumberDataFieldIndex =
vcNet1.DetectFieldIndex("NodeDataTable", "SortNumber");

       // Load data
         loadData();

        vcNet1.UpdateRowNumberFields();
        vcNet1.SaveAsEx(@"C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding);
   }
```

LinkAppearanceCollection

Read Only Property of VcNet

This property lets you access the LinkAppearanceCollection object that contains all defined link appearances.

	Data Type	Explanation
Property value	VcLinkAppearanceCollection	LinkAppearanceCollectionObject

Example Code VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
linkAppearanceCltn = VcNet1.LinkAppearanceCollection
```

Example Code C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
```

LinkCollection

Read Only Property of VcNet

This property lets you access the LinkCollection object that contains all defined links.

	Data Type	Explanation
Property value	VcLinkCollection	LinkCollection object

Example Code VB.NET

```
Dim linkCltn As VcLinkCollection
linkCltn = VcNet1.LinkCollection
```

Example Code C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
```

LinkCreationWithDialog

Property of VcNet

This property specifies whether the **Edit Data** dialog box is to appear when a new link is created. The **AllowNewNodesAndLinks** property has to be set to **True** to enable the user to create new links.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active

Example Code VB.NET

```
VcNet1.EditNewLink = False
```

Example Code C#

```
vcNet1.EditNewLink = false;
```

LinkFormatCollection

Read Only Property of VcNet

This property gives access to the LinkFormatCollection object that contains all link formats available.

	Data Type	Explanation
Property value	VcLinkFormatCollection	LinkFormatCollection object

Example Code VB.NET

```
Dim formatCollection As VcLinkFormatCollection

Set formatCollection = VcNet1.LinkFormatCollection
```

LinkPredecessorDataFieldIndex

Property of VcNet

This property lets you set or retrieve the data field which holds the identification of the predecessor node of the link. You can only set this property if data was not yet loaded.

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
Parameter:		
⇨ identifierIndex	System.Int32	Index of predecessor node {0...2}
Property value	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 = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcNet1.DataTableCollection.Update()

VcNet1.LinkPredecessorDataFieldIndex(0) =
VcNet1.DetectFieldIndex("LinkDataTable", "Id")
VcNet1.LinkSuccessorDataFieldIndex(0) = VcNet1.DetectFieldIndex("LinkDataTable",
"Id")

'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading()
```

Example Code C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcNet1.DataTableCollection.Update();

vcNet1.set_LinkPredecessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));
vcNet1.set_LinkSuccessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcNet1.EndLoading();
```

LinksDataTableName

Property of VcNet

This property lets you set or retrieve the name of the data table which contains the fields for the links. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	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 = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcNet1.DataTableCollection.Update()

VcNet1.LinkPredecessorDataFieldIndex(0) =
VcNet1.DetectFieldIndex("LinkDataTable", "Id")
VcNet1.LinkSuccessorDataFieldIndex(0) = VcNet1.DetectFieldIndex("LinkDataTable",
"Id")

'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading()
```

Example Code C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcNet1.DataTableCollection.Update();

vcNet1.set_LinkPredecessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));
vcNet1.set_LinkSuccessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcNet1.EndLoading();
```

LinkSuccessorDataFieldIndex

Property of VcNet

This property lets you set/retrieve the data field index of the successor node of a link. Setting this property is only possible if data was not loaded yet.

The property **LinkSuccessorDataFieldIndex** is an Indexed Property, which in C# is addressed by the two methods set_LinkSuccessorDataFieldIndex (identifierIndex, pvn) and get_LinkSuccessorDataFieldIndex (identifier-Index).

	Data Type	Explanation
Parameter:		
⇨ identifierIndex	System.Int32	Index of successor node {0...2}
Property value	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 = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcNet1.DataTableCollection.Update()

VcNet1.LinkPredecessorDataFieldIndex(0) =
VcNet1.DetectFieldIndex("LinkDataTable", "Id")
VcNet1.LinkSuccessorDataFieldIndex(0) = VcNet1.DetectFieldIndex("LinkDataTable",
"Id")

'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading()
```

Example Code C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcNet1.DataTableCollection.Update();

vcNet1.set_LinkPredecessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));
vcNet1.set_LinkSuccessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcNet1.EndLoading();
```

LinkTypeDataFieldIndex

Property of VcNet

This property lets you set or retrieve the name of the data field which contains the link type. Setting this property is only possible if data was not loaded yet.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which contains the link type

Example Code VB.NET

```
Dim dataTable As VcDataTable

'create Link DataTable
dataTable = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
dataTable.DataTableFieldCollection.Add("LinkType")
VcNet1.DataTableCollection.Update()
```

Example Code C#

```
VcDataTable dataTable;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
dataTable.DataTableFieldCollection.Add("LinkType");
vcNet1.DataTableCollection.Update();
```

MapCollection

Read Only Property of VcNet

This property lets you access the MapCollection object that contains a defined number of maps. The number of maps is defined by the method **VcMapCollection.SelectMaps**.

	Data Type	Explanation
Property value	VcMapCollection	MapCollection object

Example Code VB.NET

```
Dim mapCltn As VcMapCollection
mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

Example Code C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
```

MinimumColumnWidth

Property of VcNet

By this property you can assign a minimum width (unit: mm) to a column. The width chosen should correspond to the average width of a node. To make nodes utilize less space in a left-to-right orientation, you can use this property to reduce the column width further. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Int32 {1...1 000}	Minimum column width in mm **Default value:** 1

Example Code VB.NET

```
VcNet1.MinimumColumnWidth = 100
```

Example Code C#

```
vcNet1.MinimumColumnWidth = 100;
```

MinimumRowHeight

Property of VcNet

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 a node. To make nodes utilize less space in a top-down orientation, you can use this property to reduce the row height further. This property can also be set on the **General** 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 {1...1 000}	Minimum row height in mm **Default value:** 1

Example Code VB.NET

```
VcNet1.MinimumRowHeight = 100
```

Example Code C#

```
vcNet1.MinimumRowHeight = 100;
```

MouseProcessingEnabled

Property of VcNet

This property allows you to process mouse events in your own way. If you want your own processing method between the **VcMouseDown** event and the

VcMouseUp event, then set the **MouseProcessingEnabled** property to **False** for this time interval. Then VARCHART XNet will ignore all mouse movements and clicks until this property is set to **True** again.

This property also can be set in the VcMouse events.

	Data Type	Explanation
Property value	System.Boolean	Property active (True)/ not active (False) **Default value:** True

MovingCollapsedClustersAllowed

Property of VcNet

This property permits (True) or prohibits (False) the user to move collapsed clusters (only relevant for the clustering mode).

This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Boolean	Moving collapsed clusters allowed (True)/not allowed (False) **Default value:** True

Example Code VB.NET

```
Dim boole As Boolean

boole = VcNet1.GroupMovingAllowed
```

Example Code C#

```
Boolean boole = vcNet1.GroupMovingAllowed;
```

NodeAndLinkCreationAllowed

Property of VcNet

This property permits (True) or prohibits (False) the user to create new nodes and links. If this property is set to False, the user cannot activate the **CreateNodesAndLinks** mode. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active (True)/not active (False) **Default value:** True

Example Code VB.NET

```
Dim boole As Boolean

boole = VcNet1.AllowNewNodesAndLinks
```

Example Code C#

```
Boolean boole = vcNet1.AllowNewLinksAndNodes;
```

NodeAppearanceCollection

Read Only Property of VcNet

This property lets you access the NodeAppearanceCollection object that contains all defined node appearances.

	Data Type	Explanation
Property value	VcNodeAppearanceCollection	NodeAppearanceCollection object

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.BackgroundColor = Color.LightBlue
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.BackgroundColor = Color.LightBlue;
```

NodeCalendarNameDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which holds the name of a calendar if you wish to use an individual calendar for a node. Setting this property is only possible if data was not loaded yet.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the name of the calendar for the node

NodeChangeRankToPredecessorRankDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field to which the rank of the predecessor node is stored. Setting this property is only possible if data was not loaded yet.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the rank number of the predecessor node

NodeCollection

Read Only Property of VcNet

This property lets you access the NodeCollection object that contains either all nodes (vcAll) or only the marked nodes (vcMarked) or only the visible nodes (vcAllVisible), depending on the setting of **SelectNodes**.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll
```

Example Code C#

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
```

NodeColumnNumberDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which holds the column number of an activity. Setting this property is only possible if data was not loaded yet.

	Data Type	Explanation
Property value	System.Int32	Index of the data field which holds the column number of an activity

NodeCreationWithDialog

Property of VcNet

This property sets whether the **Edit Data** dialog box appears when a new node is created. The **AllowNewNodesAndLinks** property must be set to **True** to enable the user to create new nodes.

	Data Type	Explanation
Property value	System.Boolean	**Edit Data** dialog appears/does not appear.

Example Code VB.NET

```
VcNet1.NodeCreationWithDialog = False
```

Example Code C#

```
vcNet1.NodeCreationWithDialog = false;
```

NodeFormatCollection

Read Only Property of VcNet

This property lets you access he NodeFormatCollection object that contains all node formats available.

	Data Type	Explanation
Property value	VcNodeFormatCollection	NodeFormatCollection object

Example Code VB.NET

```
Dim formatCtln As VcNodeFormatCollection
formatCtln = VcNet1.NodeFormatCollection
```

Example Code C#

```
VcNodeFormatCollection formatCtln = vcNet1.NodeFormatCollection;
```

NodeRowNumberDataFieldIndex

Property of VcNet

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 data was not loaded yet.

	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

        VcNet1.NodeRowNumberDataFieldIndex =
VcNet1.DetectFieldIndex("NodeDataTable", "SortNumber")

        'Load data
        LoadData()

        VcNet1.UpdateRowNumberFields()
        VcNet1.SaveAsEx("C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding)
    End Sub
```

Example Code C#

```
private void Form1_Load(object sender, System.EventArgs e)
  {
      vcNet1.NodeRowNumberDataFieldIndex =
vcNet1.DetectFieldIndex("NodeDataTable", "SortNumber");

       // Load data
         loadData();

        vcNet1.UpdateRowNumberFields();
        vcNet1.SaveAsEx(@"C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding);
   }
```

NodesDataTableName

Property of VcNet

This property lets you set or retrieve the name of the data table which provides the fields for the nodes.

	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 = VcNet1.DataTableCollection.Add("NodeDataTable")
VcNet1.NodesDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("NodeDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;Node One;")
dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;")
VcNet1.EndLoading()
```

Example Code C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Node DataTable
dataTable = vcNet1.DataTableCollection.Add("NodeDataTable");
vcNet1.NodesDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("NodeDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;Node One;");
dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;");
vcNet1.EndLoading();
```

NodesUseCalendars

Property of VcNet

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 (True)/not active (False) **Default value:** True

Example Code VB.NET

```
VcNet1.NodesUseCalendars = False
```

Example Code C#

```
vcNet1.NodesUseCalendars = false;
```

NodeToolTipTextDataFieldIndex

Property of VcNet

This property lets you require/set the index of the data field of a node to store the tooltip texts for VMF files. This text appears when in the WebViewer the right mouse button is pressed.

This property also can be set on the **Nodes** property page.

	Data Type	Explanation
Property value	System.Int16	Index of the node data field for tooltip texts **Default value:** 4

Example Code VB.NET

```
VcNet1.NodeToolTipTextDataFieldIndex = 1
```

Example Code C#

```
vcNet1.NodeToolTipTextDataFieldIndex = 1;
```

ObliqueTracksOnLinks

Property of VcNet

This property lets you set or retrieve whether the link lines that connect the short horizontal line sections should be orthogonal or oblique. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean {True, False}	Oblique link lines (True)/orthogonal link lines (False) **Default value:** False

Example Code VB.NET

```
VcNet1.ObliqueTracksOnLinks = True
```

Example Code C#

```
vcNet1.ObliqueTracksOnLinks = true;
```

Orientation

Property of VcNet

This property lets you set or retrieve the orientation of the diagram. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	VcLayoutOrientation	From top to bottm, from left to right
	Possible Values:	
	.vcLeftToRight 0	Orientation of the net chart **from left to right**
	.vcTopToBottom 1	Orientation of the net chart **from left top to bottom**

Example Code VB.NET

```
VcNet1.Orientation = vcLeftToRight
```

Example Code C#

```
vcNet1.Orientation = vcLeftToRight;
```

PhantomDrawingWhileDraggingEnabled

Property of VcNet

This property lets you disable the display of an OLE drag phantom. Disabling the phantom makes sense, when merely the attributes of the object in the target control change, omitting to generate a new object.

You also can set this property on the **Nodes** property page.

	Data Type	Explanation
Property value	System.Boolean	Phantom does/does not occur **Default value:** True

Example Code VB.NET

```
VcNet1.OLEDragWithPhantom = False
```

Example Code C#

```
vcNet1.OLEDragWithPhantom = false;
```

Printer

Property of VcNet

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 = VcNet1.Printer.ZoomFactor
printerCuttingMarks = VcNet1.Printer.CuttingMarks
```

Example Code C#

```
int printerZoomfactor = vcNet1.Printer.ZoomFactor;
bool printerCuttingMarks = vcNet1.Printer.CuttingMarks;
```

RoundedLinkSlantsEnabled

Property of VcNet

This property lets you set or retrieve whether the slants of links of the routing type **vcLRTOrthogonal** are to be displayed as quarter circles instead of straigt lines.This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Slants of links are to be displayed/not displayed as quarter circles **Default value:** false

Example Code VB.NET

```
VcNet1.RoundedLinkSlantsEnabled = True
```

Example Code C#

```
vcNet1.RoundedLinkSlants.Enabled = true;
```

Scheduler

Read Only Property of VcNet

This property returns the VcScheduler object.

	Data Type	Explanation
Property value	VcScheduler	Returns the VcScheduler object

ShortenedLinks

Property of VcNet

This property will influence the layout of a network diagram and will be considered by the method **Arrange**. If you set this property to **True**, nodes will be placed as closely as possible near their successor nodes, thus keeping the distance between them as small as possible. If you set it to **False**, nodes will be placed as far left or up as possible. 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

```
VcNet1.ShortenedNodes = False
VcNet1.Arrange
```

Example Code C#

```
vcNet1.ShortenedLinks = false
vcNet1.Arrange;
```

StraightLinkDrawing

Property of VcNet

If this property is set to **true** the links between nodes do not lead orthogonally around objects, but cut straight through. If set, this property disables the property **ObliqueTracksOnLinks**.

	Data Type	Explanation
Property value	System.Boolean	Straight link drawing enabled (true) / disabled (false) **Default value:** false

TextEntrySupplyingEventEnabled

Property of VcNet

This property lets you activate the **VcTextEntrySupplying** event, that lets you modify the texts of the VARCHART XNet Control, for example to translate them into a different language. You can also set this property on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active

Example Code VB.NET

```
VcNet1.TextEntrySupplyingEventEnabled = True
```

Example Code C#

```
vcNet1.TextEntrySupplyingEventEnabled = true;
```

TimeUnit

Property of VcNet

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
Property value	VcTimeUnit	Time unit **Default value:** vcDay

Example Code VB.NET

```
Dim timeUnit As VcTimeUnit
timeUnit = VcNet1.TimeUnit
```

Example Code C#

```
VcTimeUnit timeUnit = vcNet1.TimeUnit;
```

ToolTipChangeDuration

Property of VcNet

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 **Default value:** -1

ToolTipDuration

Property of VcNet

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 **Default value:** -1

ToolTipPointerDuration

Property of VcNet

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 **Default value:** -1

ToolTipShowAfterClick

Property of VcNet

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) **Default value:** False

ToolTipTextSupplyingEventEnabled

Property of VcNet

This property lets you activate/deactivate the event **VcToolTipText-Supplying**. The event **VcToolTipTextSupplying** lets you edit the tooltip texts. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active/not active **Default value:** False

Example Code VB.NET

```
VcNet1.ToolTipTextSupplyingEventEnabled = True
```

Example Code C#

```
vcNet1.ToolTipTextSupplyingEventEnabled = true;
```

UngroupedNodesAllowed

Property of VcNet

This property specifies whether nodes without an entry for the group code (empty string) will not be grouped. Otherwise a special group for nodes without group code will be created.

This property is active only for the grouping mode clustering (GroupMode = vcGMClustering).

You should not modify this property any more as soon as groups are visible in the diagram.

This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	System.Boolean	Property active (True)/not active (False) **Default value:** False

Example Code VB.NET

```
VcNet1.UngroupedNodesAllowed  = True
```

Example Code C#

```
vcNet1.UngroupedNodesAllowed  = true;
```

ViewXCoordinate

Property of VcNet

This property lets you save the current scroll offset in x direction of the diagram section currently displayed and set it again if the same application is started. For the latter the zoom factor also has to be set in the same way.

	Data Type	Explanation
Property value	System.Int32	Scroll offset in x direction

ViewYCoordinate

Property of VcNet

This property lets you save the current scroll offset in y direction of the diagram section currently displayed and set it again if the same application is started. For the latter the zoom factor also has to be set in the same way.

	Data Type	Explanation
Property value	System.Int32	Scroll offset in y direction

WaitCursorEnabled

Property of VcNet

This property lets you set or returns whether a wait cursor appears on time critical operations (like SheduleProject).

The property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean	Wait cursor is set/is not set **Default value:** False

WorldView

Read Only Property of VcNet

This property lets you access the VcWorldView object that defines the world view (complete view) of the diagram.

	Data Type	Explanation
Property value	VcWorldView	World View object

Example Code VB.NET

```
Dim worldview As VcWorldView

worldview = VcNet1.WorldView
worldview.Visible = True
```

Example Code C#

```
VcWorldView worldview = vcNet1.WorldView;
worldview.Visible = true;
```

ZoomFactor

Property of VcNet

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).

	Data Type	Explanation
Property value	System.Int16 {0...1000}	Zoom factor (%)

Example Code VB.NET

```
VcNet1.ZoomFactor = 150
```

Example Code C#

```
vcNet1.ZoomFactor = 150;
```

ZoomingPerMouseWheelAllowed

Property of VcNet

This property lets you set or retrieve whether zooming by mouse wheel should be allowed to the user.

	Data Type	Explanation
Property value	System.Boolean	Zooming allowed (True)/not allowed (False)

Example Code VB.NET

```
VcNet1.ZoomingPerMouseWheelAllowed = False
```

Example Code C#

```
vcNet1.ZoomingPerMouseWheelAllowed = false;
```

Methods

Arrange

Method of VcNet

This method performs a layout of the network diagram. By doing so, the property **ShortenedLinks** will be considered.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
VcNet1.Arrange
```

Example Code C#

```
vcNet1.Arrange;
```

Clear

Method of VcNet

This method should be used only if nodes are in the chart. This methods lets you delete all graphical objects (nodes, links, calendars etc.) from the diagram. The initial state of the ini file will be restored.

	Data Type	Explanation
Return value	System.Boolean	Nodes were deleted successfully. {True}

Example Code VB.NET

```
VcNet1.Clear
```

Example Code C#

```
vcNet1.Clear;
```

CompleteViewMode

Method of VcNet

This method allows to display a diagram completely. The zoom factor automatically adapts to changements in the chart. The maximum zoom factor of 100% will not be exceeded so that the nodes by maximum are displayed in their original size. Also see property **ZoomFactor** and method **Zoom**.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
VcNet1.CompleteViewMode
```

Example Code C#

```
vcNet1.CompleteViewMode;
```

CopyNodesIntoClipboard

Method of VcNet

This method lets you copy the selected nodes from the network diagram to the clipboard. Also see methods **CutNodesIntoClipboard** and **PasteNodesFromClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
VcNet1.CopyNodesIntoClipboard
```

Example Code C#

```
vcNet1.CopyNodesIntoClipboard;
```

CutNodesIntoClipboard

Method of VcNet

This method lets you cut the nodes diagram into the clipboard. Also see **CopyNodesIntoClipboard** and **PasteNodesFromClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
VcNet1.CutNodesIntoClipboard
```

Example Code C#

```
vcNet1.CutNodesIntoClipboard;
```

DeleteLinkRecord

Method of VcNet

This method lets you delete a link by passing the link record. Also see method **Delete** of object **VcLink**.

	Data Type	Explanation
Parameter: ⇨ linkRecord	System.Object	Link record
Return value	System.Boolean	Link record was/was not not deleted successfully

Example Code VB.NET

```
VcNet1.DeleteLinkRecord "A100;A105;;"
```

Example Code C#

```
vcNet1.DeleteLinkRecord "A100;A105;;";
```

DeleteNodeRecord

Method of VcNet

This method lets you delete a node. The node will be identified by the primary key in the node record. The data field that is used for the identification of nodes is set in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter: ⇨ nodeRecord	System.Object	Node record
Return value	System.Boolean	Node record was/was not deleted successfully

Example Code VB.NET

```
VcNet1.DeleteNodeRecord "A100;;;;;;"
```

Example Code C#

```
vcNet1.DeleteNodeRecord "A100;;;;;;";
```

DetectDataTableFieldName

Method of VcNet

This property lets you retrieve the name of a data table field by its index.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	System.Int32	Index of the data table field of which the name is to be retrieved
Return value	System.String	Name of the data table field returned

Example Code VB.NET

```
'Find the name of a DataTableField
Dim fieldName As String

fieldName = VcNet1.DetectDataTableFieldName(0)
```

Example Code C#

```
//Find the name of a DataTableField
string fieldName = vcNet1.DetectDataTableFieldName(0);
```

DetectDataTableName

Method of VcNet

This property lets you retrieve the name of a data table by its index.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	System.Int32	Index of the data table of which the name is to be retrieved
Return value	System.String	Name of the data table returned

Example Code VB.NET

```
'Find the name of a DataTable
Dim tableName As String

tableName = VcNet1.DetectDataTableName(0)
```

Example Code C#

```
//Find the name of a DataTable
string tableName = vcNet1.DetectDataTableName(0);
```

DetectFieldIndex

Method of VcNet

This property lets you retrieve the index of a data table field by ist name and the name of the data table.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	System.String	Name of the data table that holds the field of which the index is to be retrieved
⇨ dataTableFieldName	System.String	Name of the data table field of which the index is to be retrieved
Return value	System.Int32	Index of the data table field returned

Example Code VB.NET

```
'Find the index of a DataTableField
Dim fieldIndex As Integer

fieldIndex = VcNet1.DetectFieldIndex("Maindata", "Name")
```

Example Code C#

```
//Find the index of a DataTableField
int fieldIndex = vcNet1.DetectFieldIndex("Maindata", "Name");
```

DumpConfiguration

Method of VcNet

This method lets you save the configuration that consist of the .INI and the .IFD file.

The method should only be used for diagnosis purposes.

	Data Type	Explanation
Parameter:		
⇨ FileName	System.String	File name (including a path, if necessary)
⇨ encoding	VcEncoding	Mode of encoding
	Possible Values:	
	.vcUnicodeEncoding 2	Saving a file in Unicode encoding makes it independent of whatever settings and hence should be the preferred mode if possible. If a file that was saved in Unicode encoding is to be loaded in Visual Basic 6 independently of the VARCHART control, it has to be treated in a special way.
Return value	System.Boolean	File was (True)/was not (False) stored successfully.

EndLoading

Method of VcNet

This method indicates the finish of the loading procedure on the methods **InsertNodeRecord** and **InsertLinkRecord**, simultaneously triggering an update of the chart.

	Data Type	Explanation
Return value	System.Boolean	Loading finished {True}

Example Code VB.NET

```
VcNet1.EndLoading()
```

Example Code C#

```
vcNet1.EndLoading();
```

ExportGraphicsToFileEx

Method of VcNet

This method lets you store a net diagram to a file without generating a **Save as** dialog box. Possible formats for saving:

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, ggf. mit eingebauten EMF)

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

When exporting to bitmap formats, setting 0 to the desired number of pixels of both, the x and the y direction, will keep the aspect ratio. If both pixel numbers equal 0, the size (in pixels) of the exported chart is calculated by VARCHART XNet as listed below:

- PNG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. The number of DPIs will be stored to the PNG file, so with a given zoom factor display software can find the correct size for display.

- GIF, TIFF, BMP, JPEG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. In addition, an internal limit of 50 MBs of memory size is required for the uncompressed source bit map in the memory; so larger diagrams may have a smaller resolution than expected.

To formats of vector graphics, no pixel number can be set, but the below coodinate spaces:

- WMF: A fixed resolution is assumed where the longer side uses coordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.

- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm in both, the x and y direction.

For further details on the different formats please read the chapter "Important Concepts: Graphics Formats".

	Data Type	**Explanation**
Parameter:		
⇨ fileName	System.String	File name (including a path, if necessary)
⇨ printOutputFormat	PrintOutputFormat	Format of the file to be stored.
	Possible Values:	
	.vcBMP 2	File will be written in the format BMP.
	.vcEMF 9	File will be written in the format EMF.
	.vcEMFPlus 12	File will be written in the format EMF+, the standard extension is EMF.

	.vcEMFWithEMFPlusIncluded 11	File will be written in the format EMF, additionally including the format EMF+. The standard extension is EMF.
	.vcEPS 3	Deprecated
	.vcGIF 4	File will be written in the format GIF.
	.vcJPG 5	File will be written in the format JPG.
	.vcPCX 6	Deprecated
	.vcPNG 7	File will be written in the format PNG.
	.vcTIF 8	File will be written in the format TIF.
	.vcVMF 0	File will be written in the format VMF.
	.vcWMF 1	File will be written in the format WMF.
	.vcWMFWithEMFIncluded 10	File will be written in the format WMF, additionally including the format EMF. The standard extension is WMF.
⇨ 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.
Return value	System.Boolean	File was (true) / was not (false) stored successfully.

Example Code VB.NET

```
VcNet1.ExportGraphicsToFile "C:\temp\export", vcVMF, 0, 0
```

Example Code C#

```
vcNet1.ExportGraphicsToFile (@"c:\Tmp\test.vmf", VcPrintOutputFormat.vcVMF,0,0);
```

GetAValueFromARGB

Method of VcNet

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the alpha value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	System.Int32	ARGB value, from which the alpha value is to be identified
Return value	SystemInt.32	Alpha value returned

Example Code VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha,red,green,blue)
alpha = VcNet1.GetAValueFromARGB(argb)
```

Example Code C#

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha,red,green,blue);
alpha = vcNet1.GetAValueFromARGB(argb);
```

GetBValueFromARGB

Method of VcNet

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the "blue" value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	System.Int32	ARGB value, from which the "blue" value is to be identified
Return value	SystemInt.32	"Blue" value returned

Example Code VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha,red,green,blue)
blue = VcNet1.GetBValueFromARGB(argb)
```

Example Code C#

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha,red,green,blue);
blue = vcNet1.GetBValueFromARGB(argb);
```

GetGValueFromARGB

Method of VcNet

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the "green" value of an ARGB value.

	Data Type	Explanation
Parameter: ⇨ argb	System.Int32	ARGB value, from which the "green" value is to be identified
Return value	SystemInt.32	"Green" value returned

Example Code VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha,red,green,blue)
green = VcNet1.GetRValueFromARGB(argb)
```

Example Code C#

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha,red,green,blue);
green = vcNet1.GetGValueFromARGB(argb);
```

GetLinkByID

Method of VcNet

This method gives access to a link by its identification which was specified on the **Administrate Data Tables** dialog. If the identification consists of more than one field (composite primary key), the multipart ID has to be noted as shown below:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ linkID	System.Object	Link identification
Return value	VcLink	Link

Example Code VB.NET

```
Dim link As VcLink

link = VcNet1.GetLinkByID(" 5")
```

Example Code C#

```
VcLink link  = vcNet1.GetLinkByID(" 5");
```

GetLinkByNodeIDs

Method of VcNet

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

Example Code VB.NET

```
Dim link As VcLink
link = VcNet1.GetLinkByNodeIDs(" 2", " 3")
```

Example Code C#

```
VcLink link = vcNet1.GetLinkByNodeIDs(" 2", " 3");
```

GetNodeByID

Method of VcNet

This method lets you access a node by its identification which was specified on the **Administrate Data Tables** dialog. If the identification consists of more than onel field (composite primary key), the multipart ID needs to be noted as shown below:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter: ⇨ nodeID	System.Object	Node identification
Return value	VcNode	Node

Example Code VB.NET

```
Dim node As VcNode
node = VcNet1.GetNodeByID("10")
```

Example Code C#

```
VcNode node = vcNet1.GetNodeByID("10");
```

GetRValueFromARGB

Method of VcNet

A color value is composed by four parts: A (alpha), R (red), G (green) and B (blue). A value of 0 in the alpha position will result in complete transparency whereas 255 represents a completely solid color. Ascending values of R, G and B show increasingly lightening colors, the ultimate values 0,0,0 and 255,255,255 representing black and white, respectively. This method retrieves the "red" value of an ARGB value.

	Data Type	Explanation
Parameter:		
⇨ argb	System.Int32	ARGB value, from which the "red" value is to be identified
Return value	SystemInt.32	"Red" value returned

Example Code VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha,red,green,blue)
red = VcNet1.GetRValueFromARGB(argb)
```

Example Code C#

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha,red,green,blue);
red = vcNet1.GetRValueFromARGB(argb);
```

IdentifyFormatField

Method of VcNet

This method lets you retrieve the format of the specified node, as well as the index of the format field at the specified position. If there is a field at the position specified, **True** will be returned, if there isn't, the method will deliver **False**.

	Data Type	Explanation
Parameter:		
⇨ x	System.Int32	X coordinate of the position
⇨ y	System.Int32	Y coordinate of the position
⇨ node	VcNode	Reference Node
⇦ format	VcNodeFormat	Identified format
⇦ formatFieldIndex	System.Int16	Index of the format field
Return value	System.Boolean	A format field exists/does not exist at the position specified

Example Code VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
   Dim foundFlag As Boolean
   Dim format As VcNodeFormat
   Dim formatFieldIndex As Integer
   foundFlag = VcNet1.IdentifyFormatField(e.X, e.Y, e.Node, format,
formatFieldIndex)
   If foundFlag Then
      MsgBox("You hit the field with the index " + CStr(formatFieldIndex))
   End If
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeLeftClicking(object sender, VcNodeClickingEventArgs e)
{
   bool foundFlag;
   VcNodeFormat format = null;
   short formatFieldIndex = new short();
   foundFlag = vcNet1.IdentifyFormatField(e.X, e.Y, e.Node, ref format, ref
formatFieldIndex);
   if (foundFlag)
      MessageBox.Show("You hit the field with the index " +
formatFieldIndex.ToString());
}
```

IdentifyObjectAt

Method of VcNet

This method lets you identify any object located in an unknown position of the diagram. The object type will be returned.

	Data Type	Explanation
Parameter:		
⇨ x	System.Int32	X coordinate of the cursor
⇨ y	System.Int32	Y coordinate of the cursor
⇦ identifiedObject	System.Object	Object identified

⇦ identifiedObjectType	VcObjectType	Type of the object identified
	Possible Values:	
	.vcObjTypeBox 15	object type **box**
	.vcObjTypeGroup 7	object type **group**
	.vcObjTypeLinkCollection 3	object type **link collection**
	.vcObjTypeNode 2	object type **node**
	.vcObjTypeNone 0	no object
Return value	System.Boolean	Object identified/no object identified

Example Code VB.NET

```
Private Sub VcNet1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles VcNet1.MouseMove

      Dim identifiedObject As Object = Nothing
      Dim identifiedObjectType As VcObjectType = VcObjectType.vcObjTypeNone
      Dim node As VcNode = Nothing
      Dim identifiedLayer As VcLayer = Nothing

      VcNet1.IdentifyObjectAt(e.X, e.Y, identifiedObject, identifiedObjectType)

      Select Case identifiedObjectType
         Case VcObjectType.vcObjTypeNodeInDiagram
            node = identifiedObject

            VcNet1.IdentifyLayerAt(e.X, e.Y, node, identifiedLayer)

            If identifiedLayer IsNot Nothing Then
               Label1.Text = "X = " & e.X & "   Y = " & e.Y & vbCrLf & _
                             "Node ID = " & node.DataField(0) & vbCrLf & _
                             "Layer Name = " & identifiedLayer.Name
            End If

         Case Else
            Label1.Text = ""
      End Select

   End Sub
```

Example Code C#

```
private void VcNet1_MouseMove(object sender, MouseEventArgs e)
      {
         object identifiedObject = null;
         VcObjectType identifiedObjectType = VcObjectType.vcObjTypeNone;
         VcNode node = null;
         VcLayer identifiedLayer = null;

         VcNet1.IdentifyObjectAt(e.X, e.Y, ref identifiedObject, ref
identifiedObjectType);

         switch (identifiedObjectType)
         {
            case VcObjectType.vcObjTypeNodeInDiagram:
               {
                  node = (VcNode)identifiedObject;

                  VcNet1.IdentifyLayerAt(e.X, e.Y,  node, ref identifiedLayer);

                  if (identifiedLayer != null)
                     label1.Text = "X = " + e.X + "   Y = " + e.Y +
                                   "\nNode ID = " + node.get_DataField(0) +
                                   "\nLayer Name = " + identifiedLayer.Name;
                  break;
               }
            default:
               {
                  label1.Text = "";
                  break;
               }
         }
      }
```

ImportConfiguration

Method of VcNet

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

Example Code VB.NET

```
VcNet1.ImportConfiguration ( "c:\VARCHART\XNet\sample.ini")
'or
VcNet1.ImportConfiguration
("http://members.tripod.de/netronic_te/xnet_sample.ini)
```

Example Code C#

```
vcNet1.ImportConfiguration (@"c:\VARCHART\XNet\sample.ini");
// or
vcNet1.ImportConfiguration
(@"http://members.tripod.de/netronic_te/xnett_sample.ini");
```

InsertLinkRecord

Method of VcNet

This method lets you generate a link. The data will be passed as a CSV string (using semicolons as separators) in accordance with the structure defined in the **DataDefinition**. The method **EndLoading** should be invoked after the process of loading (links and nodes) was completed.

	Data Type	Explanation
Parameter: ⇨ linkRecord	data field/string	Link record
Return value	VcLink	Link

Example Code VB.NET

```
VcNet1.InsertNodeRecord "A100;Activity 1;12.09.14;17.09.14;5;Planning"
VcNet1.InsertNodeRecord "A105;Activity 5;13.09.14;18.09.14;7;Testing"
VcNet1.InsertLinkRecord "A100;A105;FS;0"

VcNet1.EndLoading
```

Example Code C#

```
vcNet1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning");
vcNet1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing");
vcNet1.InsertLinkRecord("A100;A105;FS;0");
```

InsertNodeRecord

Method of VcNet

This method lets you generate a node. The data will be passed as a CSV string (using semicolons as separators) in accordance with the structure defined on the **DataDefinition** property page. The method **EndLoading** should be invoked after the process of loading (links and nodes) was completed.

	Data Type	Explanation
Parameter: ⇨ nodeRecord	data field/string	Node record
Return value	VcNode	Node

Example Code VB.NET

```
' data format: "Number;Name;Start date;Finish date;Group code;Group name"
VcNet1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning")
VcNet1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing")
```

Example Code C#

```
//data format: "Number;Name;Start date;Finish date;Group code;Group name"
vcNet1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning");
vcNet1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing");
```

Load

Method of VcNet

This method lets you load data of the selected file. In the file, data have to be saved in CSV format (using semicolons as separators) in accordance with the **DataDefinition**. At first data of nodes is read and after a line with four asterisks (****) data of links is read.

	Data Type	Explanation
Parameter:		
⇨ fileName	System.String	File name
Return value	System.Boolean {True}	No significance

Example Code VB.NET

```
VcNet1.Open "C:\ProjectData.net"
```

Example Code C#

```
vcNet1.Open "C:\ProjectData.net";
```

MakeARGB

Method of VcNet

This method lets you compose an ARGB value from the four single values of a color.

	Data Type	Explanation
Parameter:		
⇨ alpha	SystemInt.32	Alpha value
⇨ red	SystemInt.32	"Red" value
⇨ green	SystemInt.32	"Green" value
⇨ blue	SystemInt.32	"Blue" value

Return value	System.Int32	ARGB value returned

Example Code VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = FF
red = A0
green = 34
blue = AB
argb = VcNet1.MakeARGB(alpha,red,green,blue)
```

Example Code C#

```
long argb;
int alpha = FF;
int red = A0;
int green = 34;
int blue = AB;
argb = vcNet1.MakeARGB(alpha,red,green,blue);
```

PasteNodesFromClipboard

Method of VcNet

This method lets you paste the nodes from the clipboard into the diagram. Also see **CopyNodesIntoClipboard** und **CutNodesIntoClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
nodeCltn = VcNet1.NodeCollection
nodecollection.SelectNodes(VcSelectionType.vcMarked)
If nodecollection.Count = 1 Then
VcNet1.PasteNodesFromClipboard(nodecollection.FirstNode,
VcPastePosition.vcPasteAsLastChild)
End If
```

Example Code C#

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcMarked);
if (nodeCltn.Count == 1)
vcNet1.PasteNodesFromClipboard(nodeCltn.FirstNode(),
VcPastePosition.vcPasteAsLastChild);
```

PixelsToRaster

Method of VcNet

This method turns window coordinates, as they for example are returned by events, into band numbers of horizontal and vertical direction. If the band numbers are beyond the chart limits, the function will return the value **False**. Also see **RasterToPixels**.

	Data Type	Explanation
Parameter:		
⇨ x	System.Int32	Y coordinate in pixels
⇨ y	System.Int32	X coordinate in pixels
⇦ xBandNo	System.Int32	Y coordinate in band numbers
⇦ yBandNo	System.Int32	X coordinate in band numbers
Return value	System.Boolean	Converting was/was not performed successfully

Example Code VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
   Dim lineNo As Long
   Dim columnNo As Long
   'change a data field of the node to the line number
   VcNet1.PixelsToRaster(e.X, e.Y, columnNo, lineNo)
   e.Node.DataField(19) = lineNo
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeLeftClicking(object sender, VcNodeClickingEventArgs e)
{
   int lineNo = new int();
   int columnNo = new int();
   //change a data field of the node to the line number
   vcNet1.PixelsToRaster(e.X, e.Y, ref columnNo, ref lineNo);
   e.Node.set_DataField(19, lineNo);
}
```

PrintEx

Method of VcNet

This method lets you print the diagram directly. A dialog box will not be displayed. If the printing was not successful the return value indicates the reason. This could be e.g. an entry in a log file.

	Data Type	Explanation			
Return value	VcPrintResultStatus	**Possible values:**			
		Name	parameter position	description	
		vcPrintingSucceeded	0	Printing was performed successfully.	
		vcNoPrinterInstalled	1	No printer was found	neither the one specified by the call **VcPrinter.PrinterName** nor the one labeled as default printer by the Windows operating system.
		vcPrintingAbortedByUser	2	Printing was aborted by the user.	
		vcPrintingAbortedByDriver	3	Printing was aborted by the Windows printer driver.	
		vcUnprintablePageLayout	4	Printing could not be performed since the page layout did not match the printer properties such as paper size or margins.	

Example Code C#

```
VcPrintResultStatus status = VcNet1.PrintDirectEx();
if (status != VcPrintResultStatus.vcPrintingSucceeded)
   System.Diagnosis.Trace.WriteLine("Printing failed: "+status.ToString);
```

PrintToFile

Method of VcNet

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
Parameter:		
⇨ fileName	System.String	File name

Return value	Void	

Reset

Method of VcNet

This methods lets you either delete the contents of all data tables or restore the settings of the property pages carried out at design time.

	Data Type	Explanation
Parameter:		
⇨ resetAction	VcResetAction	Objects to be initialized or deleted
	Possible Values:	
	.vcEmptyAllDataTables 4	The contents of all data tables are deleted but the data tables are kept.
	.vcReloadConfiguration 2	Complete reinitialization with the INI-file. All settings and created objects expire.
Return value	System.Boolean	The objects in the diagram were deleted successfully. {True}

Example Code VB.NET

```
VcNet1.Reset(VcResetAction.vcReloadConfiguration)
```

Example Code C#

```
vcNet1.Reset(VcResetAction.vcReloadConfiguration);
```

SaveAsEx

Method of VcNet

This method lets you save the records of all data tables to a file of CSV format, using the structure defined on the property page **Data Tables** invoked by the property page **Objects**. Data tables that do not contain records will not be saved. If no file name was specified, the file most recently used by the **Open** method will be overwritten (correponding to the common **Save** function).

	Data Type	Explanation
Parameter:		
⇨ fileName	System.String	File name
⇨ encoding	VcEncoding	Mode of encoding

	Possible Values: .vcUnicodeEncoding 2	Saving a file in Unicode encoding makes it independent of whatever settings and hence should be the preferred mode if possible. If a file that was saved in Unicode encoding is to be loaded in Visual Basic 6 independently of the VARCHART control, it has to be treated in a special way.
Return value	System.Boolean	Storing was/was not performed successfully

Example Code VB.NET

```
VcNet1.SaveAs "C:\ProjectData.net"
```

Example Code C#

```
vcNet1.SaveAs "C:\ProjectData.net";
```

ScheduleProject

Method of VcNet

This method triggers a forward and backward calculation of the current project. If you pass the start date, first a forward calculation will be performed, followed by a backward calculation. If you pass a final date, first a backward calculation will be performed, followed by a forward calculation. You can pass both dates, which will add some buffer times to the activities. At least one date must be passed, otherwise an error message will occur. If a cycle amongst the nodes and links is identified, the ones affected will be marked.

	Data Type	Explanation
Parameter:		
⇨ startDate	Date/Time	Start date or Null
⇨ endDate	Date/Time	End Date or Null
Return value	System.Boolean	Scheduling was/was not successfully performed

Example Code VB.NET

```
VcNet1.ScheduleProject "21.06.04", 0
```

Example Code C#

```
vcNet1.ScheduleProject "21.06.04", 0;
```

ScrollToNode

Method of VcNet

This method allows you to scroll to the row containing a particular node to make appear on the screen.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node to the row of which is to be scrolled to
Return value	System.Boolean	Scrolling was/was not performed successfully.

Example Code VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
   'scroll the diagram so that the node is completely on screen
   VcNet1.ScrollToNode(e.Node)
End Sub
```

Example Code C#

```
       object[] objDataRecord = new object[5];

         vcNet1.ExtendedDataTablesEnabled = true;
         vcNet1.MinimumRowHeight = 1000;

         vcNet1.TimeScaleEnd = new DateTime(2010, 8, 1);
         vcNett1.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 =
vcNet1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection;
         for (int i = 1; i < 100; i++)
            {
            objDataRecord[0] = i;
            objDataRecord[1] = "Node " + i.ToString();

            dataRecordCol.Add(objDataRecord);
             }
         vcNet1.EndLoading();
         vcNet1.ScrollToNode(vcNet1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned);
```

SetImageResource

Method of VcNet

With this method, a specified name can be assigned at runtime to an image object already existing in the application.The method provides an alternative to the one available so far, where the image name specified on the XNet property pages always leads to reading the image object out of the addressed file. It should be carried out when starting the application, for instance in the method **Form_Load**. The image names can be chosen at will. To distinguish the file names, characters that are otherwise forbidden in file names, can be used, e.g, the asterisk (*). All image objects are permitted: bitmaps (BMP, JPG, GIF, PNG, TIFF) and metafiles (WMF, EMF). If the parameter **image** is set to "null", all former assignments are cancelled.

Example: Add an image or file resource (in Visual Studio in the Poject Properties: Resources/Add Resource/Add Existing File…) and enter a code line in Form_Load like the one shown below .

For bitmap resources:

*vcNet1.SetImageResource("*PlusImage", <namespace>.Properties.Resources.plusImage);*

For metafile resources:

*vcNet1.SetImageResource("*MinusImage", new Metafile(new MemoryStream(<namespace>.Properties.Resources.minusImage)));*

	Data Type	Explanation
Parameter:		
⇨ imageName	System.String	Name assigned to image
Return value	System.Drawing.Image	image

ShowAboutDialog

Method of VcNet

This method lets you open the **About** box. It contains an overview of the program and the library files currently used with the absolute path and version numbers. This feature makes the hotline support more comfortable. The overview can be selected with the help of the mouse and copied by Ctrl+C and inserted by Ctrl+V into a mail.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
VcNet1.ShowAboutDialog()
```

Example Code C#

```
vcNet1.ShowAboutDialog();
```

ShowExportGraphicsDialog

Method of VcNet

This method lets you invoke the **Save As** dialog box to export the diagram. You can store the files to the formats:

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

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

Further details on the different formats please find in the chapter **Important Concepts: Graphics Formats**.

When exporting, the size of the exported diagram will be calculated this way:

- PNG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input.

- GIF, TIFF, BMP, JPEG: a resolution of 100 dpi and a zoom factor of 100% are assumed. If alternatively a value of <= -50 is specified in the parameter SizeX, the absolute number will be used as DPI input. In addition, an internal limit of 50 MBs of memory size is required for the uncompressed source bit map in the memory; so larger diagrams may have a smaller resolution than expected.

- WMF: A fixed resolution is assumed where the longer side uses coordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.

- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm.

	Data Type	Explanation
Return value	System.Boolean	Graphics successfully (true) /not successfully (false) exported

Example Code VB.NET

```
VcNet1.ShowExportGraphicsDialog()
```

Example Code C#

```
vcNet1.ShowExportGraphicsDialog();
```

ShowLinkEditDialog

Method of VcNet

This method invokes the **Edit Link** dialog box for the link passed.

	Data Type	Explanation
Parameter: ⇨ link	VcLink	Link the data of which are to be edited
Return value	System.Boolean	Link data were edited/edition was cancelled

Example Code VB.NET

```
Private Sub VcNet1_VcLinksLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksLeftClicking
   If e.LinkCollection.Count = 1 Then
     VcNet1.ShowLinkEditDialog(e.LinkCollection.FirstLink)
   End If
End Sub
```

Example Code C#

```
private void vcNet1_VcLinksLeftClicking(object sender, VcLinksClickingEventArgs
e)
{
   if (e.LinkCollection.Count == 1)
  vcNet1.ShowLinkEditDialog(e.LinkCollection.FirstLink());
}
```

ShowNodeEditDialog

Method of VcNet

This property invokes the **Edit Data** dialog box for the node passed.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node whose data are to be edited
Return value	System.Boolean	Node data were edited/editing was cancelled.

Example Code VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
   VcNet1.ShowNodeEditDialog(node)
End Sub
```

Example Code C#

```
private void vcvcNet1_VcNodeLeftClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
   {
   vcNet1.ShowNodeEditDialog(e.Node);
   }
```

ShowPageSetupDialog

Method of VcNet

This method lets you invoke the **Page Setup** dialog.

	Data Type	Explanation
Return value	System.Boolean	No significance {True}

Example Code VB.NET

```
VcNet1.ShowPageLayoutDialog()
```

Example Code C#

```
vcNet1.ShowPageLayoutDialog();
```

ShowPrintDialog

Method of VcNet

This method invokes the **Print** dialog, considering the parameters set in the **ShowPageLayoutDialog** dialog.

	Data Type	Explanation
Return value	System.Boolean	No significance {True}

Example Code VB.NET

```
VcNet1.ShowPrintDialog()
```

Example Code C#

```
vcNet1.ShowPrintDialog();
```

ShowPrinterSetupDialog

Method of VcNet

This method lets you invoke the Windows **Printer Setup** dialog.

	Data Type	Explanation
Return value	System.Boolean	No significance {True}

Example Code VB.NET

```
VcNet1.ShowPrinterSetupDialog()
```

Example Code C#

```
vcNet1.ShowPrinterSetupDialog();
```

ShowPrintPreviewDialog

Method of VcNet

This method invokes the print preview.

	Data Type	Explanation
Return value	System.Boolean	No significance {True}

Example Code VB.NET

```
VcNet1.ShowPrintPreviewDialog()
```

Example Code C#

```
vcNet1.ShowPrintPreviewDialog();
```

SuspendUpdate

Method of VcNet

For projects comprising many nodes, updating procedures may be very time consuming if actions are repeated for each node. You can accelerate the updating procedure by using the **SuspendUpdate** method. Bracket the code that describes the repeated action between **SuspendUpdate (True)** and **SuspendUpdate (False)** as in the below code example. This will get the nodes to be updated all at once and improve the performance.

	Data Type	Explanation
Return value	System.Boolean	SuspendUpdate(True): Start of the SuspendUpdate method/ SuspendUpdate(False): end of the SuspendUpdate method

Example Code VB.NET

```
VcNet1.SuspendUpdate (True)

   If updateFlag Then
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.14" Then
            node.DataField(13) = "X"
            node.Update
            counter = counter + 1
         End If
      Next node
   Else
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.14" Then
            node.DataField(13) = ""
            node.Update
            counter = counter + 1
         End If
      Next node
   End If

VcNet1.SuspendUpdate (False)
```

Example Code C#

```
bool updateFlag = true;
VcNodeCollection nodeCltn = vcvcNet1.NodeCollection;
int counter = 0;

vcvcNet1.SuspendUpdate(true);
if (updateFlag == true)
   {
   foreach (VcNode node in nodeCltn)
      {
      if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.14")) < 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("12.09.14")) < 0)
         {
         node.set_DataField(13,"");
         node.Update();
         counter = counter + 1;
         }
      }
   }
vcNet1.SuspendUpdate(false);
```

UpdateLinkRecord

Method of VcNet

This method lets you modify the data of an existing link record. The link record will be identified by the ID defined on the **DataDefinition** property page. This method is used when external modifications in the diagram have to be carried out on the display.

	Data Type	Explanation
Parameter: ⇨ linkRecord	System.Object	Link record
Return value	VcLink	Link updated

Example Code VB.NET

```
VcNet1.UpdateLinkRecord ("A100;A105;FS;0")
```

Example Code C#

```
vcNet1.UpdateLinkRecord( "A100;A105;FS;0");
```

UpdateNodeRecord

Method of VcNet

This method lets you modify the data of an existing node record. The node record will be identified by the ID set on the **DataDefinition** property page. This method is used when external modifications in the diagram have to be carried out on the display.

	Data Type	Explanation
Parameter: ⇨ nodeRecord	System.Object	Node record
Return value	VcNode	Node updated

Example Code VB.NET

```
VcNet1.UpdateNodeRecord ("A100;Activity 1;12.09.14;18.09.14;6;Planning")
```

Example Code C#

```
vcNet1.UpdateNodeRecord ("A100;Activity 1;12.09.14;18.09.14;6;Planning");
```

Zoom

Method of VcNet

This method lets you enlarge/reduce the diagram on the display by the specified percentage factor (enlarging the diagram: zoomFactor > 100, reducing the diagram: zoomFactor < 100).

	Data Type	Explanation
Parameter: ⇨ zoomFactor	System.Int16 {1...1000}	Zoom factor {11...999}, other values will remain unconsidered
Return value	System.Boolean	Zooming was performed successfully {True}

Example Code VB.NET

```
VcNet1.Zoom 120
```

Example Code C#

```
vcNet1.Zoom 120;
```

ZoomOnMarkedNodes

Method of VcNet

This method lets you zoom in on the nodes marked.

	Data Type	Explanation
Return value	Void	

Example Code VB.NET

```
VcNet1.ZoomOnMarkedNodes
```

Example Code C#

```
vcNet1.ZoomOnMarkedNodes;
```

Events

VcBoxLeftClicking

Event of VcNet

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
Properties:		
⇨ 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
	Possible Values:	
	.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 VcNet1_VcBoxLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.Xnet.VcBoxClickingEventArgs) Handles VcNet1.VcBoxLeftClicking
   TextBox1.Text = e.Box.FieldText(1)
End Sub
```

Example Code C#

```
private void vcNet1_VcBoxLeftClicking(object sender,
NETRONIC.XNetVcBoxClickingEventArgs e)
   {
   textBox1.Text = e.Box.get_FieldText(1);
   }
```

VcBoxLeftDoubleClicking

Event of VcNet

This event occurs when the user double-clicks the left mouse button on a box. The VcBox object hit and the mouse position (x,y-coordinates) are returned.

	Data Type	Explanation
Properties:		
⇨ 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
	Possible Values:	
	.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 VcNet1_VcBoxLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcBoxClickingEventArgs) Handles VcNet1.VcBoxLeftDoubleClicking
   e.Box.FieldText(0) = TextBox1.Text
End Sub
```

Example Code C#

```
private void vcNet1_VcBoxLeftDoubleClicking(object sender,
VcNetLib.VcBoxClickingEventArgs e)
   {
   e.Box.set_FieldText(1, textBox1.Text);
   }
```

VcBoxModified

Event of VcNet

This event occurs when the modification of the box is finished. The Box object modified and the modification type are passed as parameters.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcBoxModifiedEventArgs	Object specific to the event that is being handled

Properties of the VcBoxModifiedEventArgs object

	Data Type	Explanation
Properties:		
⇨ box	VcBox	Box modified

Example Code VB.NET

```
Private Sub VcNet1_VcBoxModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcBoxModifiedEventArgs) Handles VcNet1.VcBoxModified
   MsgBox("The box has been modified")
End Sub
```

Example Code C#

```
private void vcNet1_VcBoxModified(object sender, VcNetLib.VcBoxModifiedEventArgs
e)
   {
   MessageBox.Show("The box has been modified");
   }
```

VcBoxModifying

Event of VcNet

This event occurs when the user has modified a box interactively. The modified VcBox object and the modification type are returned.

This event should be used only for reading data of the current box, but not for modifying them. For modifying them please use **VcBoxModified**.

By setting the return status to **vcRetStatFalse**, the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcBoxModifyingEventArgs	Object specific to the event that is being handled

Properties of the VcBoxModifyingEventArgs object

	Data Type	Explanation
Properties:		
⇨ box	VcBox	Box modified
⇨ modificationType	VcBoxModificationTypes	Modification type
	Possible Values:	
	.vcBMTAnything 1	any modification
	.vcBMTNothing 0	no modification
	.vcBMTTextModified 4	text modified
	.vcBMTXYOffsetModified 2	Offset modified
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcBoxModifying(ByVal box As NETRONIC.XNet.VcBox, _
                  ByVal modificationType As _
                  VcNetLib.VcBoxModificationTypes, _
                  returnStatus As Variant)

Select Case modificationType
    Case vcBMTAnything: MsgBox "Box modification"
    Case vcBMTXYOffsetModified: MsgBox "Offset changed"
    Case vcBMTTextModified: MsgBox "Box field text changed"
End Select

End Sub
```

Example Code C#

```
private void vcNet1_VcBoxModifying(obje!ct sender,
NETRONIC.XNet.VcBoxModifyingEventArgs e)
   {
   switch(e.ModificationType)
      {
      case VcBoxModificationTypes.vcBMTAnything:
         MessageBox.Show("Box modification");
         break;
      case VcBoxModificationTypes.vcBMTXYOffsetModified:
         MessageBox.Show("Offset changed");
         break;
      case VcBoxModificationTypes.vcBMTTextModified:
          MessageBox.Show("Box field text changed");
         break;
       }
   }
```

VcBoxRightClicking

Event of VcNet

This event occurs when the user clicks the right mouse button on the box. The box object and the position of the mouse (x,y-coordinates) are returned, so that you can for example display your own context menu for the box at the appropriate location.

	Data Type	Explanation
Properties:		
⇨ 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
	Possible Values: .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 VcNet1_VcBoxRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcBoxClickingEventArgs) Handles VcNet1.VcBoxRightClicking
        PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
    End Sub
```

Example Code C#

```
private void vcNet1_VcBoxRightClicking(object sender,
NETRONIC.XNet.VcBoxClickingEventArgs e)
   {
   PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
   }
```

VcDataModified

Event of VcNet

This event occurs after data were interactively modified in the chart, i.e. after the below events:

- VcBoxModified
- VcLinkCreated
- VcLinkDeleted

- VcNodeCreated
- VcNodeDeleting
- VcNodeModified

This event allows you to set a marker to the application that reminds the user or the program to save the data before closing.

	Data Type	Explanation
Properties:		
⇦ (no parameter)		No parameter

VcDataRecordCreated

Event of VcNet

This event occurs when the interactive creation of a data record is completed. The data record object, the creation type (**vcDataRecordCreated** and **vcDataRecordCreatedByResourceScheduling** only) and the information whether the data record created is the only one or the last one of a data record collection (momentarily always **True**) are returned, so that depending data can be validated.

If a link or a node was created, you can in addition react to the analogous link or node event and verify additional graphical data (s. **VcNodeCreated** and **VcLinkCreated**).

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcDataRecordCreatedEventArgs	Object specific to the event that is being handled

Properties of the VcDataRecordCreatedEventArgs object

	Data Type	Explanation
Properties:		
⇨ dataRecord	VcDataRecord	DataRecord object created
⇨ creationType	VcCreationType	Creation type of data records

	Possible Values:	
	.vcDataRecordCreated 6	Data record created by interaction
	.vcLinkCreated 2	Link created by interaction
	.vcNodeCreated 1	Node created via mouse-click
	.vcNodesAndLinksCloned 4	Selected nodes were copied via dragging the mouse and pressing the the Ctrl button
	.vcNodeWithLinkCreated 3	Nodes and links created simultanously
⇨ isLast	System.Boolean	**True**:The data record created is the only one or the last one of a data record collection. **False**:The data record created is not the only one or the last one of a data record collection.

Example Code VB.NET

```
Private Sub VcNet1_VcDataRecordCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordCreatedEventArgs) Handles VcNet1.VcDataRecordCreated
        MsgBox(e.DataRecord.AllData)
    End Sub
```

Example Code C#

```
private void vcNet1_VcDataRecordCreated(object sender,
NETRONIC.XNet.VcDataRecordCreatedEventArgs e)
   {
   MessageBox.Show(e.DataRecord.AllData.ToString());
   }
```

VcDataRecordCreating

Event of VcNet

This event occurs when the user creates a an object that generates a data record. The generated data record object is returned, so that the data can be validated and, if necessary, a data base entry can be made.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **VcDataRecordCreated**.

By setting the return status the create operation can be inhibited.

If a link or a node was created, you can in addition react to the analogous link or node event and verify additional graphical data (s. **VcNodeCreating** and **VcLinkCreating**).

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcDataRecordCreatingEventArgs	Object specific to the event that is being handled

Properties of the VcDataRecordCreatingEventArgs object

	Data Type	Explanation
Properties:		
⇨ dataRecord	VcDataRecord	DataRecord object created
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.vcRetStatFalse 0	The data record will not be created.
	.vcRetStatOK 1	The data record will be created.

Example Code VB.NET

```
Private Sub VcNet1_VcDataRecordCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordCreatedEventArgs) Handles VcNet1.VcDataRecordCreated
        MsgBox(e.DataRecord.AllData)
End Sub
```

Example Code C#

```
private void vcNet1_VcDataRecordCreated(object sender,
NETRONIC.XNet.VcDataRecordCreatedEventArgs e)
{
MessageBox.Show(e.DataRecord.AllData.ToString());
}
```

VcDataRecordDeleted

Event of VcNet

This event occurs when the deletion of an object based on a data record is completed. The data record and the information whether the deleted data record is the only one or the last one of a data record collection are returned, so that depending data can be validated.

If a link or a node was deleted, you can in addition react to the analogous link or node event and verify additional graphical data (s. **VcNodeDeleted** and **VcLinkDeleted**).

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcDataRecordDeletedEventArgs	Object specific to the event that is being handled

Properties of the VcDataRecordDeletedEventArgs object

	Data Type	Explanation
Properties:		
⇨ dataRecord	VcDataRecord	Data record deleted
⇨ isLast	System.Boolean	**True**:The data record deleted is the only one or the last one of a data record collection. **False**:The data record deleted is not the only one or the last one of a data record collection.

VcDataRecordDeleting

Event of VcNet

This event occurs when a user deletes an object by the context menu if the object was based on a data record. The data record object to be deleted is returned, so that you can still verify its data and prohibit the deletion on a negative result by setting the return status.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcDataRecordDeletingEventArgs	Object specific to the event that is being handled

Properties of the VcDataRecordDeletingEventArgs object

	Data Type	Explanation
Properties:		
⇨ dataRecord	VcDataRecord	Data record object deleted
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.vcRetStatFalse 0	The data record will not be deleted.
	.vcRetStatOK 1	The data record will be deleted.

Example Code VB.NET

```
Private Sub VcNet1_VcDataRecordDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordDeletingEventArgs) Handles VcNet1.VcDataRecordDeleting
   'deny deletion of data record with a certain value
   If e.DataRecord.DataField(0) = "1" Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

Example Code C#

```
private void vcNet1_VcDataRecordDeleting(object sender,
NETRONIC.XNet.VcDataRecordDeletingEventArgs e)
   {
   // deny deletion of data record with a certain value
   if (e.DataRecord.get_DataField(0).Equals("1"))
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcDataRecordModified

Event of VcNet

This event occurs when the modification of the box is finished.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	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
Properties:		
⇨ dataRecord	VcDataRecord	Data record modified

Example Code VB.NET

```
Private Sub VcNet1_VcDataRecordModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordModifiedEventArgs) Handles VcNet1.VcDataRecordModified
   MsgBox("The data record has been modified")
End Sub
```

Example Code C#

```
private void vcNet1_VcDataRecordModified(object sender,
NETRONIC.XNet.VcDataRecordModifiedEventArgs e)
   {
   MessageBox.Show("The data record has been modified");
   }
```

VcDataRecordModifying

Event of VcNet

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
Parameter:		
⇨ sender	VcNet	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
Properties:		
⇨ dataRecord	VcDataRecord	Data record modified
⇨ modificationType	VcModificationTypes	Modification type
	Possible Values:	
	.vcAnything 1	Modification type cannot be identified.
	.vcChangedGroup 16	Group of the node was changed (occurs with nodes only).
	.vcMoved 8	Object was moved.
	.vcNothing 0	No modification
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.vcRetStatFalse 0	The modification will be revoked.
	.vcRetStatOK 1	The modification will be accepted.

VcDataRecordNotFound

Event of VcNet

This event occurs if a depending data record was not found. The index of the field of the current data record, which holds the key to the depending data record, is returned and thus offers some information on the data record not found.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	
⇨ e	VcDataRecordNotFoundEventArgs	

	Data Type	Explanation
Properties:		
⇦ index	System.Int32	Index of the field that contains the key of the depending data record

VcDiagramLeftClicking

Event of VcNet

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
Properties:		
⇨ x	System.Int32	x Coordinate of the mouse cursor
⇨ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values: .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 VcNet1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDiagramClickingEventArgs) Handles VcNet1.VcDiagramLeftClicking
   MsgBox("x: " + e.X.ToString() + " y: " + e.Y.ToString())
End Sub
```

Example Code C#

```
private void vcNet1_VcDiagramLeftClicking(object sender,
NETRONIC.XNet.VcDiagramClickingEventArgs e)
   {
   MessageBox.Show("x: " + e.X.ToString() + " y: " + e.Y.ToString());
   }
```

VcDiagramLeftDoubleClicking

Event of VcNet

This event occurs when the user double-clicks the left mouse button on the diagram in an empty space. The position of the mouse (x,y-coordinates) is returned.

	Data Type	Explanation
Properties:		
⇨ x	System.Int32	x Coordinate of the mouse cursor
⇨ y	System.Int32	Y coordinate of the mouse cursor
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values: .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 VcNet1_VcDiagramLeftDoubleClicking(ByVal sender As Object, ByVal e
NETRONIC.XNet.VcDiagramClickingEventArgs) Handles
VcNet1.VcDiagramLeftDoubleClicking
   VcNet1.Zoom(90)
End Sub
```

Example Code C#

```
private void vcNet1_VcDiagramLeftDoubleClicking(object sender,
NETRONIC.XNet.VcDiagramClickingEventArgs e)
   {
   vcNet1.Zoom(90);
   }
```

VcDiagramRightClicking

Event of VcNet

This event occurs when the user clicks the right mouse button on the diagram, not hitting any object. The position of the mouse (x,y-coordinates) is captured, so that you can for example display your own context menu at the appropriate location. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

	Data Type	Explanation
Properties:		
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcDiagramRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDiagramClickingEventArgs) Handles VcNet1.VcDiagramRightClicking
        PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
        e.ReturnStatus = VcNetLib.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Example Code C#

```
private void vcNet1_VcDiagramRightClicking(object sender,
NETRONIC.XNet.VcDiagramClickingEventArgs e)
   {
   PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcDragCompleting

Event of VcNet

This event is triggered at the source component to finish a drag procedure. It announces the drop effect.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcDragCompletingEventArgs	Object specific to the event that is being handled

Properties of the VcDragCompletingEventArgs object

	Data Type	Explanation
Properties:		
➩ DropEffect	System.Windows.Forms.DragDropEffects	Effects of a drag and drop operation

VcDragStarting

Event of VcNet

This event lets you specify and thus, if necessary, limit the allowed DropEffects on the start of a drag-operation. In addition, the property **LeavingControlWhileDraggingAllowed** has to be set to **True**. The property is preset to the combined value **DragDropEffects.Copy Or DragDropEffects.Move**. If, for instance, a node is always to be copied and not to be moved when being dragged out of the control, the property has to be set to **DragDropEffects.Copy**.

	Data Type	Explanation
Parameter:		
➩ sender	VcNet	Reference to the object that triggered the event
➩ e	VcDragStartingEventArgs	Object specific to the event that is being handled

Properties of the VcDragStartingEventArgs object

	Data Type	Explanation
Properties:		
➩ allowedEffects	System.Windows.Forms.AllowedEffects	Allowed DropEffects

VcErrorOccurring

Event of VcNet

This event occurs when an unexpected error occurs in the code of VARCHART XNet. NETRONIC tries to avoid errors in its products; if in spite of that still an error should occur, 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
Properties:		
⇨ description	System.String	Error description
⇨ scode	System.Int32	&h800a402f (constant)
⇨ source	System.String	Name of the control (constant)
⇨ helpFile	System.String	Help file: "" (constant)
⇨ helpContext	System.Int32	Help context: 0 (constant)
⇦ cancelDisplay	System.Boolean	If True, then no normal error with number 71 (which could be catched via VcErrorOccurring) will be output.

Example Code VB.NET

```
Private Sub VcNet1_VcErrorOccuring(ByVal sender As System.Object, _
                                             ByVal e As 
NETRONIC.XNet.VcErrorOccuringEventArgs)_
                                             Handles VcNet1.VcErrorOccuring
     MsgBox(e.Code + " - " + e.Text)
End Sub
```

Example Code C#

```
private void vcNet1_VcErrorOccuring(object sender, VcErrorOccuringEventArgs e)
{
   MessageBox.Show(e.Code + " - " + e.Text);
}
```

VcFieldSelecting

Event of VcNet

This event occurs, if a field in a box was selected. The selection can be inhibited by setting the return status.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	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
Properties:		
⇨ editObject	VcObject	Object edited
⇨ editObjectType	VcObjectType	Object type

	Possible Values:	
	.vcObjTypeBox 15	object type **box**
	.vcObjTypeGroup 7	object type **group**
	.vcObjTypeLinkCollection 3	object type **link collection**
	.vcObjTypeNode 2	object type **node**
	.vcObjTypeNone 0	no object
⇨ 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	
	Possible Values:	
	.vcRetStatFalse 0	The field will not be selected.
	.vcRetStatOK 1	The field will be selected.

VcGiveFeedbackOnNodeCreating

Event of VcNet

This event occurs when node creation mode is switched on. X and Y denote the position of the mouse pointer relative to the control's origin of ordinates. If the default value **True** of **creationAllowed** is not changed, creating nodes at this cursor position is allowed . If **creationAllowed** is set to **False**, creating nodes is not allowed. This can be used to rule out from the start the creation of nodes in certain parts of the diagram (this may be the case in areas with no groups as shown in the code sample below).

	Data Type	Explanation
Properties:		
⇨ X	System.Int32	X coordinate of the mouse cursor
⇨ Y	System.Int32	Y coordinate of the mouse cursor
⇔ CreationAllowed	System.Boolean	Return status

Example Code C#

```
private void vcNet1_VcGiveFeedbackForNodeCreating(object sender,
VcGiveFeedbackForNodeCreatingEventArgs e)
{
   object obj = null;
   VcObjectType objType = VcObjectType.vcObjTypeNone;
   vcNet1.IdentifyObjectAt(e.X, e.Y, ref obj, ref objType);
   if (objType == VcObjectType.vcObjTypeNone)
      e.CreationAllowed = false;
}
```

VcGroupCreated

Event of VcNet

This event occurs when the user creates a new group, i.e. when he creates the first node with a new group code in the ActiveX. The new group object is captured, so that a validation and if necessary a data base entry can be made.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group created
⇔ returnStatus	VcReturnStatus	Return status: at the moment without function
	Possible Values:	
	.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.

VcGroupDeleting

Event of VcNet

This event occurs when the user deletes or moves the last node of a group so that the group gets empty and therefore is deleted. The group object is captured. The deletion of a group cannot be prevented by setting the return status.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group hit
⇔ returnStatus	VcReturnStatus	
	Possible Values:	
	.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 VcNet1_VcGroupDeleting(ByVal sender As Object, ByVal e As
NETRONIC.Xnet.VcGroupDeletingEventArgs) Handles VcNet1.VcGroupDeleting
   If e.Group.Name = "A" Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
      MsgBox("Group A cannot be deleted")
   End If
End Sub
```

Example Code C#

```
private void vcNet1_VcGroupDeleting(object sender,
NETRONIC.XNet.VcGroupDeletingEventArgs e)
   {
   if (e.Group.Name == "A")
      {
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      MessageBox.Show("Group A cannot be deleted");
      }
   }
```

VcGroupLeftClicking

Event of VcNet

This event occurs when the user clicks the left mouse button on a group. The group object and the mouse position (x,y-coordinates) are captured.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group hit
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcGroupLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupClickingEventArgs) Handles VcNet1.VcGroupLeftClicking
   MsgBox(e.Group.SubGroups.Count)
End Sub
```

Example Code C#

```
private void vcNet1_VcGroupLeftClicking(object sender,
NETRONIC.XNet.VcGroupClickingEventArgs e)
   {
   MessageBox.Show(e.Group.SubGroups.Count.ToString());
   }
```

VcGroupLeftDoubleClicking

Event of VcNet

This event occurs when the user double-clicks the left mouse button on a group. The group object and the mouse position (x,y-coordinates) are captured.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group hit
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcGroupLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupClickingEventArgs) Handles VcNet1.VcGroupLeftDoubleClicking
   MsgBox(e.Group.Name)
End Sub
```

Example Code C#

```
private void vcNet1_VcGroupLeftDoubleClicking(object sender,
NETRONIC.XNet.VcGroupClickingEventArgs e)
   {
   MessageBox.Show(e.Group.Name);
   }
```

VcGroupModified

Event of VcNet

This event occurs when the interactive collapsing or expanding of a clustered group is finished.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group modified
⇨ modificationType	VcGroupModificationTypes	Type of modification
	Possible Values:	
	.vcGMTCollapsing 2	Group collapsed

	.vcGMTExpanding 4	Group expanded

Example Code VB.NET

```
Private Sub VcNet1_VcGroupModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupModifiedEventArgs) Handles VcNet1.VcGroupModified
  MsgBox("The group has been modified.")
End Sub
```

Example Code C#

```
private void vcNet1_VcGroupModified(object sender,
NETRONIC.XNet.VcGroupModifiedEventArgs e)
   {
   MessageBox.Show("The group has been modified");
   }
```

VcGroupModifying

Event of VcNet

This event occurs when in the clustering mode a user interactively collapses a group (modificationType = vcGMTCollapsing) or expandes a group (vcGMTExpanding). The group object, the type of modification and the return status are returned. If you set the return status to **vcRetStatFalse**, the operation will be revoked.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group modified
⇨ modificationType	VcGroupModificationTypes	Type of modification
	Possible Values:	
	.vcGMTCollapsing 2	Group collapsed
	.vcGMTExpanding 4	Group expanded
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcGroupModifying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupModifyingEventArgs) Handles VcNet1.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 vcNet1_VcGroupModifying(object sender,
NETRONIC.XNet.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;
       }
   }
```

VcGroupRightClicking

Event of VcNet

This event occurs when the user clicks the right mouse button on a group of nodes. The group object and the mouse position (x,y-coordinates) are captured, so that you can display your own context menu at the appropriate position. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

	Data Type	Explanation
Properties:		
⇨ group	VcGroup	Group hit
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⬄ returnStatus	VcReturnStatus	
	Possible Values: .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 VcNet1_VcGroupRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupClickingEventArgs) Handles VcNet1.VcGroupRightClicking
        PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
        e.ReturnStatus = VcNetLib.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Example Code C#

```
private void vcNet1_VcGroupRightClicking(object sender,
NETRONIC.XNet.VcGroupClickingEventArgs e)
   {
   PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcHelpRequested

Event of VcNet

This event occurs if the user presses the F1 key on a dialog at run time. The application can invoke its own help system, to offer help specific to the dialog and to the application.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcHelpRequestedEventArgs	Object specific to the event that is being handled

Properties of the VcHelpRequestedEventArgs object

	Data Type	Explanation
Properties:		
⇨ DialogType	VcDialogType	Dialog for which help was requested
	Possible Values:	
	.vcEditDataRecordDialog 5400	Help was requested for the **Edit Data Record** dialog.
	.vcPageSetupDialog 4097	Help was requested for the **Page Set Up** dialog.
	.vcPrintPreviewDialog 4096	Help was requested for the **Print Preview** dialog.

VcInPlaceEditorShowing

Event of VcNet

This event occurs when the implemented editor is started.

TThe event will be activated only if the property **InPlaceEditingAllowed** is set to True.

By setting the return status to **False** this event can be inhibited so that your own editor can be started at the coordinates passed.

	Data Type	**Explanation**
Properties:		
⇨ editObject	VcObject	Object edited
⇨ editObjectType	VcObjectType	Object type
	Possible Values:	
	.vcObjTypeBox 15	object type **box**
	.vcObjTypeGroup 7	object type **group**
	.vcObjTypeLinkCollection 3	object type **link collection**
	.vcObjTypeNode 2	object type **node**
	.vcObjTypeNone 0	no object
⇨ 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	
	Possible Values:	
	.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 VcNet1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcInPlaceEditorShowingEventArgs) Handles
VcNet1.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 + VcNet1.Left
                    TextBox1.Top = e.FldRectVisible.Top + VcNet1.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 + VcNet1.Left
                    DateTimePicker1.Top = e.FldRectVisible.Top + VcNet1.Top
                    DateTimePicker1.Value = node.DataField(0)
                    DateTimePicker1.Visible = True
                    DateTimePicker1.Focus()
                Case 13     'Employee
                    ComboBox1.Left = e.FldRectVisible.Left + VcNet1.Left
                    ComboBox1.Top = e.FldRectVisible.Top + VcNet1.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
```

Example Code C#

```
private void vcNet1_VcInPlaceEditorShowing(object sender,
NETRONIC.XNet.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 + vcNet1.Left;
            textBox1.Top = e.FldRectVisible.Top + vcNet1.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 + vcNet1.Left;
            dateTimePicker1.Top = e.FldRectVisible.Top + vcNet1.Top;
            dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
            dateTimePicker1.Visible = true;
            dateTimePicker1.Focus();
            break;
         case 13: //Employee
            comboBox1.Left = e.FldRectVisible.Left + vcNet1.Left;
            comboBox1.Top = e.FldRectVisible.Top + vcNet1.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;
        }
   }
```

VcLegendViewClosed

Event of VcNet

This event occurs when the legend view popup window is closed.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	
⇨ e	VcLEgendViewClosedEventArgs	

	Data Type	Explanation
Properties:		
⇦ (no parameter)		

Example Code VB.NET

```
Private Sub VcNet1_VcLegendViewClosed(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLegendViewClosedEventArgs) Handles VcNet1.VcLegendViewClosed
   MsgBox("Do you want to close the legend view window?", MsgBoxStyle.OKCancel)
End Sub
```

Example Code C#

```
private void vcNet1_VcLegendViewClosed(object sender,
NETRONIC.XNet.VcLegendViewClosedEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to close the legend view
window?", "Closing legend view window", MessageBoxButtons.OKCancel);
   }
```

VcLinkCreated

Event of VcNet

This event occurs when the interactive creation of a link is completed. The link object, the creation type and the information whether the created link is the only link or the last link of a link collection are passed, so that a validation can be made.

	Data Type	Explanation
Properties:		
⇨ link	VcLink	Link created
⇨ creationType	VcCreationType	Creation type
	Possible Values:	
	.vcDataRecordCreated 6	Data record created by interaction
	.vcLinkCreated 2	Link created by interaction
	.vcNodeCreated 1	Node created via mouse-click
	.vcNodesAndLinksCloned 4	Selected nodes were copied via dragging the mouse and pressing the the Ctrl button
	.vcNodeWithLinkCreated 3	Nodes and links created simultanously
⇨ isLast	System.Boolean	The created link is/is not the only link or the last link of a link collection.

Example Code VB.NET

```
Private Sub VcNet1_VcLinkCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinkCreatedEventArgs) Handles VcNet1.VcLinkCreated
        MsgBox(e.Link.AllData)
End Sub
```

Example Code C#

```
private void vcNet1_VcLinkCreated(object sender,
NETRONIC.XNet.VcLinkCreatedEventArgs e)
   {
   MessageBox.Show(e.Link.AllData.ToString());
   }
```

VcLinkCreating

Event of VcNet

This event occurs when the user creates a link between two nodes. The link object is captured, so that a validation and if necessary a data base entry can be made.

This event should be used only for reading data of the current link, but not for modifying them. For modifying data please use **VcLinkCreated**

	Data Type	Explanation
Properties:		
⇨ link	VcLink	Link created
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcLinkCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinkCreatedEventArgs) Handles VcNet1.VcLinkCreated
MsgBox(e.Link.AllData)
End Sub
```

Example Code C#

```
private void vcNet1_VcLinkCreated(object sender, VcNetLib.VcLinkCreatedEventArgs
e)
{
   MessageBox.Show(e.Link.AllData.ToString());
}
```

VcLinkDeleted

Event of VcNet

This event occurs when the deletion of a link is completed. The link object and the information whether the created link is the only link or the last link of a link collection are returned, so that a validation can be made.

	Data Type	Explanation
Properties:		
⇨ link	VcLink	Link deleted
⇨ isLast	System.Boolean	The deleted link is/is not the only link or the last link of a link collection.

VcLinkDeleting

Event of VcNet

This event occurs when the user deletes a link by key or context menu. The link object is captured, so that a validation can be done. If you set the returnStatus to **vcRetStatFalse**, the link will not be deleted.

	Data Type	Explanation
Properties:		
⇨ link	VcLink	Link deleted
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values: .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 VcNet1_VcLinkDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinkDeletingEventArgs) Handles VcNet1.VcLinkDeleting
   'deny deletion of link with a certain predecessor
   If e.Link.PredecessorNode.DataField(0) = "1" Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Example Code C#

```
private void vcNet1_VcLinkDeleting(object sender,
NETRONIC.XNet.VcLinkDeletingEventArgs e)
{
   // deny deletion of link with a certain predecessor
   if (e.Link.PredecessorNode.get_DataField(0).Equals("1"))
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

VcLinkModified

Event of VcNet

This event occurs when the modification of the link specified was finished.

The node object and the information whether the created node is the only node or the last node of a node collection (always **True**) are returned, so that the data can be validated.

	Data Type	Explanation
Properties:		
⇨ link	VcLink	Link created
⇨ isLast	System.Boolean	The created link is/is not the only link or the last link of a link collection.

Example Code VB.NET

```
Private Sub VcNet1_VcLinkModified(ByVal sender As System.Object, _
ByVal e As NETRONIC.XNet.VcLinkModifiedEventArgs) _
Handles VcNet1.VcLinkModified
    'modify a record in the underlying database of the application
    modifyDataRecord(e.Link.AllData)
End Sub
```

Example Code C#

```
private void vcNet1_VcLinkModified(object sender, VcLinkModifiedEventArgs e)
{
    //modify a record in the underlying database of the application
      modifyDataRecord(e.Link.AllData);
}
```

VcLinkModifying

Event of VcNet

This event occurs when the user has modified a link. In the course of this, the position of the link or a value in the **Edit Data** dialog may have been changed. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

This event should be used only for reading data of the current link, but not for modifying them. For modifying data please use **VcLinkModified**.

	Data Type	Explanation
Properties:		
⇨ link	VcLink	Link after modification
⇨ oldlink	VcLink	Link before modification
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcLinkModifying(ByVal sender As System.Object, _
                    ByVal e As NETRONIC.XNet.VcLinkModifyingEventArgs) _
                    Handles VcNet1.VcLinkModifying
'deny any modification
e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

Example Code C#

```
private void vcNet1_VcLinkModifying(object sender, VcLinkModifyingEventArgs e)
{   //deny any modification
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

VcLinksLeftClicking

Event of VcNet

This event occurs when the user clicks the left mouse button on a link or on several overlapping links. A LinkCollection object and the mouse position (x,y-coordinates) are captured and passed.

	Data Type	Explanation
Properties:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksLeftClicking
   Dim linkCltn As VcLinksCollection
   Dim link As VcLink
   linkCltn = VcNet1.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 vcNet1_VcLinksLeftClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
   {
   VcLinkCollection linkCltn = vcNet1.LinkCollection;
   // set certain data field of all links
   foreach (VcLink link in linkCltn)
      link.set_DataField(2, "A");
   }
```

VcLinksLeftDoubleClicking

Event of VcNet

This event occurs when the user double-clicks the left mouse button on a link or on several overlapping links. A LinkCollection object and the mouse position (x,y-coordinates) are captured and passed.

	Data Type	Explanation
Properties:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⬄ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksLeftClicking
   Dim linkCltn As VcLinkCollection
   Dim link As VcLink
   linkCltn = VcNet1.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 vcNet1_VcLinksLeftClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
   {
   VcLinkCollection inkCltn = vcNet1.LinkCollection;
   // set certain data field of all links
   foreach (VcLink link in linkCltn)
      node.set_DataField(2, "A");
   }
```

VcLinksMarked

Event of VcNet

This event occurs after the operation of marking or unmarking links was finished.

	Data Type	Explanation
Properties:		
⇦ (no parameter)		No parameter

Example Code VB.NET

```
Private Sub VcNet1_VcLinksMarked(ByVal sender As System.Object, _
                                                            ByVal e As
NETRONIC.XNet.VcLinksMarkedEventArgs)_
                                                            Handles
VcNet1.VcLinksMarked
    MsgBox("Links have been successfully marked.")
End Sub
```

Example Code C#

```
private void vcNet1_VcLinksMarked(object sender, VcLinksMarkedEventArgs e)
{
   MessageBox.Show("Links have been successfully marked.");
}
```

VcLinksMarking

Event of VcNet

This event occurs when the user selects links for marking or when he unmarks marked links by a click into an empty section of the diagram. If the user marked links, the LinkCollection contains the selected nodes. If the user unmarked links by a click into an empty place, the link collection will be empty.

If you set the return status to **vcRetStatFalse**, you have to mark or unmark links yourself.

The data passed by this event can be read, but must not be modified. For modifying them please use **VcLinksMarked**.

	Data Type	Explanation
Properties:		
⇨ linkCollection	VcLinkCollection	NodeCollection that contains the nodes selected by the user. If the user clicked in the diagram, the collection is empty.

⇨ button	System.Int16	Indicates in which way the buttons were marked: **0**: by keyboard, **1**: left mouse button pressed, **2**: right mouse button pressed, **4**: mouse button pressed
⇨ shift	System.Int16	Indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers can be set, indicating that some, all, or none of the keys are depressed, respectively. When some keys are pressed, their values add up. For example, if both the Ctrl and Alt keys were pressed, the value of **shift** would equal "6".
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcLinksMarking(ByVal sender As System.Object, _
                                                            ByVal e As
NETRONIC.XNet.VcLinksMarkingEventArgs) _
                                                           Handles
VcNet1.VcLinksMarking
     If MsgBox("Mark this node?", vbYesNo, "Marking nodes") = vbNo Then
         e.ReturnStatus = VcReturnStatus.vcRetStatFalse
     End If
End Sub
```

Example Code C#

```
private void vcNet1_VcLinksMarking(object sender, VcLinksMarkingEventArgs e)
{
    if (MessageBox.Show("Mark this node?",
                                         "Marking nodes",
                                          MessageBoxButtons.YesNo) ==
DialogResult.No)
     e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

VcLinksRightClicking

Event of VcNet

This event occurs when the user clicks the right mouse button on a link or on several overlapping links. The LinkCollection object and the mouse position (x,y-coordinates) are captured and passed, so that you can display your own context menu at the appropriate position. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

	Data Type	Explanation
Properties:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit

⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcLinksRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksRightClicking
        PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
        e.ReturnStatus = VcNetLib.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Example Code C#

```
private void vcNet1_VcLinksRightClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
   {
   PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcMouseDoubleClicking

Event of VcNet

This event occurs when the user presses a mouse button twice.

Please also see the property **MouseProcessingEnabled**. If you wish to retrieve the state of the keys **Shift**, **Strg** or **Alt**, there is the static method **System.Windows.Forms.Control.ModifierKeys**.

	Data Type	Explanation
Properties:		
⇨ button	System.Int16	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ shift	System.Int16	An integer that corresponds to the status of the keys Shift, Ctrl and Alt at the moment when the event occurs. The Shift parameter is a bit field with the values: Shift key (Bit 0), Ctrl key (Bit 1) and Alt key (Bit 2). These bits correspond to the value 1, 2 and 4. Several, all or none of these bits can be set to indicate that several, all or no keys are pressed. If the Ctrl key and the Alt key are pressed, the value of Shift will be 6.
⇨ x	System.Int32	X coordinate of the mouse cursor

⇨ y	System.Int32	Y coordinate of the mouse cursor

VcMouseDown

Event of VcNet

This event occurs when the user presses a mouse button.

Please also see the property **MouseProcessingEnabled**. If you wish to retrieve the state of the keys **Shift**, **Strg** or **Alt**, there is the static method **System.Windows.Forms.Control.ModifierKeys**.

	Data Type	Explanation
Properties:		
⇨ button	System.Int16	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ shift	System.Int16	An integer that corresponds to the status of the keys Shift, Ctrl and Alt at the moment when the event occurs. The Shift parameter is a bit field with the values: Shift key (Bit 0), Ctrl key (Bit 1) and Alt key (Bit 2). These bits correspond to the value 1, 2 and 4. Several, all or none of these bits can be set to indicate that several, all or no keys are pressed. If the Ctrl key and the Alt key are pressed, the value of Shift will be 6.
⇨ x	System.Int32	X coordinate of the mouse cursor
⇨ y	System.Int32	Y coordinate of the mouse cursor

VcMouseMove

Event of VcNet

This event occurs when the user moves the mouse.

Please also see the property **MouseProcessingEnabled**. If you wish to retrieve the state of the keys **Shift**, **Strg** or **Alt**, there is the static method **System.Windows.Forms.Control.ModifierKeys**.

	Data Type	Explanation
Properties:		
⇨ button	System.Int16	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.

⇨ shift	System.Int16	An integer that corresponds to the status of the keys Shift, Ctrl and Alt at the moment when the event occurs. The Shift parameter is a bit field with the values: Shift key (Bit 0), Ctrl key (Bit 1) and Alt key (Bit 2). These bits correspond to the value 1, 2 and 4. Several, all or none of these bits can be set to indicate that several, all or no keys are pressed. If the Ctrl key and the Alt key are pressed, the value of Shift will be 6.
⇨ x	System.Int32	X coordinate of the mouse cursor
⇨ y	System.Int32	Y coordinate of the mouse cursor

VcMouseUp

Event of VcNet

This event occurs when the user releases the left mouse button after pressing.

Please also see the property **MouseProcessingEnabled**. If you wish to retrieve the state of the keys **Shift**, **Strg** or **Alt**, there is the static method **System.Windows.Forms.Control.ModifierKeys**.

	Data Type	Explanation
Properties:		
⇨ button	System.Int16	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ shift	System.Int16	An integer that corresponds to the status of the keys Shift, Ctrl and Alt at the moment when the event occurs. The Shift parameter is a bit field with the values: Shift key (Bit 0), Ctrl key (Bit 1) and Alt key (Bit 2). These bits correspond to the value 1, 2 and 4. Several, all or none of these bits can be set to indicate that several, all or no keys are pressed. If the Ctrl key and the Alt key are pressed, the value of Shift will be 6.
⇨ x	System.Int32	X coordinate of the mouse cursor
⇨ y	System.Int32	Y coordinate of the mouse cursor

VcNodeCreated

Event of VcNet

This event occurs when the interactive creation of a node is completed. The node object, the creation type and the information whether the created node is the only node or the last node of a node collection are returned, so that a validation can be made.

	Data Type	Explanation
Properties:		
⇨ node	VcNode	Node created
⇨ creationType	VcCreationType	Creation type
	Possible Values:	
	.vcDataRecordCreated 6	Data record created by interaction
	.vcLinkCreated 2	Link created by interaction
	.vcNodeCreated 1	Node created via mouse-click
	.vcNodesAndLinksCloned 4	Selected nodes were copied via dragging the mouse and pressing the the Ctrl button
	.vcNodeWithLinkCreated 3	Nodes and links created simultanously
⇨ isLast	System.Boolean	The created node is/is not the only node or the last node of a node collection.

Example Code VB.NET

```
Private Sub VcNet1_VcNodeCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeCreatedEventArgs) Handles VcNet1.VcNodeCreated
        MsgBox(e.Node.AllData)
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeCreated(object sender,
NETRONIC.XNet.VcNodeCreatedEventArgs e)
{
MessageBox.Show(e.Node.AllData.ToString());
}
```

VcNodeCreating

Event of VcNet

This event occurs when the user creates a node. The node object is captured, so that a validation can be made. This can be important if the user made changes in the activated dialog **Edit Data**.

This event should be used only for reading data of the current node, but not for modifying them. For modifying data please use **VcNodeCreated**.

	Data Type	Explanation
Properties:		
⇨ node	VcNode	Node to be created
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.vcRetStatDefault 2	The default behavior remains unchanged.
	.vcRetStatFalse 0	The default behavior will not be performed.

	.vcRetStatNoPopup 4 .vcRetStatOK 1	The popup of the context menu is inhibited. The default behavior will be performed.

Example Code VB.NET

```
Private Sub VcNet1_VcNodeCreating(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeCreatingEventArgs) Handles VcNet1.VcNodeCreating
        MsgBox("Show your own dialog")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End Sub
```

Example Code C#

```
private void vcNet1_VcNodeCreating(object sender,
NETRONIC.XNet.VcNodeCreatingEventArgs e)
   {
   MessageBox.Show("Show your own dialog");
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcNodeDeleted

Event of VcNet

This event occurs when the interactive deletion of a node is completed. The node object and the information whether the deleted node was the last one of a batch are returned for data validation.

	Data Type	Explanation
Properties:		
⇨ node	VcNode	Node deleted
⇨ isLast	System.Boolean	The deleted node is/is not last node of a batch.

VcNodeDeleting

Event of VcNet

This event occurs when the user deletes a node. The user can delete a node by the context menu. The node object is captured and passed, so that a validation can be done. If you set the returnStatus to **vcRetStatFalse**, the node will not be deleted.

	Data Type	Explanation
Properties:		
⇨ node	VcNode	Node object
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	

	.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 VcNet1_VcNodeDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeDeletingEventArgs) Handles VcNet1.VcNodeDeleting
        'deny the deletion of the last node in the chart
        If VcNet1.NodeCollection.Count = 1 Then
            e.ReturnStatus = VcReturnStatus.vcRetStatFalse
            MsgBox("The last node in the chart cannot be deleted.")
        End If
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeDeleting(object sender,
NETRONIC.XNet.VcNodeDeletingEventArgs e)
   {
   //deny the deletion of the last node in the chart
   if (vcNet1.NodeCollection.Count == 1)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      MessageBox.Show("The last node in the chart cannot be deleted.");
   }
```

VcNodeLeftClicking

Event of VcNet

This event occurs when the user clicks the left mouse button on a node. The node object and the mouse position (x,y-coordinates) are captured and passed.

	Data Type	**Explanation**
Properties:		
⇨ node	VcNode	Node object
⇨ location	VcLocation	Placed in the chart
	Possible Values:	
	.vcInDiagram 1	Located in the node area
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.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 vcNet1_VcNodeLeftClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
   {
   //change data field of the node
    e.Node.set_DataField(4,Convert.ToInt64(e.Node.get_DataField(4)));
   }
```

VcNodeLeftDoubleClicking

Event of VcNet

This event occurs when the user double-clicks the left mouse button on a node. The node object, the mouse position (x,y-coordinates) and the location in the diagram are captured and passed. After returning, the **Edit data** dialog of the node automatically will be invoked. If you set the returnStatus to **vcRetStatFalse**, you can suppress the dialog.

	Data Type	Explanation
Properties:		
⇨ node	VcNode	Node object
⇨ location	VcLocation	Placed in the chart
	Possible Values: .vcInDiagram 1	 Located in the node area
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values: .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 VcNet1_VcNodeLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftDoubleClicking
        MsgBox("Show your own dialog")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeLeftDoubleClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
   {
   MessageBox.Show("Show your own dialog");
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcNodeModifiedEx

Event of VcNet

This event occurs when the modification of the marked node was completed.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	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
Properties:		
⇨ node	VcNode	Node created
⇨ isLast	System.Boolean	The created node is/is not the only node or the last node of a node collection.

Example Code VB.NET

```
Private Sub VcNet1_VcNodeModifiedEx(ByVal sender As System.Object, _
                                  ByVal e As
NETRONIC.XNet.VcNodeModifiedExEventArgs)
                                 Handles VcNet1.VcNodeModifiedEx
     'modify a record in the underlying database of the application
     modifyDataRecord(e.Node.AllData)
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeModifiedEx(object sender,

VcNodeModifiedExEventArgs e)
{
  //modify a record in the underlying database of the application
  modifyDataRecord(e.Node.AllData);
}
```

VcNodeModifying

Event of VcNet

This event occurs when the user modifies a node. In the course of this, 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
Parameter:		
⇨ sender	VcNet	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
Properties:		
⇨ oldNode	VcNode	Node before the modification
⇨ node	VcNode	Node to be modified
⇨ modificationType	VcModificationTypes	Type of modification
	Possible Values:	
	.vcAnything 1	Modification type cannot be identified.
	.vcChangedGroup 16	Group of the node was changed (occurs with nodes only).
	.vcMoved 8	Object was moved.
	.vcNothing 0	No modification
⬄ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeModifyingEventArgs) Handles VcNet1.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

End Sub
```

Example Code C#

```
private void vcNet1_VcNodeModifying(object sender,
NETRONIC.XNet.VcNodeModifyingEventArgs e)
   {
   //revoke the modification if the node would change the group
   if (e.ModificationType == VcModificationTypes.vcChangedGroup)
      {
      MessageBox.Show("The node cannod be moved into another group.");
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      }
   }
```

VcNodeRightClicking

Event of VcNet

This event occurs when the user clicks the right mouse button on a node. The node object and the mouse position (x,y-coordinates) are captured, so that you can display a context menu at the appropriate position. If you set the returnStatus to **vcRetStatNoPopup**, the integrated menu be revoked.

This event should be used only for reading data of the current node, but not for modifying them. For modifying data please use **VcNodesMarked**.

	Data Type	Explanation
Properties:		
⇨ node	VcNode	Node object
⇨ location	VcLocation	Placed in the chart
	Possible Values: .vcInDiagram 1	 Located in the node area
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values: .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 VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking
        PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XNet.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Example Code C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
   {
   PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcNodesMarked

Event of VcNet

This event occurs after the operation of marking or unmarking nodes was finished.

	Data Type	Explanation
Properties:		
⇦ (no parameter)		No parameter

Example Code VB.NET

```
Private Sub VcNet1_VcNodesMarked(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodesMarkedEventArgs) Handles VcNet1.VcNodesMarked
   MsgBox("Nodes have been marked successfully.")
End Sub
```

Example Code C#

```
private void vcNet1_VcNodesMarked(object sender,
NETRONIC.XNet.VcNodesMarkedEventArgs e)
   {
   MessageBox.Show("Nodes have been marked successfully.");
   }
```

VcNodesMarking

Event of VcNet

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**
Properties:		
⇨ nodeCollection	VcNodeCollection	NodeCollection that contains the nodes selected by the user. If the user clicked in the diagram, the collection is empty.
⇨ button	System.Int16	Indicates in which way the buttons were marked: **0**: by keyboard, **1**: left mouse button pressed, **2**: right mouse button pressed, **4**: mouse button pressed
⇨ shift	System.Int16	Indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers can be set, indicating that some, all, or none of the keys are depressed, respectively. When some keys are pressed, their values add up. For example, if both the Ctrl and Alt keys were pressed, the value of **shift** would equal "6".
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcNodesMarking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodesMarkingEventArgs) Handles VcNet1.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 vcNet1_VcNodesMarking(object sender,
NETRONIC.XNet.VcNodesMarkingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Mark this node?", "Marking nodes",
MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcStatusLineTextShowing

Event of VcNet

This event gives you an information about a node that was touched with the mouse cursor. You can e.g. display this information in a status line. The information itself is taken from a data field you can define by the configuration file (IFD, Feld IF_ID2) und is defined by default as the field with the index 4.

	Data Type	Explanation
Properties:		
⇨ text	System.String	Text

Example Code VB.NET

```
Private Sub VcNet1_VcStatusLineTextShowing(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcStatusLineTextShowingEventArgs) Handles
VcNet1.VcStatusLineTextShowing
   TextBox1.Text = e.Text
End Sub
```

Example Code C#

```
private void vcvcNet1_VcStatusLineTextShowing(object sender,
NETRONIC.XNet.VcStatusLineTextShowingEventArgs e)
   {
   textBox1.Text = e.Text;
   }
```

VcTextEntrySupplying

Event of VcNet

This event occurs when a text is displayed and only when the property **EnableSupplyTextEntryEvent** is set to **True**. You can use this event for editing the texts of the names of days and months.

	Data Type	Explanation
Parameter:		
⇨ sender	VcNet	Reference to the object that triggered the event
⇨ e	VcTextEntrySupplyingEventArgs	Object specific to the event that is being handled

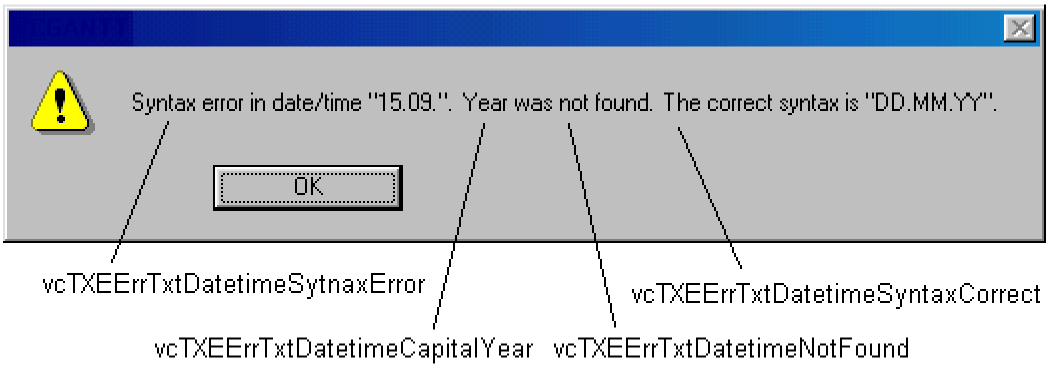
Properties of the VcTextEntrySupplyingEventArgs object

	Data Type	Explanation
Properties:		
⇨ controlIndex	VcTextEntryIndex	Text to be replaced
	Possible Values:	
	.vcTXECtxmenArrange 2150	Text in context menu: **Arrnge nodes**
	.vcTXECtxmenArrowMode 2116	Text in context menu: **Pointer mode**
	.vcTXECtxmenCopyNodes 2152	Text in context menu: **Copy nodes**
	.vcTXECtxmenCreateNodesAndLinksMode 2117	Text in context menu: **Create nodes and links mode**
	.vcTXECtxmenCutNodes 2151	Text in context menu: **Cut nodes**
	.vcTXECtxmenDeleteLink 2102	Text in context menu: **Delete link**
	.vcTXECtxmenDeleteNode 2101	Text in context menu: **Delete nodes**
	.vcTXECtxmenEditLink 2154	Text in context menu: **Edit Link**
	.vcTXECtxmenEditNode 2100	Text in context menu: : **Edit data**
	.vcTXECtxmenFilePrint 2122	Text in context menu: **Print**
	.vcTXECtxmenFilePrintPreview 2121	Text in context menu: **Print preview**
	.vcTXECtxmenFilePrintSetup 2120	Text in context menu: **Print setup**
	.vcTXECtxmenFullDiagram 2156	Text in context menu: **Restore full net**
	.vcTXECtxmenGraphicExport 2123	Text in context menu: **Export graphics**
	.vcTXECtxmenPageLayout 2119	Text in context menu: **Page setup**
	.vcTXECtxmenPasteNodes 2153	Text in context menu: **Paste nodes**
	.vcTXEDateAM 2225	text output for **a. m.**
	.vcTXEDateCW 2223	text output for **calendar week**
	.vcTXEDateDay0 2212	text output for **Monday**
	.vcTXEDateDay1 2213	text output for **Tuesday**
	.vcTXEDateDay2 2214	text output for **Wednesday**
	.vcTXEDateDay3 2215	text output for **Thursday**
	.vcTXEDateDay4 2216	text output for **Friday**
	.vcTXEDateDay5 2217	text output for **Saturday**
	.vcTXEDateDay6 2218	text output for **Sunday**
	.vcTXEDateMonth0 2200	text output for **January**
	.vcTXEDateMonth1 2201	text output for **February**
	.vcTXEDateMonth10 2210	text output for **November**
	.vcTXEDateMonth11 2211	text output for **December**
	.vcTXEDateMonth2 2202	text output for **March**
	.vcTXEDateMonth3 2203	text output for **April**
	.vcTXEDateMonth4 2204	text output for **Mai**
	.vcTXEDateMonth5 2205	text output for **June**
	.vcTXEDateMonth6 2206	text output for **July**
	.vcTXEDateMonth7 2207	text output for **August**
	.vcTXEDateMonth8 2208	text output for **September**
	.vcTXEDateMonth9 2209	text output for **October**
	.vcTXEDateOClock 2224	text output for **o'clock**
	.vcTXEDatePM 2226	text output for **p. m.**
	.vcTXEDateQuarter0 2219	text output for **first quarter**
	.vcTXEDateQuarter1 2220	text output for **second quarter**
	.vcTXEDateQuarter2 2221	text output for **third quarter**
	.vcTXEDateQuarter3 2222	text output for **fourth quarter**
	.vcTXEDlgLegArrangement 2046	Text in the **Legend Attributes** dialog: **Arrangement**
	.vcTXEDlgLegBottomMargin 2052	Text in the **Legend Attributes** dialog: **Bottom margin:**
	.vcTXEDlgLegFixedToColumns 2048	Text in the **Legend Attributes** dialog: **Fixed to columns**
	.vcTXEDlgLegFixedToRows 2047	Text in the **Legend Attributes** dialog: **Fixed to rows**
	.vcTXEDlgLegFixedToRowsAndColumns 2049	Text in the **Legend Attributes** dialog: **Fixed to rows and columns**
	.vcTXEDlgLegIdcancel 2042	**Legend Attributes** dialog: **Cancel** button
	.vcTXEDlgLegIdd 2040	Dialog **Legend Attributes**: Text in Title Bar

.vcTXEDlgLegIdok 2041	Button text in **Legend Attributes**dialog: **OK**
.vcTXEDlgLegLegendElements 2045	Text in the **Legend Attributes** dialog: **Legendelements**
.vcTXEDlgLegLegendFont 2053	**Legend Attributes** dialog: legend **Font...** button
.vcTXEDlgLegLegendTitleFont 2044	**Legend Attributes** dialog: legend title **Font** button…
.vcTXEDlgLegLegendTitleVisible 2043	Text in the **Legend Attributes** dialog: **Legend title visible**
.vcTXEDlgLegMargins 2050	Text in the **Legend Attributes** dialog: **Margins**
.vcTXEDlgLegTopMargin 2051	Text in the **Legend Attributes** dialog: **Top margin:**
.vcTXEDlgNedCaptionPrefix 2024	**Edit data** dialog, text for text line: “Node”
.vcTXEDlgNedIdapply 2027	**Edit data** dialog, **Apply** button
.vcTXEDlgNedIdcancel 2016	Text in the **Edit data** dialog: **Cancel**
.vcTXEDlgNedIdclose 2029	**Edit data** dialog: **Close** button
.vcTXEDlgNedIdd 2014	caption of the **Edit data** dialog
.vcTXEDlgNedIdhelp 2028	**Edit data** dialog: **Help** button
.vcTXEDlgNedIdok 2015	Text in the **Edit data** dialog: **OK**
.vcTXEDlgNedNamesColStr 2018	Text in the **Edit data** dialog: **Fields**
.vcTXEDlgNedTTGotoFirst 2032	**Edit data** dialog: tooltip text **Show first selected activity**
.vcTXEDlgNedTTGotoLast 2035	**Edit data** dialog, Tooltip **“Show last selected activity“**
.vcTXEDlgNedTTGotoNext 2034	**Edit data** dialog, tooltip text **Show next selected activity**
.vcTXEDlgNedTTGotoPrev 2033	**Edit data** dialog: tooltip text **Show previous selected activity**
.vcTXEDlgNedValuesColStr 2019	Text in the **Edit data** dialog: **Values**
.vcTXEErrTxtEntryTooLong 2730	Message text: "Entry is too long, %s characters are possible."
.vcTXEErrTxtWrongLongInteger 2729	Message text: "Entry is not an integer or too big."
.vcTXEPrctBtApply 2318	Button text in **Page Setup** dialog: **Apply**
.vcTXEPrctBtCancel 2302	Button text in Print Busy box: **Cancel**
.vcTXEPrctBtClose 2303	Button text in **Print Preview** dialog: **Close**
.vcTXEPrctBtFitToPage 2308	Button text in **Print Preview** dialog: **Fit To Page**
.vcTXEPrctBtNext 2305	Button text in **Print Preview** dialog: **Next**
.vcTXEPrctBtOk 2301	Button text in **Page Setup** dialog: **OK**
.vcTXEPrctBtPageLayout 2311	Button text in **Print Preview** dialog: **Page Setup**
.vcTXEPrctBtPrevious 2304	Button text in **Print Preview** dialog: **Previous**
.vcTXEPrctBtPrint 2313	Button text in **Print Preview** dialog: **Print**
.vcTXEPrctBtPrinterSetup 2312	Button text in **Print Preview** dialog: **Printer setup**
.vcTXEPrctBtSingle 2307	Button text in **Print Preview** dialog: **Single**
.vcTXEPrctBtZoomPrint 2319	Button text in **Print Preview** dialog: **Print Area…**
.vcTXEPrctDtAddCuttingMarks 2514	Text in the **Page Setup** dialog: **Show crop marks**
.vcTXEPrctDtAlignment 2526	Text in the **Page Setup** dialog: **Alignment**
.vcTXEPrctDtAlignmentItems 2583	Text in the **Page Setup** dialog: **Top left|Top|Top right|Left|Centered|Right|Bottom left|Bottom|Bottom right**
.vcTXEPrctDtApplicationName 2501	Text in Print Busy box: **Name of application**
.vcTXEPrctDtBottom 2521	Text in the **Page Setup** dialog: **Bottom**
.vcTXEPrctDtCm 2530	Text in the **Page Setup** dialog: **cm**

.vcTXEPrctDtCurrentValues 2581	Text in the **Page Setup** dialog: **Current** not active
.vcTXEPrctDtEnableTable 2558	
.vcTXEPrctDtFitToPage 2508	Text in the **Page Setup** dialog: **Fit to page counts**
.vcTXEPrctDtFoldingMarksItems 2577	Text in the **Page Setup** dialog: **Form A|Form B|Form C**
.vcTXEPrctDtFoldingMarksText 2576	Text in the **Page Setup** dialog: **Show &folding marks (DIN 824):**
.vcTXEPrctDtFooterGroup 2584	Text in the **Page Setup** dialog: **Footer line**
.vcTXEPrctDtFrameOutside 2515	Text in the **Page Setup** dialog: **Show frame outside**
.vcTXEPrctDtInch 2588	Text in the **Page Setup** dialog: **in**
.vcTXEPrctDtLeft 2520	Text in the **Page Setup** dialog: **Left**
.vcTXEPrctDtMargins 2529	Text in the **Page Setup** dialog: **Minimum sizes for sheet margins**
.vcTXEPrctDtMaxPages 2580	Text in the **Page Setup** dialog: **pages**
.vcTXEPrctDtOff 2557	Text **Off** dialog
.vcTXEPrctDtOptions 2528	Text in the **Page Setup** dialog: **Options**
.vcTXEPrctDtOutput 2531	Text in the **Page Setup** dialog: **Output**
.vcTXEPrctDtPageDescription 2562	Text in **Page Setup** dialog: **Text**
.vcTXEPrctDtPageLayout 2532	**Page Setup** dialog: Text in Title Bar
.vcTXEPrctDtPageNumberingItems 2582	Text in the **Page Setup** dialog: **Row.Column|Column.Row|Page/Count**
.vcTXEPrctDtPageNumbers 2518	Text in the **Page Setup** dialog: **Page n&umbering**
.vcTXEPrctDtPagePadding 2585	Text in the **Page Setup** dialog: **&Pad pages with space**
.vcTXEPrctDtPagePreview 2533	**Print Preview** dialog: Text in Title Bar
.vcTXEPrctDtPagesMaxHeight 2511	Text in the **Page Setup** dialog: **Maximum height**
.vcTXEPrctDtPagesMaxWidth 2510	Text in the **Page Setup** dialog: **Maximum width**
.vcTXEPrctDtPercent 2509	Text in the **Page Setup** dialog: **%**
.vcTXEPrctDtPrintDate 2564	Text in **Page Setup** dialog: **Additionally print current &date**
.vcTXEPrctDtPrintingPage 2556	Text in Print Busy Box: **Printing page %1 of %2 on**
.vcTXEPrctDtReduceExpand 2507	Text in the **Page Setup** dialog: **Zoom factor**
.vcTXEPrctDtRepeatTable 2565	Text in **Page Setup** dialog: **Repeat table**
.vcTXEPrctDtRight 2522	Text in the **Page Setup** dialog: **Right**
.vcTXEPrctDtScaling 2527	Text in the **Page Setup** dialog: **Scaling**
.vcTXEPrctDtScalingMode 2578	Text in the **Page Setup** dialog: **&Mode:**
.vcTXEPrctDtStatusBarCurrentValues 2586	Text in the **Status bar** of the **Page Setup** dialog: **Page %1 selected (in row %2, column %3)**
.vcTXEPrctDtStatusBarSelectedPage 2587	Text in the **Status bar** of the **Page Setup** dialog: **Page %1 selected (in row %2, column %3)**
.vcTXEPrctDtSuppressEmptyPages 2517	Text in the **Page Setup** dialog: **Suppress empty pages**
.vcTXEPrctDtTableColumnRange 2575	Text in the **Page Layout** dialog: **Show table columns (e.g. 1-5;7)**
.vcTXEPrctDtTop 2519	Text in the **Page Setup** dialog: **Top**
.vcTXEPrctDtZoomFactor 2579	Text in the **Page Setup** dialog: **&Zoom factor:**
.vcTXEPrctMtAdjustBottomAndTopMargin 2437	Message text: **The bottom margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the top margin will be adjusted to %2 cm.**
.vcTXEPrctMtAdjustLeftAndRightMargin 2434	Message text: **The left margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the right margin will be reduced to %2 cm.**

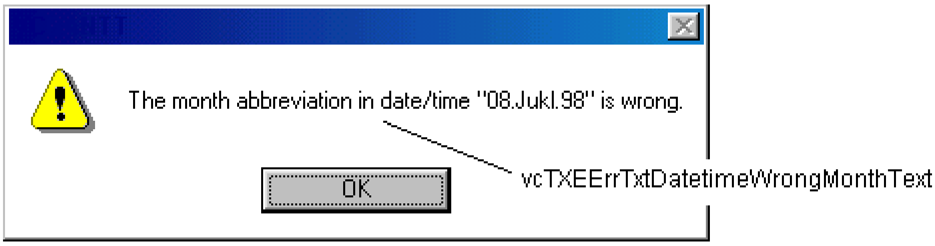
	.vcTXEPrctMtAdjustRightAndLeftMargin 2435	Message text: **The right margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the left margin will be adjusted to %2 cm.**
	.vcTXEPrctMtAdjustTopAndBottomMargin 2436	Message text: **The top margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the bottom margin will be reduced to %2 cm.**
	.vcTXEPrctMtBottomMargin 2409	Message text: **Bottom margin...**
	.vcTXEPrctMtIncompatibleVcVersion 2414	Message text: **VcVersion incompatible**
	.vcTXEPrctMtLeftMargin 2406	Message text: **Left margin is out of range and therefore will be reduced to %s cm.**
	.vcTXEPrctMtPrinterNotInstalled 2411	Message text: **Printer not installed**
	.vcTXEPrctMtPrintingNotPossible 2402	Message text: **Printing not possible at time**
	.vcTXEPrctMtRightMargin 2408	Message text: **Right margin is out of range and therefore will be reduced to %s cm.**
	.vcTXEPrctMtSelectPaperSize 2413	Message text: **Selected paper size too small**
	.vcTXEPrctMtTopMargin 2407	Message text: **Top margin is out of range and therefore will be reduced to %s cm.**
	.vcTXEPrctMtValueOutOfRange 2404	Message text: **Value out of range %1 to %2**
	.vcTXEPrctMtWillBeAdjustedTo 2410	Message text: **Will be adjusted to...**
	.vcTXERelTypeLongFF 3001	Text in the **Edit links** dialog: **Finish-to-finish (FF)**
	.vcTXERelTypeLongFS 3000	Text in the **Edit links** dialog: **Finish-to-start (FS)**
	.vcTXERelTypeLongSF 3003	Text in the **Edit links** dialog: **Start-to-finish (SF)**
	.vcTXERelTypeLongSS 3002	Text in the **Edit links** dialog:: **Start-to-start (SS)**
⇨ textEntry	System.String	Text replacing the default
⇔ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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.



Syntax error in date/time "15.09.". Year was not found. The correct syntax is "DD.MM.YY".

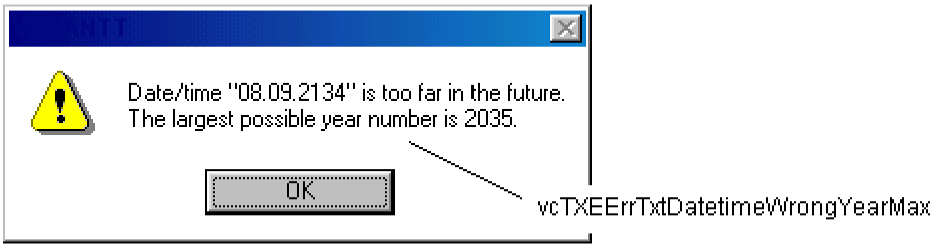
OK
vcTXEErrTxtDatetimeSytnaxError
vcTXEErrTxtDatetimeSyntaxCorrect
vcTXEErrTxtDatetimeCapitalYear
vcTXEErrTxtDatetimeNotFound

Constants of the error message ***Syntax error***



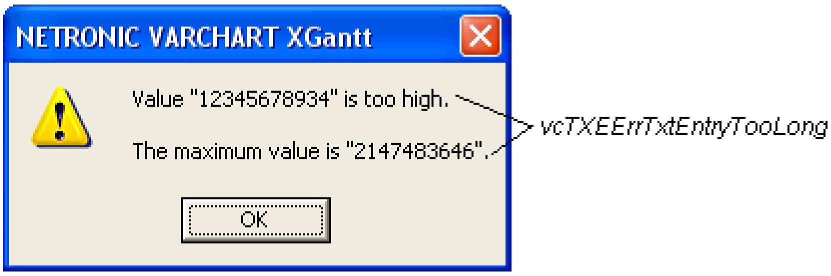
The month abbreviation in date/time "08.Jukl.98" is wrong.

Constants of the error message ***Date error, wrong month***



Date/time "08.09.2134" is too far in the future.
The largest possible year number is 2035.

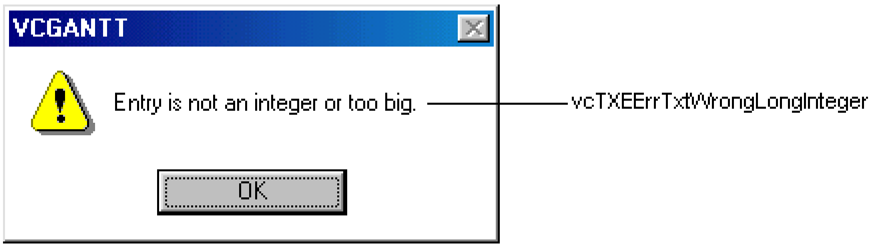
Constants of the error message ***Date error, maximum year exceeded***



Value "12345678934" is too high.

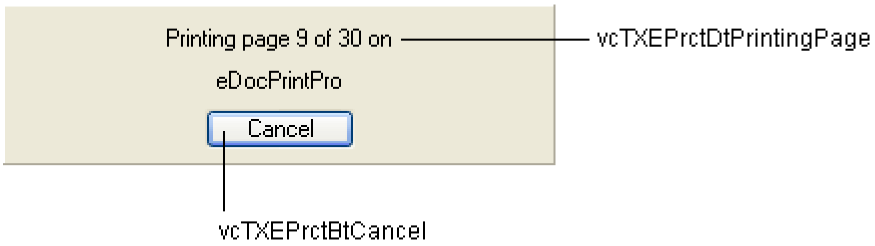
The maximum value is "2147483646".

Constants of the error message ***Entry too large***

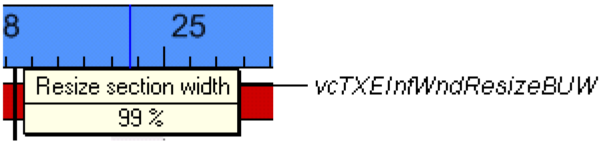


Entry is not an integer or too big.

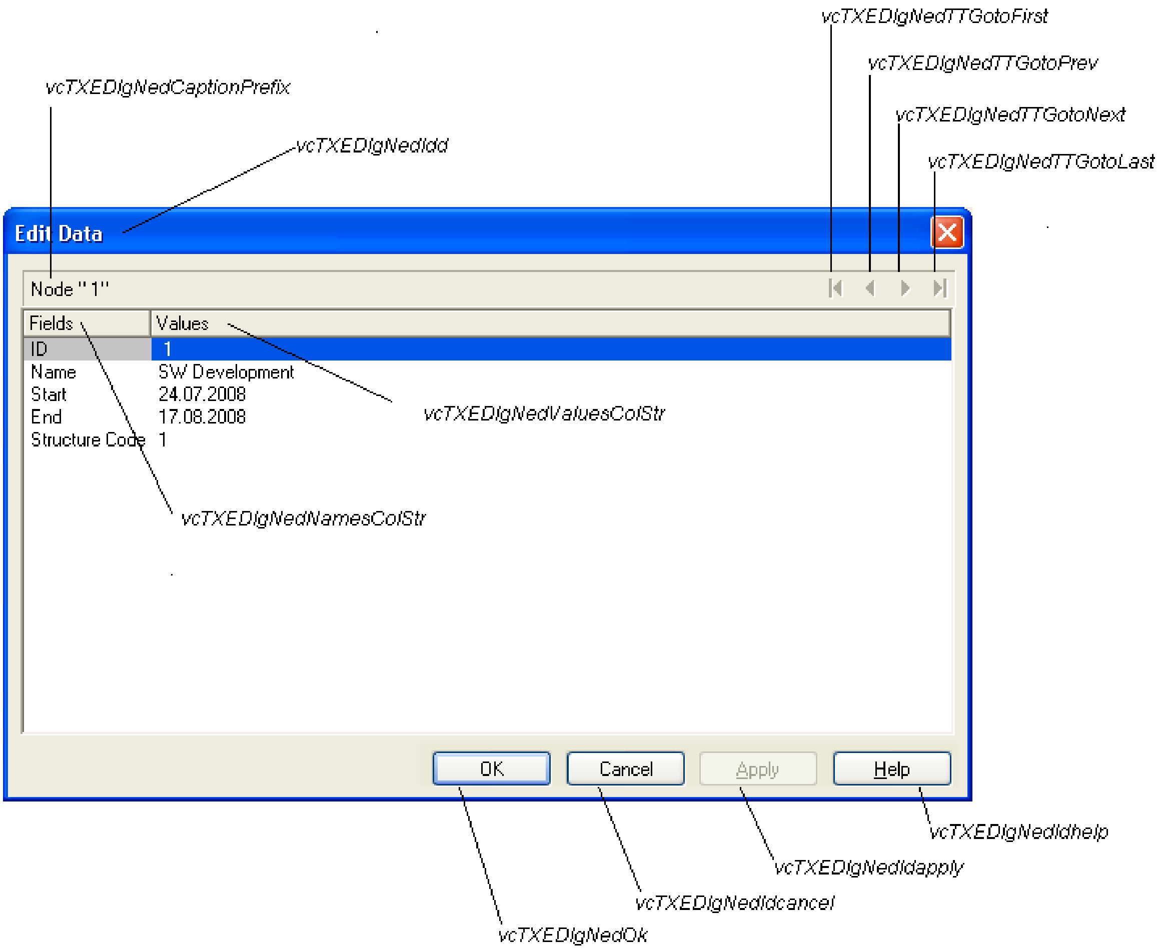
Constants of the error message ***Entry is not an integer value***



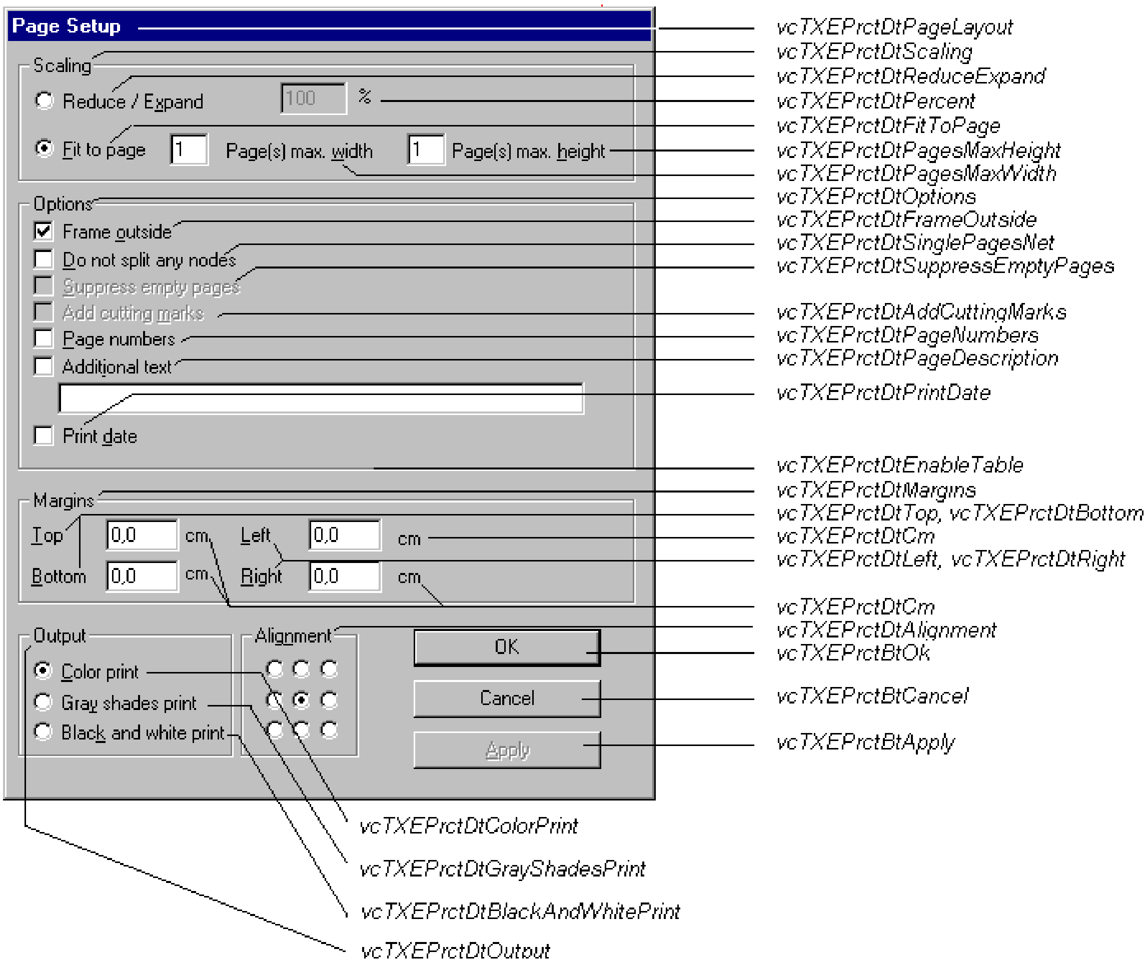
Constants of the info box ***Printing***



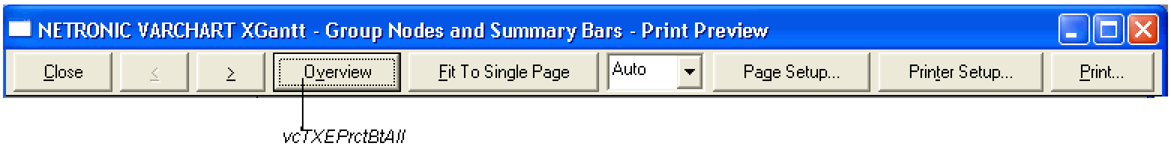
Constants of the info box ***Resizing basic unit width of time scale section***



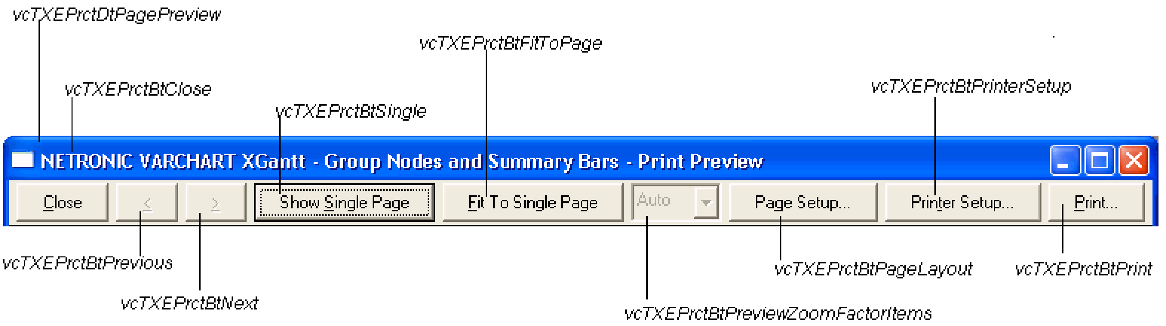
Constants of the dialogs ***Edit data*** *and* ***Edit link****, here illustrated by the* ***Edit data*** *dialog*



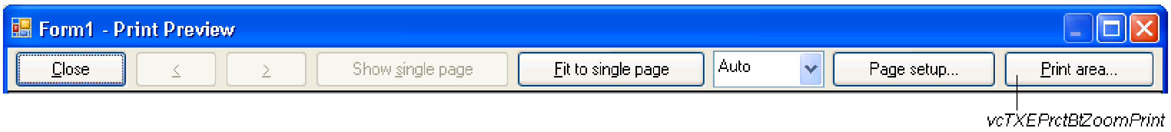
Constants of the button texts of the ***Page Setup*** *dialog*



Constants of the button texts of the ***Print Preview Overview***

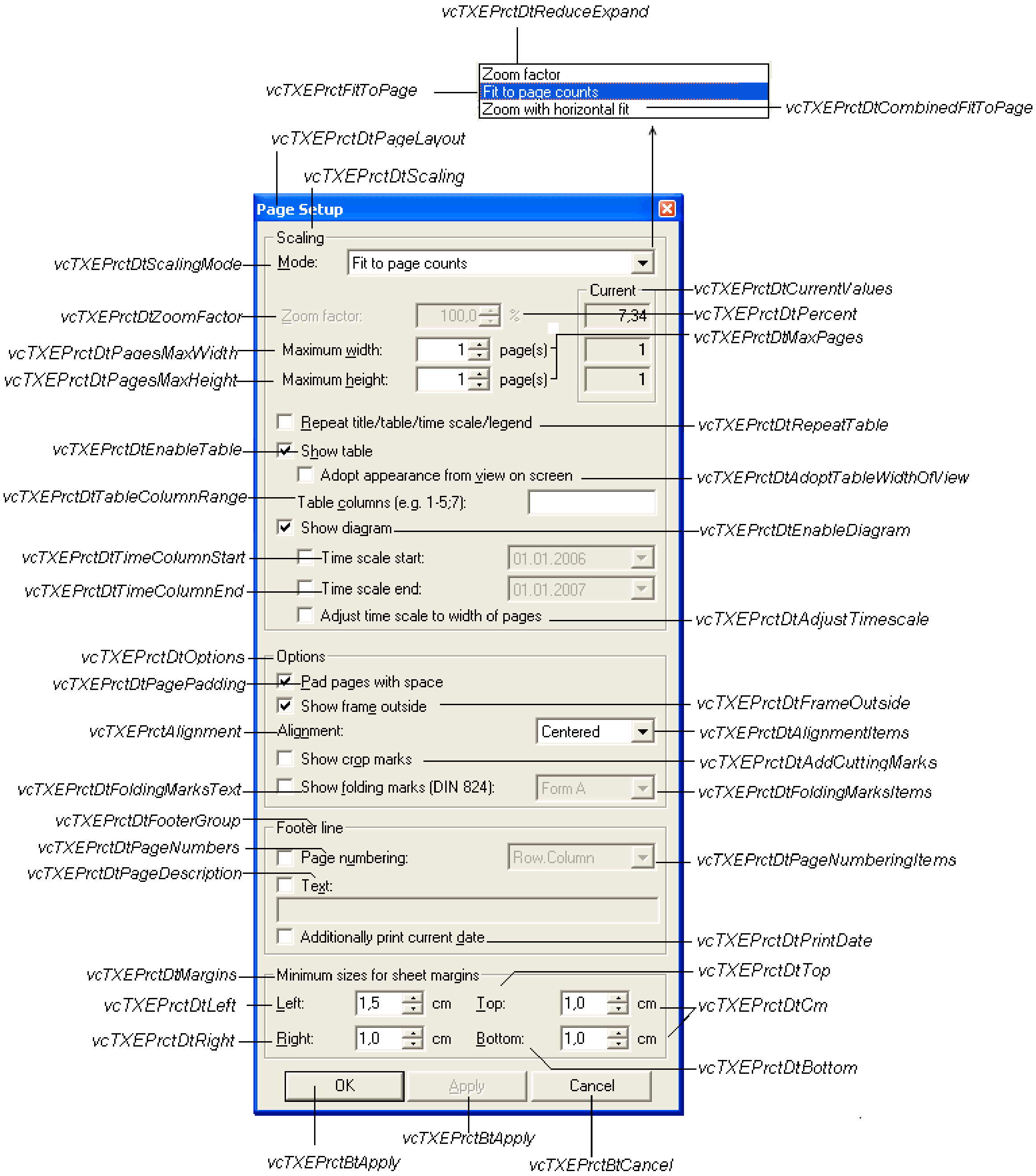


Constants of the button texts of the ***Print Preview*** *dialog*



Form1 - Print Preview

Constants of the button texts of the ***Print Preview*** *dialog*



Page Setup

Scaling

Mode: Fit to page counts

Zoom factor: 100,0 %

Maximum width: 1 page(s)

Maximum height: 1 page(s)

Current: 7,34 / 1 / 1

□ Repeat title/table/time scale/legend

☑ Show table

□ Adopt appearance from view on screen

Table columns (e.g. 1-5;7):

☑ Show diagram

□ Time scale start: 01.01.2006

□ Time scale end: 01.01.2007

□ Adjust time scale to width of pages

Options

☑ Pad pages with space

☑ Show frame outside

Alignment: Centered

□ Show crop marks

□ Show folding marks (DIN 824): Form A

Footer line

□ Page numbering: Row.Column

□ Text:

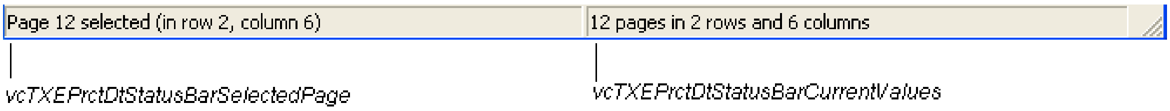
□ Additionally print current date

Minimum sizes for sheet margins

Left: 1,5 cm Top: 1,0 cm

Right: 1,0 cm Bottom: 1,0 cm

Constants of the button texts of the ***Page Setup*** *dialog*



Page 12 selected (in row 2, column 6)

12 pages in 2 rows and 6 columns

vcTXEPrctDtStatusBarSelectedPage
vcTXEPrctDtStatusBarCurrentValues

Constants of the status bar in the dialog ***Print Preview***

Example Code VB.NET

```
Private Sub VcNet1_VcTextEntrySupplying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcTextEntrySupplyingEventArgs) Handles VcNet1.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 vcNet1_VcTextEntrySupplying(object sender,
NETRONIC.XNet.VcTextEntrySupplyingEventArgs e)
   {
   switch (e.ControlIndex)
      {
      case VcTextEntryIndex.vcTXEPrctBtNext:
         e.Text = "Next page";
         break;
      case VcTextEntryIndex.vcTXEPrctBtPrevious:
         e.Text = "Previous page";
         break;
       }
   }
```

VcToolTipTextSupplying

Event of VcNet

This event only occurs after the VcNet property **ToolTipTextSupplying-EventEnabled** was set to True. It occurs when the cursor is placed on a VcNet Object. The event provides information about the object and the object type. You can use this event for editing the tooltip texts. By setting the returnStatus to **vcRetStatFalse** or by leaving the text string empty you can suppress the display of the tooltip.

	Data Type	Explanation
Properties:		
⇨ hitObject	VcObject	Object
⇨ hitObjectType	VcObjectType	Object type
	Possible Values:	
	.vcObjTypeBox 15	object type **box**
	.vcObjTypeGroup 7	object type **group**
	.vcObjTypeLinkCollection 3	object type **link collection**
	.vcObjTypeNode 2	object type **node**
	.vcObjTypeNone 0	no object
⇨ x	System.Int32	X value
⇨ y	System.Int32	Y value
⇨ toolTipText	System.String	Tooltip text, ASP editions: no restriction Other editions: 1024 characters maximum

⬄ returnStatus	VcReturnStatus	Return status
	Possible Values:	
	.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 VcNet1_VcToolTipTextSupplying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcToolTipTextSupplyingEventArgs) Handles
VcNet1.VcToolTipTextSupplying

        Dim node As VcNode
        If Convert.ToString(e.HitObject) = "VcNetLib.VcNode" Then
            node = DirectCast(e.HitObject, VcNode)
            Select Case e.HitObjectType
                Case VcObjectType.vcObjTypeNodeInDiagram
                    e.Text = Convert.ToString(node.DataField(1))
                Case VcObjectType.vcObjTypeNodeInTable
                    e.Text = Convert.ToString(node.DataField(1))
            End Select
        End If
    End Sub
```

Example Code C#

```
private void vcNet1_VcToolTipTextSupplying(object sender,
VcNetLib.VcToolTipTextSupplyingEventArgs e)
   {
   VcNode node;
   if (e.HitObject.ToString() == "NETRONIC.XNet.VcNode")
      {
      node = (VcNode)e.HitObject;
      switch(e.HitObjectType)
         {
         case VcObjectType.vcObjTypeNodeInDiagram:
            e.Text = Convert.ToString(node.get_DataField(1));
            break;
         case VcObjectType.vcObjTypeNodeInTable:
            e.Text = Convert.ToString(node.get_DataField(1));
            break;
         }
      }
   }
```

VcWorldViewClosed

Event of VcNet

This event occurs when the worldview popup window is closed.

	Data Type	Explanation
Properties:		
⇦ (no parameter)		

Example Code VB.NET

```
Private Sub VcNet1_VcWorldViewClosed(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcWorldViewClosedEventArgs) Handles VcNet1.VcWorldViewClosed
   MsgBox("Do you want to close the worldview window?", MsgBoxStyle.OKCancel)
End Sub
```

Example Code C#

```
private void vcNet1_VcWorldViewClosed(object sender,
NETRONIC.XNet.VcWorldViewClosedEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to close the worldview
window?", "Closing worldview window", MessageBoxButtons.OKCancel);
   }
```

VcZoomFactorModified

Event of VcNet

This events occurs if the user modified the size of the rectangle in the world view or if he zoomed marked objects. You can zoom smoothly by keeping the **Ctrl** key pressed while turning the mouse wheel, or in discrete steps while using the **Plus** or **Minus** keys in the number pad.

	Data Type	Explanation
Properties:		
⇦ (no parameter)		

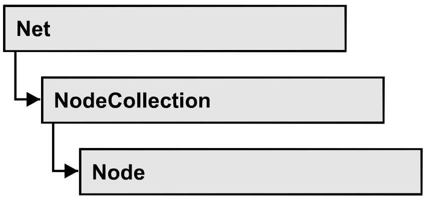
Example Code VB.NET

```
Private Sub VcNet1_VcZoomFactorModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcZoomFactorModifiedEventArgs) Handles VcNet1.VcZoomFactorModified
   MsgBox("Zoomfactor: " + VcNet1.ZoomFactor)
End Sub
```

Example Code C#

```
private void vcNet1_VcZoomFactorModified(object sender,
NETRONIC.XNet.VcZoomFactorModifiedEventArgs e)
   {
   MessageBox.Show("Zoomfactor: " + vcNet1.ZoomFactor.ToString());
   }
```

7.38 VcNode



A node is a basic element of a network diagram. Nodes can be linked to form a structure. What a node looks like is determined by NodeAppearance objects, the filters of which matching the nodes. Nodes can be generated either interactively or by the method **VcNet.InsertNodeRecord**.

Properties

- AllData
- DataField
- ID
- IncomingLinks
- Marked
- OutgoingLinks
- OutgoingLinks

Methods

- DataRecord
- Delete
- RelatedDataRecord
- 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 VcNet1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcNet1.VcNodeModifying
   Dim allDataOfNode As String
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse

   allDataOfNode = e.Node.AllData
   MsgBox(allDataOfNode)
End Sub
```

Example Code C#

```
private void vcNet1_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/retieve 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.

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
Parameter: ⇨ index	System.Int16	Index of data field
Property value	System.Object	Content of the data field

Example Code VB.NET

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking
  If MsgBox("Delete node: " + e.Node.DataField(0), MsgBoxStyle.YesNo, "Delete
node") = MsgBoxResult.Yes Then
     e.Node.Delete()
  End If
     e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Delete node: " +
e.Node.get_DataField(0), "Deleting node", MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.Yes)
      e.Node.Delete();
   else
      e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

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 = VcNet1.NodeCollection.FirstNode()

MsgBox (node.ID)
```

Example Code C#

```
VcNode node = vcNet1.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

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking
   Dim incomingLinks As VcLinkCollection
   Dim link As VcLink
   Dim predecessorNode As VcNode

   incomingLinks = e.Node.IncomingLinks
   For Each link In incomingLinks
       predecessorNode = link.PredecessorNode
       predecessorNode.Marked = True
   Next
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   VcLinkCollection incomingLinks = e.Node.IncomingLinks;
   VcNode predecessorNode;
   foreach (VcLink link in incomingLinks)
      {
      predecessorNode = link.PredecessorNode;
      predecessorNode.Marked = true;
      }
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

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 = VcNet1.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 = vcNet1.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

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

Example Code VB.NET

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking

  Dim outgoingLinks As VcLinkCollection
  Dim link As VcLink
  Dim successorNode As VcNode

   outgoingLinks = e.Node.OutgoingLinks
   For Each link In outgoingLinks
      successorNode = link.SuccessorNode
      successorNode.Marked = True
   Next
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

Example Code C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   VcLinkCollection outgoingLinks = e.Node.OutgoingLinks;
   VcNode successorNode;
   foreach (VcLink link in outgoingLinks)
      {
      successorNode = link.SuccessorNode;
      successorNode.Marked = true;
      }
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

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

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking

   If MsgBox("Delete node: " + e.Node.DataField(0), MsgBoxStyle.YesNo, "Delete
node") = MsgBoxResult.Yes Then
      e.Node.Delete()
      e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
   End If

End Sub
```

Example Code C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Delete node: " +
e.Node.get_DataField(0), "Deleting node", MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.Yes)
      {
      e.Node.Delete();
      e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
      }
   }
```

RelatedDataRecord

Method of VcNode

This property lets you retrieve a data record from a data table that is related to the node data table. The index passed by the parameter denotes the field in the data record that holds the key of the related data record.

	Data Type	Explanation
Parameter: ⇨ index	System.Int16	Index of data field that holds the key
Return value	VcDataRecord	Related data record returned

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
Return value	System.Boolean	Node was/was not updated successfully

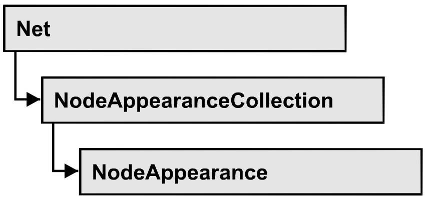
Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcNet1.NodeCollection
node = nodeCltn.FirstNode

node.DataField(12) = "Group A"
node.Update()
```

Example Code C#

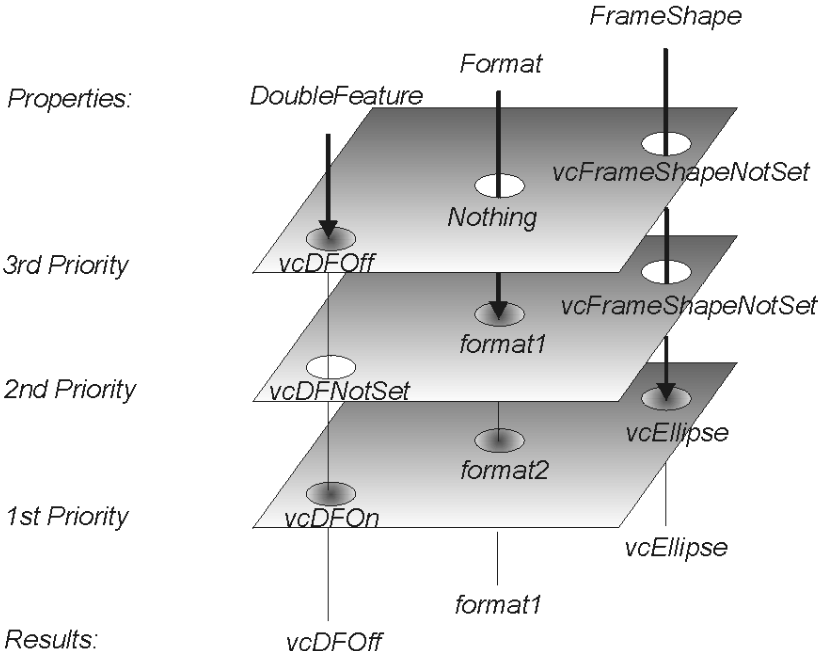
```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
node.set_DataField(12, "Group A");
node.Update();
```

7.39 VcNodeAppearance



A VcNodeAppearance object defines the appearance of a node, if the node data comply with the conditions defined by the filters assigned. Different node appearances can be set in the **Node appearances** dialog box that you reach via the **Nodes** property page.

The sketch below shows the influence of NodeAppearance objects on the appearance of nodes. The node appearances matching the nodes are displayed in descending order of priority. A property that has not been set to a NodeAppearance object will give way to a property of a NodeAppearance object that is next in the descending hierarchy.



Properties

- BackgroundColor
- LineColorDataFieldIndex
- BackgroundColorMapName
- DoubleFeature
- FilterName
- FormatName
- FrameAroundFieldsVisible

- FrameShape
- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- Pattern
- PatternColor
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- PileEffect
- Shadow
- ShadowColor
- Specification
- StrikeThrough
- StrikeThroughColor
- ThreeDEffect
- VisibleInLegend

Methods

- PutInOrderAfter

Properties

BackgroundColor

Property of VcNodeAppearance

This property lets you set or retrieve the background color of a node. 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 set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **-1** (see sketch at VcNode-Appearance object).

	Data Type	Explanation
Property value	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	ARGB color values

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance

nodeAppearance.BackColor = RGB(100, 100, 100)
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance;
nodeAppearance.BackColor = RGB(100, 100, 100);
```

BackgroundColorDataFieldIndex

Property of VcNodeAppearance

This property lets you set or retrieve the data field index to be used with a 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 VcNodeAppearance

This property lets you set or retrieve the name of a map for the background color. If set to "" or if the property **BackColorDataFieldIndex** is set to **-1**, then no map will be used.

	Data Type	Explanation
Property value	System.String	Name of the color map

DoubleFeature

Property of VcNodeAppearance

This property lets you set or retrieve a double lining around the node. If set to **vcDFNotSet**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **vcDFNotSet** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	VcAppearanceDoubleFeature	Types of double frames
	Possible Values:	
	.vcDFNotSet -1	Flag of DoubleFeature not set
	.vcDFOff 0	Flag of DoubleFeature set off
	.vcDFOn 1	Flag of DoubleFeature set on

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.DoubleFrame = VcAppearanceDoubleFrame.vcDFOn
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.DoubleFrame = VcAppearanceDoubleFrame.vcDFOn;
```

FilterName

Property of VcNodeAppearance

This property lets you set/require the name of the filter of the node appearance object. There are special filters which can not be modified:

- <ALWAYS>: always valid (for default node appearance always set)
- <NEVER>: never valid <INTERFACE-COLLAPSED>: valid for interface nodes in subdiagrams

	Data Type	Explanation
Property value	System.String	Name of filter

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim filterOfNodeApp As String

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
filterOfNodeApp = nodeAppearance.filtername
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("Blue");
string filterOfNodeApp = nodeAppearance.FilterName;
```

FormatName

Property of VcNodeAppearance

This property lets you set or retrieve a format to/from the nodeAppearance object. When set to **Nothing**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **Nothing** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	System.String	Name of a NodeFormat object or empty string

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
MsgBox(nodeAppearance.FormatName)
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
MessageBox.Show(nodeAppearance.FormatName);
```

FrameAroundFieldsVisible

Read Only Property of VcNodeAppearance

With this property you can specify whether the frame lines around fields shall be visible or not. This does not concern the outer frame line of the shape so that the effects of the property may vary depending on the frame shape. It has, for example, no effect on the type **vcRectangle**.

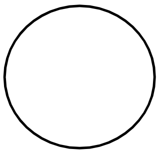
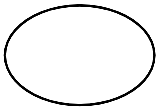
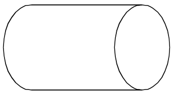
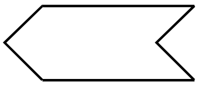
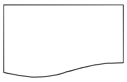
This feature can also be set in the dialog **Edit Node Appearance**.

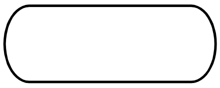
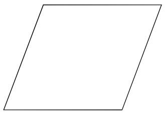
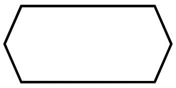
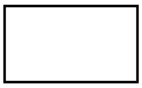
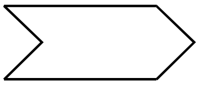
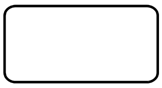
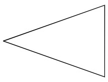
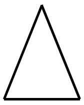
	Data Type	Explanation
Property value	VcAppearanceFrameAroundFieldsVisible	Frame around fields **Default value:** -1
	Possible Values:	
	.vcFFVNotSet -1	frame line around fields not set
	.vcFFVOff 0	Flag of FrameAroundFields set off
	.vcFFVOn 1	Flag of FrameAroundFields set on

FrameShape

Property of VcNodeAppearance

This property lets you set ot retrieve the frame shape to/from the node appearance. When set to **vcFrameShapeNotSet**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **vcFrameShapeNotSet** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	AppearanceFrameShapeEnum	Frame shape
	Possible Values:	
	.vcCircle 11	circular Frame shape 
	.vcEllipse 12	elliptical frame shape 
	.vcFile 19	frame shape horizontal cylinder 
	.vcFrameShapeNotSet -1	frame shape not set
	.vcLeftArrow 17	frame arrow shaped, pointing left 
	.vcListing 20	frame shape document 
	.vcNoFrameShape 1	no frame shape

	.vcOval 4	oval frame shape 
	.vcParallelogram 9	frame shape parallelogram 
	.vcPointed 7	frame shape pointed at vertical sides 
	.vcRectangle 2	rectangular frame shape 
	.vcRightArrow 18	frame arrow shaped, pointing right 
	.vcRounded 3	rectangular frame shape, rounded 
	.vcTriangleLeft 10	triangular frame shape, pointing left 
	.vcTriangleUp 13	triangular frame shape, pointing up 

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcEllipse
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcEllipse;
```

LegendText

Property of VcNodeAppearance

This property lets you set or retrieve the legend text of a node appearance. When set to "", the content of the **Name** property will be displayed.

	Data Type	Explanation
Property value	System.String	Legend text of the node appearance **Default value:** " " (content of the property **Name**)

LineColor

Property of VcNodeAppearance

This property lets you assign/retrieve the line color to the node appearance. When set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **-1** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB color values or **-1**

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.LineColor = Color.LightBlue
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.LineColor = Color.LightBlue;
```

LineColorDataFieldIndex

Property of VcNodeAppearance

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 VcNodeAppearance

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 VcNodeAppearance

This property lets you set or retrieve the line thickness of a NodeAppearance 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.

If you set this property to **-1**, it will give way to the property of a NodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **-1** (see sketch at VcNodeAppearance object).

	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 **Default value:** As defined on property page

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.NodeAppearanceByName("Standard")
nodeAppearance.LineThickness = 3
```

Example Code C#

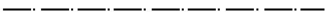
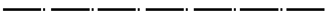
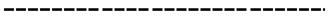
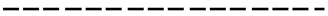
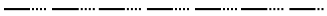
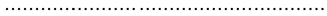
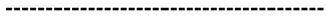
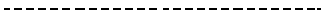
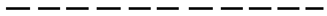
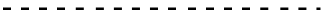
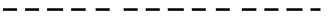
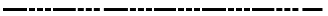
```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("Standard");
nodeAppearance.LineThickness =3;
```

LineType

Property of VcNodeAppearance

This property lets you assign/retrieve the line type to the node appearance. If set to **vcNotSet**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **vcNotSet** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	VcLineType	Line type
	Possible Values:	
	.vcDashed 4	Line dashed
	.vcDashed 4	Line dashed
	.vcDashedDotted 5	Line dashed-dotted
	.vcDashedDotted 5	Line dashed-dotted
	.vcDotted 3	Line dotted
	.vcDotted 3	Line dotted
	.vcLineType0 100	Line Type 0 ──────────────
	.vcLineType1 101	Line Type 1 — — — — — — — — — — — —
	.vcLineType10 110	Line Type 10 —···—···—···—···—···—···
	.vcLineType11 111	Line Type 11 —·-—·-—·-—·-—·-—·-
	.vcLineType12 112	Line Type 12 —··—··—··—··—··—··—
	.vcLineType13 113	Line Type 13 ——·——·——·——·——·——

.vcLineType14 114	Line Type 14 
.vcLineType15 115	Line Type 15 
.vcLineType16 116	Line Type 16 
.vcLineType17 117	Line Type 17 
.vcLineType18 118	Line Type 18 
.vcLineType2 102	Line Type 2 
.vcLineType3 103	Line Type 3 
.vcLineType4 104	Line Type 4 
.vcLineType5 105	Line Type 5 
.vcLineType6 106	Line Type 6
.vcLineType7 107	Line Type 7 
.vcLineType8 108	Line Type 8 
.vcLineType9 109	Line Type 9 
.vcNone 1	No line type assigned
.vcNone 1	No line type
.vcNotSet -1	No line type assigned
.vcSolid 2	Line solid
.vcSolid 2	Line solid

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance

nodeAppearance.LineType = vcDotted
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance;
nodeAppearance.LineType = vcDotted;
```

Name

Property of VcNodeAppearance

This property lets you set or retrieve the name of a node appearance.

	Data Type	Explanation
Property value	System.String	Name of the node appearance

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim nameNodeApp As String

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
NodeApp = nodeAppearance.Name

nameNodeAppName = nodeAppearance.name
```

Example Code C#

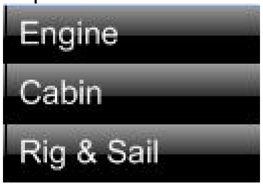
```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
string nameNodeApp = nodeAppearance.Name;
```

Pattern

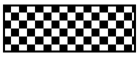
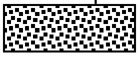
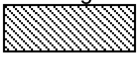
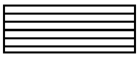
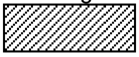
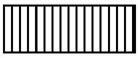
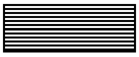
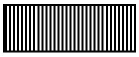
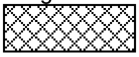
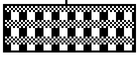
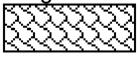
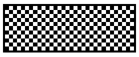
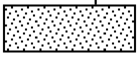
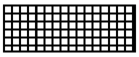
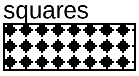
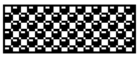
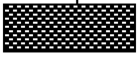
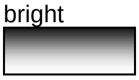
Property of VcNodeAppearance

This property lets you set or retrieve the pattern of the node. If in the property **PatternMapName** a map is specified, this map will control the pattern in dependance on the data. If set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that was not set to the value **-1** (see sketch at VcNodeAppearance object).

As a matter of fact, the values from vc05PercentPattern through vc90PercentPattern correspond to 2001 through 2011.

	Data Type	Explanation
Property value	VcFillPattern	Pattern type
	Possible Values: .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 ▥
.vcDashedDownwardDiagonalPattern 2024	Dashed diagonal lines from top left to bottom right ▧
.vcDashedHorizontalPattern 2026	Dashed horizontal lines ▤
.vcDashedUpwardDiagonalPattern 2025	Dashed diagonal lines from bottom left to top right ▨
.vcDashedVerticalPattern 2027	Dashed vertical lines ▥
.vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small ▩
.vcDiagonalBrickPattern 2032	Diagonal brick pattern ▩
.vcDivotPattern 2036	Divot pattern ░
.vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines ░
.vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines ▦
.vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right ▧
.vcHorizontalBrickPattern 2033	Horizontal brick pattern ▦
.vcHorizontalGradientPattern 52	Horizontal color gradient ▒
.vcHorizontalPattern 3	Horizontal lines ▤

.vcLargeCheckerboardPattern 2044	Checkerboard pattern showing squares of twice the size of vcSmallChecker-BoardPattern 
.vcLargeConfettiPattern 2029	Confetti pattern, large 
.vcLightDownwardDiagonalPattern 2012	Diagonal lines slanting to from top left to bottom right; spaced 50% closer than vcBDiagonalPattern 
.vcLightHorizontalPattern 2019	Horizontal lines spaced 50% closer than vcHorizontalPattern 
.vcLightUpwardDiagonalPattern 2013	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern 
.vcLightVerticalPattern 2018	Vertical lines spaced 50% closer than vcVerticalPattern 
.vcNarrowHorizontalPattern 2021	Horizontal lines spaced 75% closer than vcHorizontalPattern 
.vcNarrowVerticalPattern 2020	Vertical lines spaced 75% closer than vcVerticalPattern 
.vcNoPattern 1276	No fill pattern
.vcOutlinedDiamondPattern 2045	Diagonal cross-hatch pattern, large 
.vcPlaidPattern 2035	Plaid pattern 
.vcShinglePattern 2039	Diagonal shingle pattern 
.vcSmallCheckerBoardPattern 2043	Checkerboard pattern 
.vcSmallConfettiPattern 2028	Confetti pattern 
.vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern 
.vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares 
.vcSpherePattern 2041	Checkerboard of spheres 
.vcTrellisPattern 2040	Trellis pattern 
.vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright 

.vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark ▨
.vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright ▨
.vcVerticalGradientPattern 62	Vertical color gradient ▨
.vcVerticalPattern 2	Vertical lines ▥
.vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark ▨
.vcWavePattern 2031	Horizontal 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 VcNodeAppearance

This property lets you set or retrieve the pattern color of the node. 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 set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that was not set to the value **-1** (see sketch at VcNodeAppearance object).

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	ARGB value ({0...255},{0...255},{0...255},{0...255})

PatternColorDataFieldIndex

Property of VcNodeAppearance

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 VcNodeAppearance

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 VcNodeAppearance

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	

PatternMapName

Property of VcNodeAppearance

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 map

PileEffect

Property of VcNodeAppearance

This property lets you set or retrieve the number of node piles in the chart. If set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that was not set to the value **-1** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	System.Int32	Number of nodes piled or **-1**

Example Code VB.NET

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Piles = 2
```

Example Code C#

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = vcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Piles = 2
```

Shadow

Property of VcNodeAppearance

This property lets you assign/retrieve, whether the node appearance has a shadow. When set to **vcShNotSet**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **vcShNotSet** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	VcAppearanceShadow	Shadow settings
	Possible Values:	
	.vcShNotSet -1	Flag of Shadow not set
	.vcShOff 0	Flag of Shadow set off
	.vcShOn 1	Flag of Shadow set on

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.Shadow = VcAppearanceShadow.vcShOn
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.Shadow = VcAppearanceShadow.vcShOn;
```

ShadowColor

Property of VcNodeAppearance

This property lets you set or retrieve the shadow color of the node. 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	ARGB value

Specification

Read Only Property of VcNodeAppearance

This property lets you retrieve the specification of a node appearance. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a node appearance by the method **VcNodeAppearanceCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the node appearance

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
MsgBox(nodeAppearance.Specification)
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
MessageBox.Show(nodeAppearance.Specification);
```

StrikeThrough

Property of VcNodeAppearance

This property lets you assign/retrieve the strike through pattern of the node appearance. When set to **vcStrikeThrrough**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **vcStrikeThrough** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	VcAppearanceStrikeThrough	Strike through pattern or **-1**

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.StrikeThrough = VcAppearanceStrikeThrough.vcBackslashed
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.StrikeThrough = VcAppearanceStrikeThrough.vcBackslashed;
```

StrikeThroughColor

Property of VcNodeAppearance

This property lets you assign or retrieve the color of the strike through pattern of the node appearance. When set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **-1** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB color values or **-1**

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.StrikeThroughColor = Color.LightBlue
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.StrikeThroughColor = Color.LightBlue;
```

ThreeDEffect

Property of VcNodeAppearance

This property lets you assign/retrieve a 3D effect to/from the node appearance object. When set to **vc3DNotSet**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, that is next in the descending hierarchy and that has not been set to the value **vc3DNotSet** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	VcAppearanceThreeDEffect	3DEffect setting

Example Code VB.NET

```
Dim format As VcTableFormat

format = VcNet1.LeftTable.TableFormatCollection.FormatByName("StandardList")
format.ThreeDEffect = True
```

Example Code C#

```
VcTableFormat format =
vcNet1.LeftTable.TableFormatCollection.FormatByName("StandardList");
format.ThreeDEffect = true;
```

VisibleInLegend

Property of VcNodeAppearance

This property lets you set or retrieve whether a node appearance object is to be visible in the legend. This property also can be set by the **Administrate Node Appearances** dialog.

	Data Type	Explanation
Property value	System.Boolean	Node appearance visible in legend (True)/ invisible in legend (False) **Default value:** True

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.NodeAppearanceByName("Standard")

nodeAppearance.VisibleInLegend = False
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("Standard");
nodeAppearance.VisibleInLegend = false;
```

Methods

PutInOrderAfter

Method of VcNodeAppearance

This method lets you set the node appearance behind a node appearance specified by name, within the NodeAppearanceCollection. If you set the name to "", the node appearence will be put in the first position. The order of the node appearances within the collection determines the order by which they apply to the nodes.

	Data Type	Explanation
Parameter:		
refNodeAppearanceName	System.String	Name of the node appearance behind which the current node appearance is to be put.

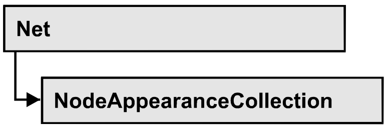
Example Code VB.NET

```
Dim nodeAppCltn As VcNodeAppearanceCollection
Dim nodeApp1 As VcNodeAppearance
Dim nodeApp2 As VcNodeAppearance

nodeAppCltn = VcGantt1.NodeAppearanceCollection()
nodeApp1 = nodeAppCltn.Add("nodeApp1")
nodeApp2 = nodeAppCltn.Add("nodeApp2")
nodeApp1.PutInOrderAfter("nodeApp2")
nodeAppCltn.Update()
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppCltn = vcGantt1.NodeAppearanceCollection;
VcNodeAppearance nodeApp1 = nodeAppCltn.Add("nodeApp1");
VcNodeAppearance nodeApp2 = nodeAppCltn.Add("nodeApp2");
nodeApp1.PutInOrderAfter("nodeApp2");
nodeAppCltn.Update();
```

7.40 VcNodeAppearanceCollection



An object of the type VcNodeAppearanceCollection automatically contains all available node appearances. You can access a node appearance using the method **NodeAppearanceByName**. The **Count** property lets you retrieve the number of node appearances in the collection. With **For Each nodeAppearance In NodeAppearanceCollection** you can access all node appearances.

Properties

- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstNodeAppearance
- GetEnumerator
- NextNodeAppearance
- NodeAppearanceByIndex
- NodeAppearanceByName
- Remove

Properties

Count

Read Only Property of VcNodeAppearanceCollection

By this property you can retrieve the number of node appearance objects in the collection.

	Data Type	Explanation
Property value	System.Int32	Number of NodeAppearance objects

Example Code VB.NET

```
MessageBox.Show(VcNet1.NodeAppearanceCollection.Count)
```

Example Code C#

```
        MessageBox.Show(vcNet1.NodeAppearanceCollection.Count.ToString());
```

Methods

Add

Method of VcNodeAppearanceCollection

By this method you can create a new node appearance as a member of the NodeAppearanceCollection. If the name was not used before, the new node appearance object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. All attributes of the new node appearance by default are set to transparent.

	Data Type	**Explanation**
Parameter: ⇨ newName	System.String	Node appearance name
Return value	VcNodeAppearance	New node appearance object

Example Code VB.NET

```
newNodeAppearance = VcNet1.NodeAppearanceCollection.Add("nodeapp1")
```

Example Code C#

```
newNodeAppearance = vcNet1.NodeAppearanceCollection.Add("nodeapp1");
```

AddBySpecification

Method of VcNodeAppearanceCollection

This method lets you create a node appearance by using a node appearance specification. This way of creating allows node appearance objects to become persistent. The specification of a node appearance can be saved and re-loaded (see VcNodeAppearance property **Specification**). In a subsequent session the node appearance can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ nodeAppearanceSpecification	System.String	Node appearance specification
Return value	VcNodeAppearance	New node appearance object

Copy

Method of VcNodeAppearanceCollection

By this method you can copy a node appearance. When the node appearance has come into existence and if the name for the new node appearance did not yet exist, the new node appearance object will be returned. Otherwise "Nothing" (Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	System.String	Name of the node appearance to be copied
⇨ newName	System.String	Name of the new node appearance
Return value	VcNodeAppearance	Node appearance object

FirstNodeAppearance

Method of VcNodeAppearanceCollection

This method can be used to access the initial value, i.e. the first node appearance object of a collection, and to continue in a forward iteration loop by the method **NextNodeAppearance** for the objects following. If there is no node appearance in the collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNodeAppearance	First node appearance object

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
While Not nodeAppearance Is Nothing
   MessageBox.Show(nodeAppearance.Name)
   nodeAppearance = nodeAppearanceCltn.NextNodeAppearance
End While
```

Example Code C#

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = vcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
While Not nodeAppearance Is Nothing
   MessageBox.Show(nodeAppearance.Name)
   nodeAppearance = nodeAppearanceCltn.NextNodeAppearance
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
while (nodeAppearance != null)
{
   MessageBox.Show(nodeAppearance.Name);
   nodeAppearance = nodeAppearanceCltn.NextNodeAppearance();
}
```

GetEnumerator

Method of VcNodeAppearanceCollection

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 appearance objects included.

	Data Type	Explanation
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim nodeApp As VcNodeAppearance

For Each nodeApp In VcNet1.NodeAppearanceCollection
   Debug.Print nodeApp.Name
Next
```

Example Code C#

```
Dim nodeApp As VcNodeAppearance

For Each nodeApp In vcNet1.NodeAppearanceCollection
   Debug.Print nodeApp.Name
Next
```

NextNodeAppearance

Method of VcNodeAppearanceCollection

This method can be used in a forward iteration loop to retrieve subsequent node appearance from a collection after initializing the loop by the method **FirstNodeAppearance**. If there is no node appearance left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNodeAppearance	Succeeding node appearance object

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
While Not nodeAppearance Is Nothing
      ListBox1.Items.Add("Name: " + nodeAppearance.Name)
   nodeAppearance = nodeAppearanceCltn.NextNodeAppearance
End While
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
while (nodeAppearance != null)
   {
     listBox1.Items.Add("Name: " + nodeAppearance.Name);
   nodeAppearance = nodeAppearanceCltn.NextNodeAppearance();
   }
```

NodeAppearanceByIndex

Method of VcNodeAppearanceCollection

This method lets you retrieve a nodeAppearance object by its index. If a node appearance of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	System.Int16	Index of the node appearance
Return value	VcNodeAppearance	Node appearance object returned

NodeAppearanceByName

Method of VcNodeAppearanceCollection

This method lets you retrieve a NodeAppearance object by its name. If a NodeAppearance object of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ nodeAppearanceName	System.String	Name of the node appearance object

Return value	VcNodeAppearance	Node appearance object

Example Code VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.NodeAppearanceByName("NodeAppearanceOne")
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcCircle
```

Example Code C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("NodeAppearanceOne");
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcCircle;
```

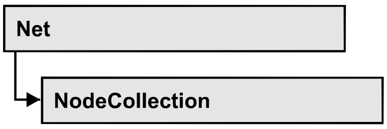
Remove

Method of VcNodeAppearanceCollection

This method lets you delete a node appearance. If the node appearance is being used in a different object, it cannot be deleted. Then **False** will be returned, otherwise **True**.

	Data Type	Explanation
Parameter: ⇨ name	System.String	Name of the node appearance
Return value	System.Boolean	Node appearance deleted (True)/not deleted (False)

7.41 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 = VcNet1.NodeCollection
MsgBox("Number of nodes: " + nodeCltn.Count)
```

Example Code C#

```
VcNodeCollection nodeCltn = vcNet1.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 Node-Collection, 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 = VcNet1.NodeCollection
node = nodeCltn.FirstNode
```

Example Code C#

```
VcNodeCollection nodeCltn = vcNet1.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
Return value	VcNode	Succeeding node

Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcNet1.NodeCollection
node = nodeCltn.FirstNode
While Not node Is Nothing
   node.Marked = False
   node = nodeCltn.NextNode
End While
```

Example Code C#

```
VcNodeCollection nodeCltn = vcNet1.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
Parameter:		
⇨ selType	VcSelectionType	Nodes to be selected
	Possible Values:	
	.vcAll 0	All objects in the diagram will be selected
	.vcAllLinksCausingCycles 7	If this selection type is chosen, the link collection will contain all links that cause the existence of cycles. If these links are deleted, cycles will cede to exist in this chart.
	.vcAllLinksInCycles 6	If this selection type is chosen, the link collection will contain all links that participate in forming cycles. Cycles are chains of nodes and links of which the beginning and end join.
	.vcAllVisible 1	All visible objects will be selected
	.vcMarked 2	All marked objects will be selected
Return value	System.Int32	Number of nodes selected

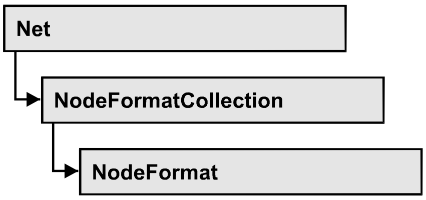
Example Code VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcSelected)
```

Example Code C#

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcSelected);
```

7.42 VcNodeFormat



An object of the type VcNodeFormat defines the contents and the format of nodes. At run time, node formats are administered and edited in the **Administrate Node Formats** dialog box that you reach via the **Nodes** property page.

Properties

- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification
- WidthOfExteriorSurrounding

Methods

- CopyFormatField
- GetEnumerator
- RemoveFormatField

Properties

FieldsSeparatedByLines

Property of VcNodeFormat

This property lets you set or retrieve whether fields inside the node are to be separated by lines.

	Data Type	Explanation
Property value	System.Boolean	Fields inside the node separated by lines (True)/ not separated by lines (False)

Example Code VB.NET

```
Dim nodeFormatCltn As VcNodeFormatCollection
Dim nodeFormat As VcNodeFormat

nodeFormatCltn = VcNet1.NodeFormatCollection
nodeFormat = nodeFormatCltn.FormatByName("FormatOne")
nodeFormat.FieldsSeparatedByLines = True
```

Example Code C#

```
VcNodeFormatCollection nodeFormatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat nodeFormat = nodeFormatCltn.FormatByName("FormatOne");
nodeFormat.FieldsSeparatedByLines = true;
```

FormatField

Read Only Property of VcNodeFormat

This property gives access to a VcNodeFormatField object by its index. The index has to be in the range from 0 to FormatFieldCount-1.

	Data Type	Explanation
Parameter:		
index	System.Int16 0 ... .FormatFieldCount-1	Index of the node format field
Property value	VcNodeFormatField	Node format field

FormatFieldCount

Read Only Property of VcNodeFormat

This property allows to determine the number of fields in a node format.

	Data Type	Explanation
Property value	System.Int16	Number of fields of the node format

Example Code VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.FormatFieldCount)
```

Example Code C#

```
      VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.FormatFieldCount.ToString());
```

Name

Property of VcNodeFormat

This property lets you set or retrieve the name of the node format.

	Data Type	Explanation
Property value	System.String	Name of the node format

Example Code VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.Name)
```

Example Code C#

```
VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.Name);
```

Specification

Read Only Property of VcNodeFormat

This property lets you retrieve the specification of a node 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 node format by the method **VcNodeFormatCollection.AddBySpecification**.

	Data Type	Explanation
Property value	System.String	Specification of the node format

Example Code VB.NET

```
Dim nodeFormatCltn As VcNodeFormatCollection
Dim nodeFormat As VcNodeFormat

nodeFormatCltn = VcNet1.NodeFormatCollection
nodeFormat = nodeFormatCltn.FirstNodeFormat
MsgBox(nodeFormat.Specification)
```

Example Code C#

```
VcNodeFormatCollection nodeFormatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat nodeFormat = nodeFormatCltn.FirstNodeFormat();
MessageBox.Show(nodeFormat.Specification);
```

WidthOfExteriorSurrounding

Property of VcNodeFormat

This property lets you set or retrieve the distance between nodes or between a node and the margin of the chart. Unit: mm. The default is 3 mm. If you choose a value smaller than this, graphical elements in the chart may overlap. You should use values below the default only if there are good reasons for it.

	Data Type	Explanation
Property value	System.Int16 0 ... 9	Distance between nodes or between a node and the margin of the chart. Unit: mm.

Methods

CopyFormatField

Method of VcNodeFormat

This method allows to copy a node format field. The new VcNodeFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	VcFormatFieldPosition	Position of the new node format field
	Possible Values:	
	.vcAbove 1	above
	.vcBelow 3	below
	.vcLeftOf 0	left of
	.vcOutsideAbove 9	outside, above
	.vcOutsideBelow 11	outside, below
	.vcOutsideLeftOf 8	outside, left of
	.vcOutsideRightOf 12	outside, right of
	.vcRightOf 4	right of
⇨ refIndex	System.Int16	Index of the reference node format field
Return value	VcNodeFormatField	Generated node format field object

GetEnumerator

Method of VcNodeFormat

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 node format fields included.

	Data Type	Explanation
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim format As VcNodeFormat
For Each format In VcNet1.NodeFormatCollection
    Debug.Write(format.Name)
Next
```

Example Code C#

```
foreach (VcNodeFormat format in vcNet1.NodeFormatCollection)
     Console.Write(format.Name);
```

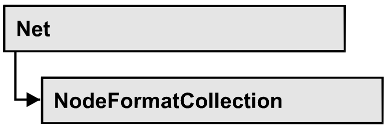
RemoveFormatField

Method of VcNodeFormat

This method lets you remove a node format field by its index. After that, the program will set all node format field indexes newly in order to number them consecutively.

	Data Type	Explanation
Parameter:		
⇨ index	System.Int16	Index of the node format field to be deleted

7.43 VcNodeFormatCollection



An object of the type VcNodeFormatCollection automatically contains all node formats available to a link. You can access all objects in an iterative loop by **For Each format InNode FormatCollection** or by the methods **First...** and **Next...**. You can retrieve a single node format by the method **FormatByName**. The property **Count** will return the number of node formats contained in the collection. By using you can retrieve all node formats.

Properties

- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- GetEnumerator
- NextFormat
- Remove

Properties

Count

Read Only Property of VcNodeFormatCollection

This property lets you retrieve the number of node formats in the node format collection.

	Data Type	Explanation
Property value	System.Int32	Number of node formats

Example Code VB.NET

```
Dim formatCltn As VcNodeFormatCollection
Dim numberOfFormats As Integer

formatCltn = VcNet1.NodeFormatCollection
numberOfFormats = formatCltn.Count
```

Example Code C#

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
int numberOfFormats = formatCltn.Count;
```

Methods

Add

Method of VcNodeFormatCollection

By this method you can create a node format as a member of the NodeFormatCollection. If the name has not been used before, the new VcNodeFormat object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

The node format has the following properties by default:

- only one field
- WidthOfExteriorSurrounding: 3 mm

The field has the following properties:

- Type: vcFFTText
- TextDataFieldIndex: IDMinimumWidth specified on the **General** property page: 3000
- Alignment: vcFFACenter
- BackColor: -1 (transparent)
- TextFontColor: RGB(0,0,0) (black)
- TextFont: Arial, 10, normal
- LeftMargin, RightMargin, TopMargin, BottomMargin: 0,3 mm

- MinimumTextLineCount, MaximumTextLineCount: 1

	Data Type	Explanation
Parameter: ⇨ newName	System.String	Name of the node format
Return value	VcNodeFormat	Node format object

Example Code VB.NET

```
newNodeFormat = VcNet1.NodeFormatCollection.Add("nodeformat1")
```

Example Code C#

```
newNodeFormat = vcNet1.NodeFormatCollection.Add("nodeformat1");
```

AddBySpecification

Method of VcNodeFormatCollection

This method lets you create a node format by using node format specification. This way of creating allows node format objects to become persistent. The specification of a node format can be saved and re-loaded (see VcNodeFormat property **Specification**). In a subsequent session the node format can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter: ⇨ formatSpecification	System.String	Node format specification
Return value	VcNodeFormat	New node format object

Copy

Method of VcNodeFormatCollection

By this method you can copy a node format. If the node format that is to be copied exists, and if the name for the new node format does not yet exist, the new node format object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ fromName	System.String	Name of the node format to be copied

⇨ newName	System.String	Name of the new node format
Return value	VcNodeFormat	Node format object

FirstFormat

Method of VcNodeFormatCollection

This method can be used to access the initial value, i.e. the first node format of a node format collection and then to continue in a forward iteration loop by the method **NextFormat** for the formats following. If there is no node format in the node format collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Return value	VcNodeFormat	First node format

Example Code VB.NET

```
Dim format As VcNodeFormat

format = VcNet1.NodeFormatCollection.FirstFormat
```

Example Code C#

```
VcNodeFormat format = vcNet1.NodeFormatCollection.FirstFormat;
```

FormatByIndex

Method of VcNodeFormatCollection

This method lets you access a format by its index. If a format 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 node format
Return value	VcNodeFormat	Node format object returned

Example Code VB.NET

```
Dim formatCltn As VcNodeFormatCollection

formatCltn = VcNet1.NodeFormatCollection
format = formatCltn.FormatByIndex(0)
format.WidthOfExteriorSurrounding = 2
```

Example Code C#

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat format = formatCltn.FormatByIndex(0);
format.WidthOfExteriorSurrounding = 2;
```

FormatByName

Method of VcNodeFormatCollection

By this method you can retrieve a node format by its name. If a node format of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ formatName	System.String	Name of the node format
Return value	VcNodeFormat	Node format

Example Code VB.NET

```
Dim formatCltn As VcNodeFormatCollection
Dim format As VcNodeFormat

formatCltn = VcNet1.NodeFormatCollection
format = formatCltn.FormatByName("Standard")
```

Example Code C#

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat format = formatCltn.FormatByName("Standard");
```

GetEnumerator

Method of VcNodeFormatCollection

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 formats included.

	Data Type	Explanation
Return value	VcObject	Reference object

Example Code VB.NET

```
Dim format As VcNodeFormat

For Each format In VcNet1.NodeFormatCollection
   Debug.Write( format.Name)
Next
```

Example Code C#

```
foreach (VcNodeFormat format In vcNet1.NodeFormatCollection)
   Console.Write(format.Name);
```

NextFormat

Method of VcNodeFormatCollection

This method can be used in a forward iteration loop to retrieve subsequent node formats from a node 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	VcNodeFormat	Subsequent node format

Example Code VB.NET

```
Dim formatCltn As VcNodeFormatCollection
Dim format As VcNodeFormat

formatCltn = VcNet1.NodeFormatCollection
format = formatCltn.Firstformat

While Not format Is Nothing
     ListBox1.Items.Add(format.Name)
      format = formatCltn.NextFormat
End While
```

Example Code C#

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat format = formatCltn.FirstFormat;

while (format != null)
   {
   listBox1.Items.Add(format.Name);
   format = formatCltn.NextFormat();
   }
```

Remove

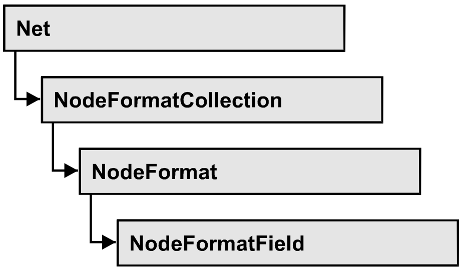
Method of VcNodeFormatCollection

This method lets you delete a node format. If the node format is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter: ⇨ name	System.String	Node format name

Return value	System.Boolean	Node format deleted (True)/not deleted (False)

7.44 VcNodeFormatField



An object of the type VcNodeFormatField represents a field of a VcNodeFormat-Object. A node format field does not have a name as many other objects, but it has an index that defines its position in the node format.

Properties

- Alignment
- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- BottomMargin
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName
- RightMargin

- TextAndGraphicsCombined
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontDataFieldIndex
- TextFontMapName
- TopMargin
- Type

Properties

Alignment

Property of VcNodeFormatField

This property lets you set or retrieve the alignment of the content of the node format field.

	Data Type	Explanation
Property value	VcFormatFieldAlignment	Alignment of the field content
	Possible Values:	
	.vcFFABottom 28	Bottom
	.vcFFABottomLeft 27	Bottom left
	.vcFFABottomRight 29	Bottom right
	.vcFFACenter 25	Center
	.vcFFALeft 24	Left
	.vcFFARight 26	Right
	.vcFFATop 22	Top
	.vcFFATopLeft 21	Top left
	.vcFFATopRight 23	Top right

BackgroundColor

Property of VcNodeFormatField

This property lets you set or retrieve the background color of the node 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 node format field shall have the background color of the node format, select the value **-1**.

If in the property **BackColorMapName** a map is specified, the map will set the background color in dependence on data.

	Data Type	Explanation
Property value	System.Drawing.Color	ARGB color values ({0...255},{0...255},{0...255},{0...255}) **Default value:** -1

BackgroundColorDataFieldIndex

Property of VcNodeFormatField

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 VcNodeFormatField

This property lets you set or retrieve the name of a color map (type vcColorMap) for the background color. If set to "", no map will be used. If the name of a map 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

BottomMargin

Property of VcNodeFormatField

This property lets you set or retrieve the width of the bottom margin of the node format field.

	Data Type	Explanation
Property value	System.Int16 0...9	Width (in mm) of the bottom margin of the node format field

ConstantText

Property of VcNodeFormatField

This property allows the node format field to display a constant text, if the node 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 VcNodeFormatField

This property lets you retrieve the name of the node format to which this field belongs.

	Data Type	Explanation
Property value	System.String	Name of the node format

GraphicsFileName

Property of VcNodeFormatField

*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 node format field. The graphics file name has to denote an existing graphics file.

	Data Type	Explanation
Property value	System.String	Name of the graphics file

GraphicsFileNameDataFieldIndex

Property of VcNodeFormatField

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 node format field the graphics that is specified for the corresponding node format will be displayed. If a valid data field index is specified, but no map is specified, the graphics file name will be loaded from the specified data field.

	Data Type	Explanation
Property value	System.Int16	Index of the data field

GraphicsFileNameMapName

Property of VcNodeFormatField

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 VcNodeFormatField

This property lets you set or retrieve for the type **vcFFTGraphics** the height of the graphics in the node format field.

	Data Type	Explanation
Property value	System.Int16 0 ... 99	Height (in mm) of the graphics

Index

Read Only Property of VcNodeFormatField

This property lets you retrieve the index of the node format field in the associated node format.

	Data Type	Explanation
Property value	System.Int16	Index of the node format field

LeftMargin

Property of VcNodeFormatField

This property lets you set or retrieve the width of the left margin of the node format field.

	Data Type	Explanation
Property value	System.Int16 0...9	Width (in mm) of the left margin of the node format field

MaximumTextLineCount

Property of VcNodeFormatField

This property lets you set or retrieve the maximum number of lines in the node format field, if the node format field is of the type **vcFFTText**. Also see the property **MinimumTextLineCount**.

	Data Type	Explanation
Property value	System.Int16 0 ... 9	Maximum number of lines

MinimumTextLineCount

Property of VcNodeFormatField

This property lets you set or retrieve the minimum number of lines in the node 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 0 ... 9	Minimum number of lines

MinimumWidth

Property of VcNodeFormatField

This property lets you set or retrieve the minimum width of the node 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 0 ... 99	Minimum width (in mm) of the node format field

PatternBackgroundColorAsARGB

Read Only Property of VcNodeFormatField

This property lets you set or retrieve the background color of the node format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the node format, select the value **-1**.

	Data Type	Explanation
Property value	System.Int32	ARGB color values ({0...255},{0...255},{0...255},{0...255})

PatternBackgroundColorDataFieldIndex

Read Only Property of VcNodeFormatField

This property lets you set or retrieve the data field index to be used with a color map specified by the property **PatternBackgroundColorMapName**. If you set this property to -1, no map will be used.

	Data Type	Explanation
Property value	System.Int32	Data field index

PatternBackgroundColorMapName

Read Only Property of VcNodeFormatField

This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **PatternBackgroundColorDataFieldIndex**, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **BackColor** will be used.

	Data Type	Explanation
Property value	System.String	Name of the color map

PatternColorAsARGB

Read Only Property of VcNodeFormatField

This property lets you set or retrieve the pattern color of the node format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

	Data Type	Explanation
Property value	System.Int16	ARGB color values ({0...255},{0...255},{0...255},{0...255})

PatternColorDataFieldIndex

Read Only Property of VcNodeFormatField

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

Read Only Property of VcNodeFormatField

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.

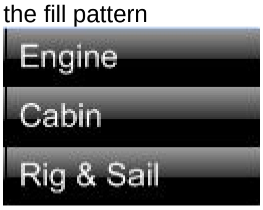
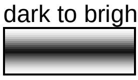
	Data Type	Explanation
Property value	System.String	Name of the color map

PatternEx

Property of VcNodeFormatField

This property lets you set or retrieve the pattern of the field background of the node 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

PatternExDataFieldIndex

Read Only Property of VcNodeFormatField

This property lets you set or retrieve the data field index to be used together with the property **PatternExMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	System.Int32	Data field index

PatternExMapName

Read Only Property of VcNodeFormatField

This property lets you set or retrieve the name of a font map (type vcPatternMap). If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **PatternExDataFieldIndex**, then the pattern is controlled by the map. If no data field entry applies, the pattern that is specified in the property **PatternEx** will be used.

	Data Type	Explanation
Parameter:		
⇨ Rückgabewert	System.String	Name of the pattern map
Property value	System.String	Name of the pattern map

RightMargin

Property of VcNodeFormatField

This property lets you set or retrieve the width of the right margin of the node format field.

	Data Type	Explanation
Property value	System.Int16 0...9	Width (in mm) of the right margin of the node format field

TextAndGraphicsCombined

Property of VcNodeFormatField

This property lets you set or retrieve whether the node field is a combi field. (See also **Edit Node Format** dialog.)

	Data Type	Explanation
Property value	System.Boolean	Combi field (True)/ no combi field (False)

TextDataFieldIndex

Property of VcNodeFormatField

This property lets you set or retrieve the index of the data field, the content of which is to be displayed in the node 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 VcNodeFormatField

This property lets you set or retrieve the font color of the node 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	System.DrawingFont	Font type of the node format

TextFontColor

Property of VcNodeFormatField

This property lets you set or retrieve the font color of the node 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 node format **Default value:** -1

TextFontDataFieldIndex

Property of VcNodeFormatField

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 VcNodeFormatField

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 VcNodeFormatField

This property lets you set or retrieve the width of the top margin of the node format field.

	Data Type	Explanation
Property value	System.Int16 0...9	Width (in mm) of the top margin of the node format field

Type

Property of VcNodeFormatField

This property lets you enquire the type of the node format field.

	Data Type	Explanation
Property value	VcFormatFieldType	Type of the node format field
	Possible Values:	
	.vcFFTGraphics 64	Graphics
	.vcFFTText 36	Text

7.45 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

- AbsoluteBottomMarginInInches
- AbsoluteLeftMarginInCM
- AbsoluteLeftMarginInInches
- AbsoluteRightMarginInCM
- AbsoluteRightMarginInInches
- AbsoluteTopMarginInCM
- AbsoluteTopMarginInInches
- Alignment
- CurrentHorizontalPagesCount
- CurrentVerticalPagesCount
- CurrentZoomFactor
- CuttingMarks
- DefaultPrinterName
- DocumentName
- FitToPage
- FoldingMarksType
- MarginsShownInInches
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation
- PageDescription
- PageDescriptionString
- PageFrame
- PageNumberMode
- PageNumbers
- PagePaddingEnabled
- PaperSize
- PrintDate
- PrinterName
- PrintPreviewWithFirstPage

- TitleAndLegendOnAllPages
- VcCalendarGrid
- ZoomFactorAsDouble

Properties

AbsoluteBottomMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute height of the bottom margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	System.Double	Absolute height of the bottom margin of the page in inches **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Example Code C#

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5;   // 0.5 inches
```

AbsoluteLeftMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute width of the left margin of the pages to be printed in cm. 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 in cm **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Example Code C#

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5;   // 1.5 cm
```

AbsoluteLeftMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute width of the left margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	System.Double	Absolute width of the left margin of the page in inches **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Example Code C#

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5;   // 0.5 inches
```

AbsoluteRightMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute width of the right margin of the pages to be printed in cm. 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 in cm **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Example Code C#

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5;   // 1.5 cm
```

AbsoluteRightMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute width of the right margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	System.Double	Absolute width of the right margin of the page in inches **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Example Code C#

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5;   // 0.5 inches
```

AbsoluteTopMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute height of the top margin of the pages to be printed in cm. 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 in cm **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Example Code C#

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5;   // 1.5 cm
```

AbsoluteTopMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute height of the top margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	System.Double	Absolute height of the top margin of the page in inches **Default value:** 0

Example Code VB.NET

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Example Code C#

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5;   // 0.5 inches
```

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 **Default value:** vcPCenterCenter
	Possible Values:	
	.vcPBottomCenter 28	Vertical alignment: bottom; horizontal alignment: center
	.vcPBottomLeft 27	Vertical alignment: bottom; horizontal alignment: left
	.vcPBottomRight 29	Vertical alignment: bottom; horizontal alignment: right
	.vcPCenterCenter 25	Vertical alignment: center; horizontal alignment: center
	.vcPCenterLeft 24	Vertical alignment: center; horizontal alignment: left
	.vcPCenterRight 26	Vertical alignment: center; horizontal alignment: right
	.vcPTopCenter 22	Vertical alignment: top; horizontal alignment: center
	.vcPTopLeft 21	Vertical alignment: top; horizontal alignment: left
	.vcPTopRight 23	Vertical alignment: top; horizontal alignment: right

Example Code VB.NET

```
VcNet1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft
```

Example Code C#

```
vcNet1.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
Property value	System.Int32	Current number of pages counted in horizontal direction

CurrentVerticalPagesCount

Read Only Property of VcPrinter

This property lets you retrieve the actual number of pages in vertical direction onto which the chart is to be printed. Also see **CurrentHorizontalPagesCount** and **MaxVerticalPagesCount**.

	Data Type	Explanation
Property value	System.Int32	Current number of pages counted in vertical direction

CurrentZoomFactor

Read Only Property of VcPrinter

This property lets you retrieve the actual zoom factor for the setting **FitToPage = False** (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 **Default value:** False

Example Code VB.NET

```
VcNet1.Printer.CuttingMarks = True
```

Example Code C#

```
vcNet1.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

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 **Default value:** " "

FitToPage

Property of VcPrinter

This property lets you set or retrieve, whether (True) the diagram is to 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

```
VcNet1.Printer.FitToPage = True
```

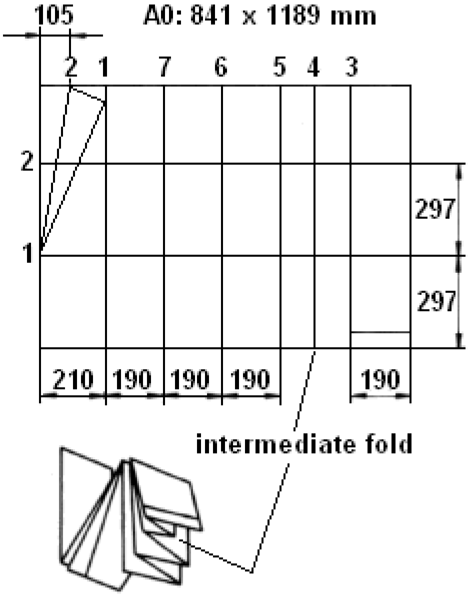
Example Code C#

```
vcNet1.Printer.FitToPage = true;
```

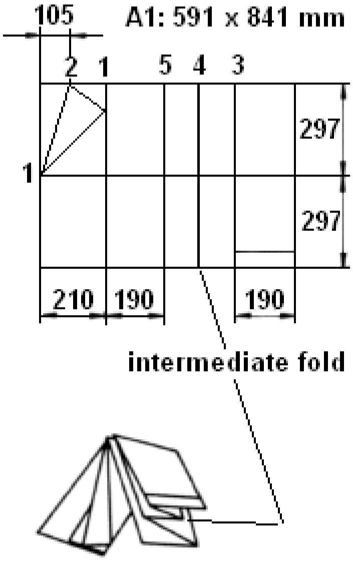
FoldingMarksType

Property of VcPrinter

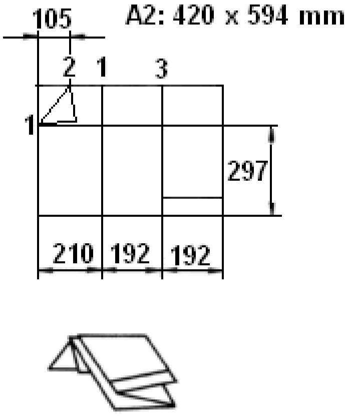
This property lets you set or retrieve folding marks according to DIN 824. The folding marks allow to fold paper sheets of the German DIN-A standard:



Folding of the DIN-A-0 format

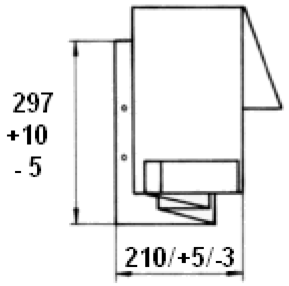
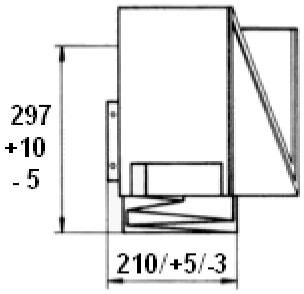
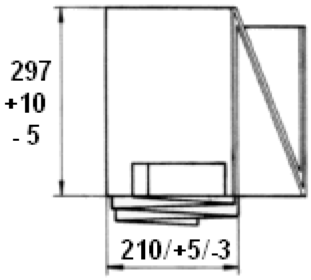


Folding of the DIN-A-1 format



Folding of the DIN-A-2 format

	Data Type	Explanation
Property value	VcFoldingMarksType	Folding marks **Default value:** vcFMTNone
	Possible Values:	

.vcFMTDIN824FormA 65	Folding marks according to DIN824-A: the drawing can be punched and filed directly to a folder.  DIN 824-A way of folding
.vcFMTDIN824FormB 66	Folding marks according to DIN824-B: the chart can be punched and filed to a folder by a flexi filing fastener.  DIN 824-B way of folding
.vcFMTDIN824FormC 67	Folding marks according to DIN824-C: the folded chart is not to be punched but to be put into a sheet protector.  DIN 824-C way of folding
.vcFMTNone 0	No folding marks

MarginsShownInInches

Property of VcPrinter

This property lets you set or retrieve whether the measuring unit of the margins in the <b"Page Layout dialog shall be switched to inches. (At present only possible at runtime).

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	System.Boolean	Measuring unit of the margins in the **Page Layout** dialog in inches (True)/ in cm (False) **Default value:** False

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
Property value	System.Int32	Maximum number of pages counted in horizontal direction **Default value:** 1

Example Code VB.NET

```
VcNet1.Printer.MaxHorizontalPagesCount = 4
```

Example Code C#

```
vcNet1.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
Property value	System.Int32	Maximum number of pages counted in vertical direction **Default value:** 1

Example Code VB.NET

```
VcNet1.Printer.MaxVerticalPagesCount = 4
```

Example Code C#

```
vcNet1.Printer.MaxVerticalPagesCount = 4;
```

Orientation

Property of VcPrinter

This property lets you set or retrieve the orientation of the output.

	Data Type	Explanation
Property value	VcOrientation	Orientation **Default value:** VcPortrait
	Possible Values: .vcLandscape 42 .vcPortrait 41	 Printing orientation **landscape** Printing orientation **portrait**

Example Code VB.NET

```
VcNet1.Printer.Orientation = VcOrientation.vcLandscape
```

Example Code C#

```
vcNet1.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 **PageDescription-String** property.

	Data Type	Explanation
Property value	System.Boolean	Page description is (True) / is not printed (False) **Default value:** False

Example Code VB.NET

```
VcNet1.Printer.PageDescription = True
```

Example Code C#

```
vcNet1.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 with 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
Property value	System.String	Page description **Default value:** Empty string ""

Example Code VB.NET

```
VcNet1.Printer.PageDescriptionString = "Net-Graphics"
```

Example Code C#

```
vcNet1.Printer.PageDescriptionString = "Net-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
Property value	System.Boolean	Page frame is (True) / is not (False) displayed **Default value:** True

Example Code VB.NET

```
VcNet1.Printer.PageFrame = True
```

Example Code C#

```
vcNet1.Printer.PageFrame = true;
```

PageNumberMode

Property of VcPrinter

This property lets you set or retrieve in which way the page numbers are to be displayed: "Page N of M pages" or "x.y" (row no./column no.).

	Data Type	Explanation
Property value	VcPageNumberMode	Mode of page numbering **Default value:** vcPRowColumn
	Possible Values: .vcPageNOfM 1597 .vcPRowColumn 1596	 "Page N of M pages" "x.y" (row no./column no.).

Example Code VB.NET

```
Dim printer As VcPrinter
printer.Orientation = VcOrientation.vcLandscape
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM
printer.PageNumbers = True
printer.FitToPage = False
VcNet1.ShowPrintPreviewDialog()
```

Example Code C#

```
VcPrinter printer = vcNet1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM;
printer.PageNumbers = true;
printer.FitToPage = false;
vcNet1.ShowPrintPreviewDialog();
```

PageNumbers

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) a page number is printed. The mode of page numbering is set with the help of the property **PageNumberMode**.

	Data Type	Explanation
Property value	System.Boolean	Page numbers are (True) / are not (False) printed **Default value:** False

Example Code VB.NET

```
VcNet1.Printer.PageNumbers = True
```

Example Code C#

```
vcNet1.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 **Default value:** True

Example Code VB.NET

```
VcNet1.Printer.PagePaddingEnabled = True
```

Example Code C#

```
vcNet1.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
	Possible Values:	
	.vcDIN_A2 66	DIN A2
	.vcDIN_A3 8	DIN A3
	.vcDIN_A4 9	DIN A4
	.vcISO_C 24	ISO C
	.vcISO_D 25	ISO D
	.vcISO_E 26	ISO E
	.vcUS_LEGAL 5	US LEGAL
	.vcUS_LETTER 1	US LETTER

Example Code VB.NET

```
VcNet1.Printer.PaperSize = VcPaperSize.vcDIN_A3
```

Example Code C#

```
vcNet1.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

```
VcNet1.Printer.PrintDate = True
```

Example Code C#

```
vcNet1.Printer.PrintDate = true;
```

PrinterName

Read Only Property of VcPrinter

This property lets you set or retrieve the name of the currently selected printer. You can use this property for saving and restoring the state of the printer object.

If you transfer an empty string when setting the property, the system printer will be used.

<Tip:> Please note that the name of network printers has to be written in UNC notation, e.g. "\\server01\printer5".

	Data Type	Explanation
Property value	**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

VcNet1.ShowPrintPreviewDialog()
```

Example Code C#

```
VcPrinter printer = vcNet1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PrintPreviewWithFirstPage = true;
printer.FitToPage = false;

vcNet1.ShowPrintPreviewDialog();
```

TitleAndLegendOnAllPages

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) the title and the legend should appear on each page. Besides, it specifies whether the pages are to be splitted in a way which avoids nodes to be cut.

	Data Type	Explanation
Property value	**System.Boolean**	**Title and legend are repeated on each page (True). Title and legend are output only once and cut, if necessary (False).** **Default value: False**

Example Code VB.NET

```
VcNet1.Printer.RepeatTitleAndLegend = True
```

Example Code C#

```
vcNet1.Printer.RepeatTitleAndLegend = true;
```

VcCalendarGrid

Property of VcPrinter

This property lets you set or retrieve the absolute height of the bottom margin of the pages to be printed in cm. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation

Example Code VB.NET

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Example Code C#

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5;   // 1.5 cm
```

ZoomFactorAsDouble

Property of VcPrinter

This property lets you set or retrieve the zoom factor for the setting FitToPage = False to enlarge or downsize the output (zoom factor = 100: original size, zoom factor > 100: enlargement, zoom factor < 100: reduction).

	Data Type	Explanation
Property value	System.Double	Zoom factor of the diagram

Example Code VB.NET

```
VcNet1.Printer.ZoomFactor = 150
```

Example Code C#

```
vcNet1.Printer.ZoomFactor = 150
```

7.46 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 VcNet1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs) Handles
VcNet1.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 + VcNet1.Left
                    TextBox1.Top = e.FldRectVisible.Top + VcNet1.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 + VcNet1.Left
                    DateTimePicker1.Top = e.FldRectVisible.Top + VcNet1.Top
                    DateTimePicker1.Value = node.DataField(0)
                    DateTimePicker1.Visible = True
                    DateTimePicker1.Focus()
                Case 13     'Employee
                    ComboBox1.Left = e.FldRectVisible.Left + VcNet1.Left
                    ComboBox1.Top = e.FldRectVisible.Top + VcNet1.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 vcNet1_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 + vcNet1.Left;
           textBox1.Top = e.FldRectVisible.Top + vcNet1.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 + vcNet1.Left;
           dateTimePicker1.Top = e.FldRectVisible.Top + vcNet1.Top;
           dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
           dateTimePicker1.Visible = true;
           dateTimePicker1.Focus();
            break;
         case 13: //Employee
           comboBox1.Left = e.FldRectVisible.Left + vcNet1.Left;
            comboBox1.Top = e.FldRectVisible.Top + vcNet1.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
Property value	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 + VcNet1.Top
```

Example Code C#

```
dateTimePicker1.Top = e.FldRectVisible.Top + vcNet1.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;
```

7.47 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	SystemInt.32	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	SystemInt.32	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) **Default value:** 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	System.Int32	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	SystemInt.32	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	SystemInt.32	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	SystemInt.32	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	System.Int32	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	System.DateTime	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
Property value	System.DateTime	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
Property value	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
Property value	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	SystemInt.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	SystemInt.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
Parameter:		
⇨ startDate	System.DateTime	Desired start date
⇨ endDate	System.DateTime	Desired end date

Return value	System.Boolean	The project data were successfully calculated (true) / were not calculated (False)

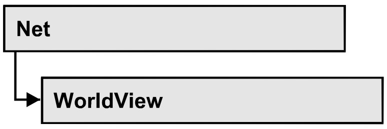
Example Code VB.NET

```
' Vorwärtsberechung (ASAP)
VcScheduler.ScheduleProject(2.5.2017, newDate(0))

' Rückwärtsberechnung (JIT)
VcScheduler.ScheduleProject(newDate(0), 2.5.2017)
```

Example Code C#

```
// Vorwärtsberechung (ASAP)
vcScheduler.ScheduleProject(2.5.2017, newDate(0));

// Rückwärtsberechnung (JIT)
vcScheduler.ScheduleProject(newDate(0), 2.5.2017);
```

7.48 VcWorldView



An object of the type **VcWorldView** designates the world view window.

Properties

- Border
- Height
- HeightActualValue
- Left
- LeftActualValue
- MarkingColor
- Mode
- ParentHWnd
- ScrollBarMode
- Top
- TopActualValue
- UpdateBehaviorName
- Visible
- Width
- WidthActualValue

Properties

Border

Property of VcWorldView

This property lets you set or retrieve whether the world view has a frame (not valid for the **vcPopupWindow** mode). The color of the frame is **Color.Black**.This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Boolean	World view with a border line (True)/without border line (False) **Default value:** True

Example Code VB.NET

```
VcNet1.WorldView.Mode = VcWorldViewMode.vcNotFixed
VcNet1.WorldView.Border = True
```

Example Code C#

```
vcNet1.WorldView.Mode = VcWorldViewMode.vcNotFixed;
vcNet1.WorldView.Border = true;
```

Height

Property of VcWorldView

This property lets you retrieve the vertical extension of the world view. It can also be set in the modes **vcFixedAtTop** and **vcFixedAtBottom**.

The coordinates are to be specified as pixels, referring to the screen.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Height of the world view **Default value:** 100

Example Code VB.NET

```
VcNet1.WorldView.Height = 100
```

Example Code C#

```
vcNet1.WorldView.Height = 100;
```

HeightActualValue

Read Only Property of VcWorldView

This property lets you retrieve the vertical extension of the world view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

	Data Type	Explanation
Property value	System.Int32 {0, ...}	Actual height of the world view {0, ...} **Default value:** 100

Example Code VB.NET

```
VcNet1.LegendView.Height = 300
```

Example Code C#

```
vcNet1.LegendView.Height = 100;
```

Left

Property of VcWorldView

This property lets you retrieve the left position of the world view. It can also be set in the modes **vcNotFixed** and **vcPopupWindow**.

The coordinates are to be specified as pixels, referring to the screen.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Left position of the world view **Default value:** 0

Example Code VB.NET

```
VcNet1.WorldView.Left = 200
```

Example Code C#

```
vcNet1.WorldView.Left = 200;
```

LeftActualValue

Read Only Property of VcWorldView

This property lets you retrieve the left position of the world view which actually ist displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

	Data Type	Explanation
Property value	System.Int32	Actual left position of the world view {0, ...} **Default value:** 0

Example Code VB.NET

```
VcNet1.LegendView.LeftActualValue = 150
```

Example Code C#

```
vcNet1.LegendView.LeftActualValue = 150;
```

MarkingColor

Property of VcWorldView

This property lets you set or retrieve the line color of the rectangle that indicates the selected section in the World View. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** RGB(0, 0, 255)

Example Code VB.NET

```
VcNet1.WorldView.MarkingColor = Color.Red
```

Example Code C#

```
vcNet1.WorldView.MarkingColor = Color.Red;
```

Mode

Property of VcWorldView

This property lets you set or retrieve the world view mode. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	VcWorldViewMode	Mode of the world view **Default value:** vcPopupWindow
	Possible Values:	

	.vcFixedAtBottom 4	The world view is displayed on the bottom of the control window. The reference system of the coordinates is the control. With this value set, the height can be specified, whereas the position and the width are fixed.
	.vcFixedAtLeft 1	The world view is displayed on the left side of the VARCHART .NET control window. Then the width can be specified, whereas the position and the height are fixed.
	.vcFixedAtRight 2	The world view is displayed on the right side of the control window. The reference system of the coordinates is the control. With this value set, the width can be specified, whereas the position and the height are fixed.
	.vcFixedAtTop 3	The world view is displayed on the top of the control window. The reference system of the coordinates is the control. With this value set, the height can be specified, whereas the position and the width are fixed.
	.vcNotFixed 5	The world view is a child window of the current parent window of the VARCHART .NET control. It can be positioned at any position with any extension. The reference system of the coordinates is the parent window. The child window does not have a frame of its own and cannot be moved interactively by the user. The parent window can be modified via the property **VcWorldView.ParentHWnd**.
	.vcPopupWindow 6	The world view is a popup window with its own frame. The reference system of the coordinates is the screen.The user can modify its position and extension, open it via the default context menu, and close it via the **Close** button in the frame.

Example Code VB.NET

```
VcNet1.WorldView.Mode = VcWorldViewMode.vcFixedAtBottom
```

Example Code C#

```
vcNet1.WorldView.Mode = VcWorldViewMode.vcFixedAtBottom;
```

ParentHWnd

Property of VcWorldView

In the **vcNotFixed** mode this property lets you set the HWnd handle of the parent window, for example, if the world view is to appear in a frame window implemented by your own. By default, the frame window is positioned on the HWnd handle of the parent window of the VARCHART Windows Forms main window. This property can be used only at run time.

	Data Type	Explanation
Property value	OLE_HANDLE	Handle

ScrollBarMode

Property of VcWorldView

This property lets you set or retrieve the scroll bar mode of the world view. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	VcWorldViewScrollBarMode	Scrollbarmode **Default value:** NoScrollBar
	Possible Values:	
	.vcAutomaticScrollBar 3	Display of a horizontal or vertical scrollbar if required.
	.vcHorizontalScrollBar 1	Display of a horizontal scrollbar if required.
	.vcNoScrollBar 0	The chart is always displayed completely without scrollbars.
	.vcVerticalScrollBar 2	Display of a vertical scrollbar if required.

Example Code VB.NET

```
VcNet1.WorldView.ScrollBarMode = vcAutomaticScrollbar
```

Example Code C#

```
vcNet1.WorldView.ScrollBarMode = vcAutomaticScrollBar;
```

Top

Property of VcWorldView

This property lets you retrieve the top position of the world view. It can also be set in the modes **vcNotFixed** and **vcPopupWindow**.

The coordinates are to be specified as pixels, referring to the screen.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Top position of the world view **Default value:** 0

Example Code VB.NET

```
VcNet1.WorldView.Top = 20
```

Example Code C#

```
vcNet1.WorldView.Top = 20;
```

TopActualValue

Read Only Property of VcWorldView

This property lets you enquire the top position of the world view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

	Data Type	Explanation
Property value	System.Int32	Actual top position of the world view {0, ...}

Example Code VB.NET

```
VcNet1.LegendView.TopActualValue = 40
```

Example Code C#

```
vcNet1.LegendView.TopActualValue = 40;
```

UpdateBehaviorName

Read Only Property of VcWorldView

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	System.String	Name of the UpdateBehavior

Visible

Property of VcWorldView

This property lets you enquire/set whether the world view is visible or not. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Boolean	World view visible (True)/not visible (False) **Default value:** False

Example Code VB.NET

```
VcNet1.WorldView.Visible = True
```

Example Code C#

```
vcNet1.WorldView.Visible = true;
```

Width

Property of VcWorldView

This property lets you retrieve the horizontal extent of the world view. It can also be set in the modes **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow**.

The coordinates are to be specified as pixels, referring to the screen.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	System.Int32	Horizontal extension of the world view **Default value:** 100

Example Code VB.NET

```
VcNet1.WorldView.Width = 200
```

Example Code C#

```
vcNet1.WorldView.Width = 200;
```

WidthActualValue

Read Only Property of VcWorldView

This property lets you retrieve the horizontal extent of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or width is preset.

	Data Type	Explanation
Property value	System.Int32 {0, ...}	Actual horizontal extension of the world view {0, ...} **Default value:** 100

Example Code VB.NET

```
VcNet1.LegendView.WidthActualValue = 600
```

Example Code C#

```
vcNet1.LegendView.WidthActualValue = 600;
```

8 Index