VARCHART XNet

ActiveX Edition 5.2 User's and Reference Guide

NETRONIC

VARCHART XNet ActiveX Edition

Version 5.2

User's Guide

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

Last Revision: 27 April 2020

Table of Contents

1 Introduction

1.1 General Information on VARCHART XNet

VARCHART XNet is an element of the product group VARCHART-X on offer. This product group contains ActiveX controls that were developed using NETRONIC's VARCHART function library (VARCHART XGantt, VARCHART XNet, VARCHART XTree).

VARCHART XNet lets you implement an initial graphical representation of your data in a matter of minutes. You can easily adapt VARCHART XNet on request of your customers.

VARCHART XNet lets you display scenarios such as computer networks, work flow or class diagrams. For project management charts an integrated scheduling module is available.

Larger amounts of data can be loaded and stored to files via the application programming interface (API). Single data can be inserted, modified or deleted by user interaction.

The data of links and of nodes are kept separate. By default they are stored in two different tables. In the new version it is possible to define up to 90 data table.

The data formats of the different VARCHART ActiveX controls can be made compatible (depending on the settings in the respective data table), allowing the easy exchange of data or the combination of controls within an application.

The structure of the data format is defined during design mode of the VARCHART ActiveX control.

VARCHART ActiveX controls can easily be configured - in design mode via the property pages, during runtime by the programming interface. A large number of events offers a variety of options to customize default interactions.

> Functionalities

- It allows to assign different node appearances of different priorities to a node.
- It offers data-controlled allocation of graphical attributes via filters, in order to e.g. display nodes of the same group in yellow.
- Node formats allow a variety of settings to node appearances.

- Positions of nodes and links can be kept in data fields to be stored and restored.
- Left-to-right and top-to-bottom orientation are available. The layout algorithm, to be activated by an API call, keeps the clarity of a chart at a maximum by reducing the number of crossings to a minimum.
- Nodes can be grouped after criteria and groups can be sorted. Nodes can be grouped by a data field individually defined. Groups can easily be identified by their title. These features allow a clear presentation of even large amounts of data.
- Users can interactively create, edit, delete or move nodes and links.
- Continuous zooming of diagrams is available. Sections of your diagram can be zoomed to full screen size and moved within the diagram via the scroll bars to let you view other parts in the same enlargement.
- If you move a node behind a margin of the control form, the diagram will be scrolled automatically (Autoscrolling).
- Auxiliary nodes can be positioned, that is, nodes can be placed in the same rank as their predecessors.
- The appearance of nodes can be modified by a variety of options (e.g. nodes of the critical path in red).
- A title and a legend can be displayed in the charts for output (formats: VMF, WMF, JPG, BMP, EPS, GIF, PCX, PNG, TIF). (See Chapter "Important Terms: Viewer Metafile (*.vmf)".)
- Paging and page preview are integrated in the printing functionality and allow an immediate output of all charts. A chart can be partitioned into pages and viewed by the preview. A partitioned chart can be reassembled and any section of it can be zoomed.
- The VARCHART ActiveX control can be inserted into a HTML page so that it will be visible in a browser. (Further information you will find in this introduction in the chapter "ActiveX Controls in Browser Environment".)

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

1.2 Technical Requirements

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

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

1.3 Installation

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

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

- c:\windows32\system\regsvr32 "c:\program files\varchart\xnet\vcnet.ocx"

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

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

The same file will be used for uninstalling. You can start the uninstalling procedure by selecting **Start** –> **Programs** –> **Varchart** and then **UninstallXNet**.

You can remove the registration entry yourself by using the command

- c:\windows32\system\regsvr32 -u "c:\program files\varchart\xnet\-vcnet.ocx"

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.

1.4 Delivery

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

VARCHART XNet files:

- *vcnet.ocx* (version 5.0)
- *vcpane32u.dll* (version 5.5)
- *vcprct32u.dll* (version 5.5)
- *vcwin32u.dll* (version 5.5)
- *vxcsv32u.dll* (version 1.320)
- **Microsoft libraries:**
- *gdiplus.dll*
- *mfc100u.dll*
- *msvcp100.dll*
- *msvcr100.dll*

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

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

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

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

1.5 Data Exchange by VARCHART XNet

At present, the data exchange by the VARCHART ActiveX controls is performed via variants. You can address a file or communicate via the API. Via the API, you can either enter or read a complete data record, or address the data as properties of the VcNode or of the VcLink object by fields.

1.5.1 Definition of the Interface

By default 21 data fields are available in the **Maindata** table and 7 data fields are available in the **Relations** table. In the dialob **Edit Data Table ""** you can create new fields  , delete fields  or copy fields 

You can name the fields and specify their data type (alphanumeric, integer, date/time). For date fields, you must specify a date format (e.g. DD.MMM.YYYY). You are recommended not to modify the date types once they have been created since formats and nodes might then base on wrong data types. This can cause errors.

1.5.2 The Structure of CSV Files

Please enter a single data record per row for each node (Maindata table) and separate the data field contents by semicolons. If also links are to be retrieved (Relations table), enter "****" into the row after the Maindata rows. After that you can enter the link records into the following rows.

Note: The CSV format (separation by semicolons) saves texts and values only. At the moment, CSV-Files are always written in ANSI. In the example below, the structure of a CSV file is shown:

Example Code

```
1;1.;;;SWDevelopment;A;;GroupA;6;0;100;03.11.00;10.11.00;;0;;
2;1.2;;;Design&Concept;C;;GroupC;10;0;50;02.11.00;18.11.00;;0;;
3;1.2.1;;;Requirements;A;;GroupA;5;0;50;02.11.00;07.11.00;;0;;
```

1.5.3 Using CSV Files

You can open a file by the **Open** method and save it by the **SaveAsEx** method. If you do not enter a name when using the **SaveAs** method, the name specified last by using the **Open** method will be used.

Note: CSV-Files may be retrieved and written in ANSI as well as in Unicode (automatic recognition when read).

Example Code

```
VcNet1.Open "c:\data\example1.net"
...
VcNet1.SaveAs ""
' or
VcNet1.SaveAs "c:\data\example2.net"
```

1.5.4 Transferring Node and Link Data to VARCHART XNet via API

When you use the call interface, each node has to be passed by the **InsertNodeRecord** method. Links are passed by the **InsertLinkRecord** method. The method **EndLoading** shows the end of loading and triggers the update of the diagram.

Example Code

```
Dim data as String
data = "1;1.;;;SWDevelopment;A;;GroupA;6;6;100;03.11.00;10.11.00;;0;;"
VcNet1.InsertNodeRecord data
VcNet1.EndLoading
```

1.5.5 Retrieving Node Data from VARCHART XNet

Via the property **NodeCollection** a VcNodeCollection object is generated. By this object, all nodes (**vcAll**), all visible nodes (**vcAllVisible**) or the nodes marked (**vcMarked**) can be retrieved.

By the methods **FirstNode** and **NextNode** you can retrieve single node objects. The method **SelectNodes** lets you limit the choice of nodes. By the method **AllData** you can retrieve all data fields of a node or specify a data field by the method **DataField**.

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode
Dim value As String

Set nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes vcAll
Set node = nodeCltn.FirstNode
Do Until node Is Nothing
   '
   ' Access to field 0 of each node
   '
   value = node.DataField(0)
   '
   ' Access to all data
   '
   value = node.AllData
   Set node = nodeCltn.NextNode
```

```
Loop
```

1.5.6 Retrieving Link Data from VARCHART XNet

The property **LinkCollection** lets you retrieve a VcLinkCollection object. By the methods **FirstLink** and **NextLink** you can retrieve single link objects. By the method (**AllData**) you can retrieve all data fields of a link or specify a data field by the method (**DataField**).

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim value As String

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink
Do Until link Is Nothing
   '
   ' Access to field 0 of each link
   '
   value = link.DataField(0)
   '
   ' Access to all data
   '
   value = link.AllData
   Set link = linkCltn.NextLink
Loop
```

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

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

Visual Studio 6.0:

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

Visual Studio 7.0:

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

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

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

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

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

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

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

After that, remove the class that has not been deselected before from the project and instead of this, insert the NETRONIC file *xnet.cpp* from the subdirectory MFC of the installation directory of the VARCHART ActiveX control. The corresponding header file (*xnet.h*) you will also find there.

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

Example Code

```
#include "xnet.h"
using namespace XNet;
```

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

Example Code

```
XNet::CVcTitle title = VcNet1.GetTitle();
```

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

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

1.7 VARCHART ActiveX in HTML Pages

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

1.7.1 Restrictions

Compared to other applications, there are some restrictions:

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

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

1.7.2 Implementation Including Direct Embedding

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

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

Example Code

```
<OBJECT ID="VcNet1" WIDTH=700 HEIGHT=350
 CLASSID="CLSID:3C415F1E-CFBA-11D2-B467-02608C4302A9"
 CODEBASE="vcnet.cab#version=4,000,0,0">
</OBJECT>
```

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

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

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

Example Code

```
VcNet1.ConfigurationName =
"http://www.netronic_test.com/xnet_sample.ini"
```

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

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

A little web application is delivered amongst the programming samples.

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

Example Code

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

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

1.7.3 Implementation Including Indirect Embedding

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

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

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

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

1.7.4 Trouble-Shooting

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

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

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

> References for solving problems and for further technical information:

OBJECT Tag which specifies component FileVersion and #Version

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

How To Implement IObjectSafety in Visual Basic 6.0 Controls

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

Mozilla ActiveX Project

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

MeadCo Neptune

www.meadroid.com/neptune

Microsoft Cabinet SDK

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

Microsoft IExpress

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

Code Download Log Viewer (CDLLOGVW)

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

Microsoft Script Debugger

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

Code signing

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

Certification authorities

VeriSign: www.verisign.com/developer

Thawte: www.thawte.com

GeoTrust: www.geotrust.com

GlobalSign: www.globalsign.net

Signcode tool

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

Signtool tool

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

1.8 Support and Advice

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

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

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

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 www.netronic.com

www.netronic.com

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

- Support hotline
- Detailed expert advice to questions of application
- Quick fixing of possible bugs in the software
- Upgrade to a new VARCHART XNet release 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 network 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.

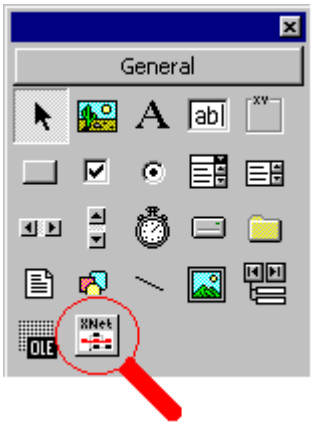
We use Visual Basic 6.0 as developing environment for the samples.

2.2 Adding VARCHART XNet to the Toolbox

For adding VARCHART XNet to the toolbox proceed as following:

1. In the **Project** menu of Visual Basic, choose the **Components** option.
2. On the record card **Controls**, choose **NETRONIC VARCHART XNet** from the list and confirm your choice by **OK**.

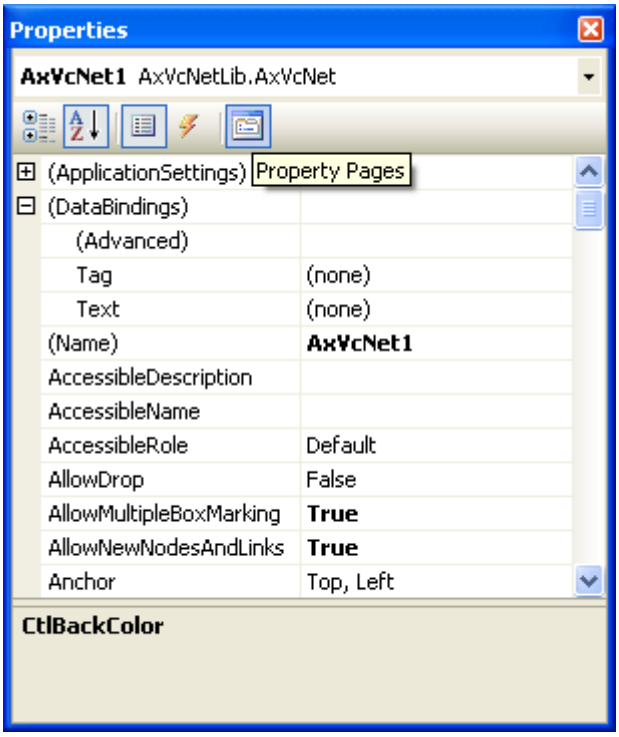
Once the VARCHART XNet control has been successfully added to the toolbox, its icon will be displayed in the toolbox.



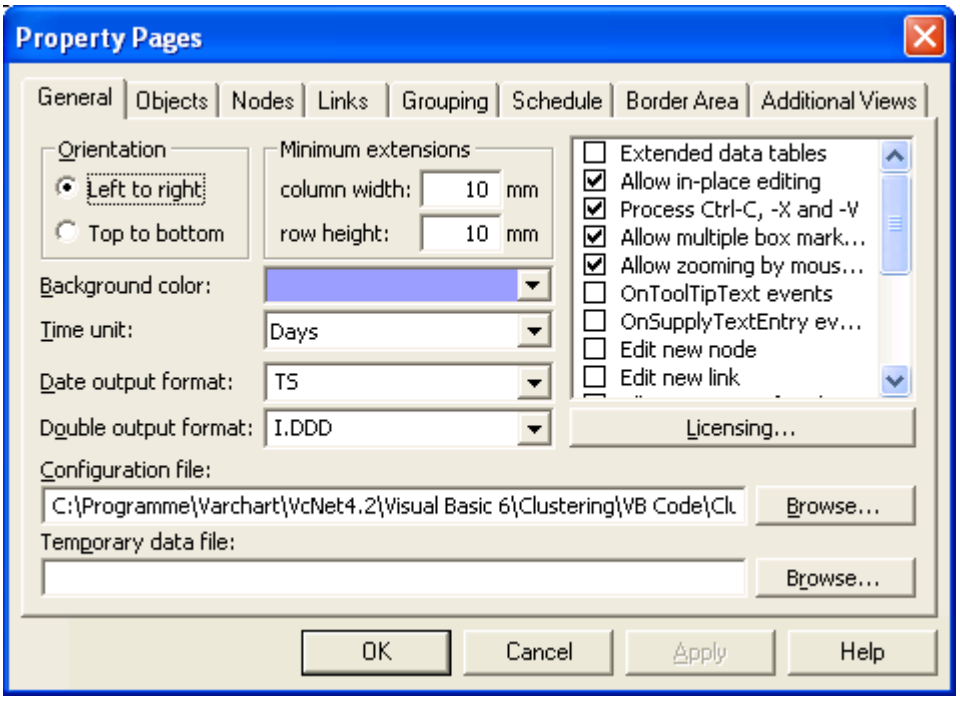
2.3 Placing the VARCHART XNet Control on a Form

To place the VARCHART XNet control in a Visual Basic form please click on it in the toolbox after inserting VARCHART XNet in the toolbox and then, using the mouse, draw a frame at the position in the form where you want the VARCHART XNet control to appear. The VARCHART XNet control will be displayed in the size drawn. You can certainly readjust the size by mouse.

In the **Properties** dialog of the control, you can activate the VARCHART XNet property pages by setting the **Custom** entry, or, later in versions of the programming environment, an icon is offered for this.



Alternatively, you can mark the VARCHART XNet control in the form, press the right mouse button and select the **Properties** menu item from the context menu popping up.



Note: Here and in the example code, the inserted VARCHART XNet control is called **VcNet1**.

2.4 Automatic Scaling of VARCHART XNet

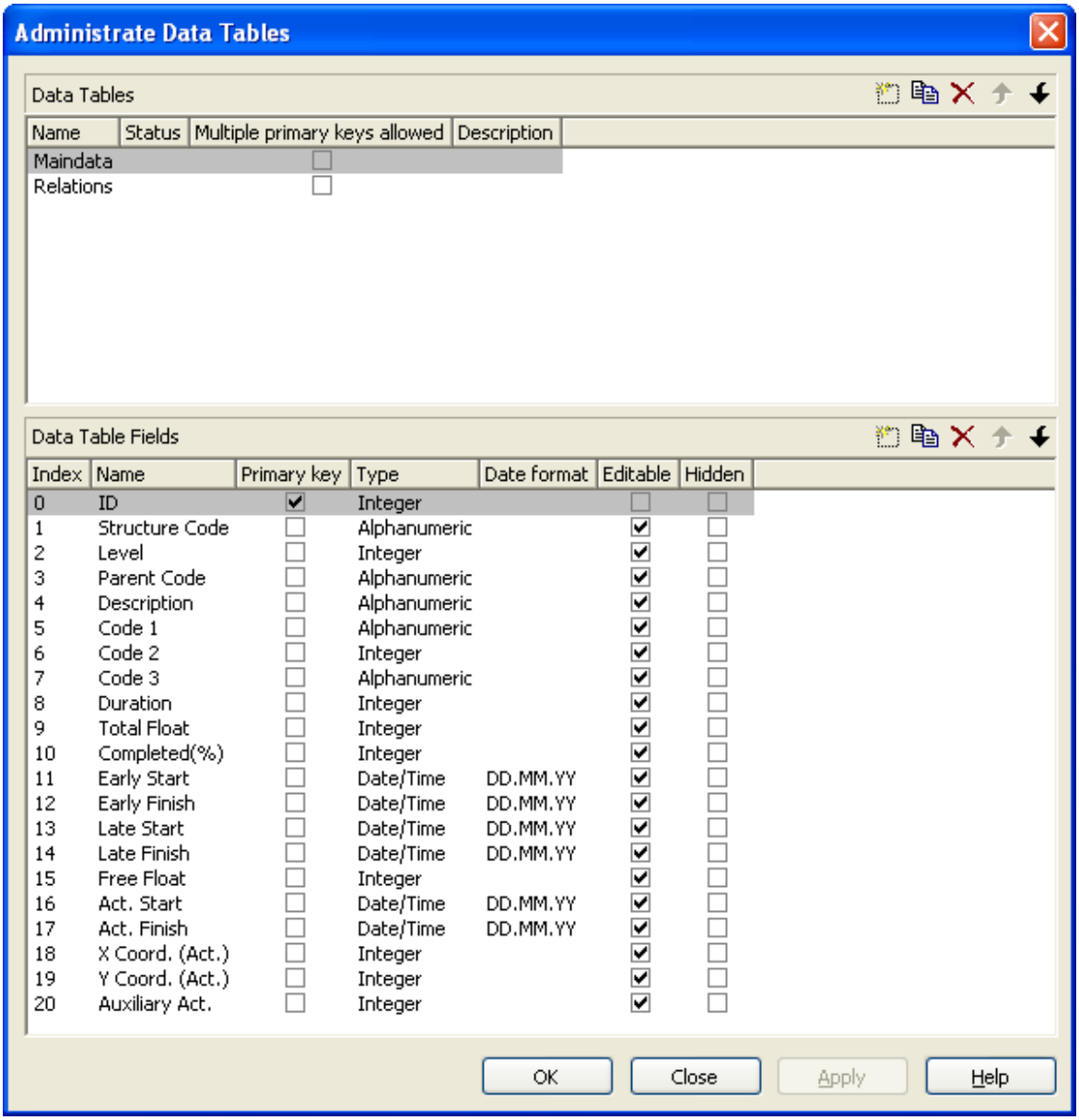
If you wish the bottom and right-hand side of the VARCHART XNet control to be adjusted to the full size of the window during runtime, add the below code:

Example Code

```
Private Sub Form_Resize()
   If ScaleWidth - VcNet1.Left > 0 And _
            ScaleHeight - VcNet1.Top > 0 Then
      VcNet1.Width = ScaleWidth - VcNet1.Left
      VcNet1.Height = ScaleHeight - VcNet1.Top
   End If
End Sub
```

2.5 Preparing the Interface

Prepare the interface now by defining the data fields of the **Maindata** and the **Relations** table. Please click on the button **Data tables...** on the **Objects** property page and open the corresponding dialog.



Please select the data table **Maindata** in the upper list. In the lower list, you can create new fields  , delete fields  or copy fields  . The name can be edited by double-clicking on it. You can select a type from a select box that appears after clicking on the type.

The field of the index "0" by default is named "ID" and is of the type **alphanumeric**. To adapt your interface to this sample, please re-name the

field into "number" and select the data type **Integer**. The ID should **not** be editable to avoid it to be overwriting it in the **Edit Data** default dialog.

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.

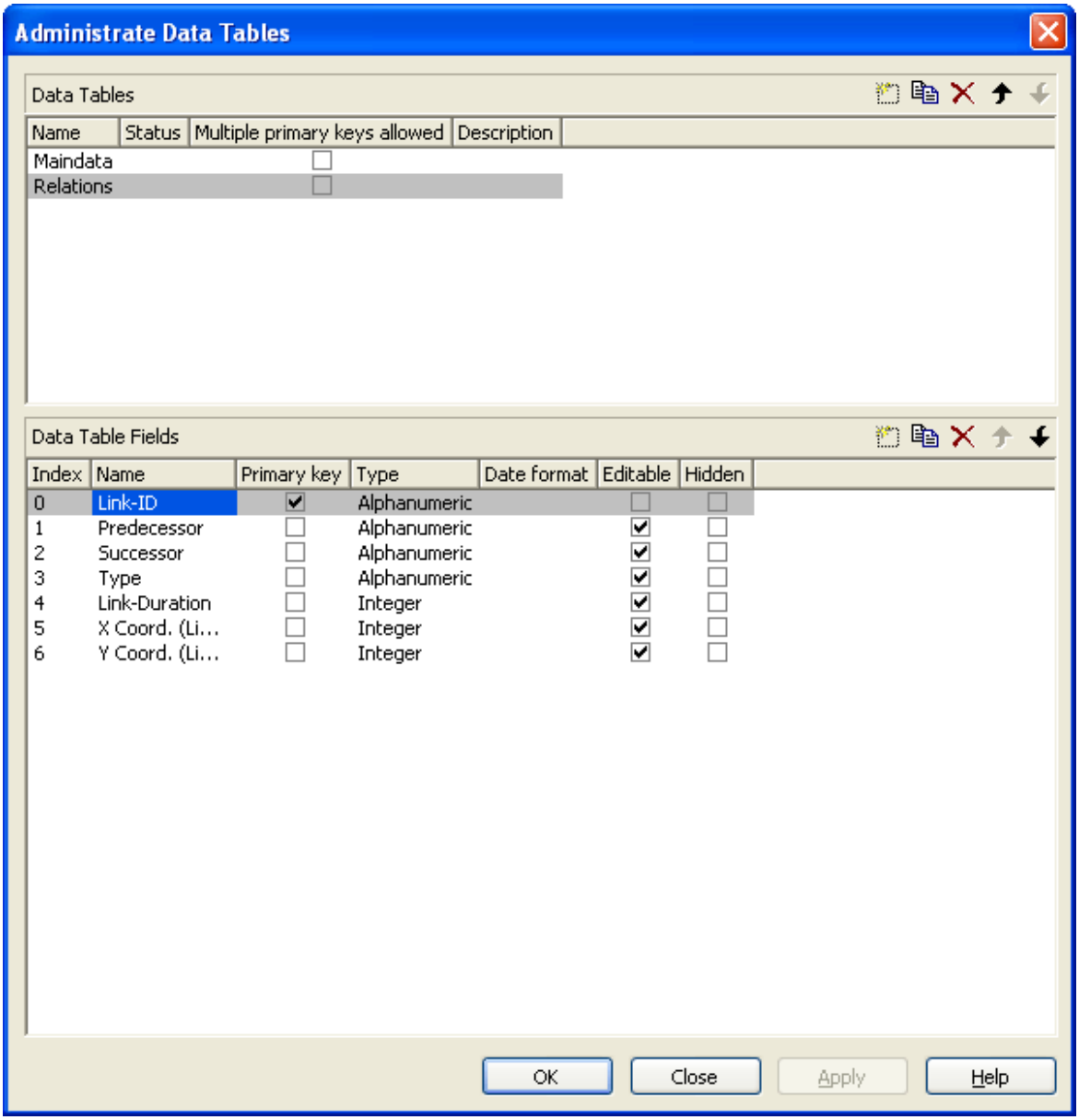
The **Date/Time** fields allow to enter a format. Please select "DD.MM.YY".

To select a field that the node is to be identified by tick the check box **Primary key** for the field <bID.

Fields of the Maindata table:

Index	Name	Primary key	Type	Date format
0	Nummer	True	Integer	
1	Structure code	False	alphanumeric	
2	Level	False	Integer	
3	Parent node	False	alphanumeric	
4	Name	False	alphanumeric	
5	Group code	False	alphanumeric	
6	Code	False	Integer	
7	Group name	False	alphanumeric	
8	Duration	False	Integer	
9	Float	False	Integer	
10	completed (%)	False	Integer	
11	Early start	False	Date/Time	DD.MM.YYYY
12	Early finish	False	Date/Time	DD.MM.YYYY
13	Late start	False	Date/Time	DDD.MM.YYYY
14	Late finish	False	Date/Time	DD.MM.YYYY
15	Free float	False	Integer	
16	Calculated Start	False	Date/Time	DD.MM.YYYY
17	Calculated Finish	False	Date/Time	DD.MM.YYYY
18	X-Coord. (node)	False	Integer	
19	Y-Coord. (node)	False	Integer	
20	Auxiliary node	False	alphanumeric	

Please change to the **Relations** table now.



Tick the check box **Primary key** and deactivate the option "Editable" for the field "Link-ID".

Note: A name that already exists in the table will not be accepted. Instead, the former name will reappear.

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

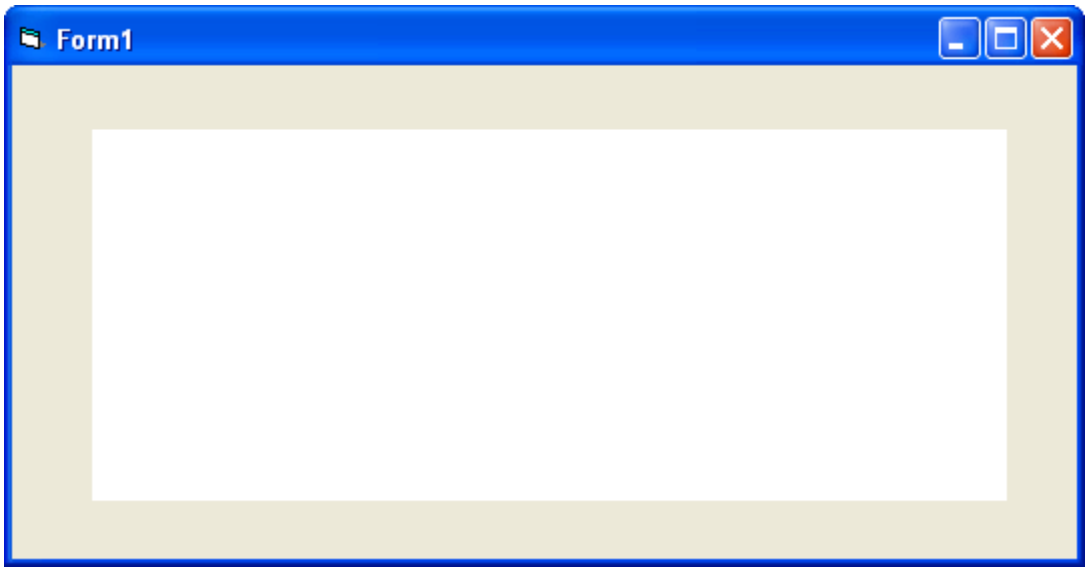
They will as well be stored by clicking on the **OK** button and by changing to a different property page, thus being available to other property pages immediately.

Fields of the Relations table:

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

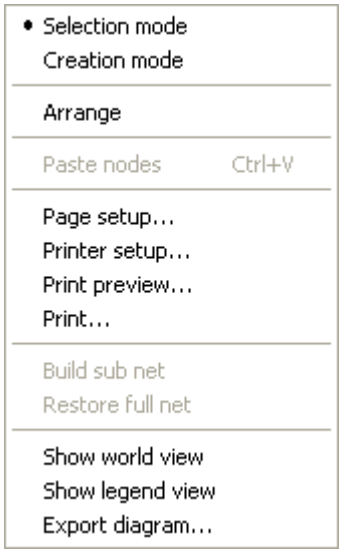
2.6 Your First Run

Start the program by the menu items **Run – Start**, the function key F5 or the appropriate Visual Basic icon ( ). The generated form shows an empty chart.



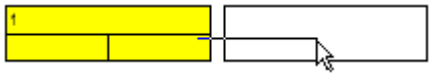
> Creating Nodes and Links

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 button on an empty area in the diagram and select the menu item **Creation mode** from the context menu popping up.



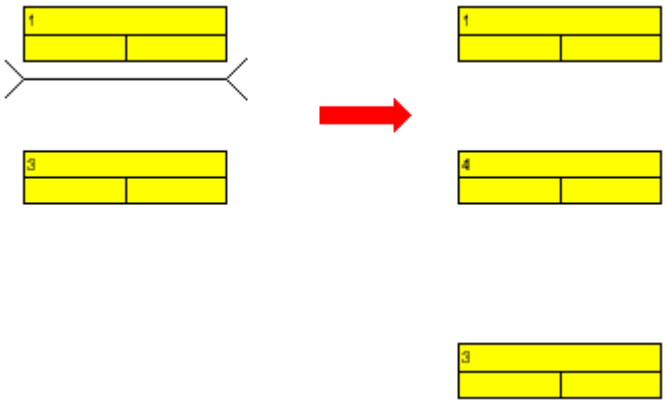
In Creation mode the cursor will transform into a rectangular frame. You can create a node by pressing the left mouse button in an empty area of the

diagram. A link you can generate by dragging the mouse from a node to a different one while keeping the left mouse button depressed. During the dragging operation, the cursor will transform into an arrow that draws a line.



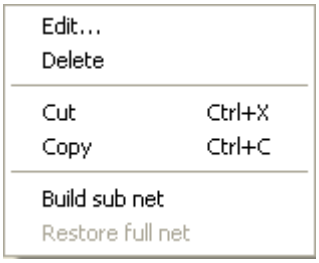
As soon as you release the mouse button, the link will occur. If you drag the mouse between a node and an empty place, both a node and a link will be generated.

If you place the mouse between two nodes that are close together, the 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 button while the bone cursor is showing, the two nodes will shift apart and a node will be inserted in the gap.

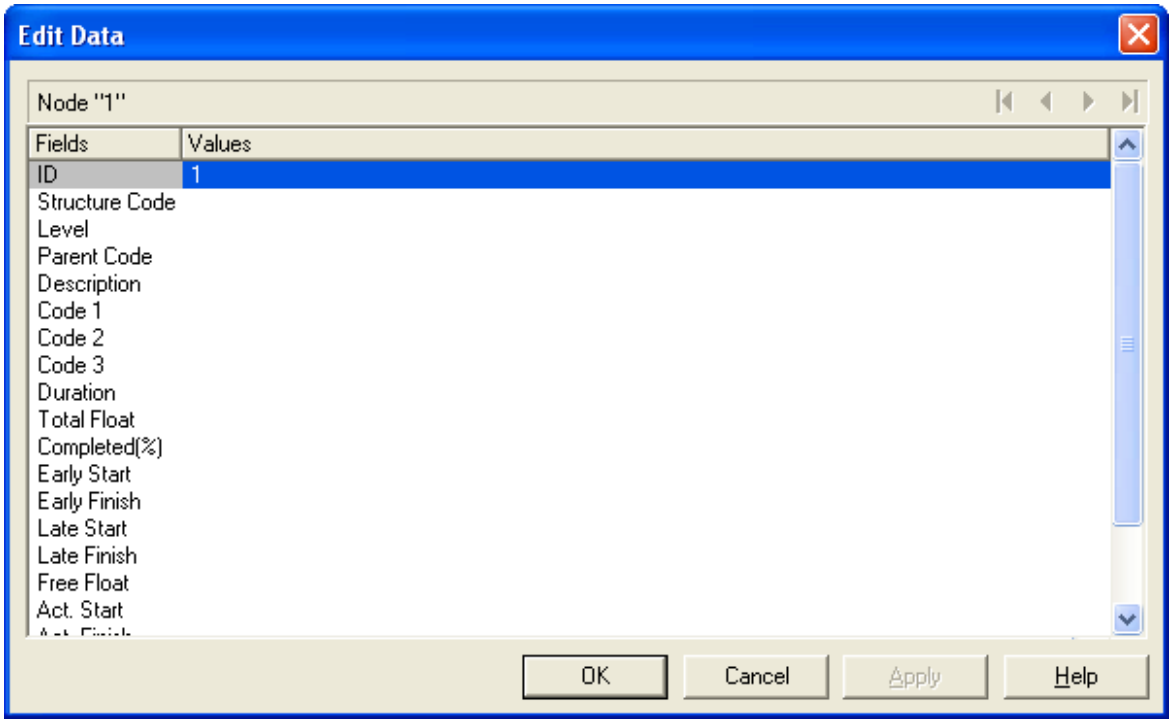


> Editing Nodes

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 to edit, cut, copy or delete nodes.



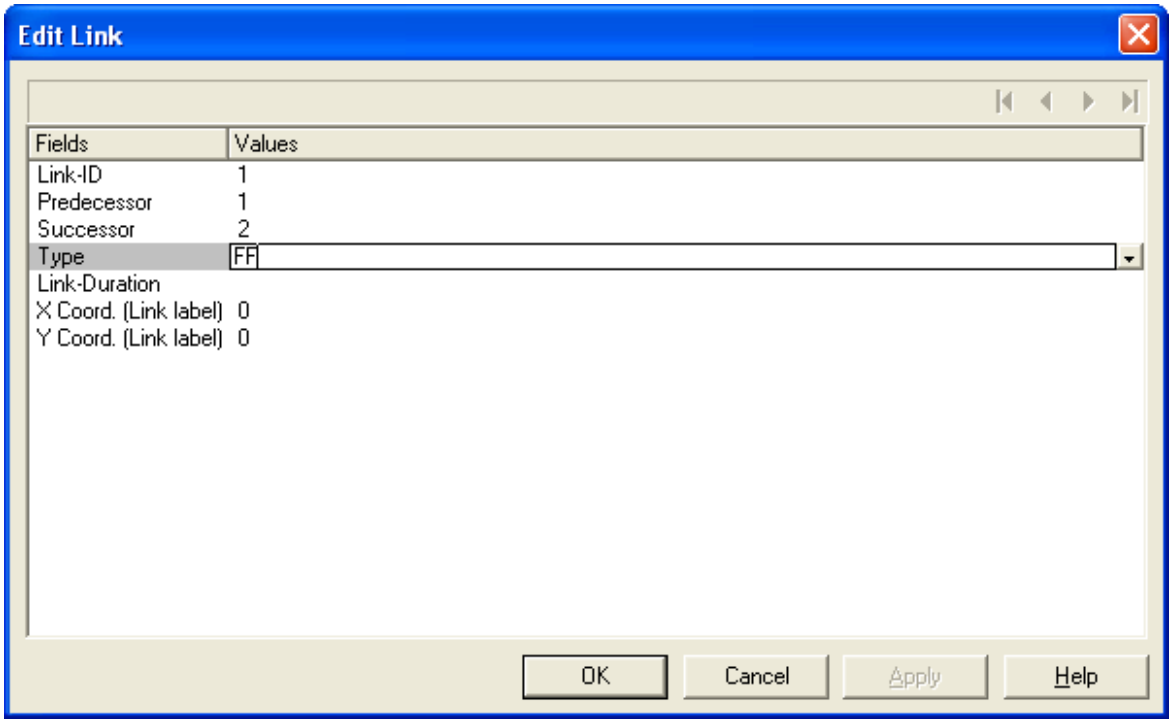
Now invoke the **Edit Data** dialog on a node. You will find the data fields that you defined on the **DataDefinition** property page. The data fields that were defined as **hidden** will not appear in this dialog. The data fields that were defined as **read only** cannot be edited in this dialog.



> Edit Links

You can generate a link by dragging the mouse from a node to a different node in creation mode. 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



This dialog allows to edit the data of the link.

> Moving Nodes and Links interactively

Nodes and links can be moved via the mouse. For this, switch to Selection mode, in case the cursor is in a different mode. Place the cursor onto a node or a link and press the left mouse button. The cursor will turn into a little square and four arrows. You can move the node or link selected as long as you keep the left mouse button depressed. When moving a node, links joining will automatically follow.

> Back to Design Mode

Finish the first run by closing the form.

2.7 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:

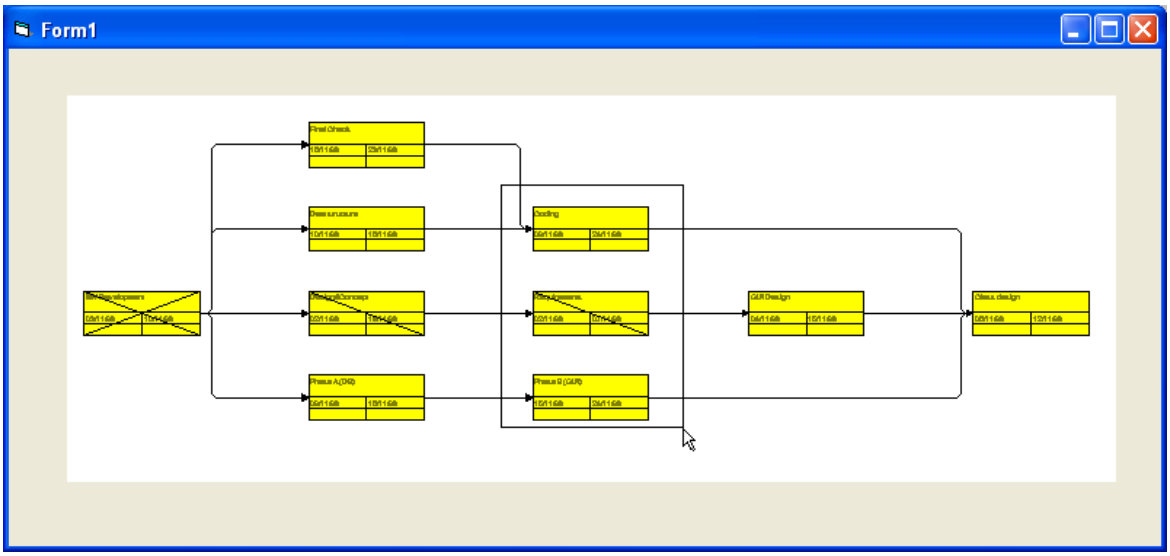
Example Code

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

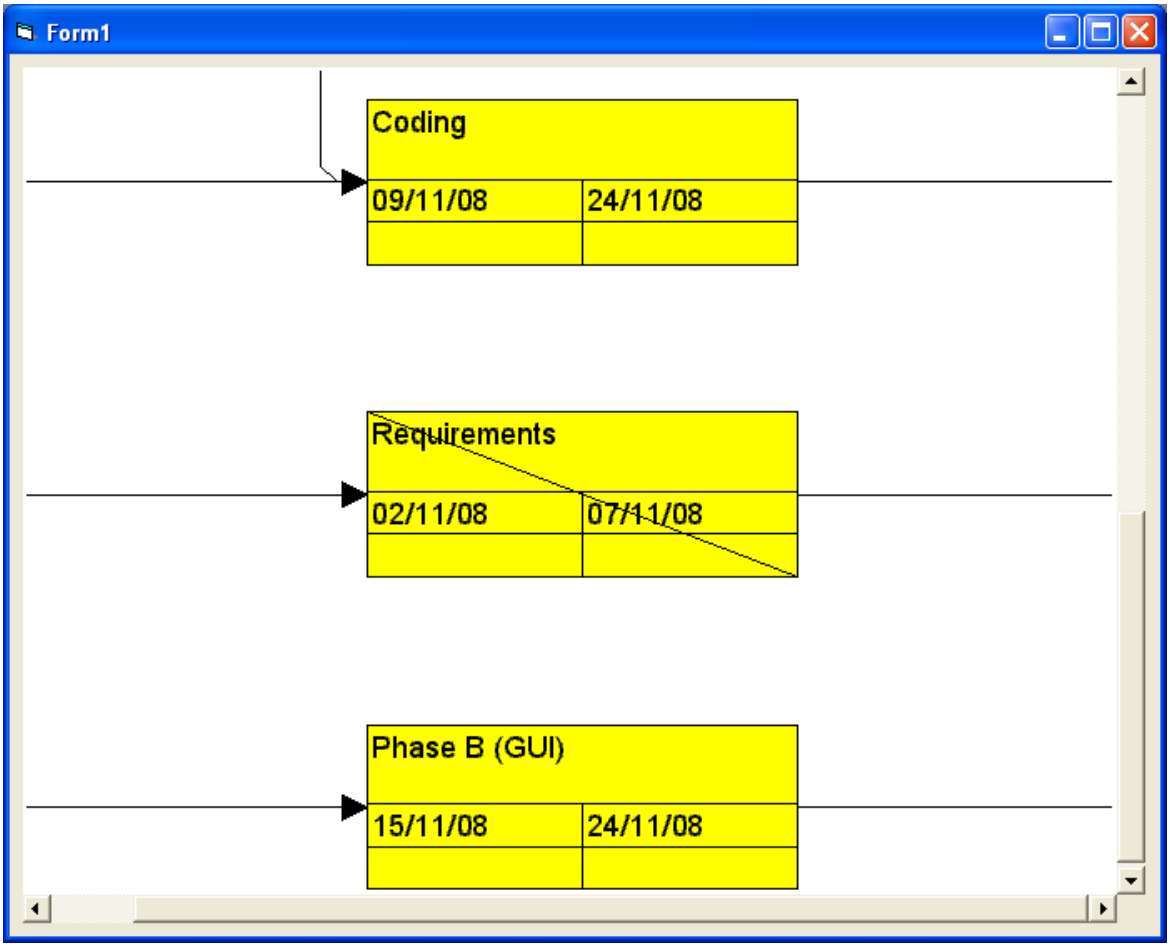
The path depends on the installation of your program. Please save the project now. If you start the program, the nodes and links of the project will be displayed.

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.



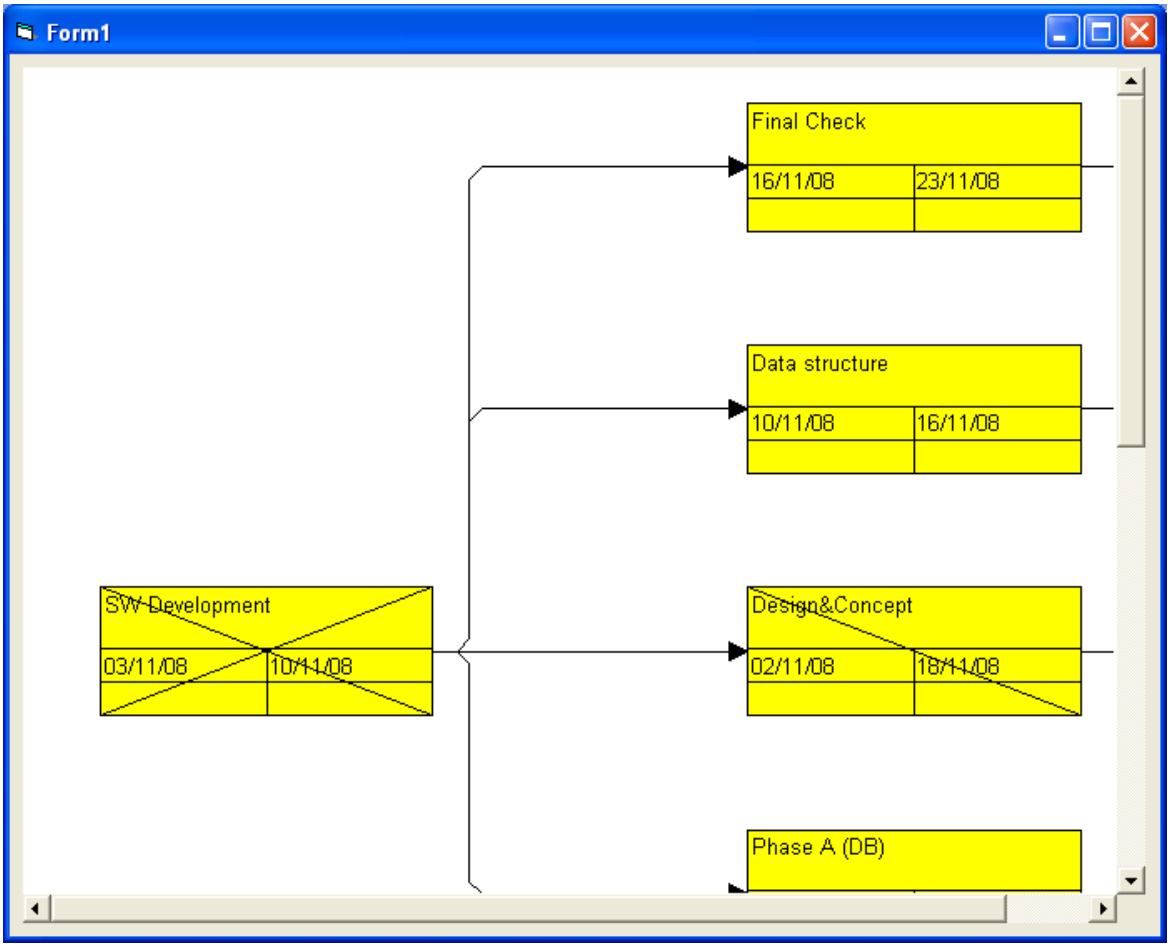
The marked section will be zoomed to full screen size. Use the scrollbars to move through the section and to other parts of the diagram magnified to the same scale.



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
```

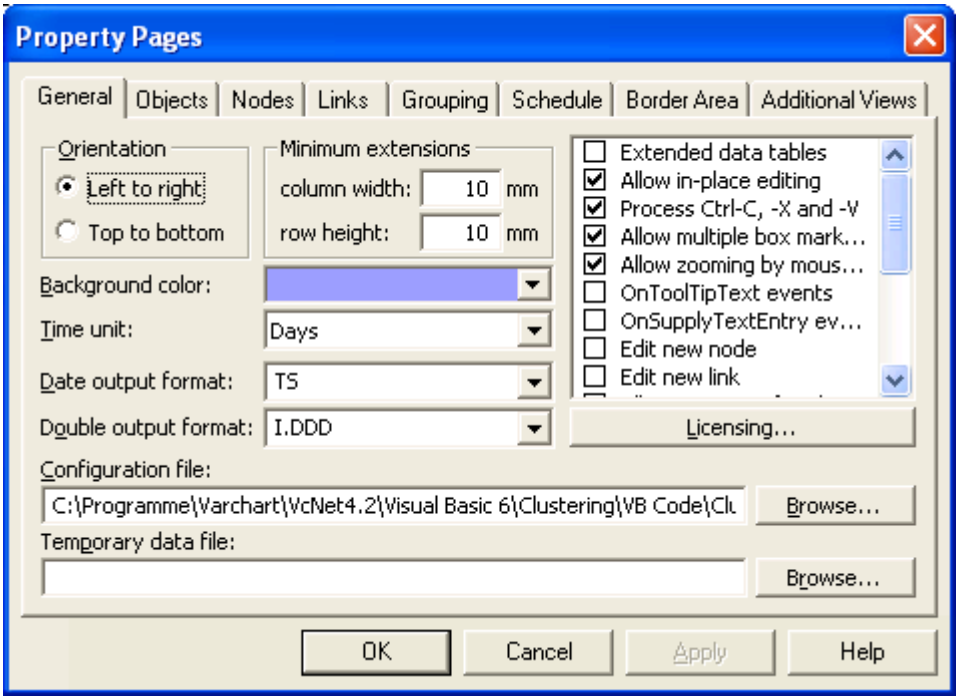


Return to design mode. If you want VARCHART XNet to completely cover the form, please verify the following:

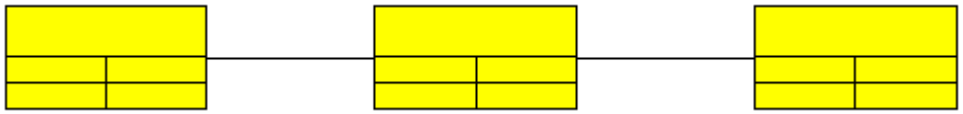
- Make sure that the properties **Top** and **Left** are set to 0. This will position VARCHART XNet to the top left corner of the form.
- Set the VARCHART XNet properties **Width** and **Height** to the form values **ScaleWidth** and **ScaleHeight**. In case you are having VARCHART XNet rescaled automatically,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

2.8 Setting the Orientation of a Diagram

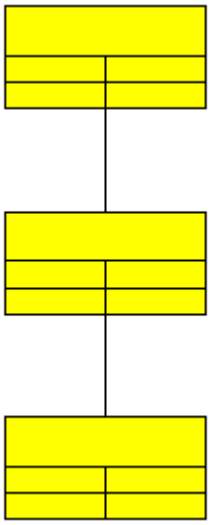
On the **General** property page, you can enter the basic and general settings of VARCHART XNet.



At first, please select by **Orientation** whether the nodes should be directed from left to right or from top to bottom. Try both orientations in the sample file *tutorial.net* 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 by **Execute - Start**, by the F5 key or by the corresponding Visual Basic icon ( ). 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.

2.9 Selecting a Project Data File for Design Mode

During design mode, on the **General** property page you can enter or select the name of a file via the field **Temporary data file** in order to load node and link data and to control them directly via the component. By the **Browse** button you can open the Windows dialog **Load/Save** that will display the preset file type **Data files** (*.net).

Please select the delivered sample file *tutorial.net* and confirm your selection by clicking on the **Apply** button. From now on, the nodes defined in that file will be displayed in the form.

Now please try how modifcations of the settings are displayed during design mode. Please open the **Nodes** property page and click on the **Appearances** button. The **Node appearances** dialog will open, with the "Standard" node appearance marked. Please click on the **Edit** button to get to **Edit Node Design** dialog. Modify some settings here, such as the background color, the line color, the line type etc. Any changement will be displayed in the preview window. When you confirm your modifications by the **OK** button in the dialog and the **Apply** button on the property page, the modifications of the appearance will be displayed in the form.

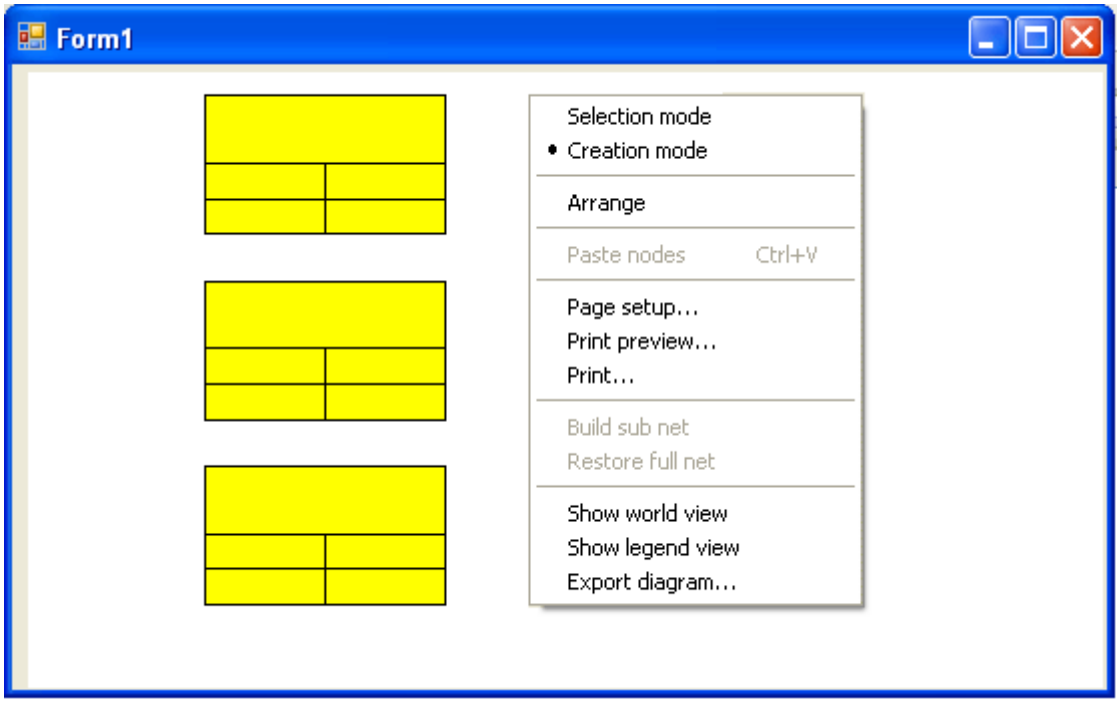
Please note that the data file selected will be valid for design mode only. During runtime, a file has to be opened by the **Open** method.

Alternatively, you can insert nodes and links into a network diagram by the methods **InsertNodeRecord** and **InsertLinkRecord**.

2.10 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 by mouse clicks. 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 was 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 by pressing 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.

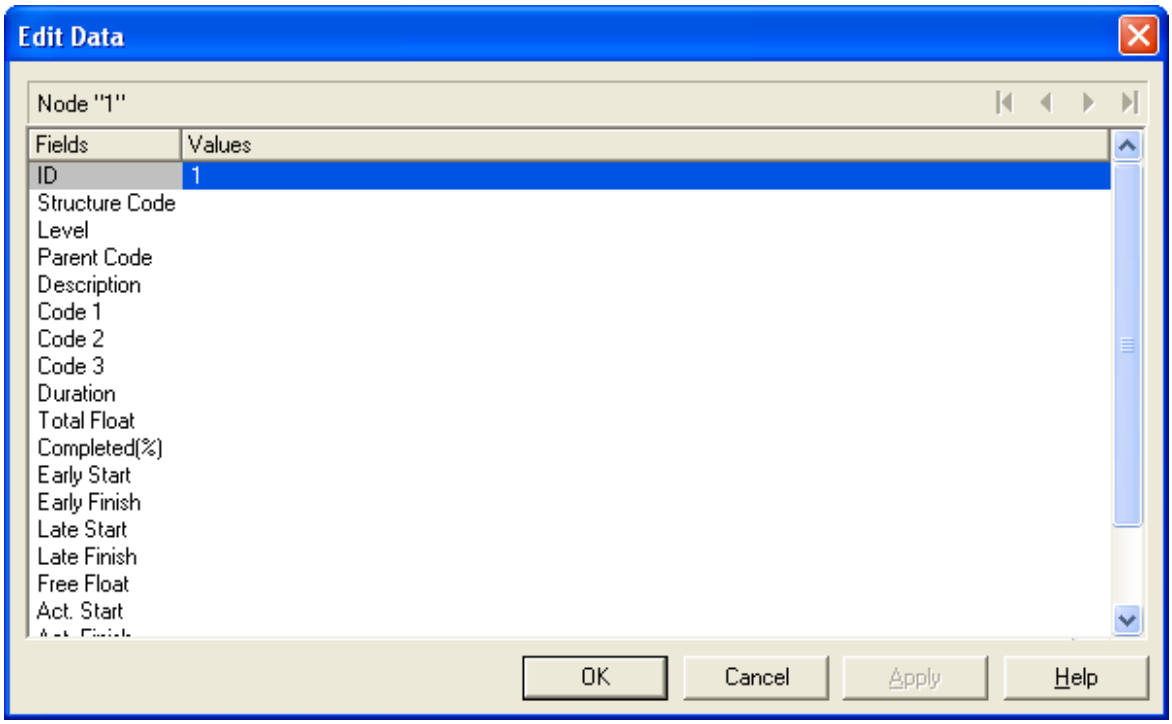


If you deactivate the option **Allow creation of nodes and links**, the user will be unable to generate nodes and links, even when the creation mode is switched on. Nodes and links can then only be loaded by API calls.

Please activate the options **Edit new nodes** and **Edit new links** now.

Start the application and change to creation mode. Click the right mouse button on a node. The context menu for nodes appears. Choose the **Edit** menu item. Then the **Edit Data** dialog will open. The data fields that you defined on the **DataDefinition** property page (not defined as hidden) are displayed in the **Edit Data** dialog. You can edit the data fields and the values

of the node record. When you press the **OK** button, the node will be generated from the values set.



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.

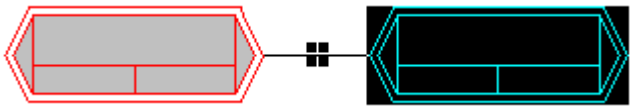
2.11 Marking Nodes and Links

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** combo 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:

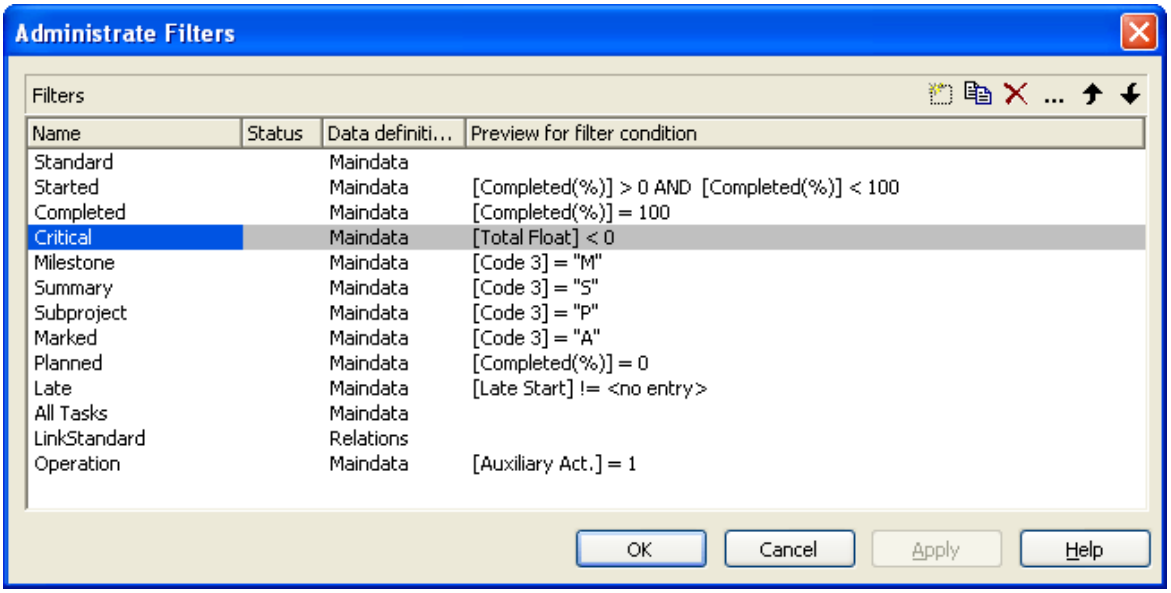


2.12 Setting Filters for Nodes

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

When you use a filter in a node appearance, only those nodes will show the appearance that match the filter conditions.

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.



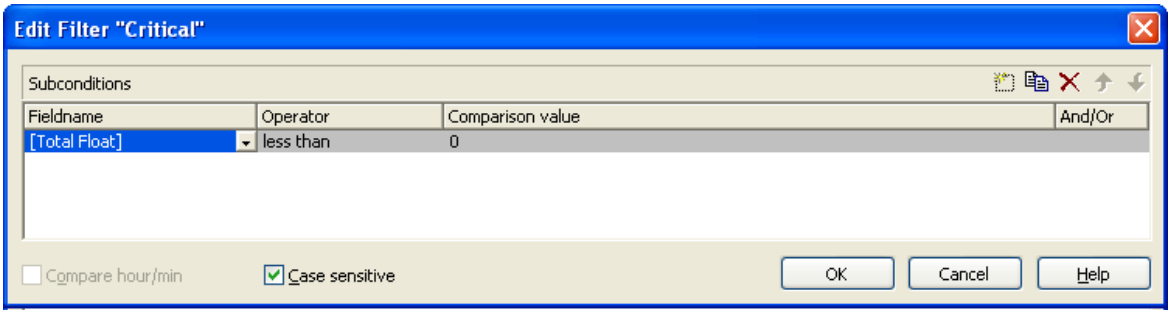
> Buttons in the "Administrate Filters" dialog box

- 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 into "Critical".

Now edit the new filter. Click on the **Edit filter** button to get to the **Edit Filter** dialog box. Please enter the below settings:



The head line indicates the name of the current filter.

The **Code name** field displays the data field the value of which is compared to the **Comparison value**. Please select the field "Total Float".

The **Operator** field displays the current operator. The type of operator available depends on the type of data field selected. Please select the operator "<" 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 "0".

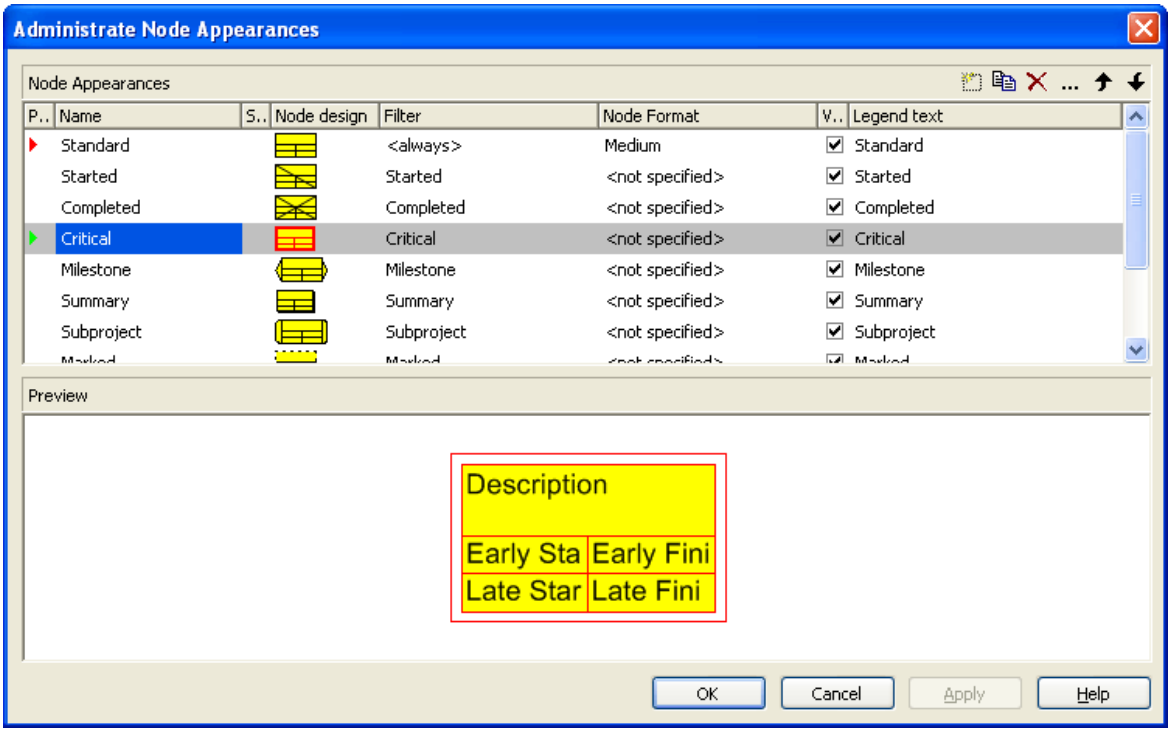
In the **And/Or** column you can choose the operators to combine the condition of the current row with the one in the row below, if necessary.

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

2.13 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.



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**

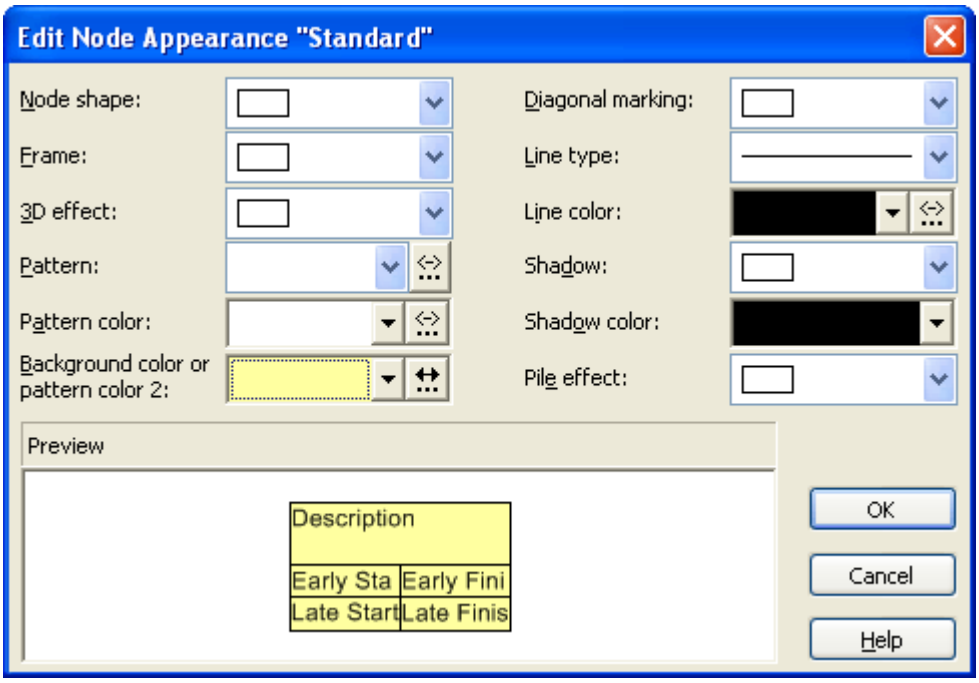
 **Edit node appearance**

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

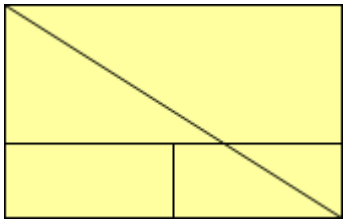
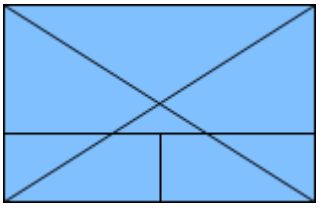
> Using node appearances and filters

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, specify the node format and the filter to be combined with the node appearance.



Please enter the below settings:

Node appearance	Started	Finished
Filter	Started	Finished
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.

- 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 pale yellow background and a downward strike-through pattern.

- Finally, please enter "100" into "Completed(%)": The node will show the "Finished" appearance, that has a crossed-lines strike-through pattern and a blue blackground.

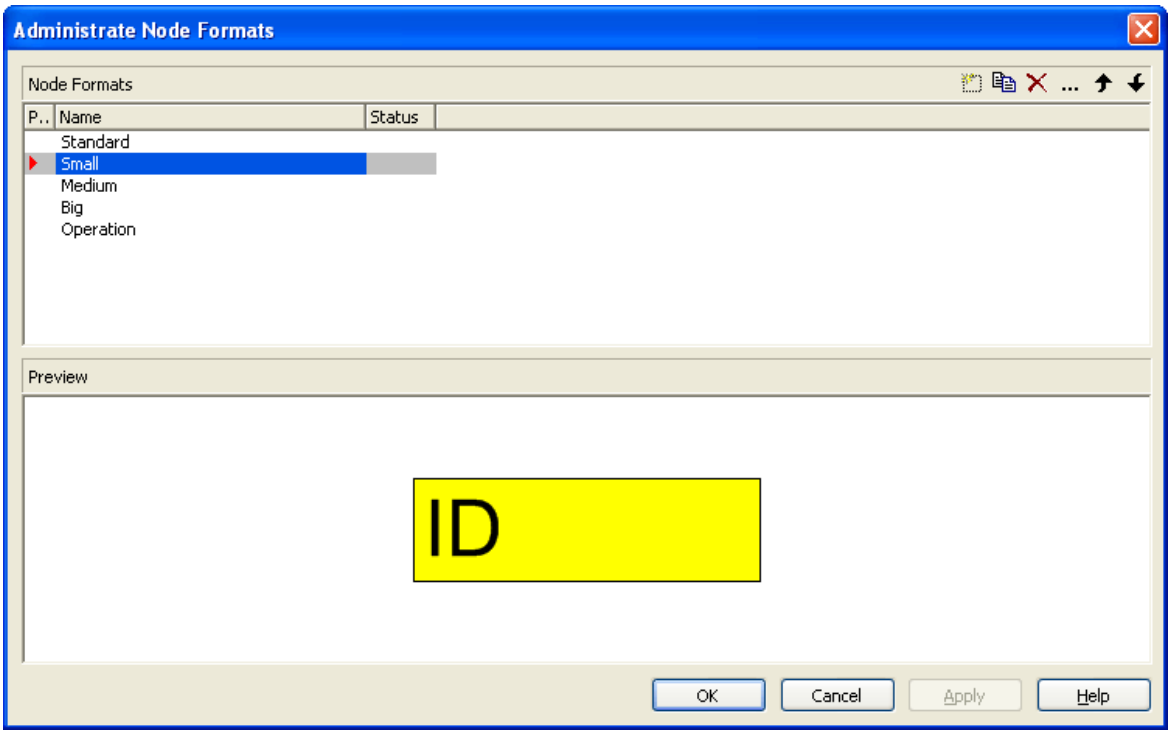
> Specifying the node appearance in dependence on its data

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

2.14 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.



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

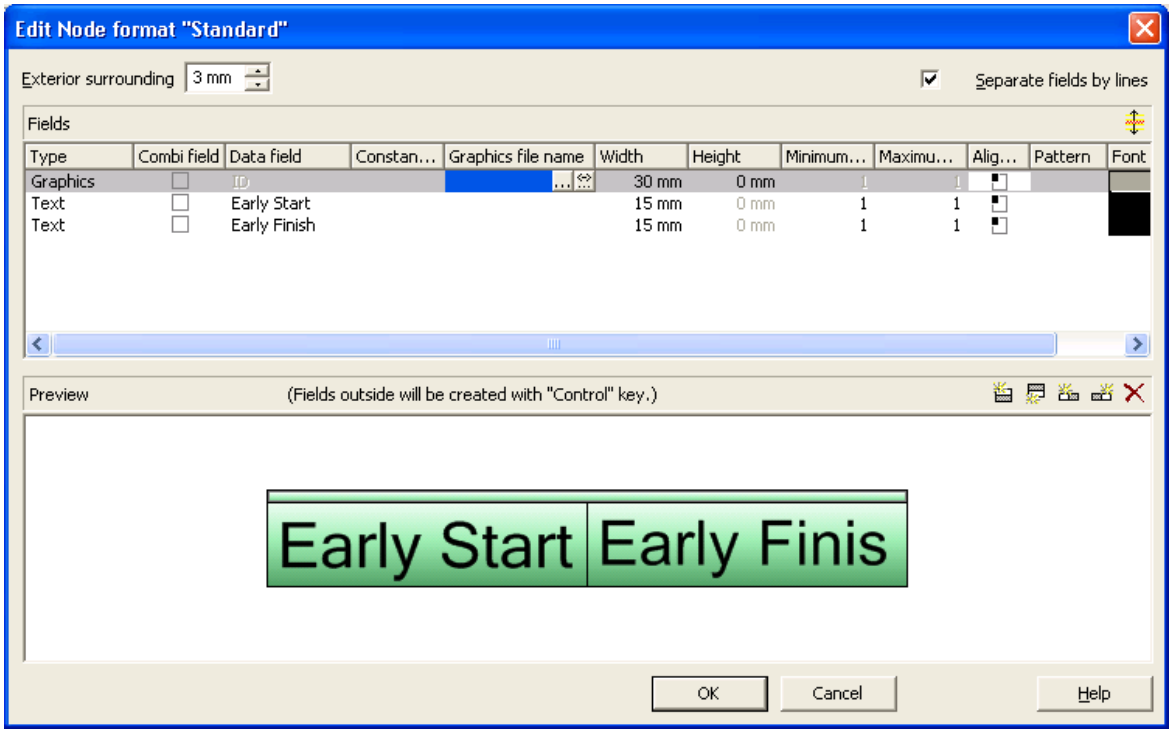
Delete node format

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.



In this dialog box you can set 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 the content of which is to be displayed in the field marked, or a constant text
- for the type **graphics**: the name and directory of the graphics file to be displayed in the marked field
- the width and height of the marked field
- the maximum number of text text lines to be displayed in the marked field
- the alignment of the text/graphics in the marked field
- the background color of the marked field
- the fill pattern of the marked field
- the font attributes of the marked 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.

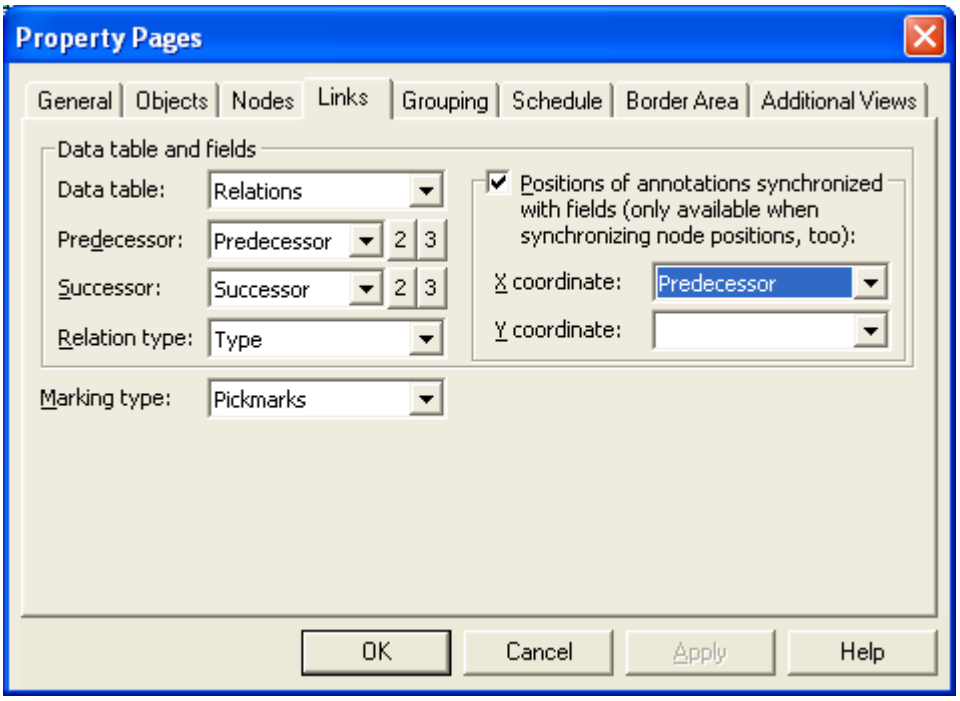
If a mapping has been configured, a symbol is displayed besides the symbol file name ().

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

2.15 Setting the Link Appearances

In the above paragraphs you learned how to apply filters to nodes. Beside to node appearances, filters can also be applied to link appearances. You can, for example, set 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.



In the **Appearances** table, each line displays the **Name**, the **Filter** and the **Line type** of a link appearance.

In the lowest line, you find an entry "New...". Please double-click on that entry to edit its name and type "FF link". As soon as you click on a different field, the appearance will be created and added to the list. The appearance will show 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 combo box by clicking on its arrow-down button. Up to now, a single filter named "LinkStandard" exists.

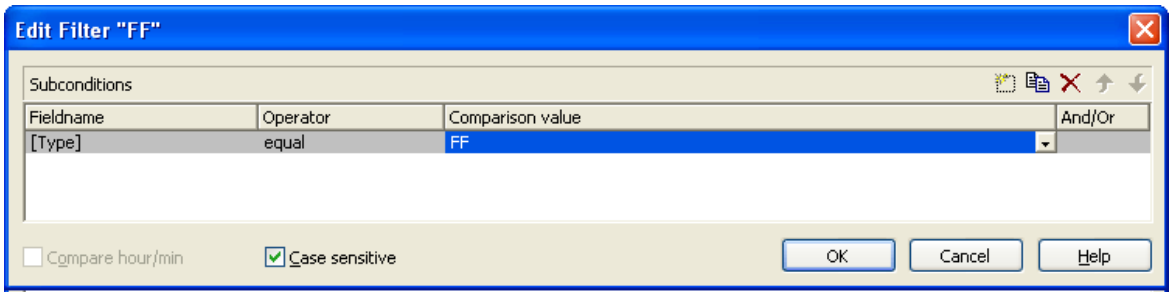
Please create a new filter now by clicking on the **Edit** button of the **Filter** field. The **Administrate Filters** dialog will open, that lets you generate new filters, or copy, edit and delete existing ones. Please click on the "Add filter" button to create a new filter. Rename this filter into "FF". It is designed to select for finish-finish links.

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

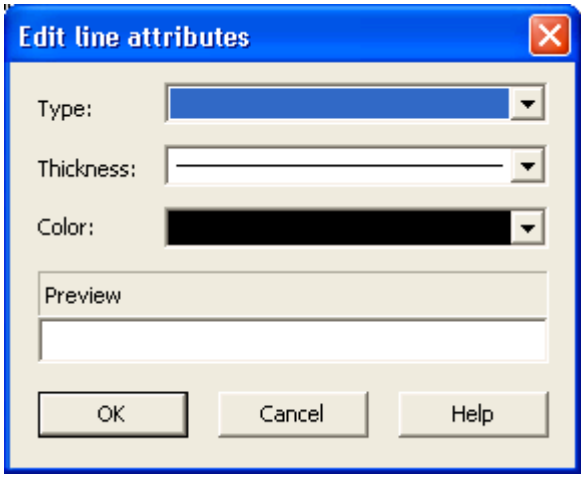
Please define the criterion "Link type equals FF" now. All links of the type "FF" will now adopt the link appearance associated with this filter.

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.

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



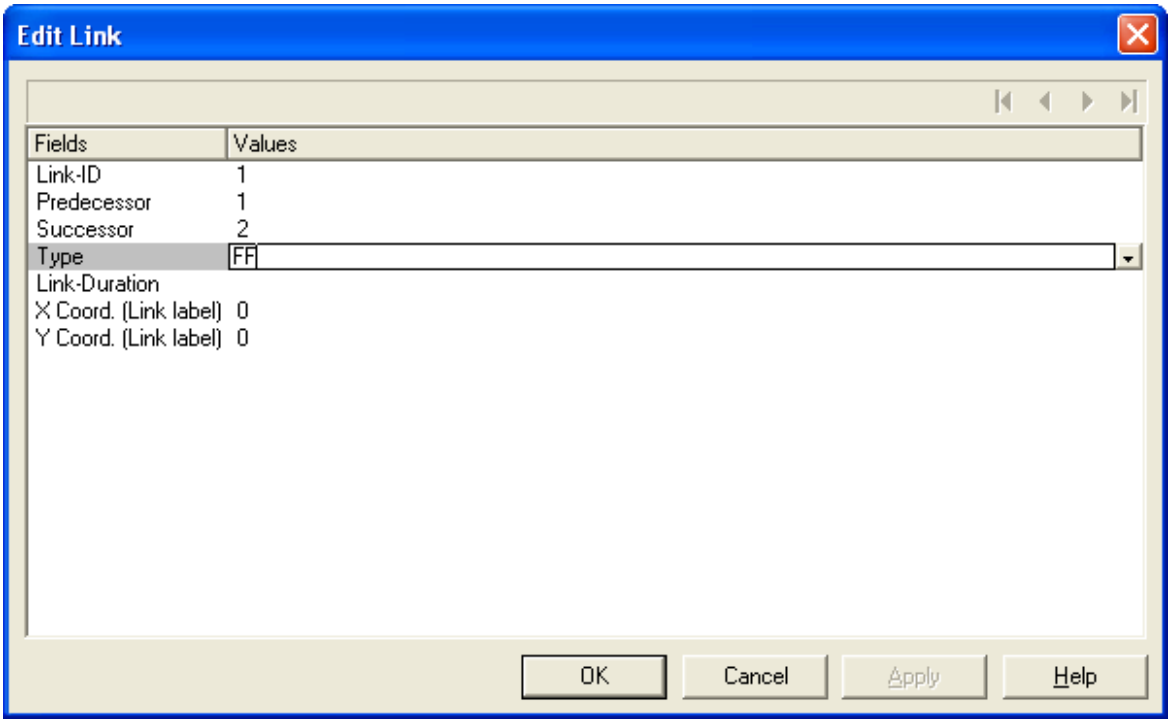
Please set the line attributes for the "FF link" link appearance now. When you click on the entry of the field **Line type**, an **Edit** button will occur by which you can get to the **Edit line attributes** dialog. There you can set the color, type and thickness of the line.



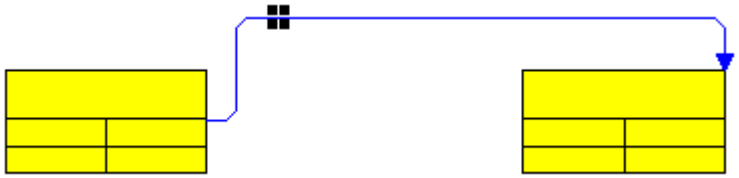
Select a blue color for the links that fulfil the criterion of the "FF link" filter, i.e. that are of the FF type.

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

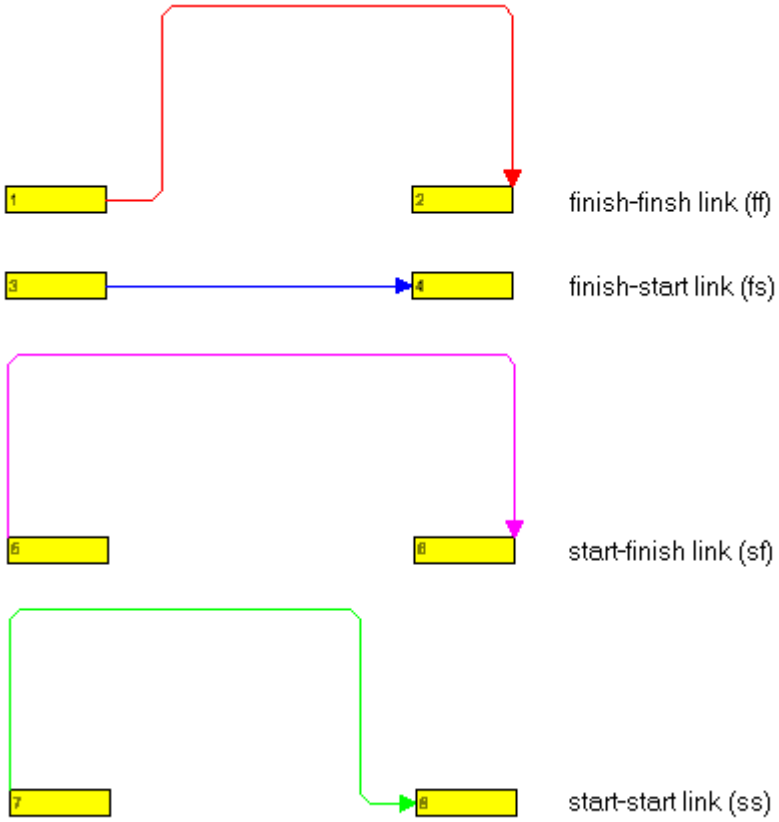
Click on the link by the right mouse button and select the **Edit** menu item from the context menu. It will open the **Edit Link** dialog.



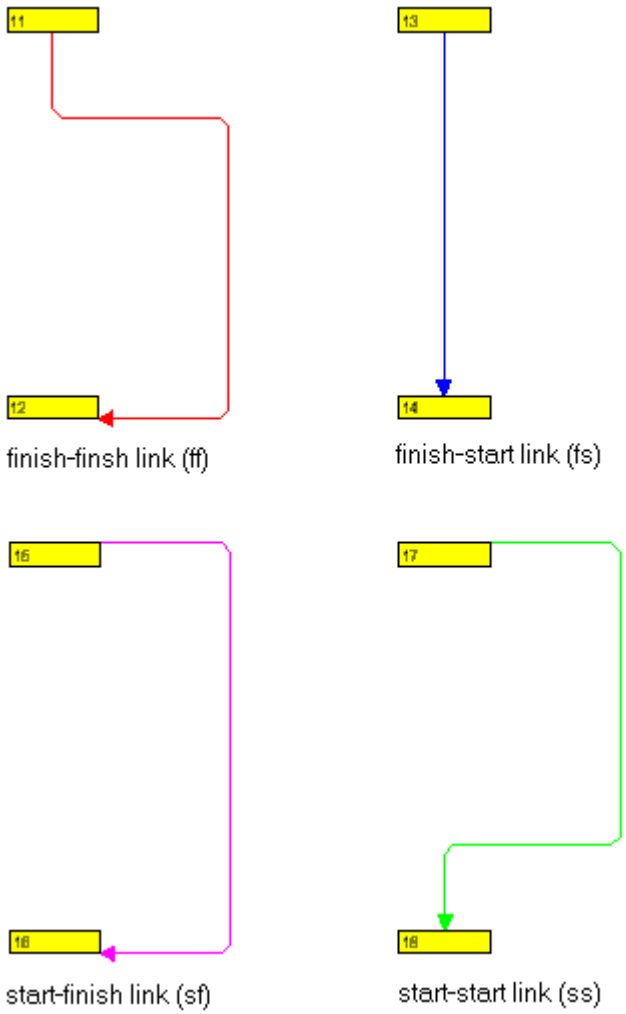
Enter "FF" into the **Type** and confirm by clicking on the **OK** button. The marked link will turn into a finish-finish relation and will be immediately diplayed as a blue line.



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



Left-to-right orientation



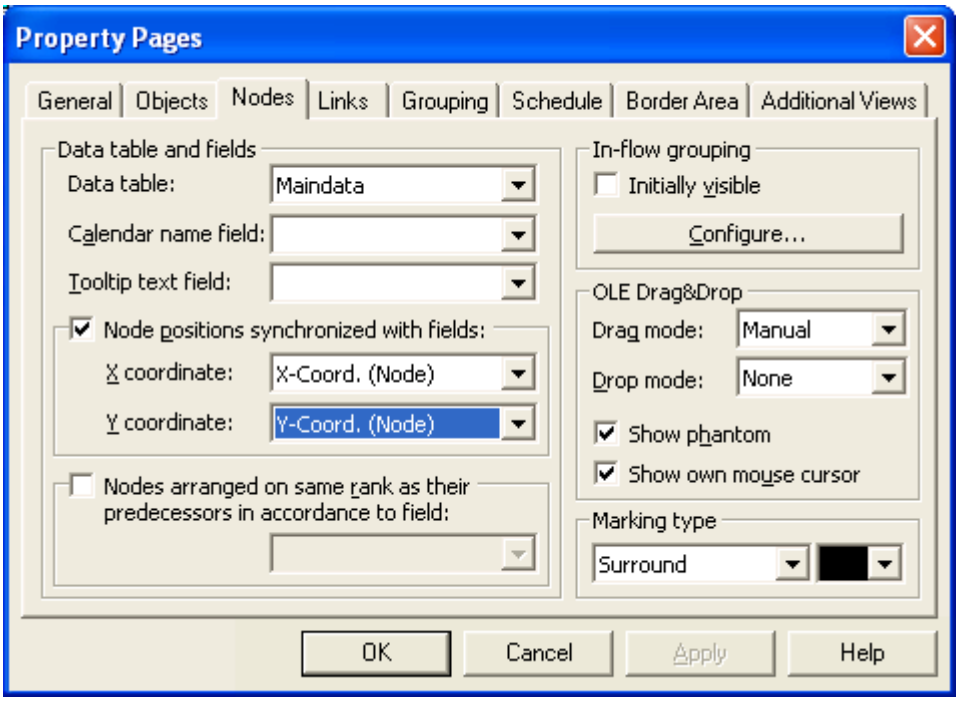
Top-to-bottom orientation

2.16 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.

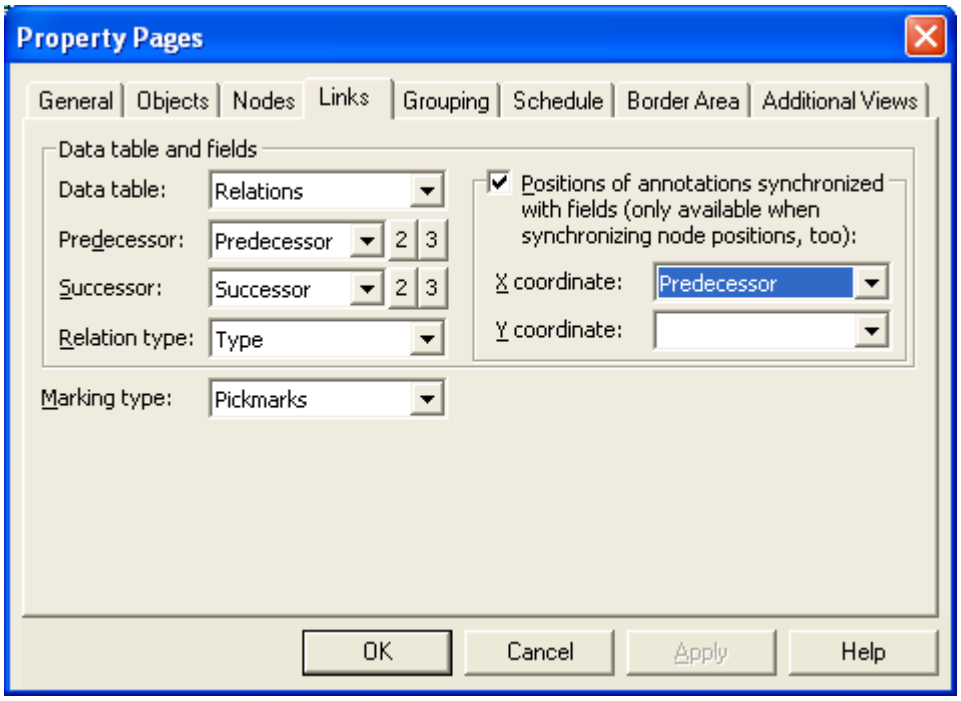
To synchronize node positions with their data fields, please activate the check box **Node positions synchronized with data fields** on the **Nodes** property page and select the following data fields:

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



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)"

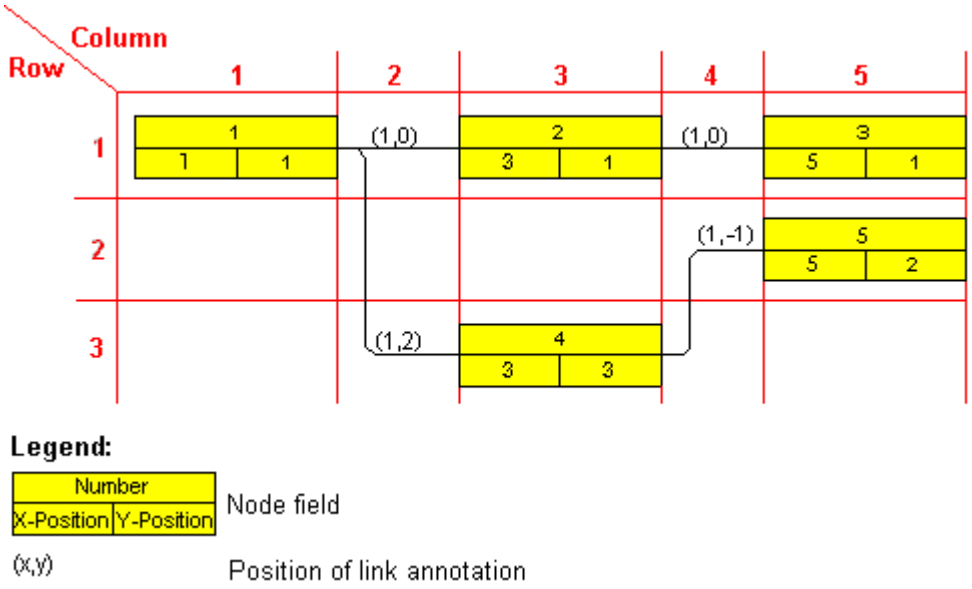


Positions of nodes and of link annotations are stored as coordinates in a matrix.

- 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

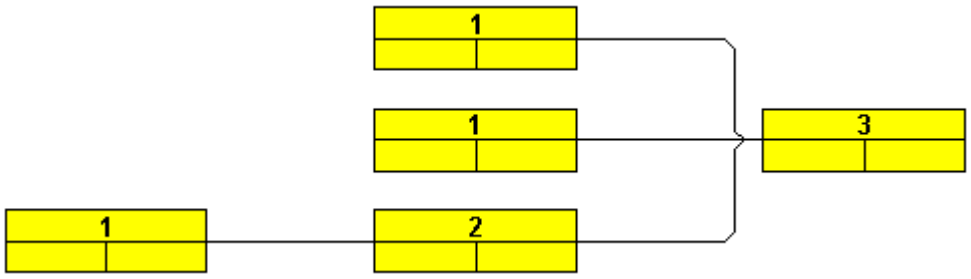
```
private void Clustering_FormClosing(object sender, FormClosingEventArgs
e)
{
   vcNet1.SaveAsEx(@"....", VcEncoding.vcUnicodeEncoding);
}
```

2.17 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

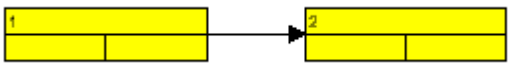
This is how ranks of nodes work:

- 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).

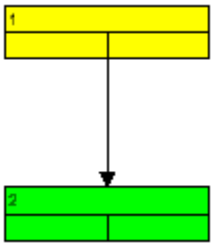
Ranks are calculated by clicking on the **Arrange** item of the diagram context menu. They serve as a base to the layout algorithm to position the nodes in the overall orientation. If cyclic structures exist in the chart, VARCHART XNet will identify them by a separate algorithm and ignore them temporarily.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

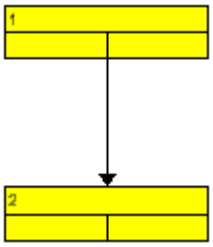


Rank 1 and rank 2 holding a node each

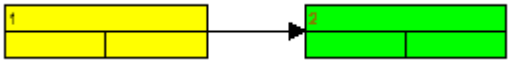


*The rank number of the second node was diminished by 1. Then **Arrange** was invoked.*

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



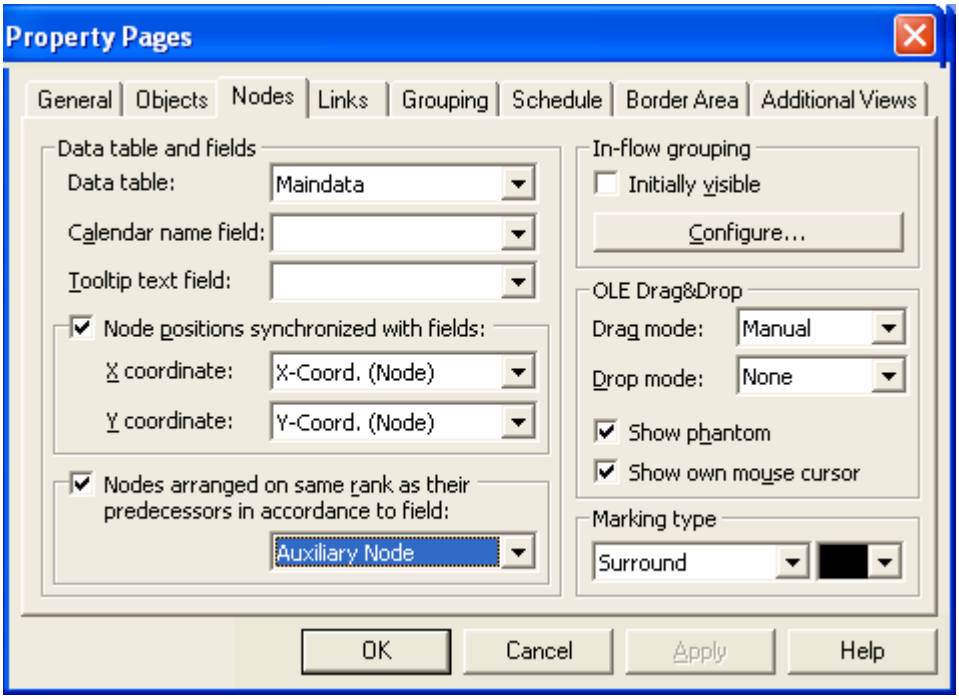
*The rank number of the second node was diminished by 1. After this, **Arrange** was invoked.*

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**.

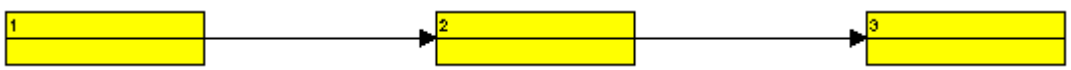
Value in the field "Auxiliary nodes"	Top-to-bottom orientation	Left-to-right orientation
0	The rank number of the auxiliary node is not diminished.	The rank number of the auxiliary node is not diminished.
1	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears left or right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

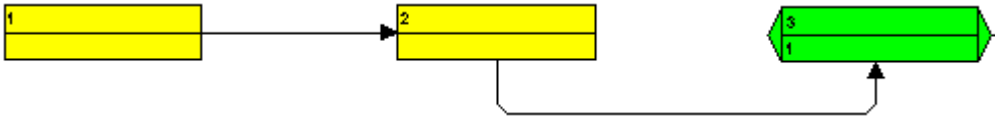
Please follow the below example of placing auxiliary nodes in the same rank as their predecessors. Select the **Left to right** orientation on the **General** property page. Then tick the check box **Nodes arranged on same rank as their predecessor in accordance to data field** on the **Nodes** property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



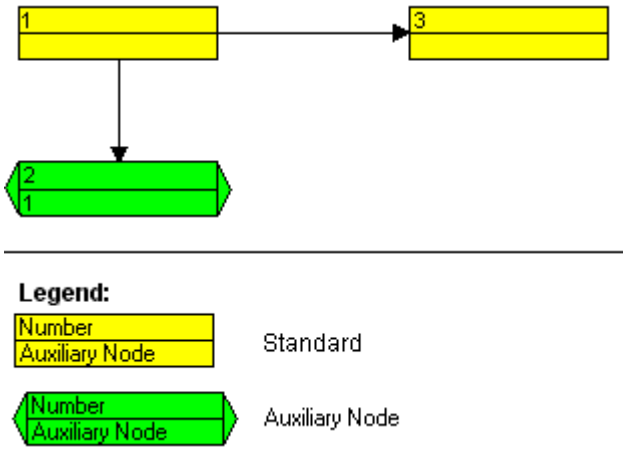
Run the program, generate some nodes and link them as shown in the below picture:



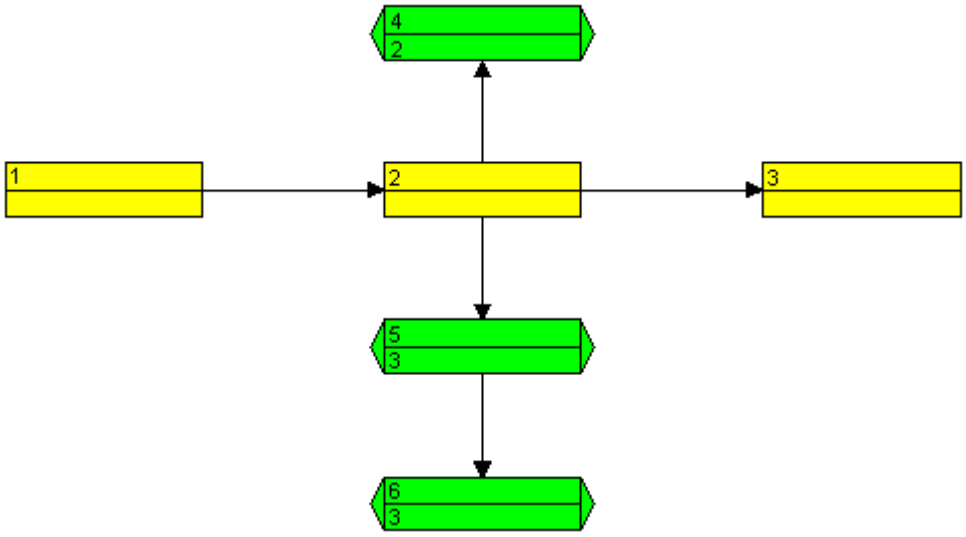
Double-click on the third node and enter the value "1" into the "Auxiliary node" field of the **Edit Data** dialog. This will be the result:



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

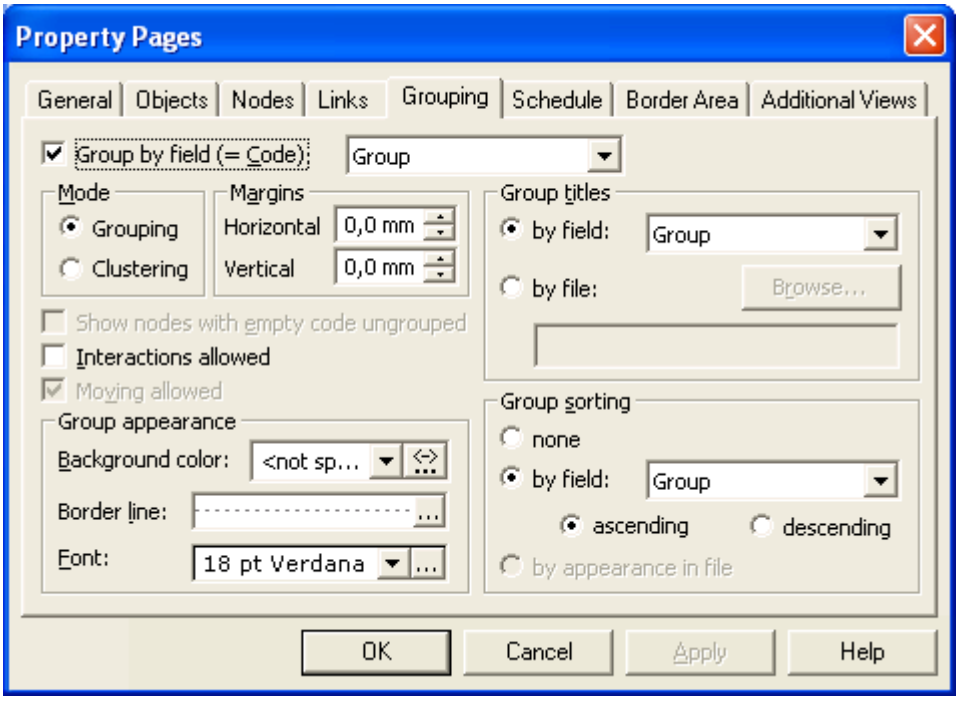


The node the rank of which was reduced, will be placed below instead of right of its predecessor. The picture below shows the ranking of different auxiliary nodes (left-to-right orientation):



2.18 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.



You can group nodes by the field that you select 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. 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 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.

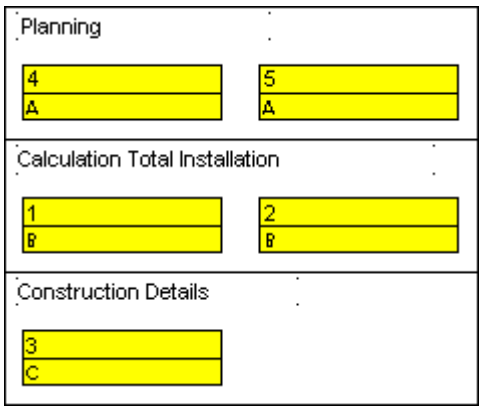
For generating and editing nodes during runtime, as an example in this tutorial please use the tables

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:

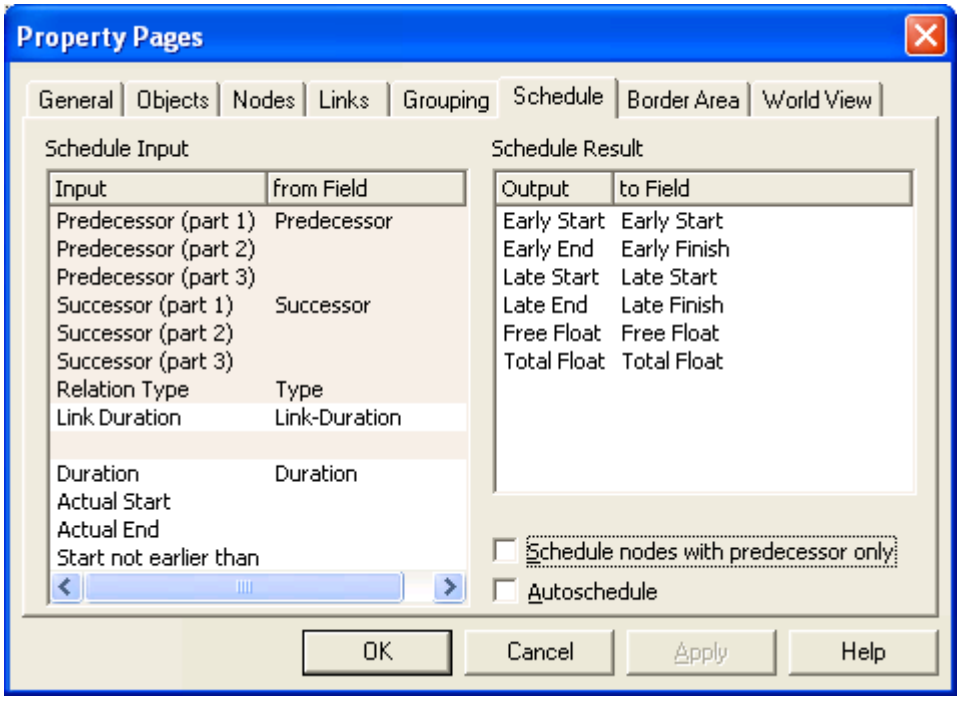


Please move a node to a different group. The value in the "Group code" field will automatically change. You can verify this by invoking the **Edit Data** dialog of the node to view the field.

2.19 Setting the Scheduling Options in 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.



Please select in the **Schedule Input** table for each item of the **from Field** column a field from the combo 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 combo 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 by API calls, 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
```

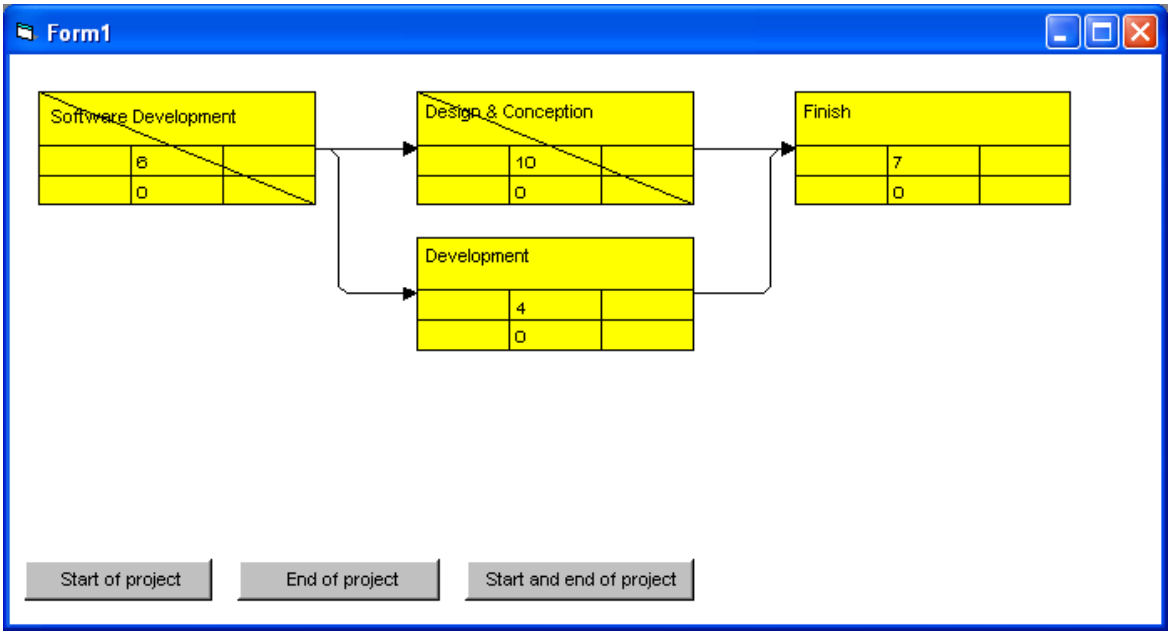
```
Private Sub Command2_Click()
VcNet1.ScheduleProject 0, "01.02.2000"
End Sub

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
```

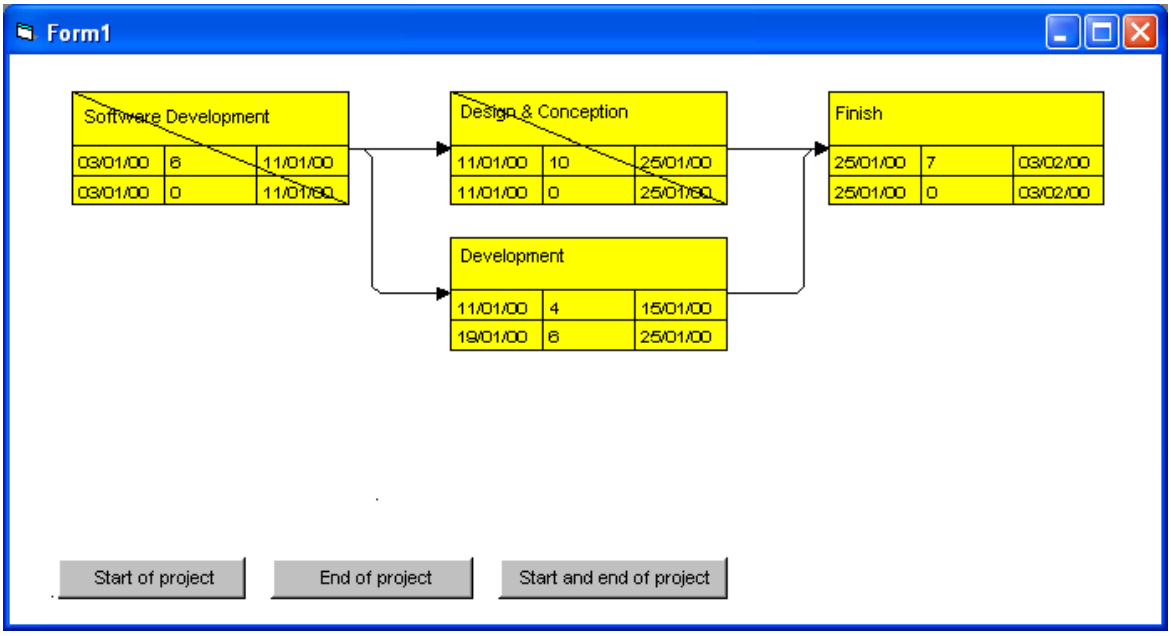
Please start the program now. The nodes and links loaded by the API are displayed:



Node format used: "Big"

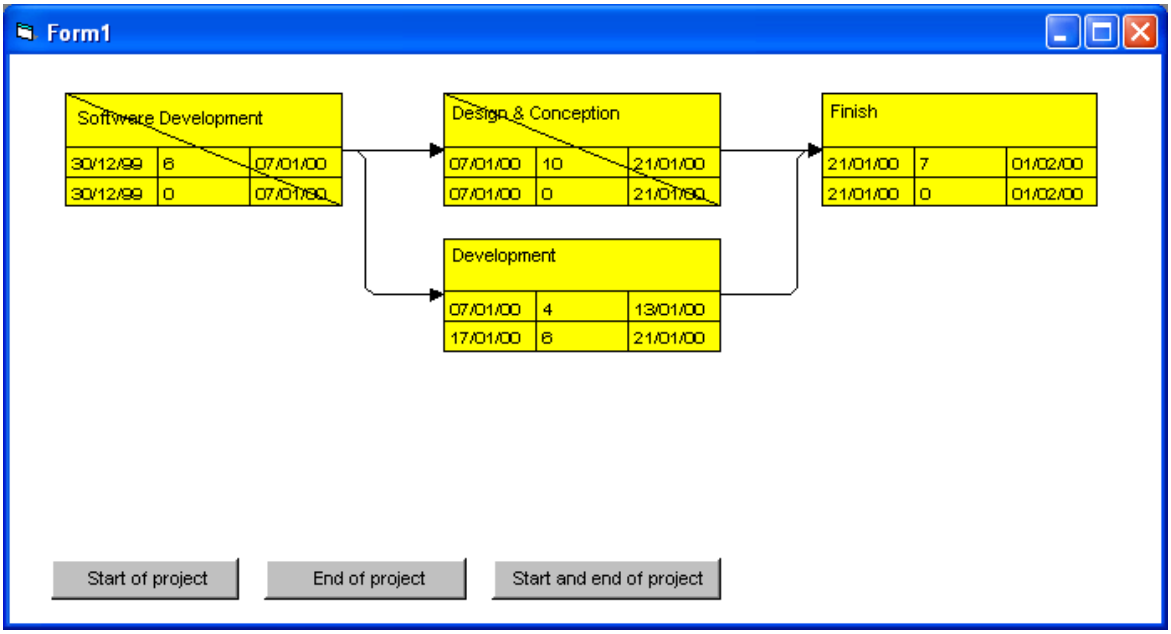
Name		
Early start	Duration	Early finish
Late start	Float	Late finish

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

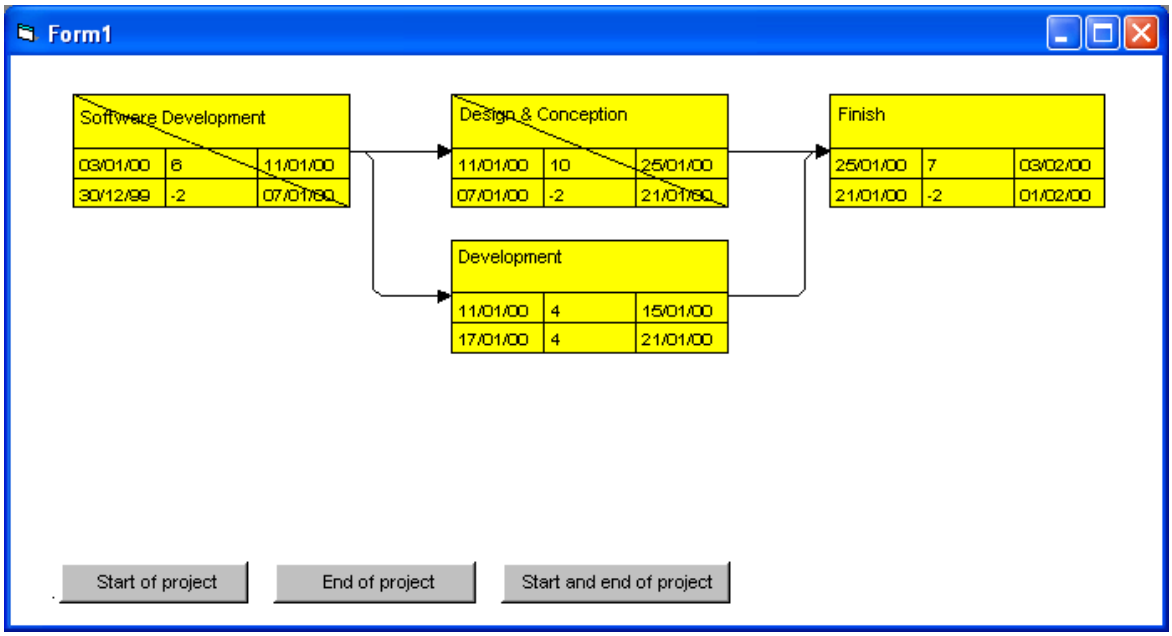


Please note that an internal calendar will be considered where weekends represent work-free periods. The internal calendar is used if you ticked the **Scheduler uses internal calendar** check box on the **General** property page.

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



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



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

2.20 Printing the Diagram

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

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

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

The method **PrintDirect** of the object 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 **PageLayout** 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 see chapter 5.12 "Setting up Pages".

2.21 Exporting a Diagram

Your diagram can be exported as a graphics file:

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

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

2.22 Saving the Configuration

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

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

> How to save your current configuration:

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

> How to re-load a configuration:

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

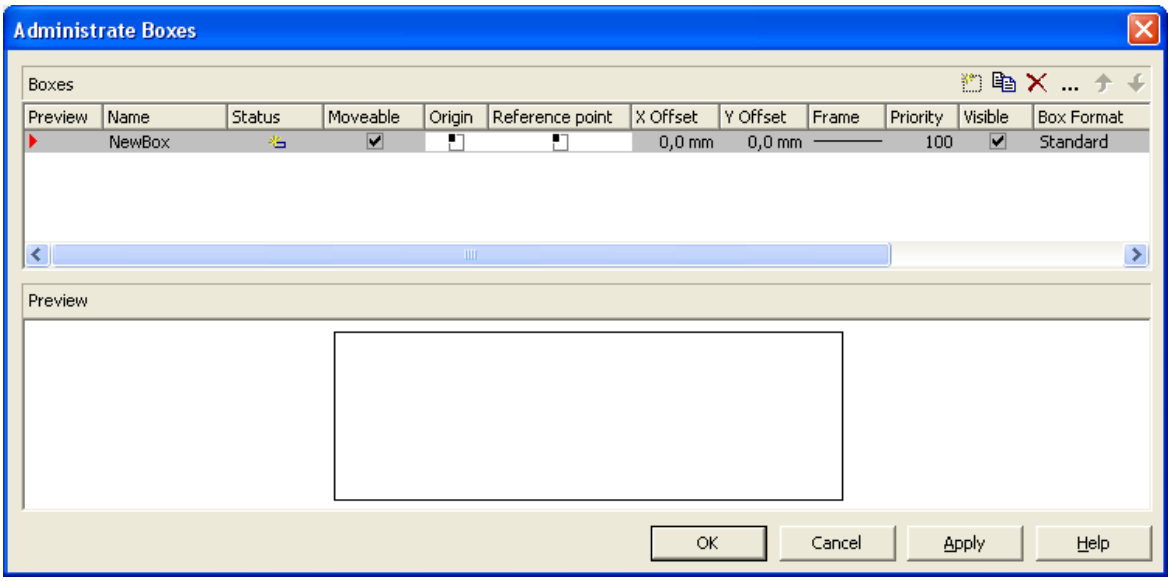
Note: The settings of the configuration file are loaded only once. VARCHART XNet will not load them for a second time from the same file. Instead, the settings will be loaded from the internal storage, which are the same as those in the configuration file.

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

3 Important Concepts

3.1 Boxes

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



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

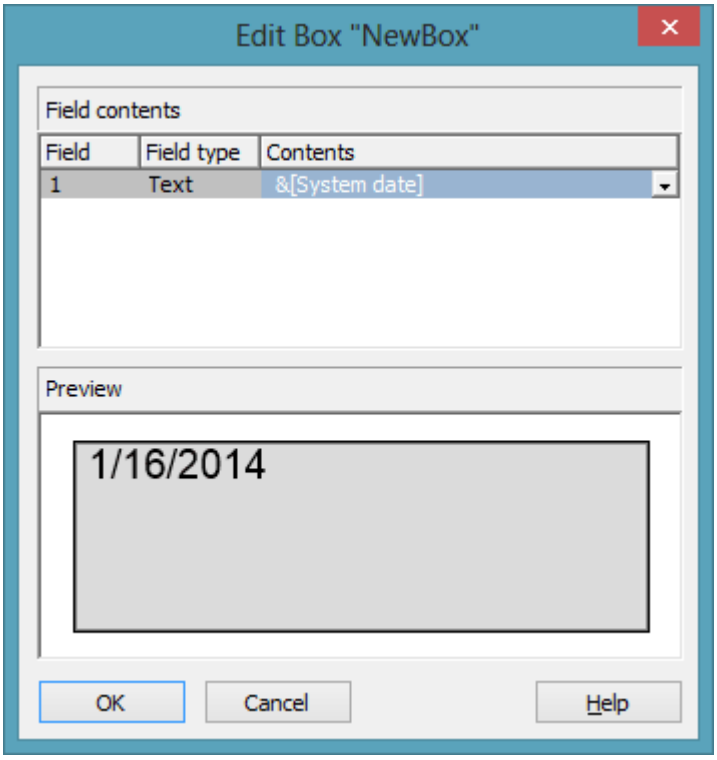
To a box you can set the below features:

- its name
- whether it can be moved in the diagram at run time
- its origin (the point to which the reference point refers in x and y direction)
- its reference point (the point to which the origin refers in x and y direction)
- 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 relation to other diagram objects (nodes, grids, etc.)

- whether the box should be visible
- the box format

> Editing boxes

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

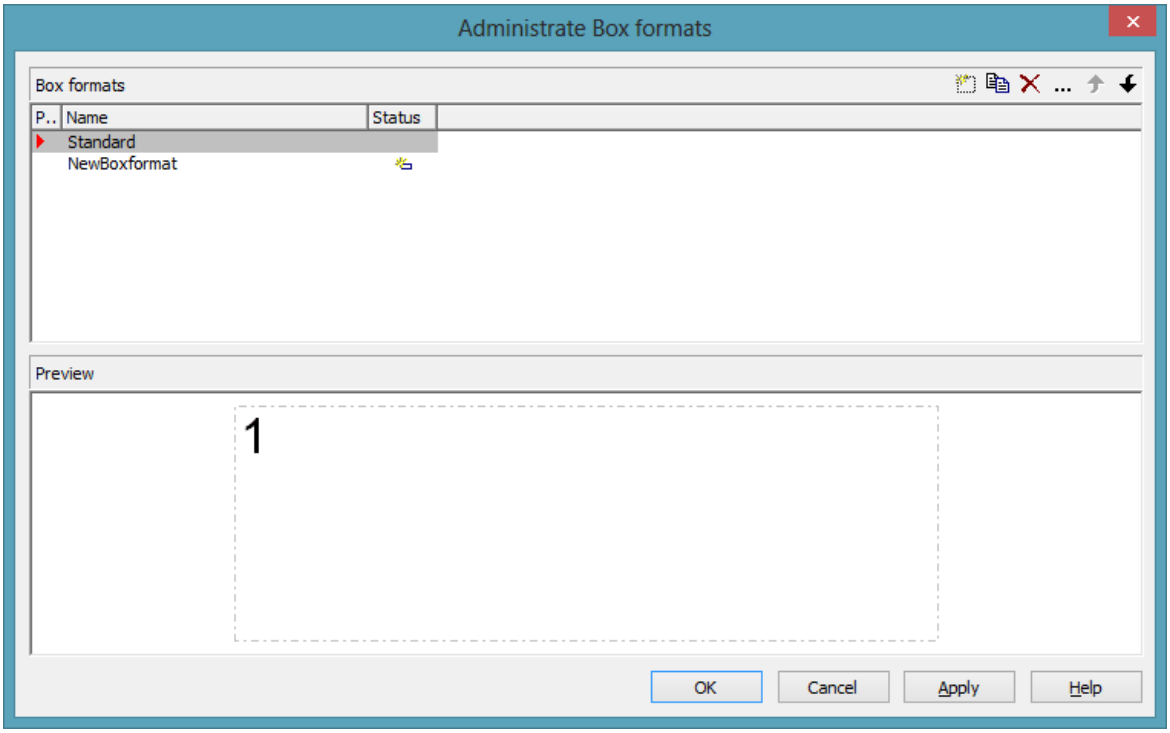


The **Field** column contains the numbers of the box fields. The number of fields depends on the selected box format (see further below).

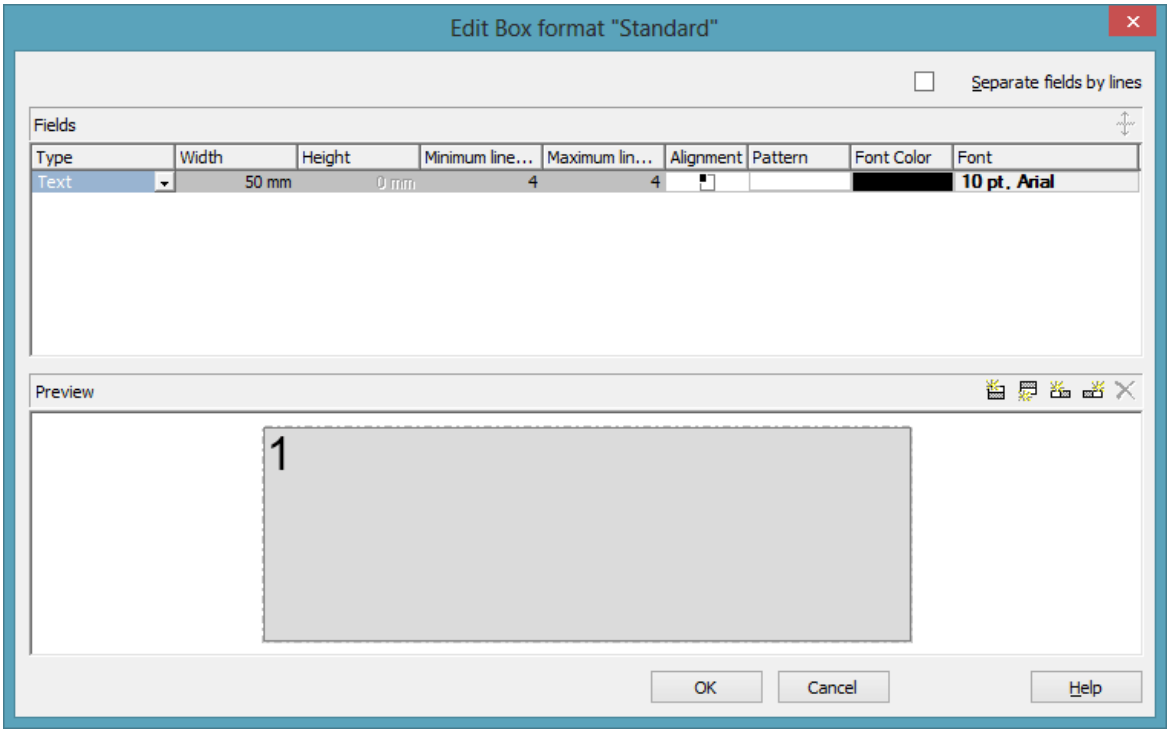
The **Field type** column displays the field types (text or graphics).

You can type the contents of the field or a graphics file name into the **Contents** column. If a text field contains more than one line, you can use "\n" to set line breaks (Example: "Line1\nLine2"). If you do not set line breaks, the lines will automatically be divided where blanks are.

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



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



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

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

- field type (text or graphics)

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

3.2 Data

Nodes and links can be generated interactively or by the API. If you generate them by the API, you can either load their data by a file using the method **Open** or you can create new ones by using the methods **InsertNodeRecord** and **InsertLinkRecord**. A record is transferred as a string or in a Variant array, with its data fields defined according to the settings of the data defintion. The data fields are separated by semicolons. If a semicolon is to be passed as data, it needs to be enclosed by quotation marks. In Visual Basic **+Chr$(34)+** is used instead of quotation marks.

Example Code

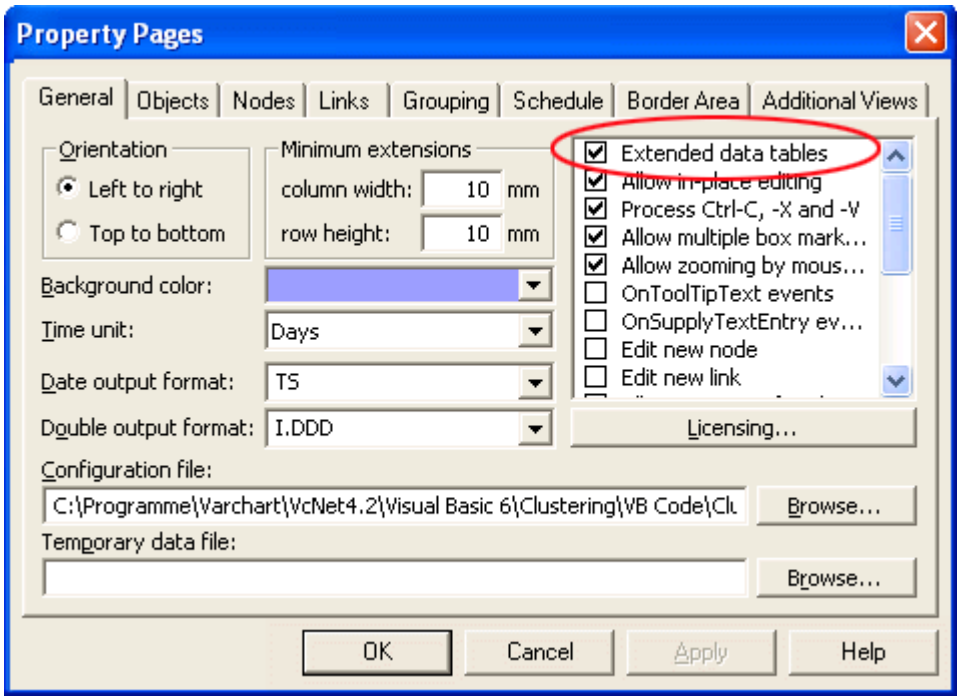
```
VcNet1.InsertNodeRecord "1;1.;;;" + Chr$(34) + " Company A; Department B
" + Chr$(34) + ""
```

The data is saved to a file via the **SaveAs** method. To use customized saving procedures in your application, you can retrieve the data of each every node by the **NodeCollection** and the data of each link between nodes by the **LinkCollection**.

3.3 Data Tables

As a data base for the graphical display of network diagrams 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. Similar to data bases, the data is structured in data sets that depend on each other, which avoids data redundancies.

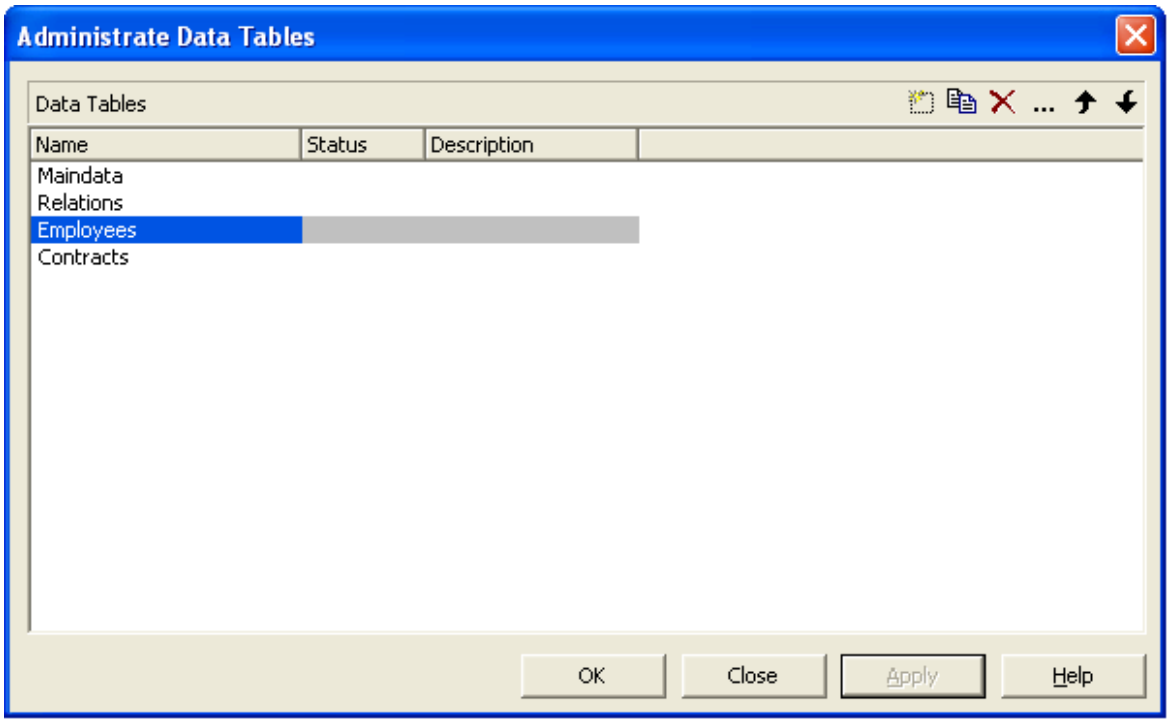
For reasons of compatibility to existing applications VARCHART XNet continues to operate in the previous mode by default. Only by activating the corresponding option at design time or at run time the extended data tables can be used. You can find the option **Extended data tables** on the property page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



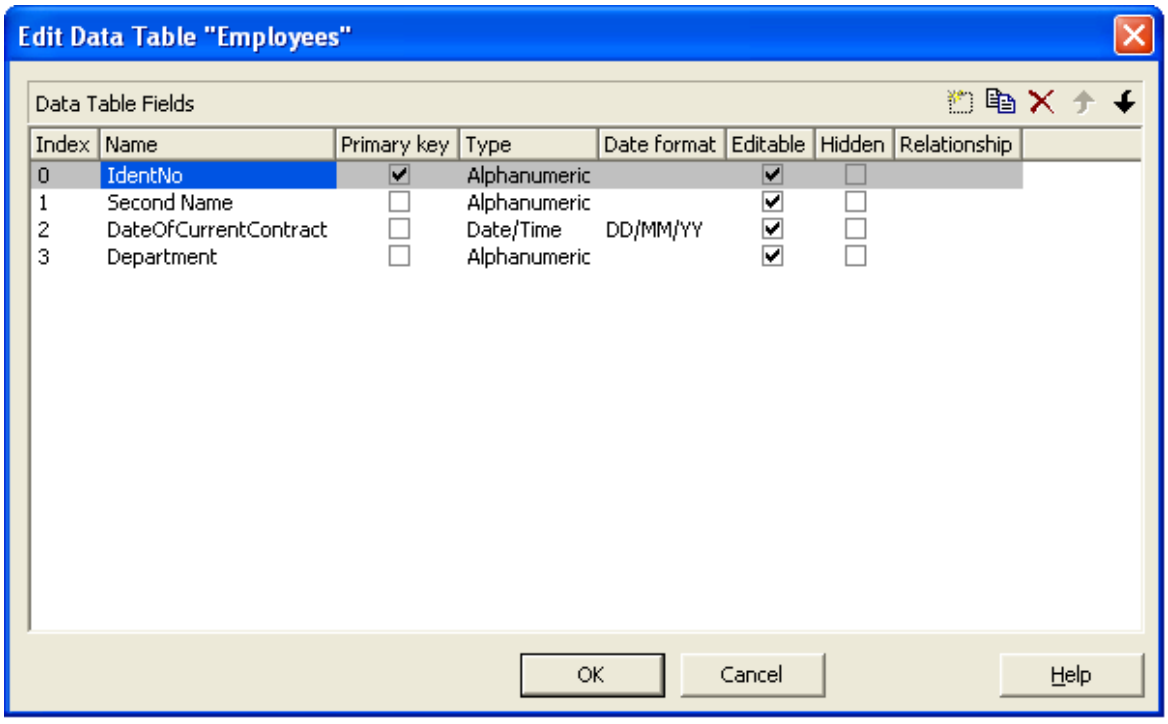
In the programming interface, the extended data tables are switched on at runtime by setting the VcNet property **ExtendedDataTables** to **True**.

> Handling Data Tables

By default, the data tables **Maindata** and **Relations** exist. On the property page **Objects** you can click on the button **Data Tables…** to get to the dialog **Administrate Data Tables**. Generating new data tables requires to have switched on the **extended data tables** mode before. The data tables **Employee** and **Contracts** in the picture below were created by clicking on  .



By the ... **Edit** button you can get to the **Edit Data Table** dialog. You can generate new fields by , delete existing fields by or copy fields by , as shown below.



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

you need to adapt the programming code that accesses the corresponding field.

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

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

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

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

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

In the sample below a relation is created between the tables **Employees** and **Contracts** by setting **Employees:IdentNo** in the column **Relationship**.

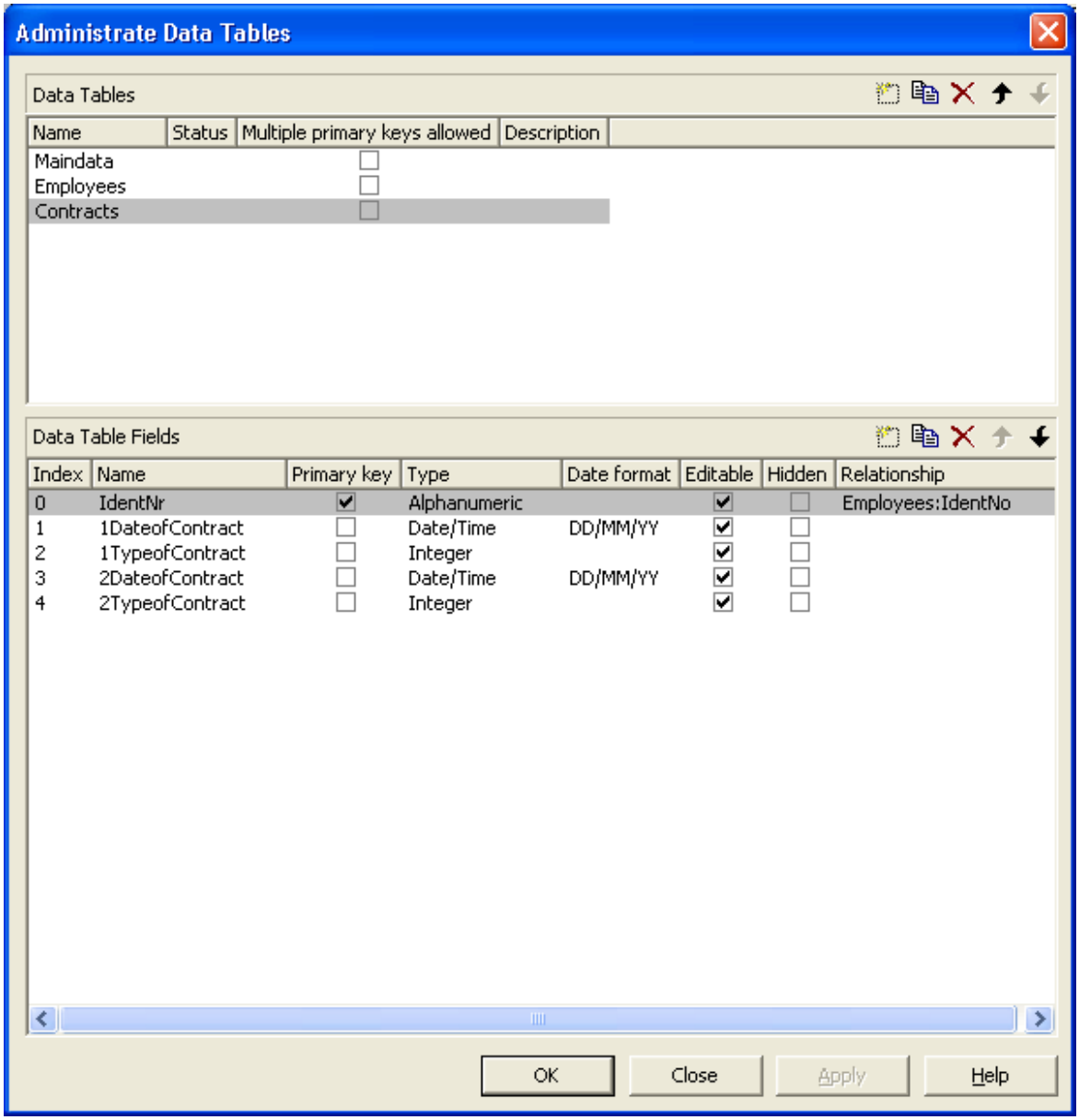


Table Employees:

IdentNo	SecondName	Department
1	Miller	Research and Development
2	Smith	Aministration

Table Contracts:

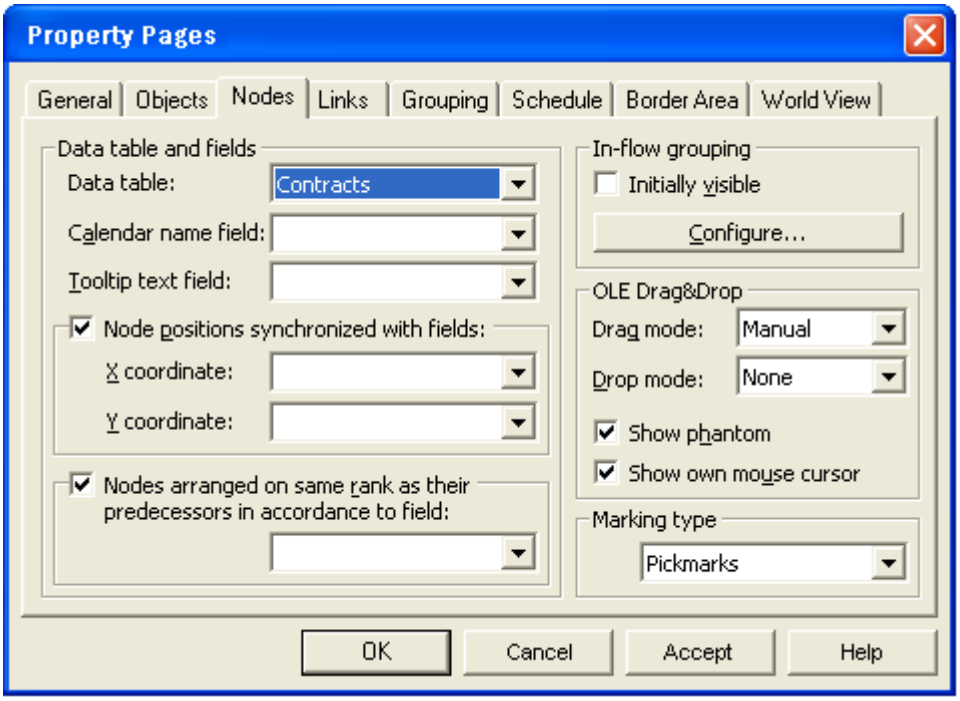
ID	ContractDate	SalaryLevel	StaffIdentNo
1	1996/08/01	3	1
1	2006/06/01	4	1

ID	ContractDate	SalaryLevel	StaffIdentNo
2	1994/03/01	1	2
2	1996/05/01	2	2
2	2001/10/01	3	2

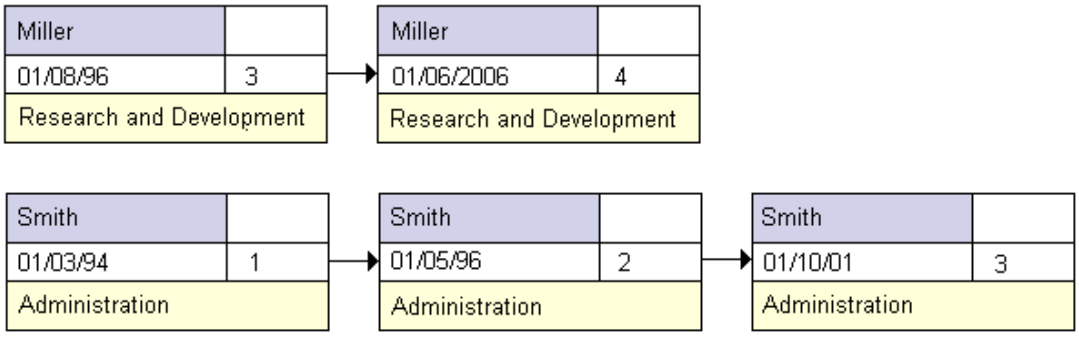
Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.DataTableByName("Employees")

dataTable.DataRecordCollection.Add ("1;Miller;Research and Development")
dataTable.DataRecordCollection.Add ("2;Smith;Administration")

Set dataTable = dataTableCltn.DataTableByName("Contracts")
dataTable.DataRecordCollection.Add ("1;1996/08/01;3;1")
dataTable.DataRecordCollection.Add ("1;2006/06/01;4;1")
dataTable.DataRecordCollection.Add ("1;1994/03/01;1;2")
dataTable.DataRecordCollection.Add ("1;1996/05/01;2;2")
dataTable.DataRecordCollection.Add ("1;2001/10/01;3;2")

VcNet1.EndLoading
```

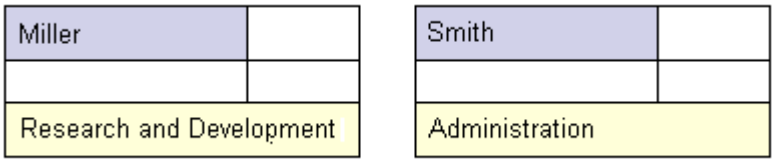
Depending on which table is displayed on the property page **Node** in the **Data table** section, the graphical display of the nodes may derive from various bases. When creating nodes interactively, the base is the table to which new data records are added automatically. The corresponding rows displayed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



This is the result in the network chart if the table **Contracts** was taken as base. The names originate from the main table **Employee**.



If the table **Employees** instead of **Contracts** is used, the visible data in XNet will consist of two entries only.



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	VcDataTable
VcDefinitionTable	VcDataTableFieldCollection
VcDefinitionField	VcDataTableField
	VcDataRecord

> Creating and modifying data records

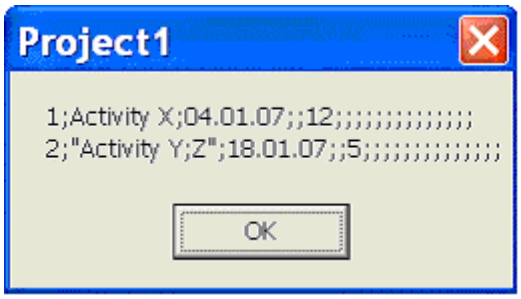
After having defined the data table fields, you can add data records to a table by the API. There are two ways of adding data to your records. We recommend the common practice of defining an array of the type variant with the number of its elements corresponding to the number of the data table fields.

Example Code

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

```
'...

Dim dataRec1 As VcDataRecord
Dim dataRec2 As VcDataRecord

Dim content As String

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

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

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

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

VcNet1.EndLoading

'...

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

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

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

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

'...

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

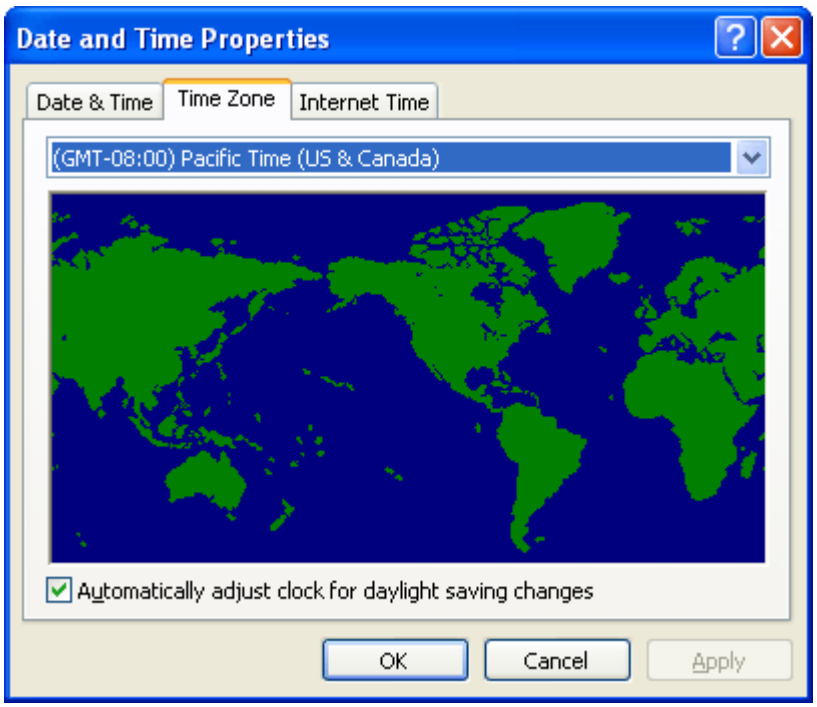
This is the output:


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

3.4 Dates and Daylight Saving Time

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

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



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

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

3.5 Events

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

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

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

> Return Status

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

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

3.6 Filters

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

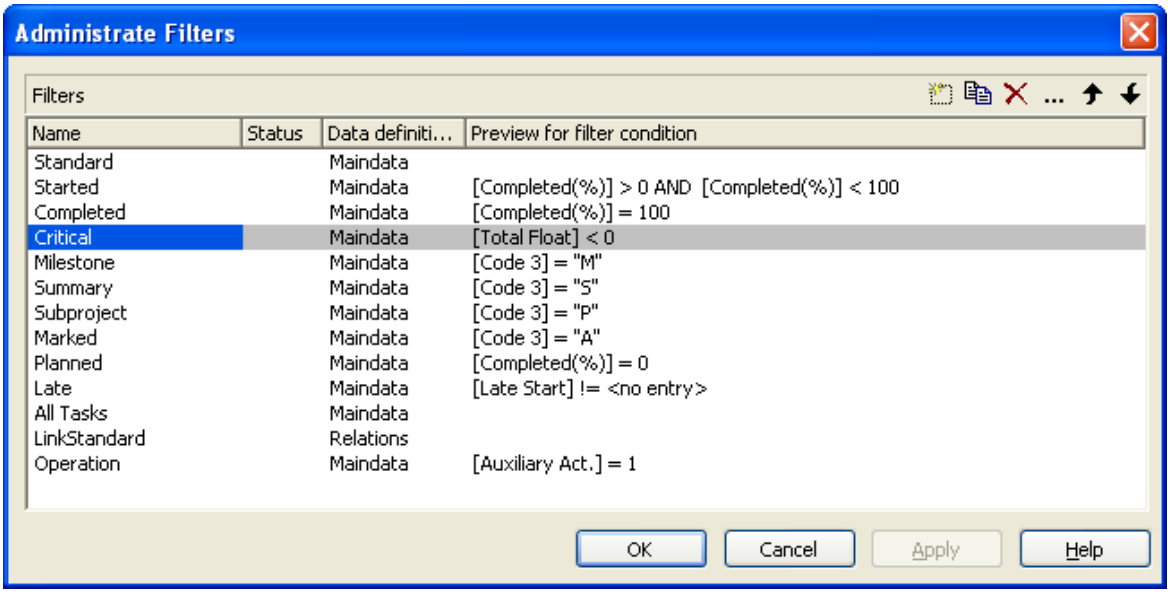
When you apply a filter, the data of the activity or link is compared with the criteria of the filter. Activities that fulfill the filter criteria will be selected.

For example, you can define a filter that specifies "All nodes starting after January 2009".

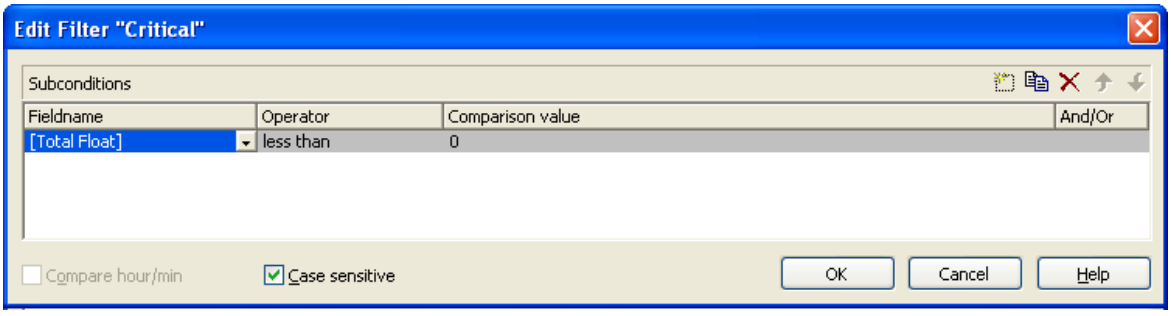
Filters can only be handled in design mode.

You can get to the **Administrate Filters** dialog via the **Objects** property page, and in terms of filters that apply to links also via the **Links** property page.

By using the **Administer Filters** dialog box you can rename, create, copy, delete or edit filters.



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



3.7 Graphics Formats

VARCHART supports the below graphics formats, which is important to exporting charts, affecting mainly the calls **VcNet1.ShowGraphicsExport-Dialog** 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 beween the chosen (supported) graphics format and the graphic's display quality in the control (after the import) or in an external viewer program (after the export). Please find below a description of the advantages and restrictions of the individual graphics formats. Basically there are two different types:

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

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

> WMF (Windows Metafile Format)

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

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

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

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

For the description of the format please see:

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

On the limitations of the format see:

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

> EMF (Enhanced Metafile Format)

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

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

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

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

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

For the format description please see:

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

> EMF+ (Enhanced Metafile Format Plus)

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

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

For the format description please see:

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

> GIF (Graphics Interchange Format)

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

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

> JPEG (Joint Photographic Experts Group)

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

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

> BMP (Windows Bitmap)

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

> TIFF (Tagged Image File Format)

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

> PNG (Portable Network Graphics)

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

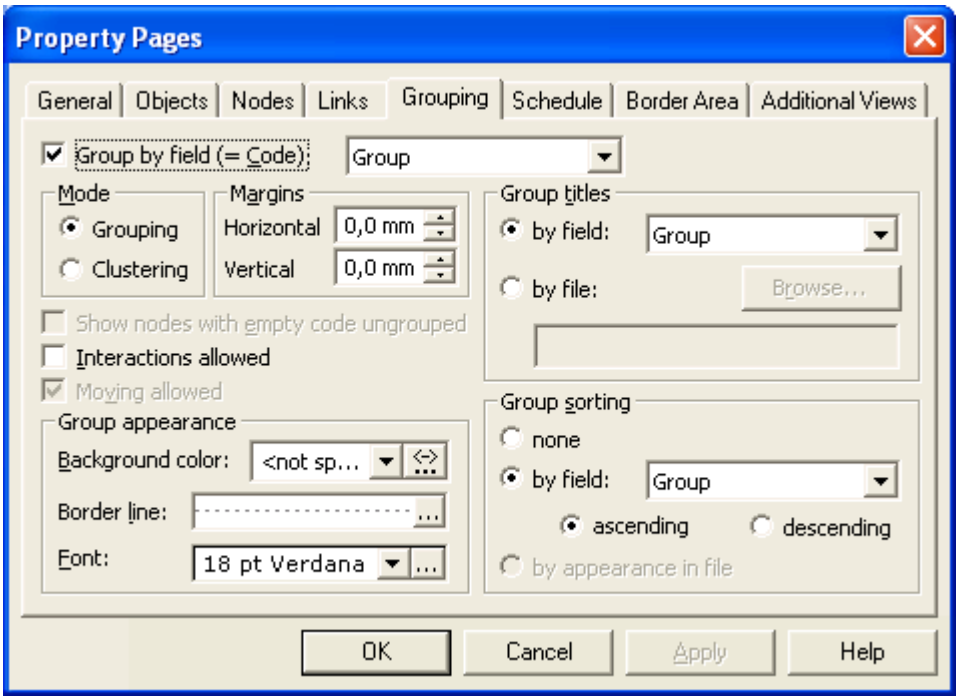
From version 4.2 onward the free library **libpng** is used which is freely available, in order to set a resolution and thus store bitmaps of any size. It has to be taken into account though that very large PNG files may cause problems when loaded, since usually PNG files get completely unpacked in the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

For the format description please see:

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

3.8 Grouping

Many applications require to group nodes and to highlight the groups. For example, nodes are 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.



The nodes are grouped by the field that you select 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

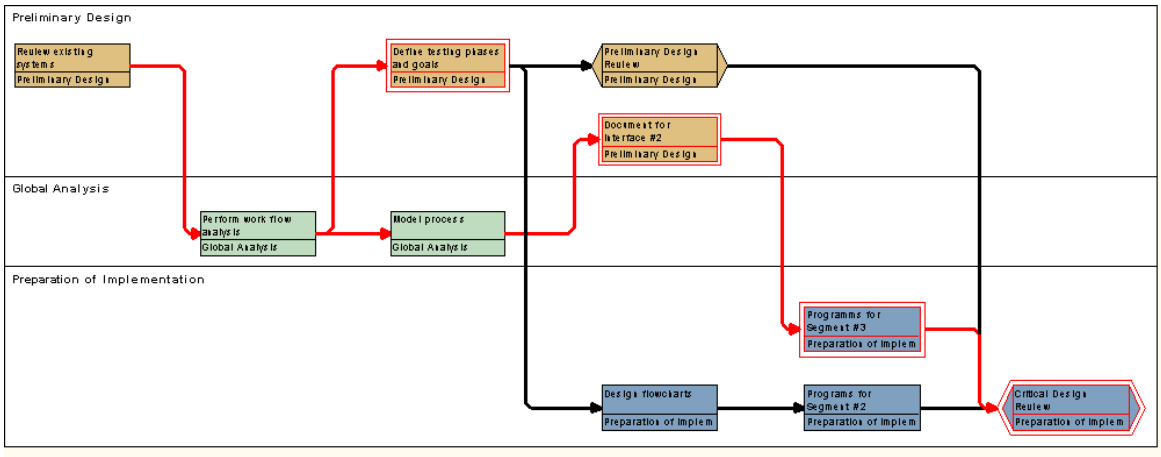
blank = separator

"Group A" = full text

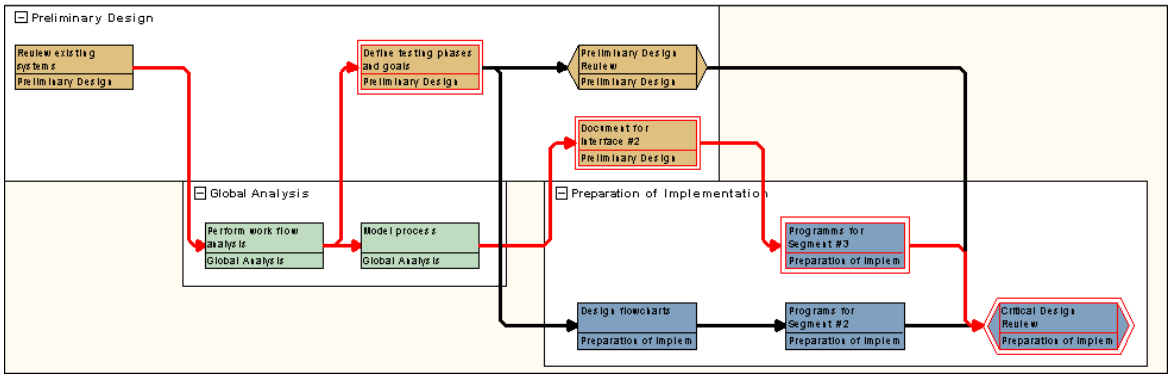
> Modes: Grouping and Clustering

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.GroupModus**).

> 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:

- **OnGroupCreate**
- **OnGroupDelete**
- **OnGroupLClick**
- **OnGroupLDblClick**
- **OnGroupModify**
- **OnGroupModifyComplete**
- **OnGroupRClick**

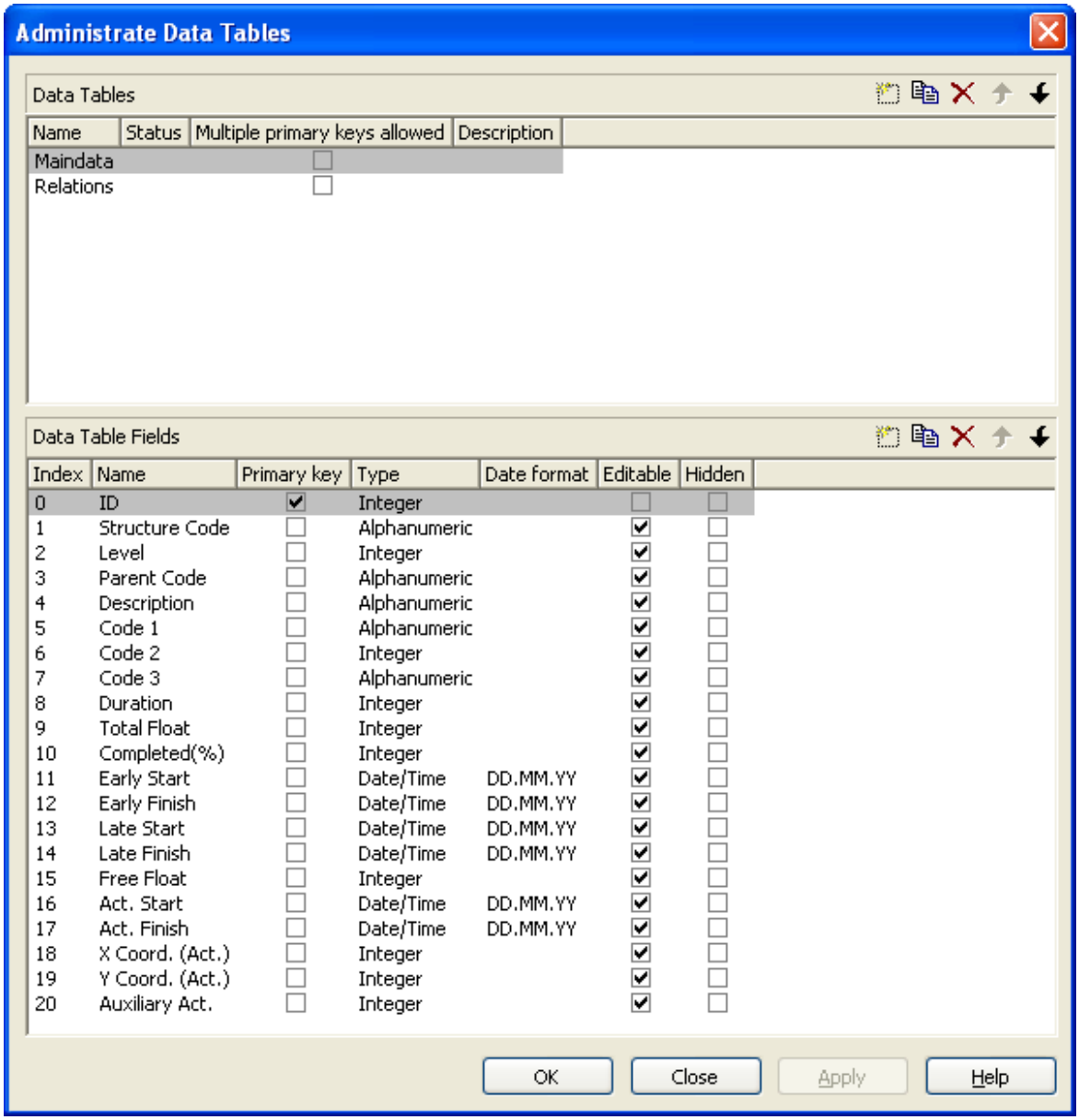
3.9 Identification

On the **Objects** property page, please select the **Data Table** property page. Please select a data table from the top list, for example the **Maindata** table, which describes nodes.

In the bottom list you can select a data field to be used for the identification of nodes. To most applications it is useful to also asign it the state of a primary key.

The identification for example is used in data fields of links to identify predecessor or successor nodes. As another example, the identification may serve to refer nodes and links correctly to a data base that works in the background.

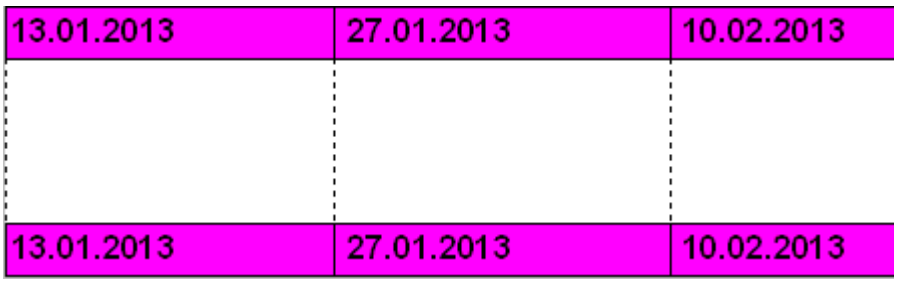
Identifications are also used to access nodes and links by the methods **GetNodeByID** and **GetLinkByID**, respectively.


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 ID Integer
1 Structure Code Alphanumeric
2 Level Integer
3 Parent Code Alphanumeric
4 Description Alphanumeric
5 Code 1 Alphanumeric
6 Code 2 Integer
7 Code 3 Alphanumeric
8 Duration Integer
9 Total Float Integer
10 Completed(%) Integer
11 Early Start Date/Time DD.MM.YY
12 Early Finish Date/Time DD.MM.YY
13 Late Start Date/Time DD.MM.YY
14 Late Finish Date/Time DD.MM.YY
15 Free Float Integer
16 Act. Start Date/Time DD.MM.YY
17 Act. Finish Date/Time DD.MM.YY
18 X Coord. (Act.) Integer
19 Y Coord. (Act.) Integer
20 Auxiliary Act. Integer
OK
Close
Apply
Help

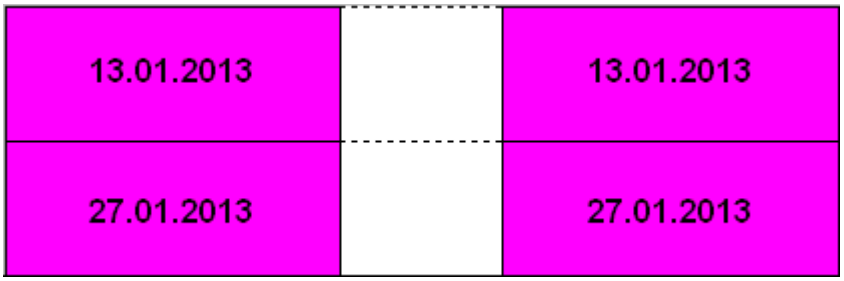
3.10 In-Flow Grouping

In-flow grouping arranges nodes according to a criterion "in flow" with the orientation of the network. The in-flow direction corresponds to the settings of the orientation,so if it was set to **left-to-right**, the nodes will be sorted horizontally; if it was set to **top-to-bottom**, the nodes will show a vertical arrangement.

The criterion can be a temporal one (date field) or a code (data field). The size of time intervalls you can define:

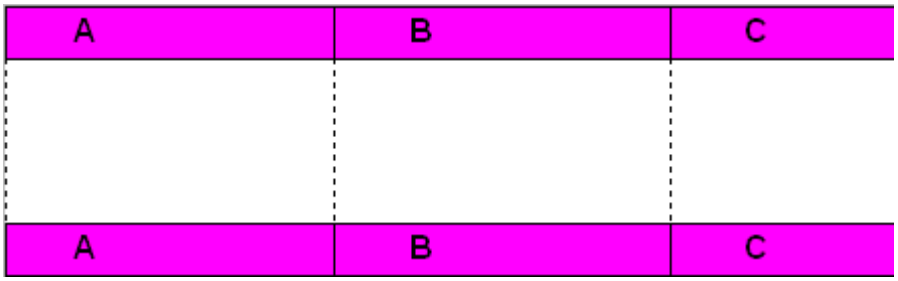


Example of a time-oriented network in a left-to-right-orientation

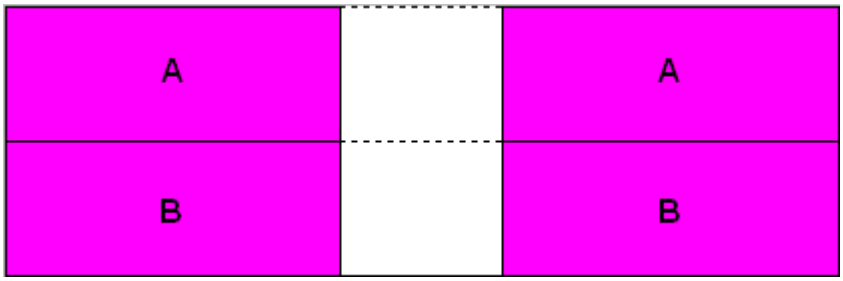


Example of a time-oriented network in a top-to-bottom orientation

This picture shows grouping by code:

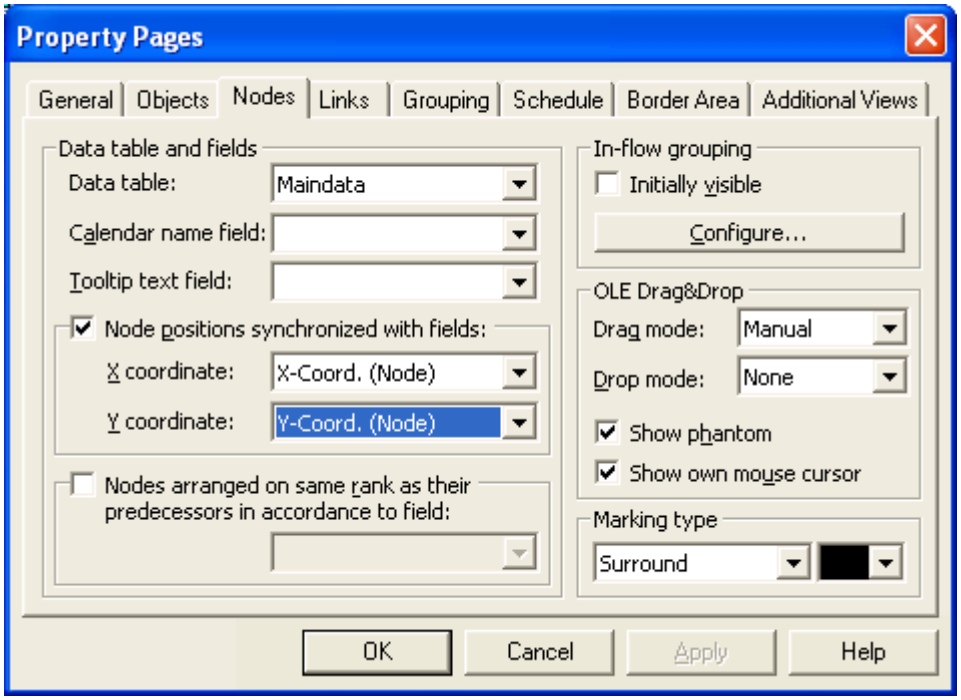


Example of in-flow grouping by code in a left-to-right-orientation

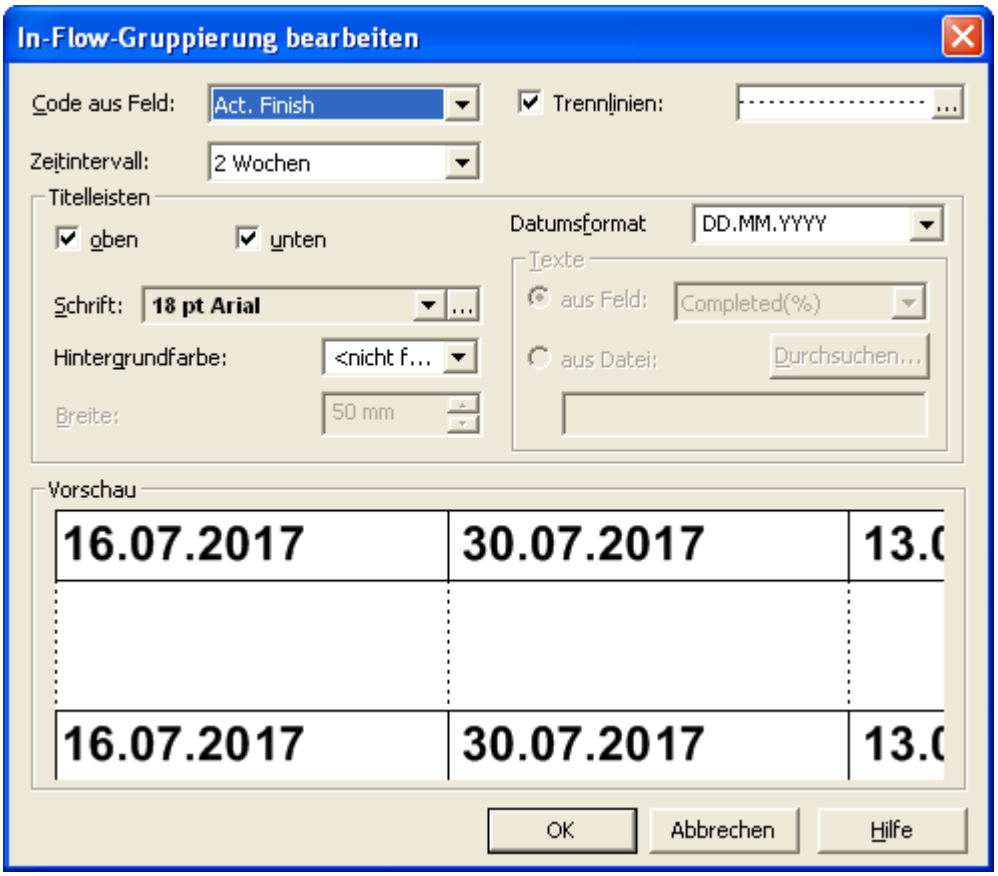


Example of in-flow grouping by code in a top-to-bottom orientation

In-Flow-Grouping can be activated on the **Nodes** property page in the **In-flow grouping** area by the **Initially visible** check box:



The criteria and the layout you can set by pressing the **Configure** button. The **Edit In-Flow Grouping** dialog will pop up:

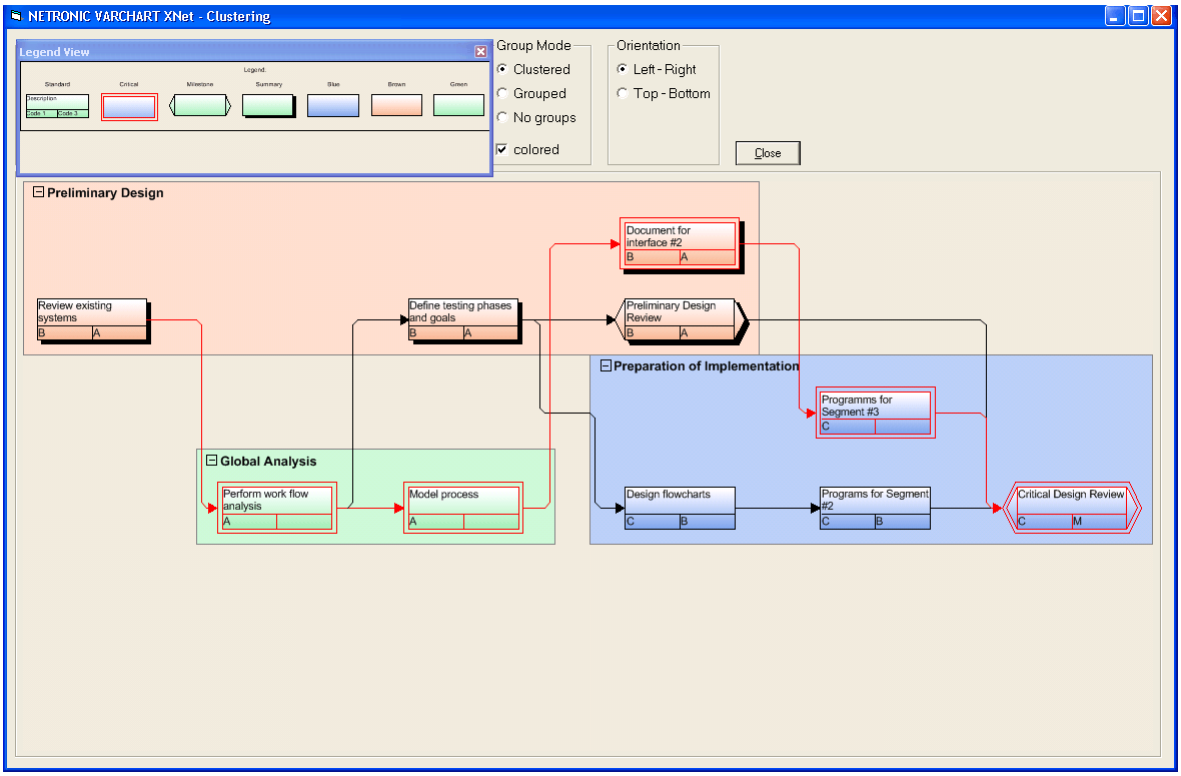


If you choose an in-flow grouping with a left-to-right orientation, you can display a title bar at the top and/or bottom of the diagram area. For a top-to-bottom orientation you can display title bars on the left and/or right of the diagram.

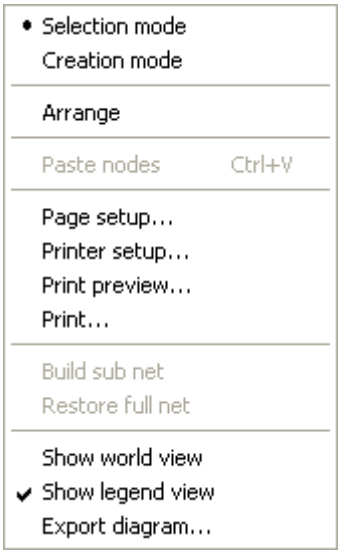
Beside, you can set whether vertical or horizontal separation lines should be used to mark group borders. You can specify colors and annotations to the title bars as well as the design of the separation lines.

3.11 Legend View

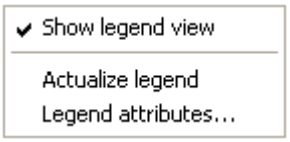
The legend view is an additional window that lets you display a legend on the screen. The layout of the legend can be specified with the legend attributes of **VcBorderBox** or in the dialog **Legend attributes** which can be reached from the **Border area** property page.



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



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



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

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

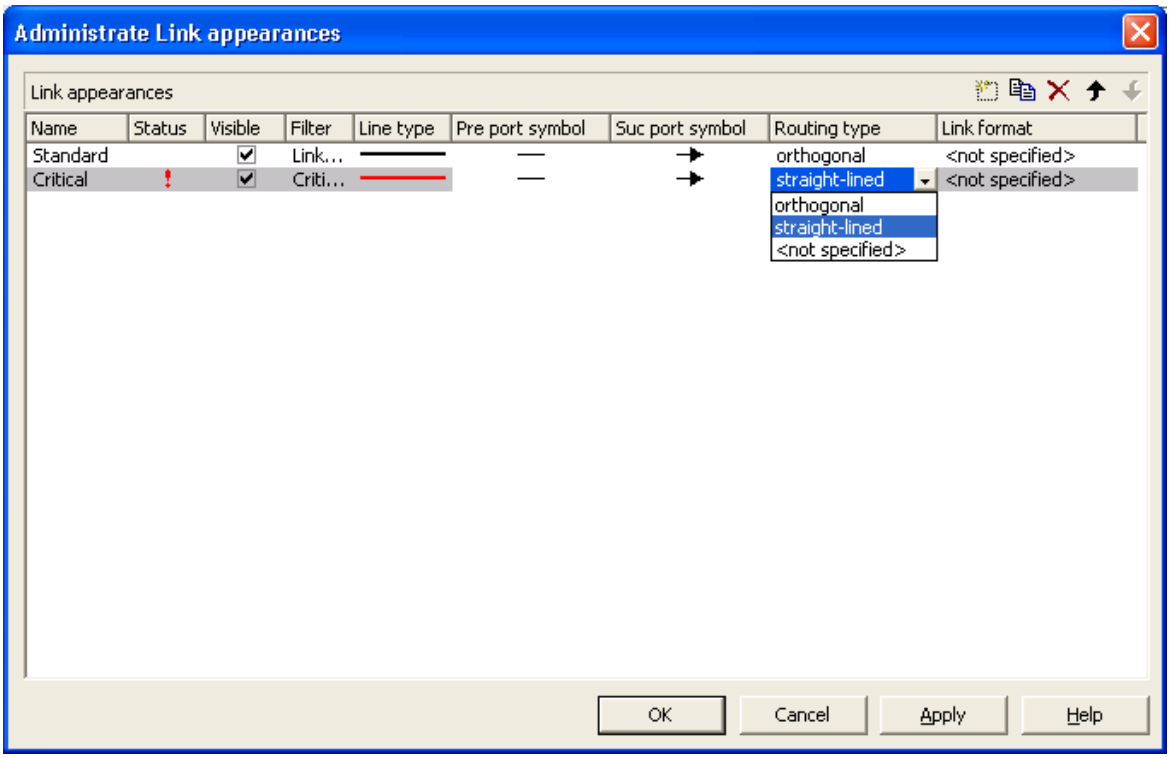
On the **Additional Views** property page you can set the properties of the Legend View. For details please see **The Additional Views Property Page** in the chapter **Property Pages and Dialog Boxes** .

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

3.12 Link Appearance

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

> Defining a Link Appearance



> Deleting Link Appearances

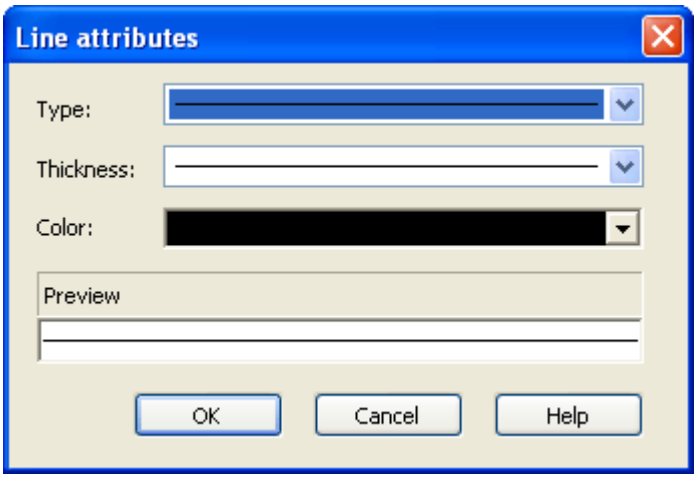
You can delete a link appearance from the **Appearances** list by the **Del** key.

> 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

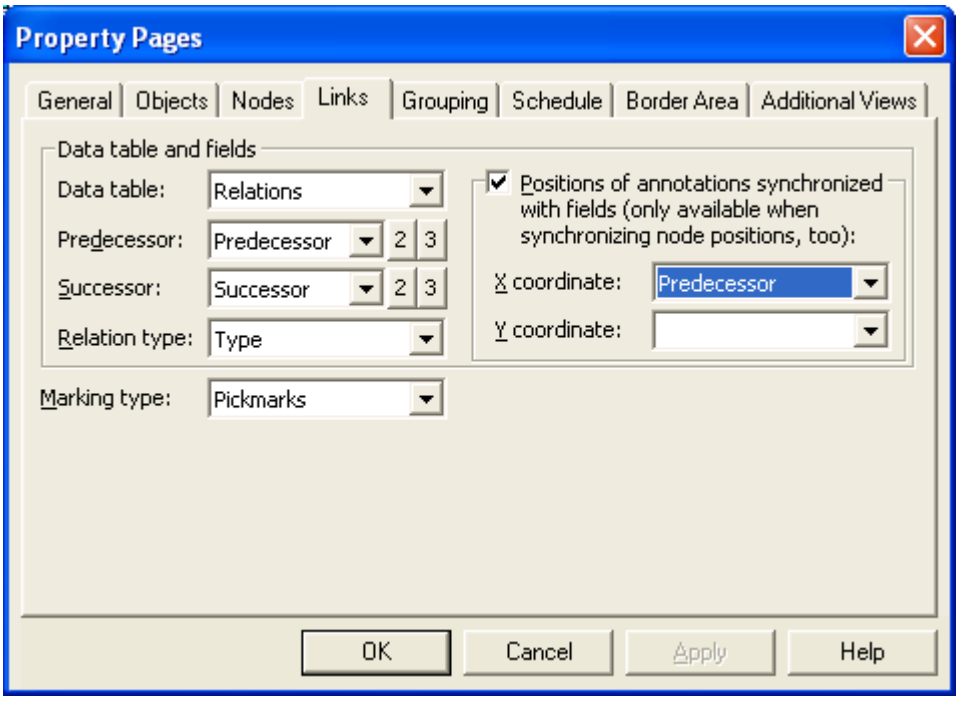
If you click on the entry of the field **Line type**, an **Edit** button will appear by which you can get to the **Line Attributes** dialog. There you can set the color, type and thickness of the line.



3.13 Links

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

> Data Fields for the Links

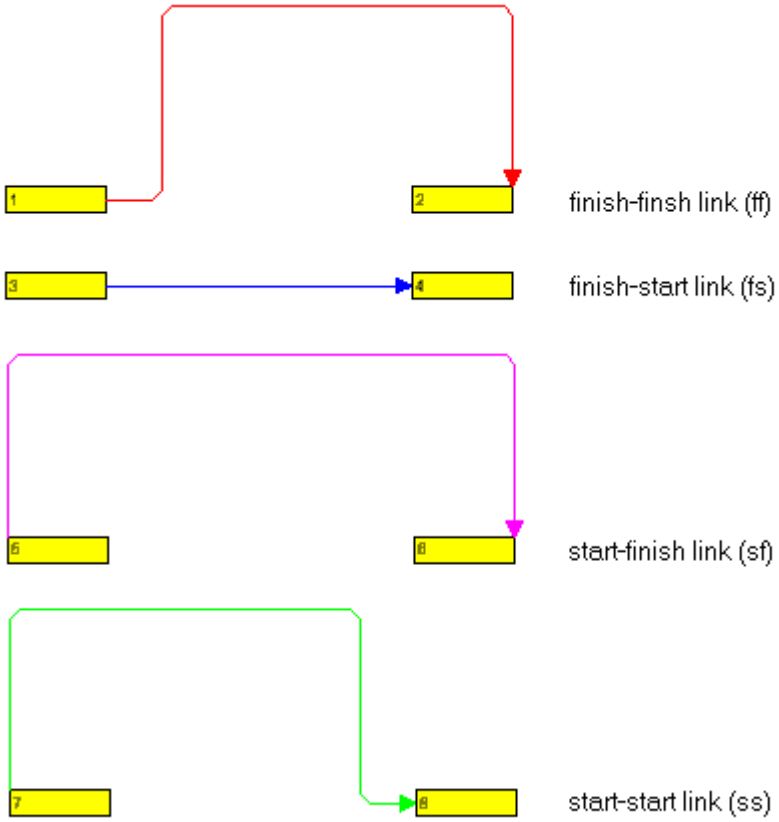


On the **Link** property page you can specify the data fields to which the identifications of the predecessor and successor nodes and the relation types are to be stored.

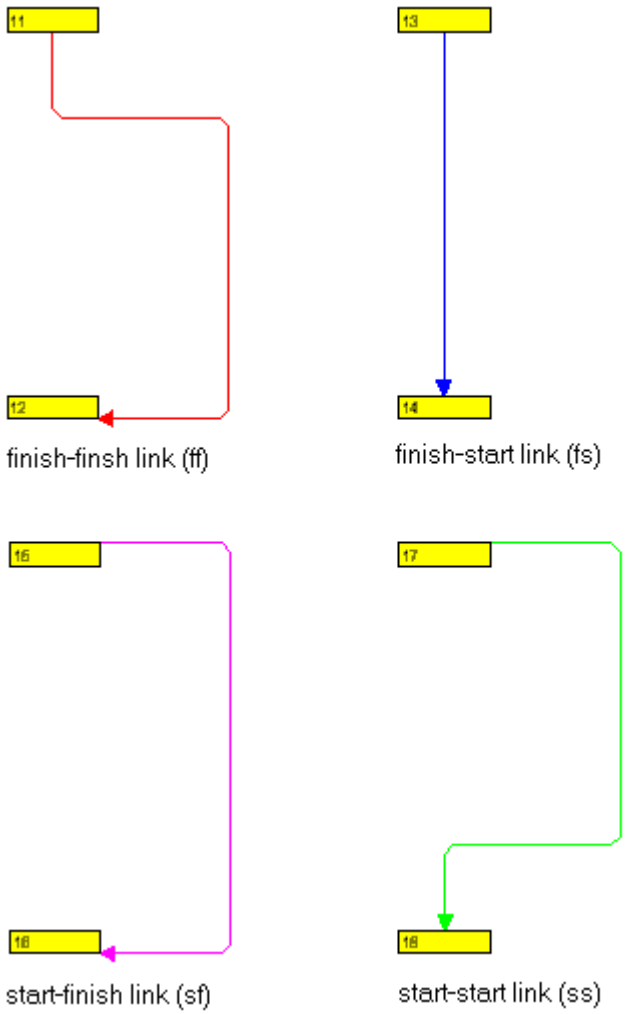
> Types of Links

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



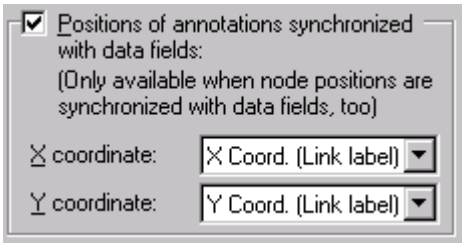
Top-to-bottom orientation

> 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").

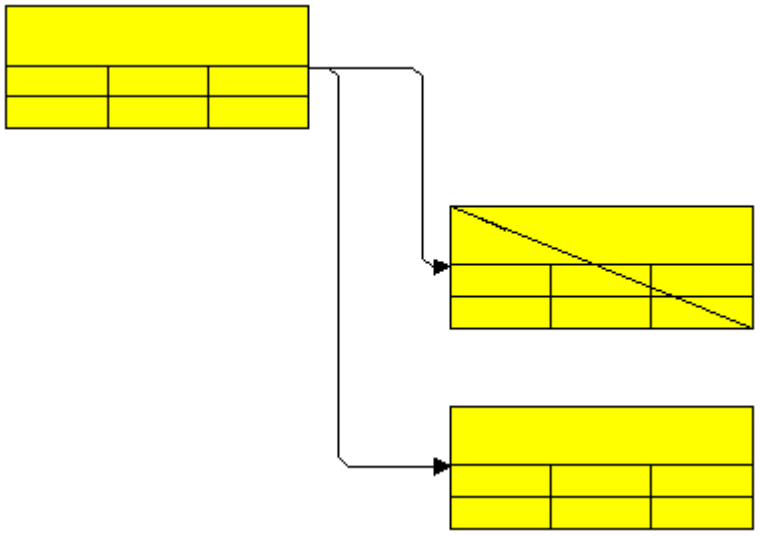


The values of the above data fields you can retrieve and, if necessary, modify via the dialog **Edit Link**. You can find an example of positions of node and link annotations in the "Nodes" paragraph of this chapter.

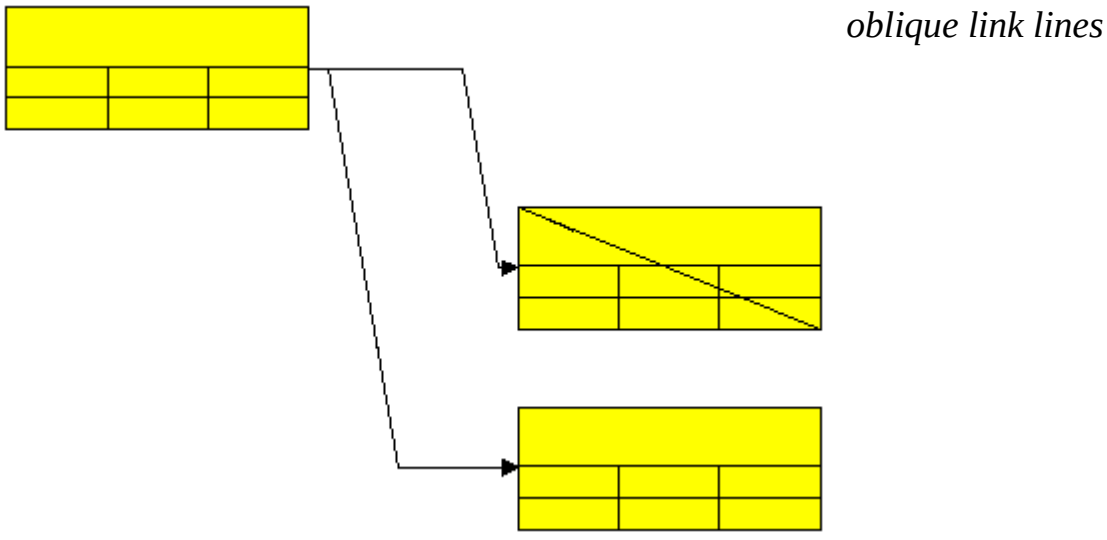
> Orthogonal/Oblique Link Lines

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 **OnLinkCreate**.

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

> Events

You can react to the events:

- **OnLinkCreate**
- **OnLinkCreateComplete**
- **OnLinkDelete**
- **OnLinkDeleteComplete**
- **OnLinkLClickCltn**

- **OnLinkLDblClickCltn**
- **OnLinkModify**
- **OnLinkModifyComplete**
- **OnLinkModifyEx**
- **OnLinkRClickCltn**

3.14 Localization of Text Output

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

To do so, set the VcNet property **EnableSupplyTextEntryEvent** to **True** to activate the event.

Example Code

```
VcNet1.EnableSupplyTextEntryEvent = True
```

Alternatively, you can tick the **OnSupplyTextEntry events** check box on the **General** porperty page. Then catch the **OnSupplyTextEntry** event and pass the text to be displayed.

Example Code

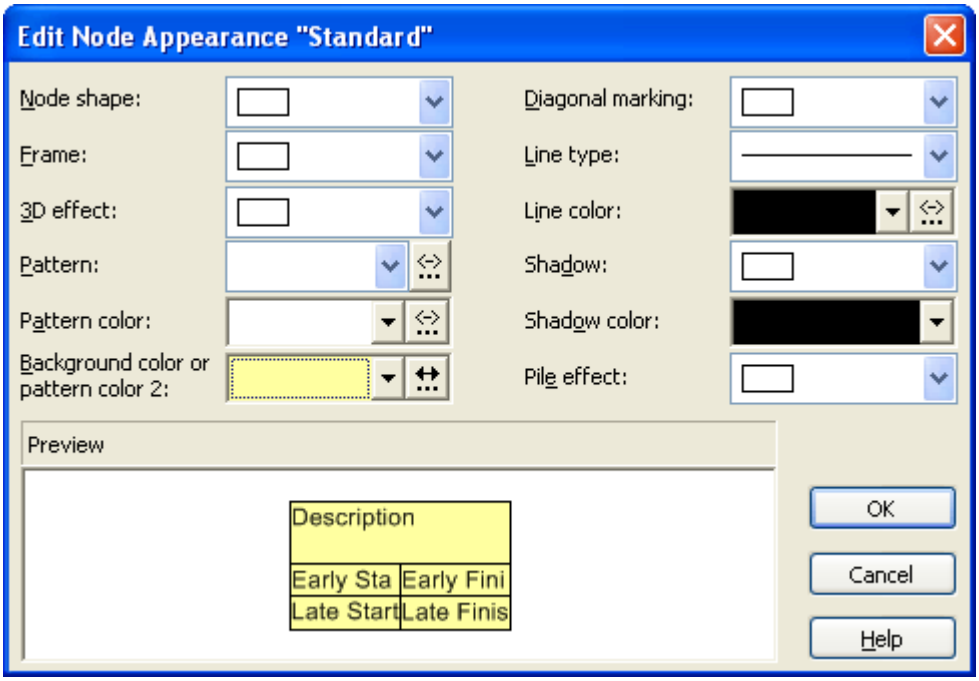
```
Private Sub VcNet1_OnSupplyTextEntry(ByVal controlIndex As _
                                VcNetLib.TextEntryIndexEnum, _
                                TextEntry As String, _
                                returnStatus As Variant)
    Select Case controlIndex
        Case vcTXERibCW
            TextEntry = "Semaine"
        Case vcTXERibDay0
            TextEntry = "Lundi"
        Case vcTXERibMon8
            TextEntry = "Septembre"
        Case vcTXERibQuar2
            TextEntry = "2. Quart."
        End Select
End Sub
```

3.15 Maps

Maps serve to set properties in dependence on the contents of data fields. By using maps, you can avoid having to define many similar filters and layers. Also node appearances and node formats can be assigned to the nodes in dependence on their data.

> Node Appearance in Dependence on Node Data

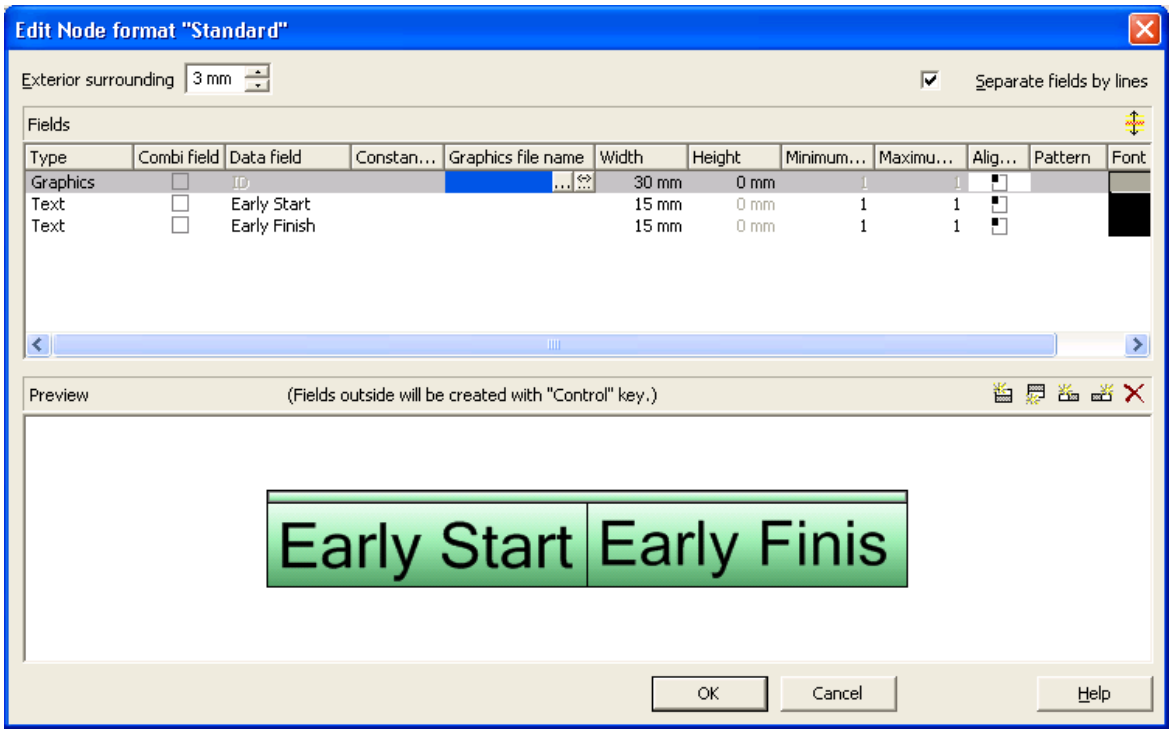
For each node appearance the background color and the line color can be set by a map. In the **Edit Node Appearance** dialog box, please click on the second button besides the **Background color** field or **Line color** field ( ).



You will get to 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.

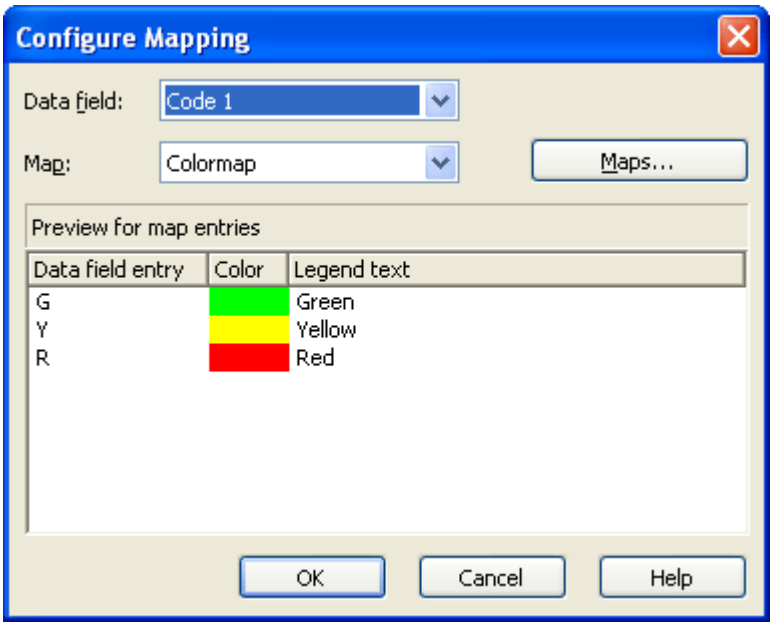


To allocate data field entries of the type graphics to graphics files, in the **Graphics File** field please click on the right-hand button. The **Configure Mapping** dialog box will open.

After finishing the configuering, a symbol ( ) will be displayed besides the symbol file name as soon as you leave the **Graphics File** field.

> Configure Mapping

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



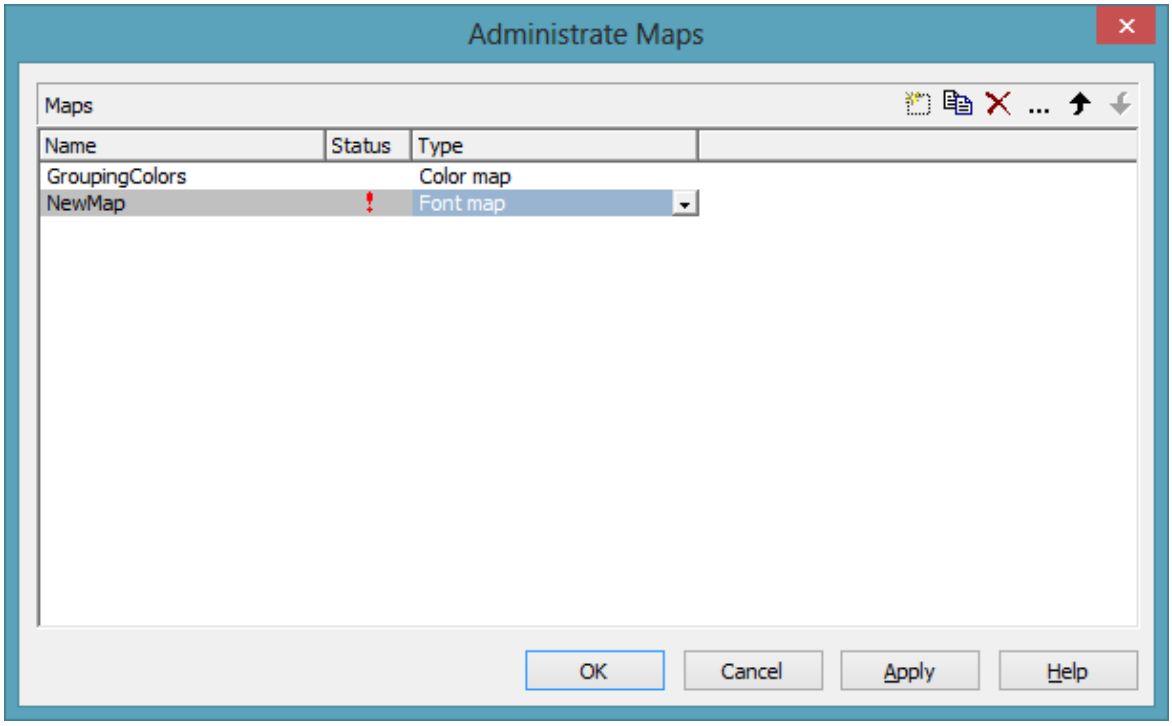
From the first combobox, select the **Data field** which a map is to be assigned. From the second combobox, select the **Map** that assigns a graphics file or a color respectively and a legend text to the data field entries.

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

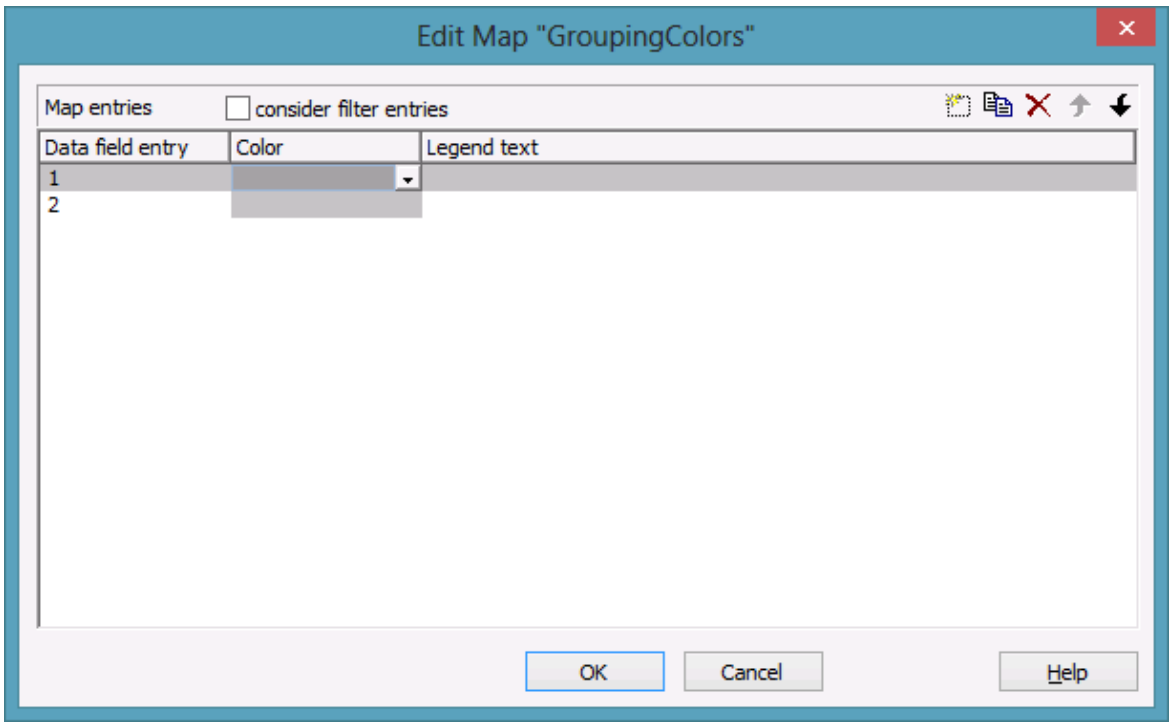
In the **Administrate Maps** dialog which can be invoked by clicking the **Maps** button or by clicking the **Maps** button of the **Objects** property page, you can modify the name and the type of a map by directly entering the corresponding data fields. By clicking the corresponding buttons on the 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.



> Editing Maps

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



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

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

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

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

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

> Adjusting the Map during Runtime

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

3.16 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 **Allow creation of nodes and links** has been chosen, the user will be able to create new nodes interactively by a mouse click.

If in addition the check box **Edit new node** 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 via the API by the **InsertNodeRecord** method. Any interactively created node will invoke the event **OnNode-Create**.

> Marking Nodes

On the **Nodes** property page you can set a pattern 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 **OnNodesMarkEx**. The end of an marking/demarking operation will invoke the event **OnNodesMarkComplete**.

> 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 **OnNodeDelete**.

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

> Events

You can react to the events:

- **OnNodeCreate**
- **OnNodeCreateCompleteEx**
- **OnNodeDelete**
- **OnNodeLClick**
- **OnNodeLDblClick**
- **OnNodeModify**
- **OnNodeModifyComplete**
- **OnNodeModifyEx**
- **OnNodeRClick**
- **OnNodesMarkComplete**
- **OnNodesMarkEx**

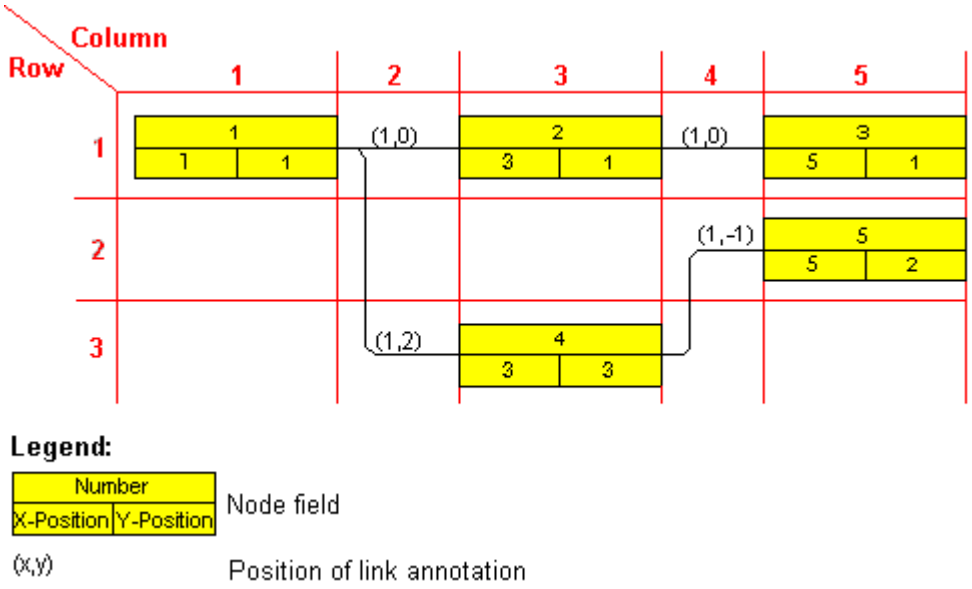
> Positions of Nodes

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.

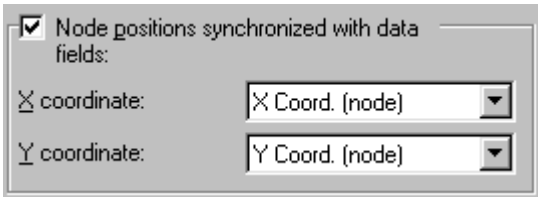


> Saving and Loading Node Positions

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"



> The Rank of a Node

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

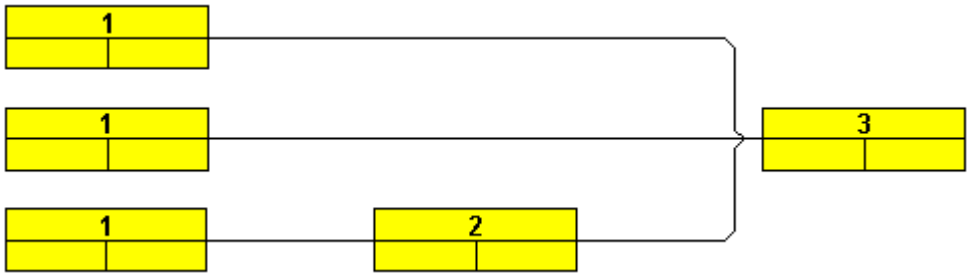
The rank of an unpreceded node equals 1. The rank of a node that has predecessors equals 1 plus the rank number of the predecessor of the top rank.

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).



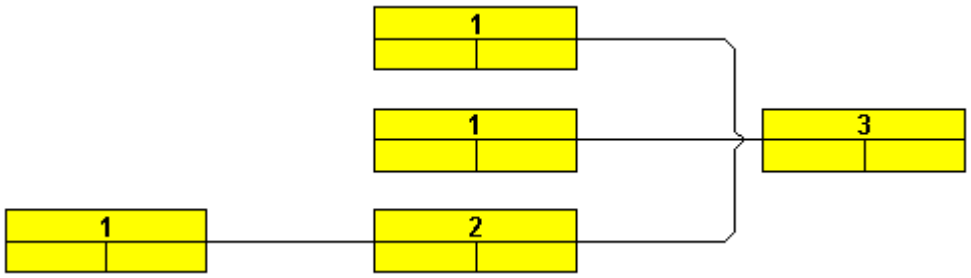
Ranks of nodes oriented from left to right

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.

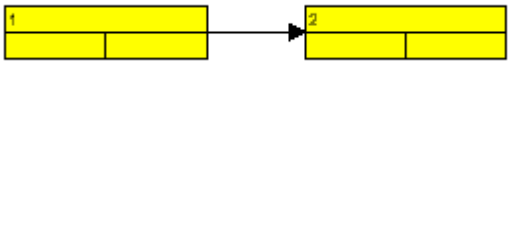
By the property **ShortenedLinks** or by ticking the **Shorten links on arrange** check box on the **General** property page the positions will be positioned far right or far below to reduce the length of links to a minimum.



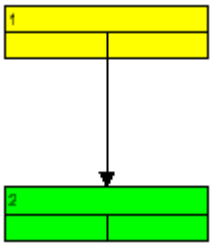
> Auxiliary Nodes

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.

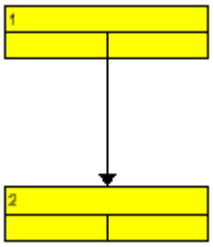


Nodes of rank 1 resp. rank 2

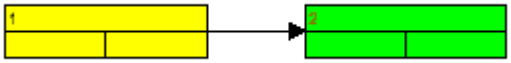


*The rank number of the second node was diminished by 1. Then the command **Arrange** was invoked.*

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.

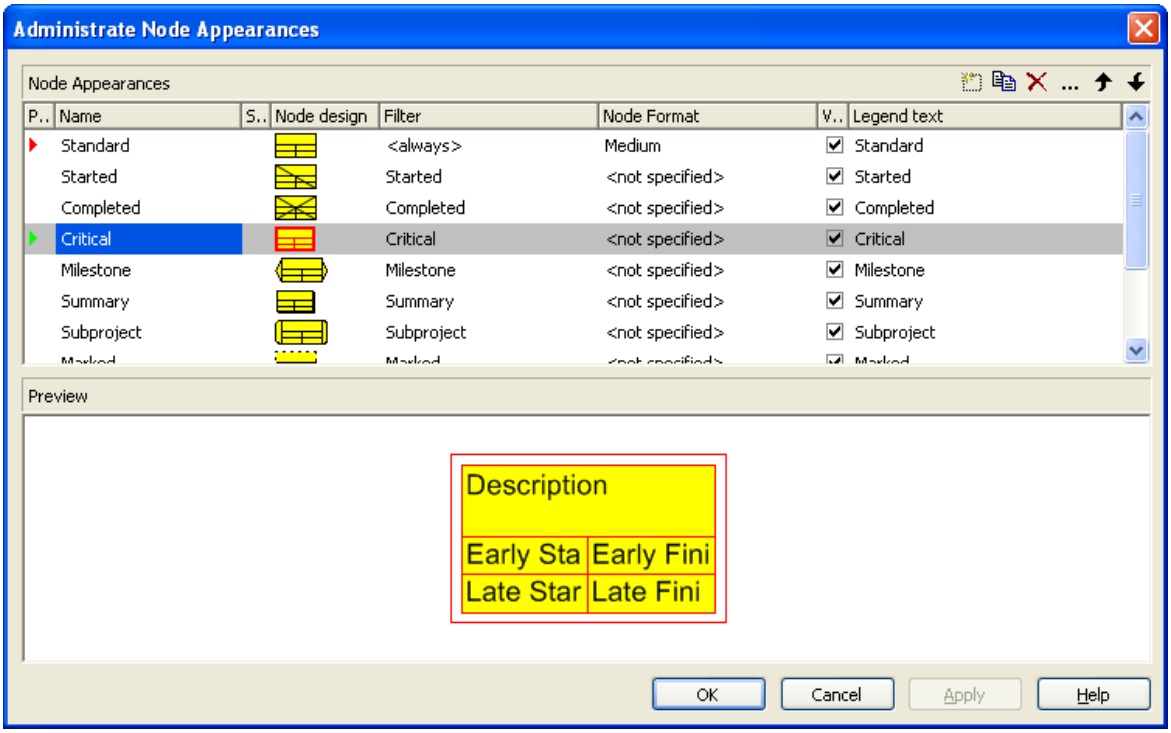
Value in the field "Auxiliary Nodes"	Top-to-bottom orientation	Left-to-right orientation
0	The rank number of the auxiliary node is not diminished.	The rank number of the auxiliary node is not diminished.
1	The rank number of the auxiliary node is diminished by 1. The auxiliary node appears left or right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

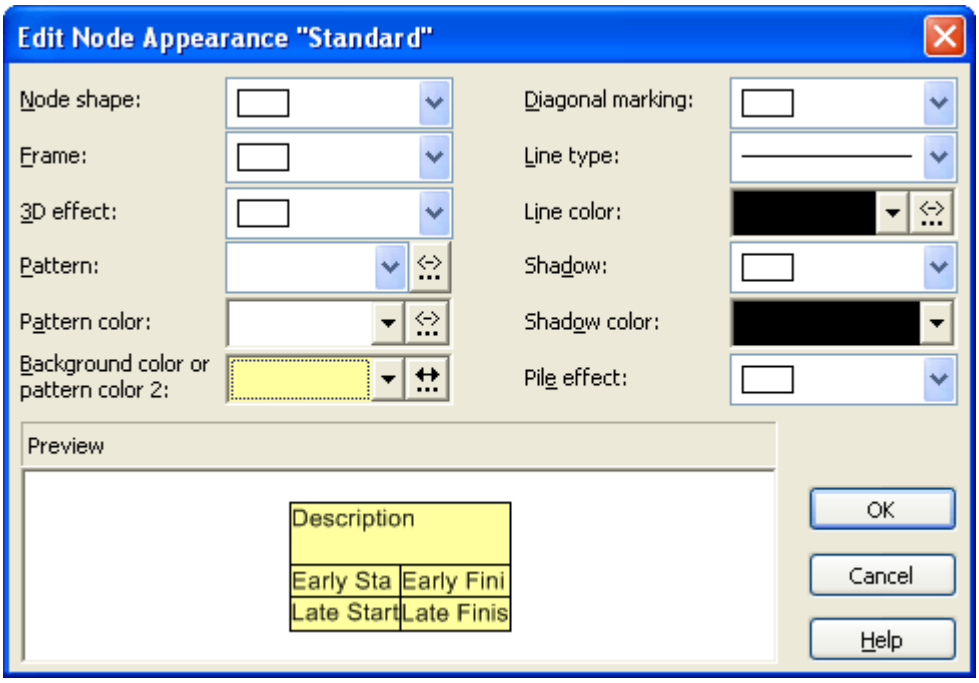
3.17 Node Appearance

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 **Administer Node Appearances** dialog. There you can edit, copy or delete node appearances or create new node appearances or modify the order of working off.



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.

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



If a node fulfills 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.

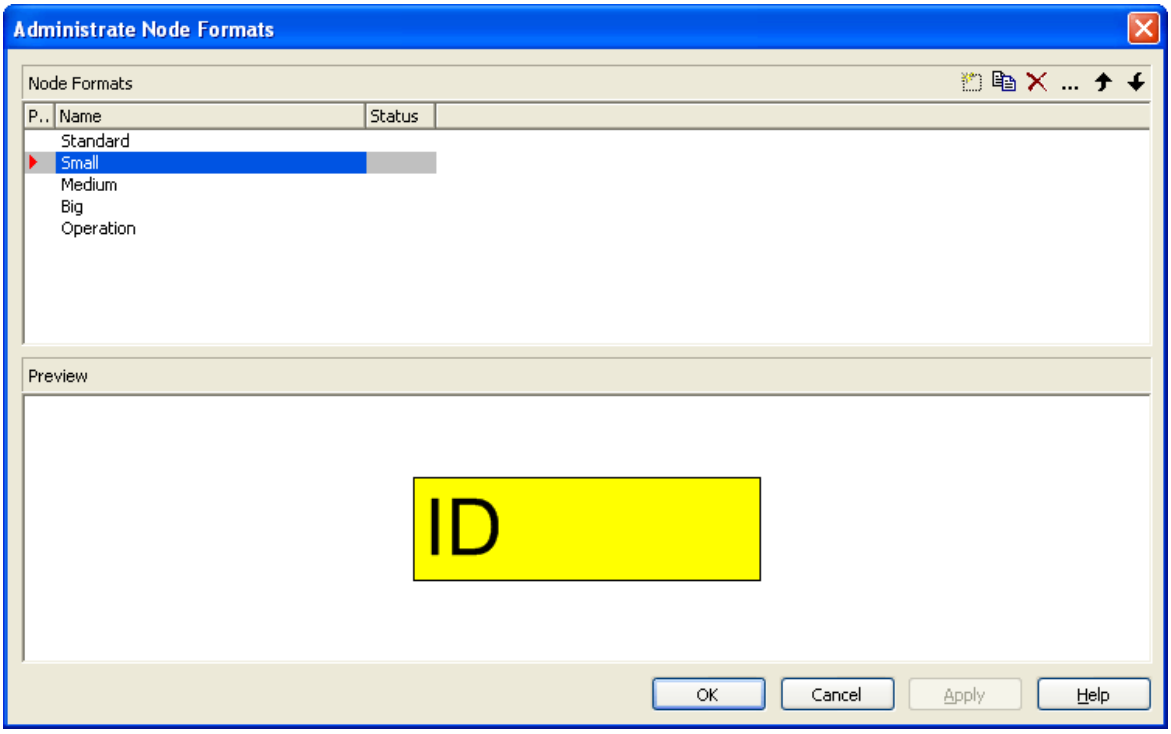
You can modify the order of working off the node appearances by clicking on the arrow buttons.

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 Terms: Maps".

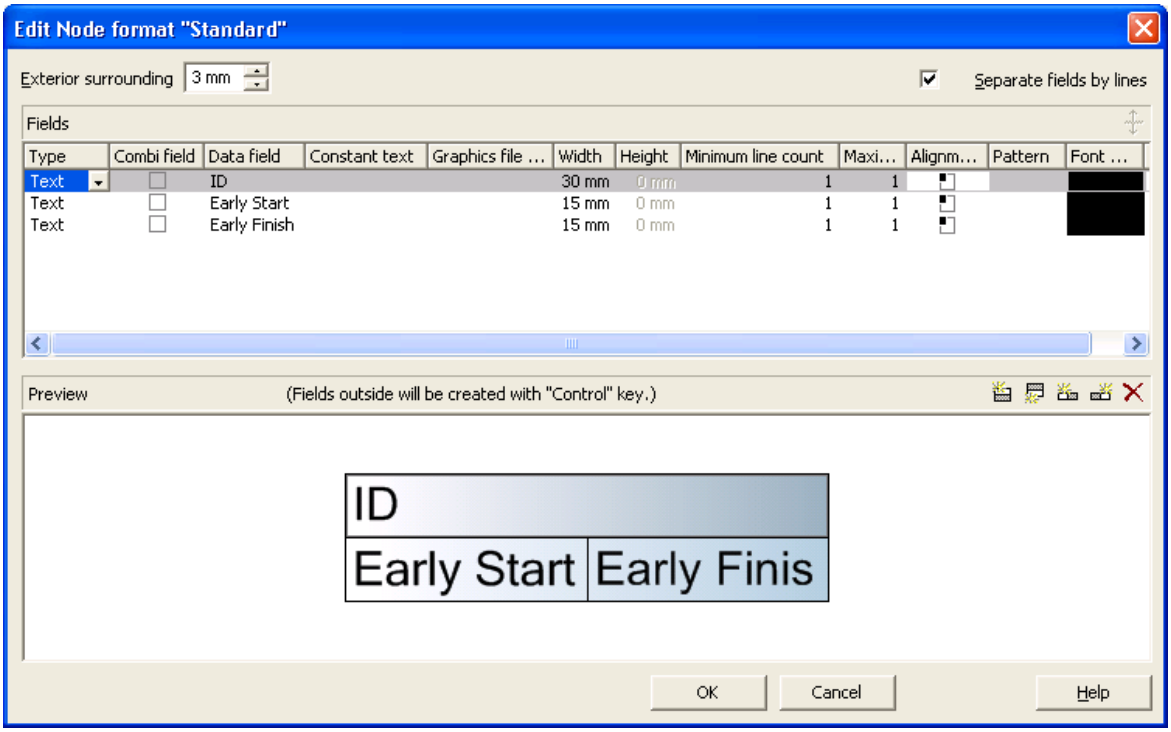
3.18 Node Format

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 by the **Node formats** button in the **Objects** property page.



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



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 the 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 Terms: Maps".

3.19 OLE Drag & Drop

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

Via OLE Drag & Drop nodes and their links or subnets can be moved. The drag & drop mode is either started automatically or can be started manually by the VcNet method **OLEDrag**.

> OLE Drag Mode

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

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

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

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

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

> OLE Drop Mode

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

There are three options:

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

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

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

> Displaying a Phantom During an OLE Drag & Drop Operation

The check box **Show phantom** lets you disable the display of an OLE drag phantom. Disabling the phantom is useful for dropping operations between different controls, where merely the attributes of the moved object change, omitting to generate a new object.

> Show Own Mouse Cursor

The check box **Show own mouse cursor** 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 via the event **OLEGiveFeedback**. If you do this, two competing cursors will exist in the target control, and will start to flicker. This you can avoid by disabling the target cursor.

Beside, if the cursor is enabled and the property **vcOLEDropManual** 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.

> Events

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

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

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

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

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

dataObject.SetData Empty, myClipFormat

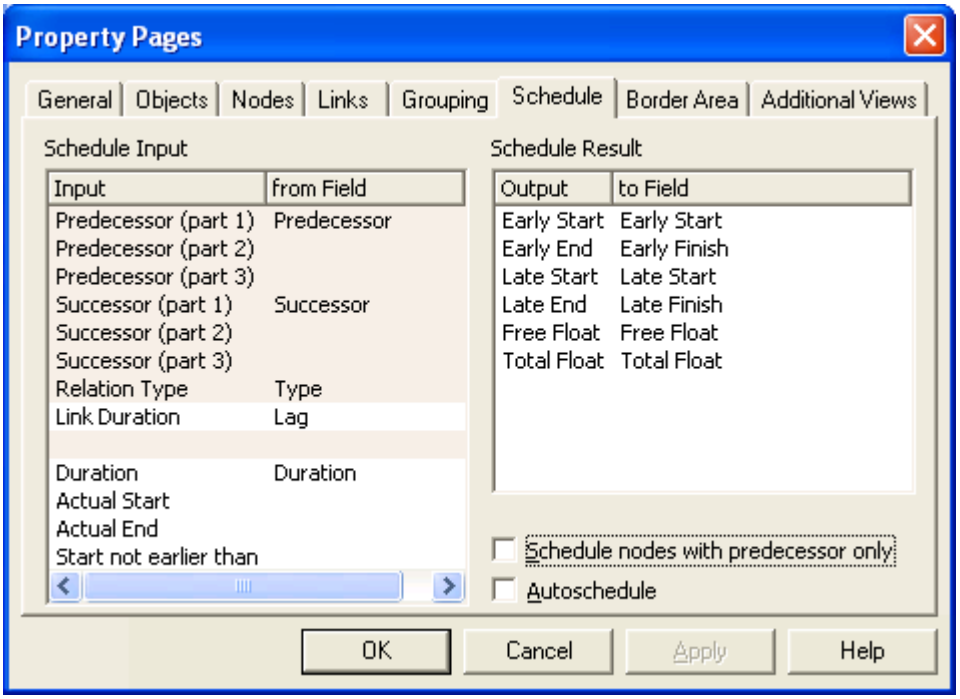
On a request for the existence of a format using **dataObject.GetFormat** the target application will answer **True**. A **DataObject.GetData** call to the source control will trigger the **OLESetData** event which then allows to pass the desired formats.

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

3.20 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 input (**Schedule Input**) and for 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.



The **Schedule Input** lets you select data fields from which the data is loaded. The scheduler uses data fields of the maindata and relations table as input fields for calculating dates, but outputs to data fields of the maindata table only.

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.

3.21 Status Line Text

The **OnStatusLineText** 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 events **OnToolTipText** and **OnToolTipTextAsVariant** let you edit tooltips (None, Node) that occur during runtime, in order to, for example, translate them into a different language or to suppress them.

The event **OnToolTipTextAsVariant** is required if you use a Script language that does not allow to return strings, e.g. VBScript. To activate the event, set the VcNet property **ShowToolTip** to **True**.

Example Code

```
VcNet1.ShowToolTip = True
```

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

3.23 Unicode

To display Unicode characters on the property pages at design time, an appropriate font has to be set by following the menu of the operating system through **Start / Settings / Control Panel / Display / Appearance** to the **Window** field.

Besides, only those characters can be displayed that belong to the language set by the menu items **Start / Settings / Control Panel / Regional and Language options** .

All objects in a VARCHART component which contain texts can display Unicode characters if an appropriate font was set in the corresponding property **Font**.

A Unicode font can be assigned to context menus, tooltips and run time dialogs by the property **DialogFont** of the **DummyObject** object.

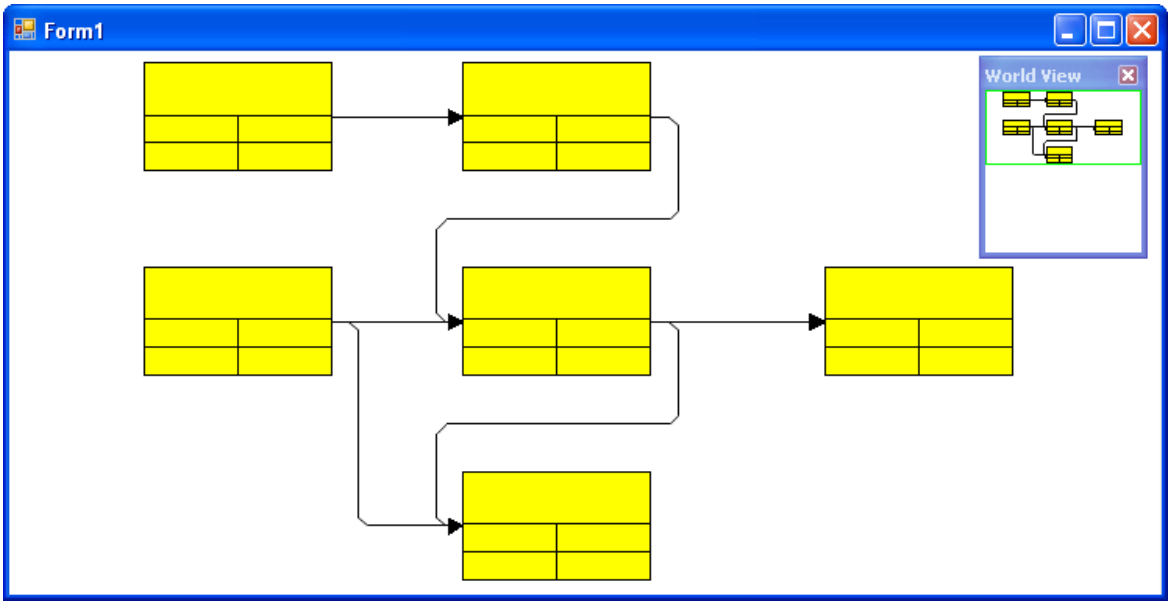
You will find an overview of all available fonts, which contain at least part of all unicode characters in "Wazu Japa's Gallery of Unicode Fonts" (http://www.wazu.jp/index.html). Detailed information on the Unicode standard is also offered on the homepage of the Unicode Consortium (http://www.unicode.org) and on Microsoft's GlobalDev Homepage (http://www.microsoft.com / globaldev / getwr / steps / wrg_unicode.mspx). In Windows 2000 and XP you can find out about the characters contained in the built-in fonts under **Start / Programs / Accessories / System Tools / Character Map**.

When importing CSV files, the method **VcGantt.Load** automatically recognizes whether there is a Unicode or an ANSI file.

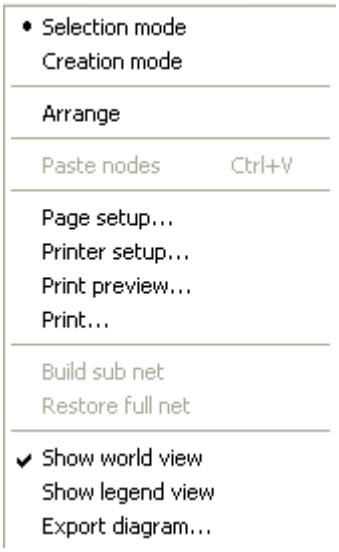
Note: The development environments of Visual Studio 6 are not able to use Unicode characters in source code files. Internally however, the strings of VB6 are displayed in Unicode. If you use Visual C++ combined with MFC you have to set the Defines_UNICODE and UNICODE to use strings in Unicode. The version Visual Studio .NET 2002 and later versions allow to edit source code files in Unicode coding. When saving a file, you need to select the coding type "Unicode".

3.24 World View

The world view is an additional window that displays the diagram completely. A frame indicates the diagram section actually displayed in the main window. If you move the frame or modify 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 change if you scroll or zoom the section in the main window.



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



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

The properties of the World View can also be specified by the API property **VcNet.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.

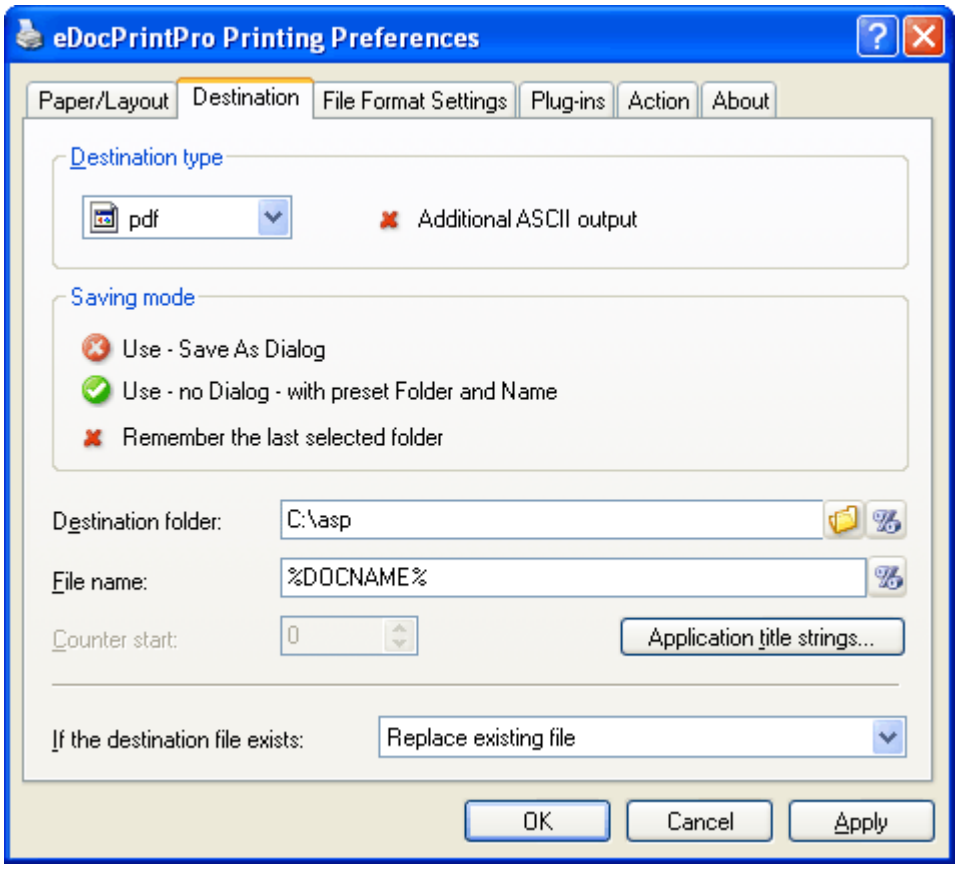
To be suitable a PDF printing driver has to fulfill the below requirements concerning controlling and print quality:

- 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:



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

Example Code

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

Very few printing drivers require a different program code:

Example Code

```
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

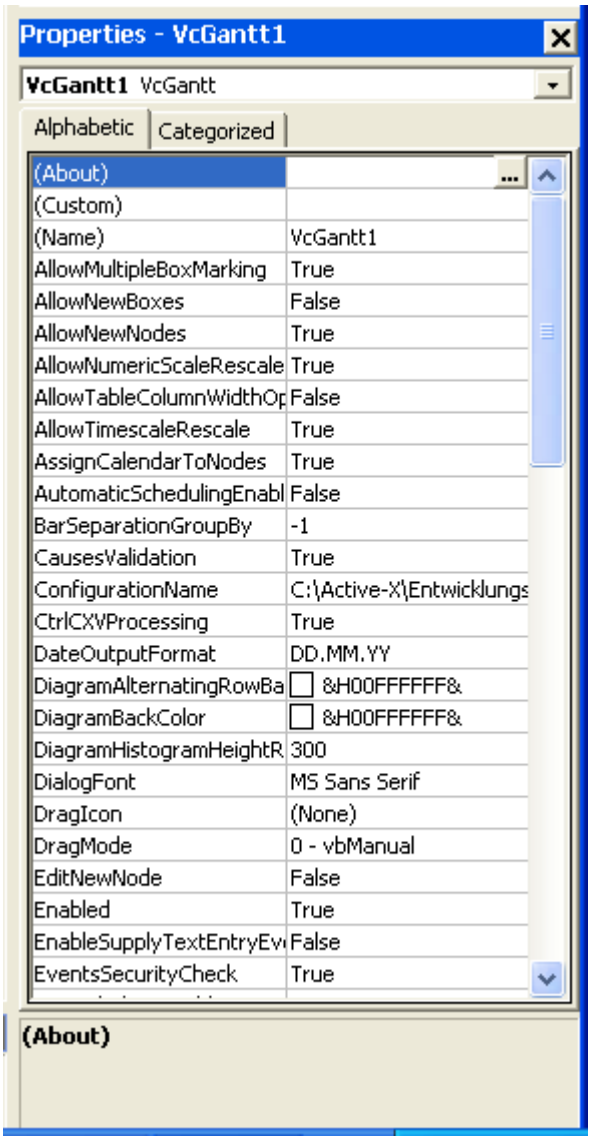
4.1 General Information

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.

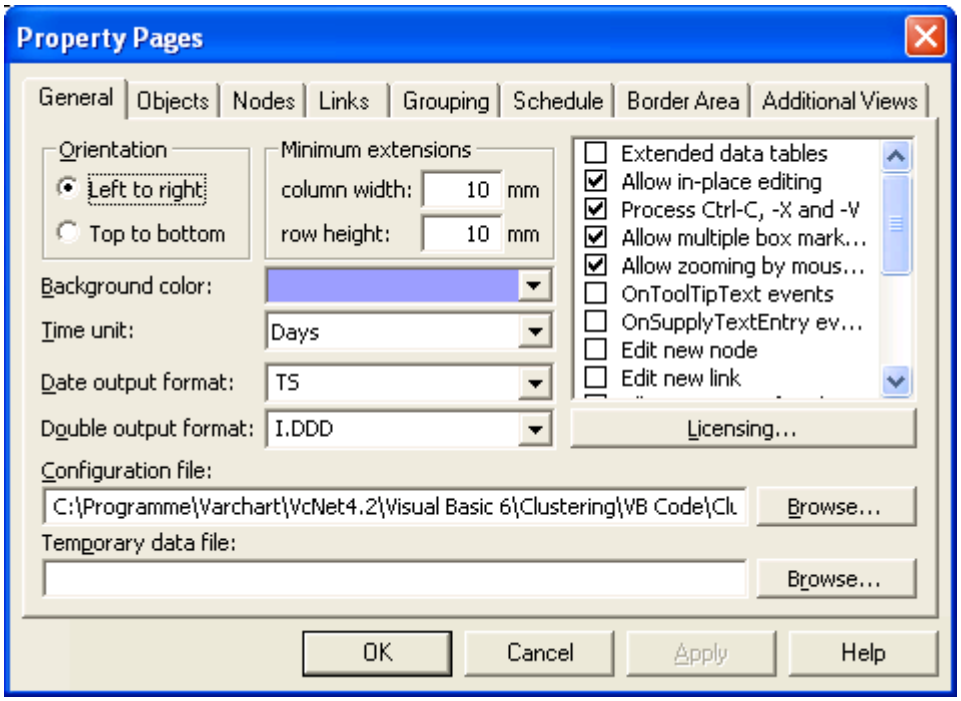
or

- In the **Properties** box of the control (to be invoked by the F4 key) click on the right icon in the icon bar  .



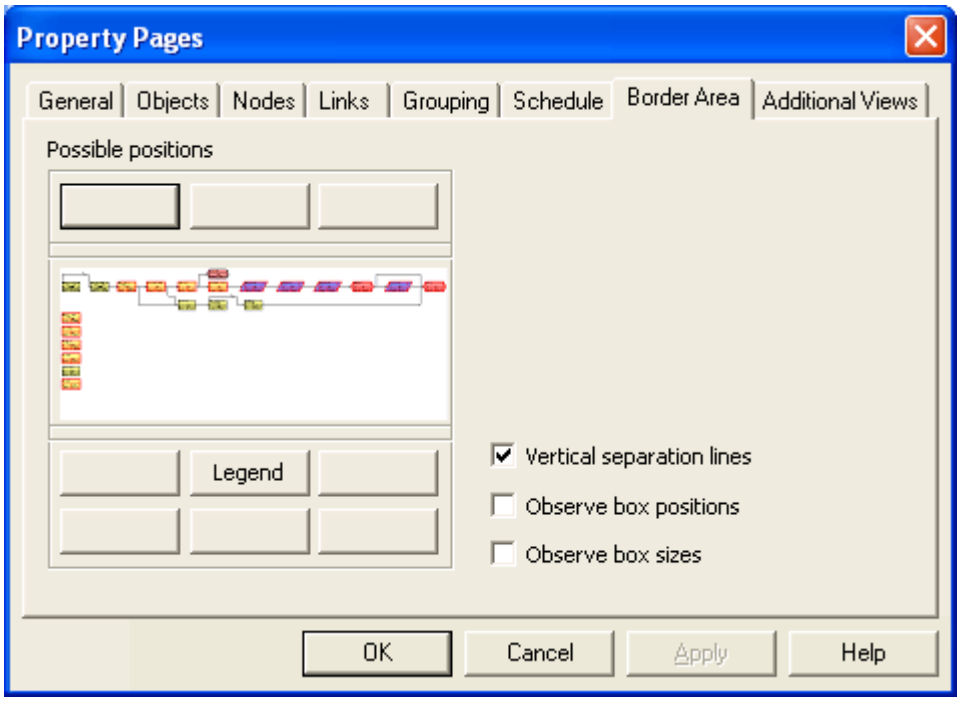
More information about the functions of property pages and dialog boxes you can obtain by either clicking on the **Help** button or by pressing the **F1** key of your keyboard. This will open the corresponding online help file.

4.2 The "General" Property Page



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

4.3 The "Border Area" Property Page



Possible positions

There are three areas above and six areas below the diagram which you can utilise for texts, graphics or a legend. These areas are displayed only in the print preview and in the print output. Click on one of the buttons above/below the diagram to reach the **Specification of texts, graphics and legend** dialog box.

Vertical separation lines

Activate this check box, if the areas for texts, graphics or the legend are to be separated by vertical lines.

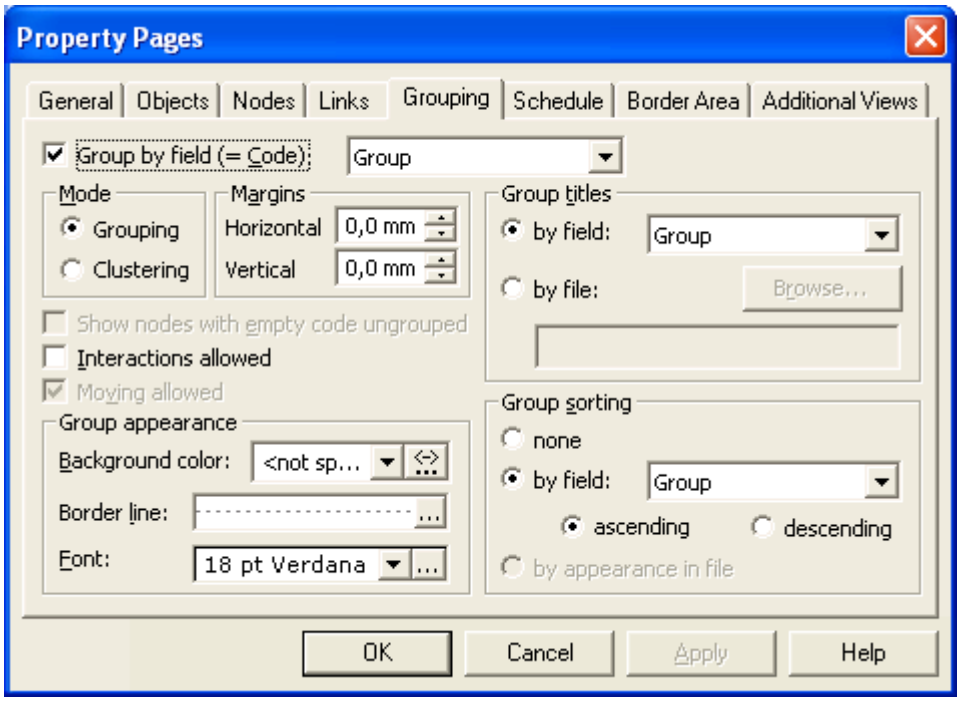
Observe box position

Activate this check box, if the box positions are to be considered as exactly as possible. Otherwise the available space will be divided proportionally between all elements in the row.

Observe box size

Activate this check box, if the box sizes are to be considered as exactly as possible. Possibly the chart will be enlarged and/or the texts in the boxes will be cropped.

4.4 The "Grouping" Property Page



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 the Plus or Minus symbol beside the group title).

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 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 VARCHART ActiveX control.

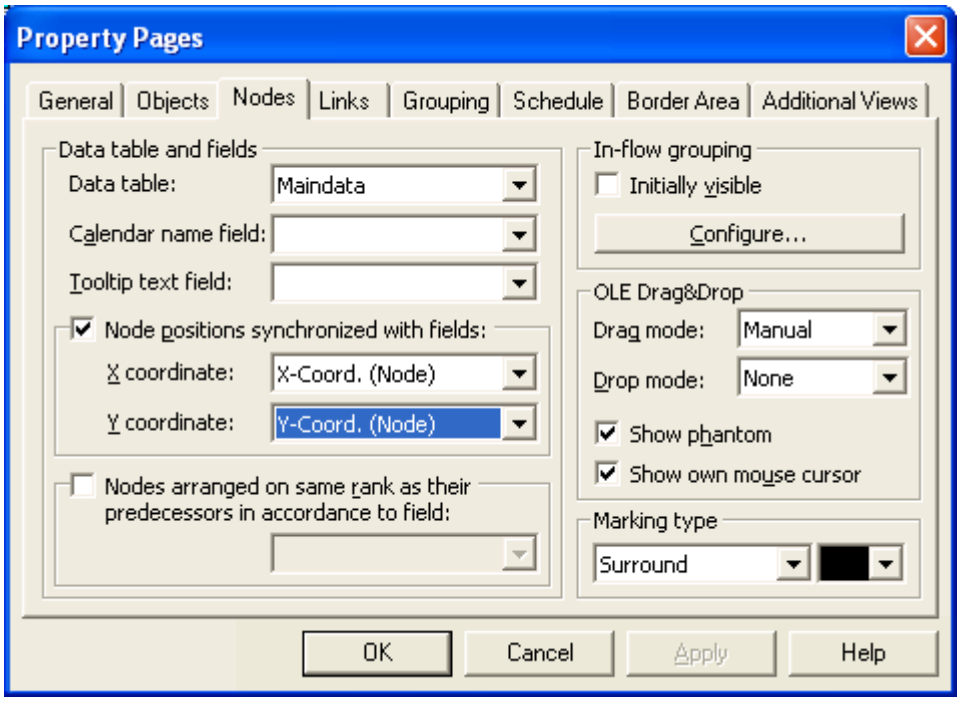
Group sorting

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



Data table

Select the data table to be used for the visualisation of the nodes.

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

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** needs to be activated. Beside, the calendars have to be created before loading the nodes.

This feature can also be set by the property **VcNet.NodeCalendarName-DataFieldIndex**.

Tooltip text field

The data field specified here is shown as a tooltip if you show a VMF file using the WebViewer and there right-click on a node. No further settings are required.

The VMF (Viewer Metafile) format is a vector format that allows to store a chart independently of pixel resolution. Files of the VMF format can be

displayed by the GRANEDA WebViewer on any platform using Java compatible internet browsers.

To show tooltips in your VARCHART ActiveX application, activate the check box **OnToolTipText events** on the **General** property page or set the property **ShowToolTip** to **True** and in the **OnToolTipText** event, specify the data fields to be displayed.

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

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 field

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.

Before you can select the field, it needs to have been generated. If the field doesn't exist, please create it on the **General** property page. 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.

Value of the data field	Top-to-bottom orientation	Left-to-right orientation
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 Terms: Nodes".

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.

Drag mode

With this property you can set/retrieve, whether dragging a node beyond the limits of the current VARCHART XNet control is allowed.

- If you select **Manual** you need to invoke the method **OLEDrag** to trigger dragging the node.
- If you select **Automatic**, dragging a node beyond the control limits will be started automatically.

On the start of dragging, the source component will fill the DataObject with the data it contains and will set the **effects** parameter before initiating the OLEStartDrag event, as well as other source-level OLE Drag & Drop events. This gives you control over the drag/drop operation and allows you to intercede by adding other data formats.

VARCHART XNet by default uses the clipboard format CF_TEXT (corresponding to the vbCFText format in Visual Basic), that can be retrieved easily.

During dragging, the user can decide whether to shift or to copy the object by using the Ctrl key.

OLE drag & drop operations in VARCHART XNet are compatible to the ones in Visual Basic. Methods, properties and events have identical names and meanings as the default objects of Visual Basic.

Drop mode

By this property you can set/retrieve, whether a node from a different VARCHART XNet control can be dropped to the current control.

- Dropping will not be allowed if you select **None**.
- If you select **Manual**, you will receive the event **OLEDragDrop** that enables you to process the data received by the object dropped, e.g. to generate a node or to read a file. If the source and the target component are identical, you will receive either the event **OnNodeModifyEx** or **OnNodeCreate** as with OLE Drag&Drop switched off.
- If you select **Automatic**, the dropping will automatically be processed by the control, generating a node in the place of the dropping, if possible.

Show phantom

This property lets you disable the display of an OLE drag phantom. Disabling the phantom is useful if generating a new object is omitted but merely the attributes of the object in the target control are modified.

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

Show own mouse cursor

This property lets you enable or disable the mouse cursor in the target control during an OLE drag operation. OLE Drag & Drop allows to set the cursor in the source control by the event **OLEGiveFeedback**. If you set it, two competing cursors will exist in the target control, that may appear to flicker. You can avoid the flickering by disabling the target cursor by this check box.

Beside, if the cursor is enabled and the property **OLEDropManual** is set, objects cannot be dropped outside the joining ports of a node. If you disable the cursor, you can drop objects outside the joining ports.

This feature can also be set by the property **VcNet.OLEDragWithOwn-MouseCursor**.

Marking type

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

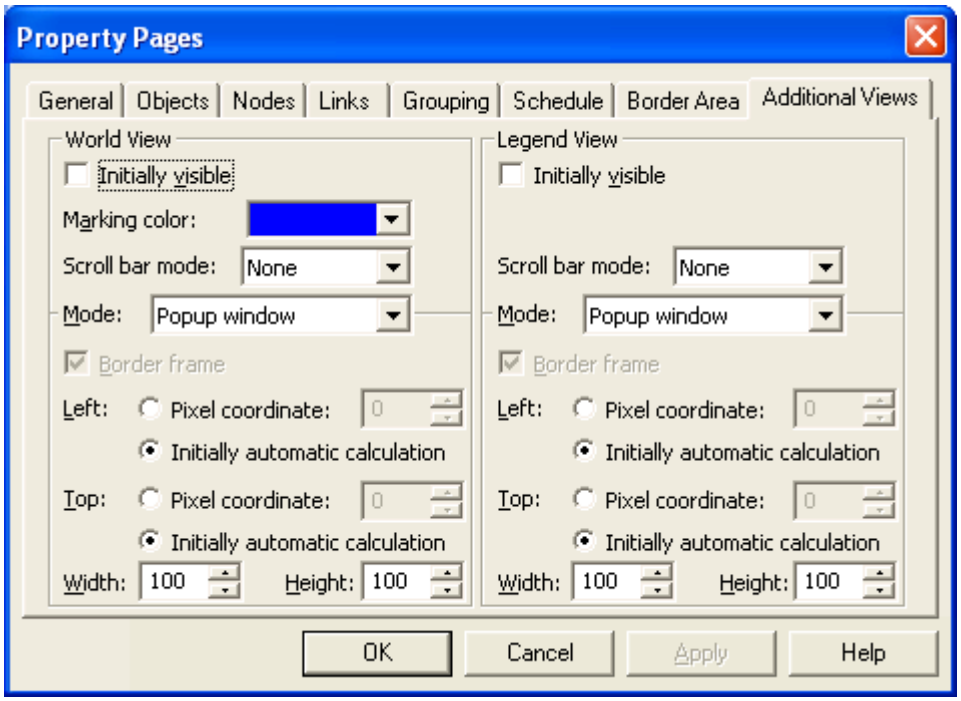
- No Mark
- Surround

- Surround inside
- Invert
- Pickmarks
- Pickmarks inside

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

Beside, you can select a color for the marking type set.

4.6 The "Additional Views" Property Page



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 frame that indicates the displayed section in the World View.

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

Scroll bar mode

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

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

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

Mode

Select the view mode. The below options are available:

- **Left fixed:** The view is displayed on the left side of the VARCHART ActiveX control window. Only the width can be set, whereas the position and the height are fixed.
- **Right fixed:** The view is displayed on the right side of the VARCHART ActiveX control window. Only the width can be set, whereas the position and the height are fixed.
- **Top fixed:** The view is displayed on the top of the VARCHART ActiveX control window. Only the height can be set, whereas the position and the width are fixed.
- **Bottom fixed:** The view is displayed on the bottom of the VARCHART ActiveX control window. Only the height can be set, whereas the position and the width are fixed.
- **Position not fixed:** The view is a child window of the current parent window of the VARCHART ActiveX. It can be positioned at any position

and be of any extension. The parent window can be modified by the property **VcWorldView.ParentHWnd**.

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

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

Border frame

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

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

Left

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

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

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

Top

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

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

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

Width

*Not active if the mode **Top fixed/Bottom fixed** was selected.* Select the horizontal extension of the view. Note that the pixel coordinate is a system (device) coordinate.

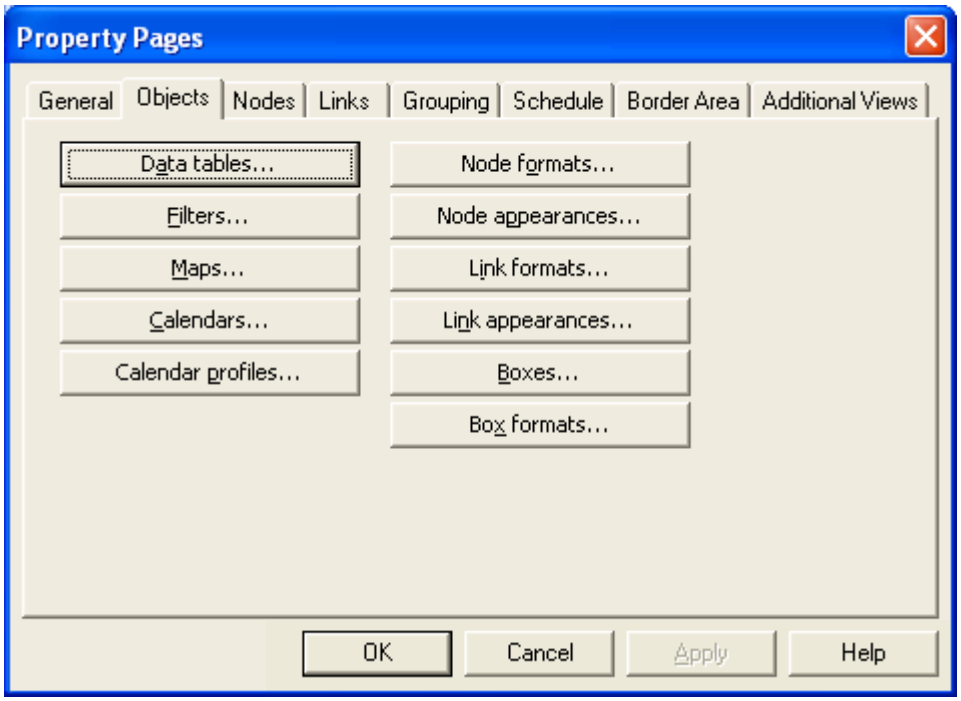
This property can also be set by the API calls **VcWorldView.Width** and **VcLegendView.Width**

Height

*Not active if the mode **Left fixed/Right fixed** was selected.* Select the vertical extension of the view. Note that the pixel coordinate is a system (device) coordinate.

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

4.7 The "Objects" Property Page



Data tables

Opens the dialog **Administrate Data Tables**.

Filters

This button lets you open the **Administrate Filters** dialog box.

Maps

This button will open the dialog **Administrate Maps**.

Calendars

Opens the dialog **Specify Calendars**.

Node formats

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

Node appearances

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

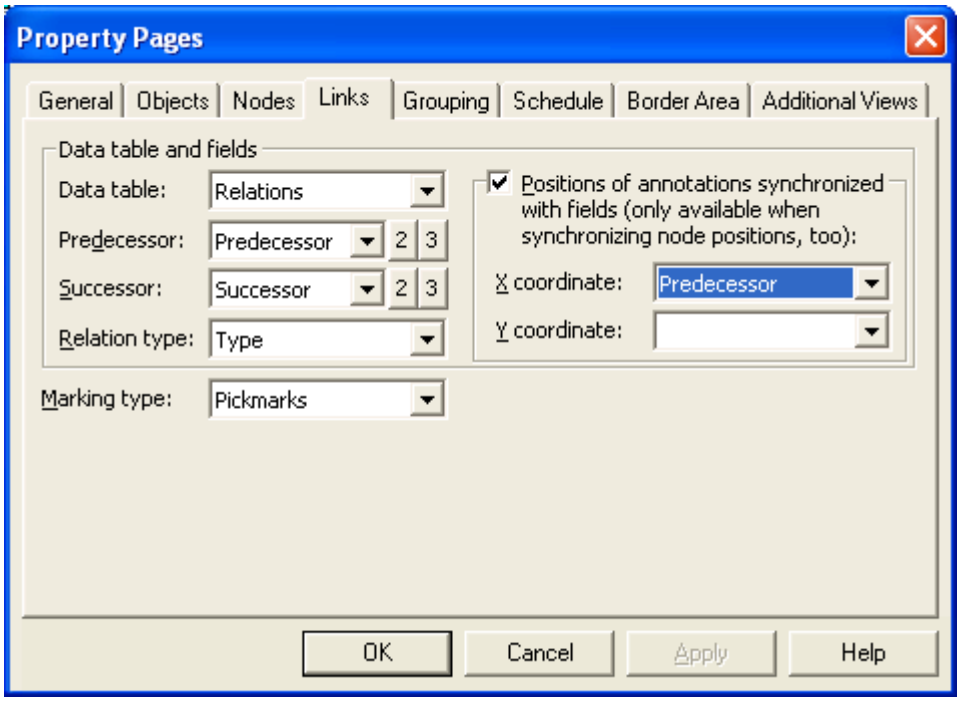
Boxes

Opens the dialog **Administrate Boxes**.

Box formats

Opens the dialog **Administrate Box Formats**.

4.8 The "Links" Property Page



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 of the links. This feature can also be set by the property **VcNet.LinksDataTableName**.

Predecessor

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

Successor

This field lets you select a data field from the **Relations** table that the identification of the successor node of the link is stored to.

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:

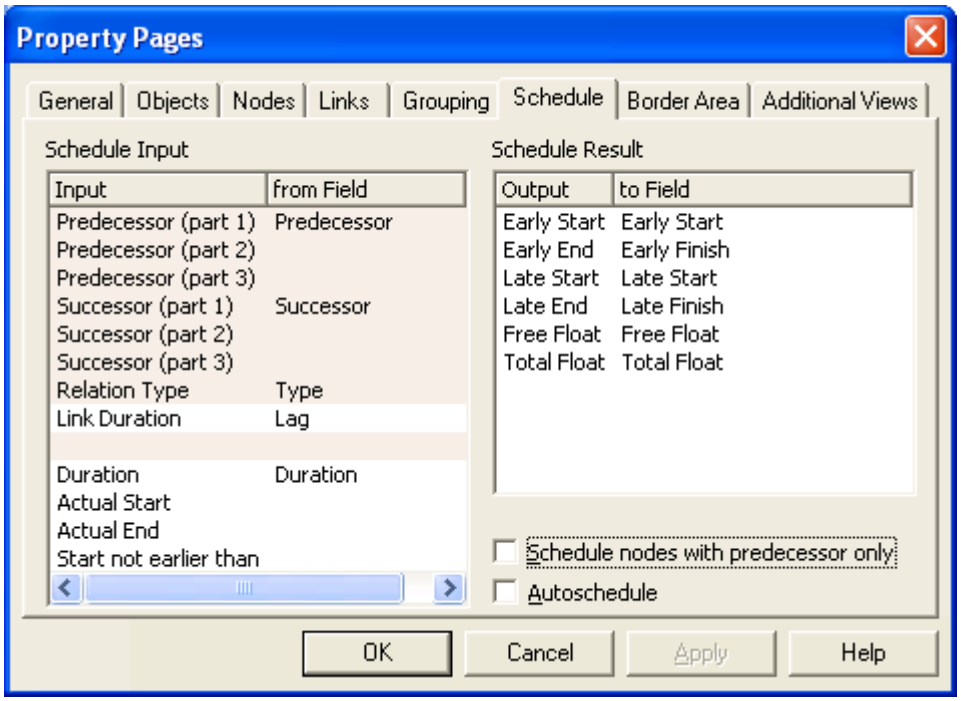
- Surround
- Invert
- No Mark
- Pickmarks

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



This property page lets you adapt VARCHART XNet's date calculation settings to your interface by specifying which data fields you want to use for the input (**Schedule Input**) and output (**Schedule Result**) of the scheduler.

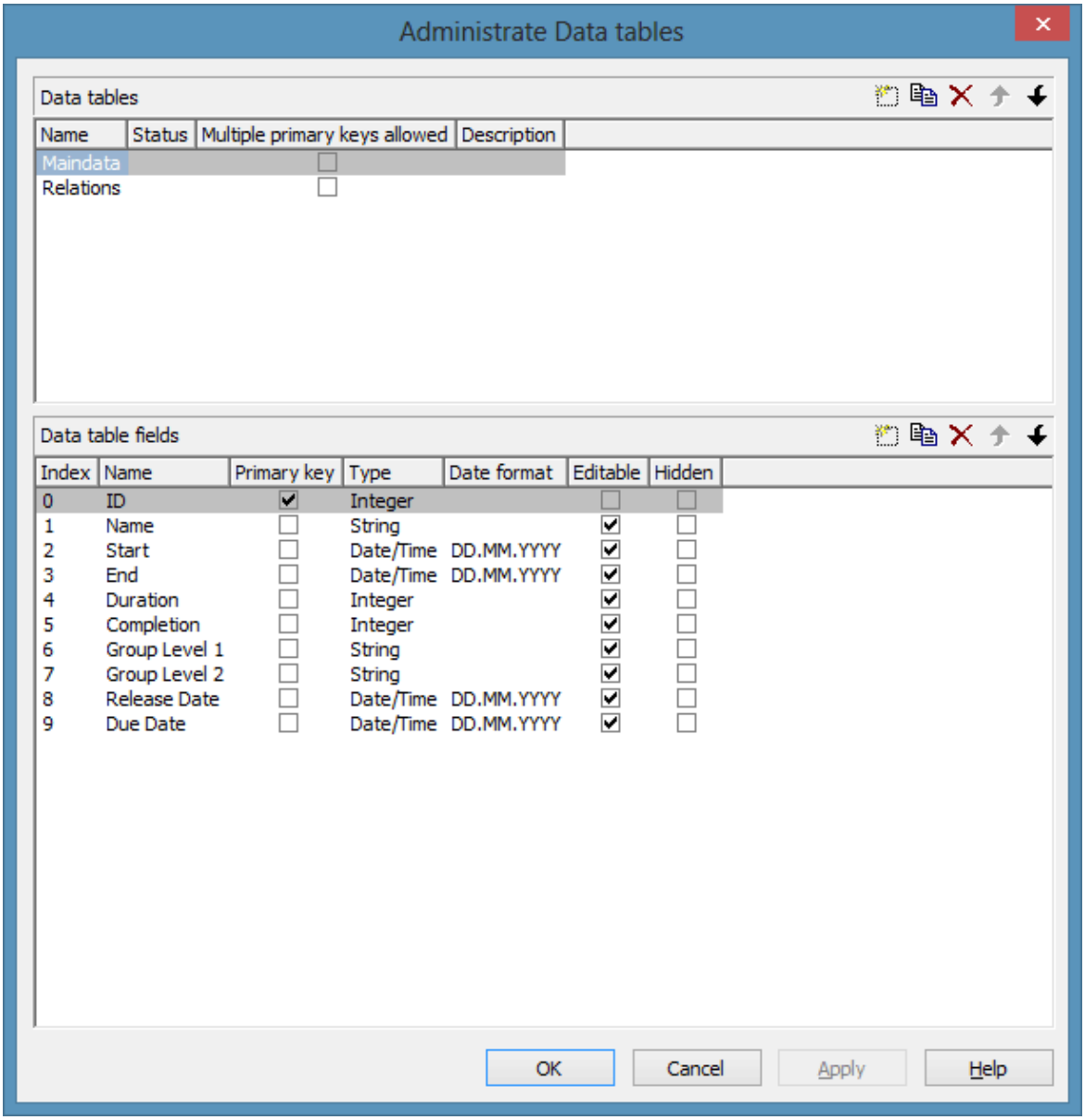
Schedule Input

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

Schedule Result

Specify for each result to which field it should be stored. The scheduler only outputs to data fields from the **Maindata** table. The early/late start and end dates plus the total float and free float are calculated from the duration of the activities, the logical dependencies and the project start.

4.10 The "Administrate Data Tables" Dialog Box



You can get to this dialog via the property page **Objects**. This dialog lets you create and edit data tables and their data fields.

Data tables

- **Name:** Lists the names of all existing data tables. The names can be edited.

- **Status:** In the **Status** column each data table that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.
- **Multiple primary keys allowed:** Here you can define whether the primary key for your table consists of **one** or **more (maximum 3** fields. As soon as you have checked the box **Multiple primary keys allowed** you can select up to three data fields for the primary key in the **Data table fields** section. The box **Multiple primary keys allowed** can only be unchecked if no more than one field is selected as primary key in the **Data table fields** section.
- **Description:** Here you can describe the data table.

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

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

Data Table Fields

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

- **Index:** The index of the data fields cannot be modified, since internally, it serves as a reference. In the API, data fields are referred to by the index.
- **Name:** This column displays the names of the fields of the data table. You can modify the field names after clicking on them.
- **Primary Key:** This check box allows to select a data field from the column to be the primary key of the data record.
- **Type:**This field allows to set the data type of the data field selected. You can choose between:

  String

  Integer

  Date/Time

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

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

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

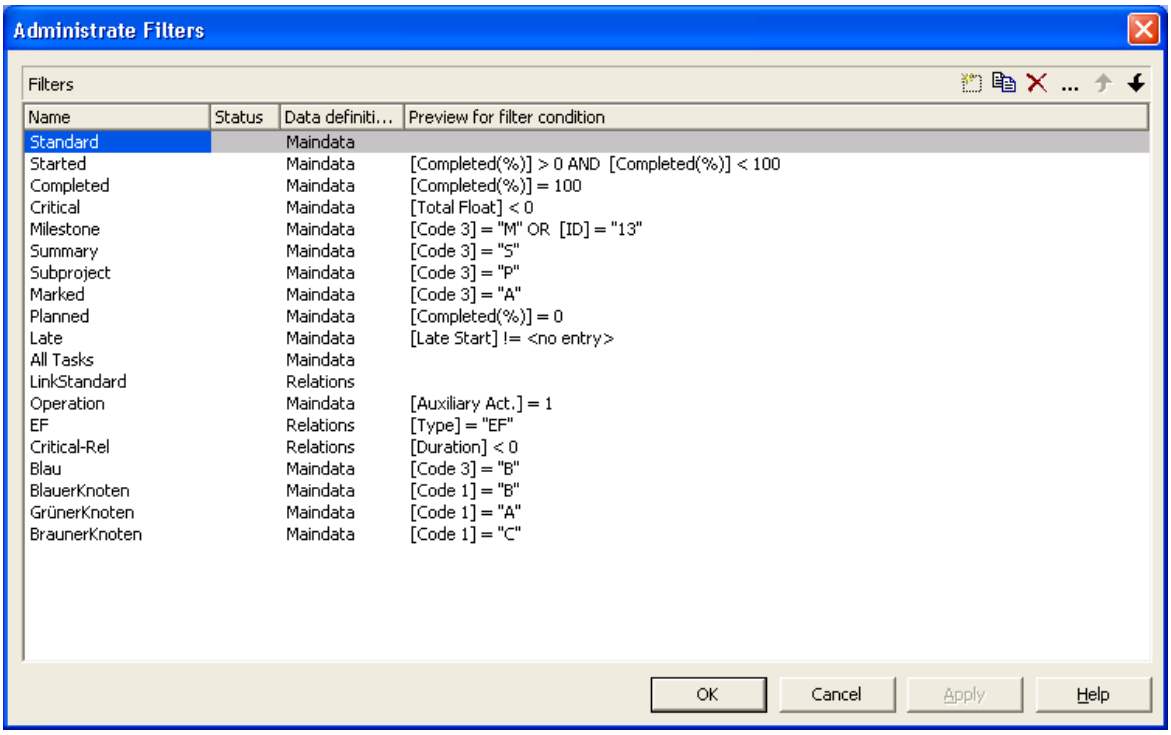
The date format set here only is relevant for entering data, but not for displaying data.

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

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

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

4.11 The "Administrate Filters" Dialog Box



You can get to this dialog box

- by the **Objects** property page
- by the **Administrate Node Appearances** dialog box
- by the **Administrate Link Appearances** dialog box

Name

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

Status

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

Data definition table

This column displays the data definition table (**Maindata** or **Relations**) associated with a filter (see property page **DataDefinition**).

Preview for filter condition

This column displays the conditions of the filters. Conditions cannot be edited in this dialog. To modify the filter condition, click on the **Edit filter** button.

Add filter

 A new filter is created. You can modify its default name by double-clicking and editing it. New filters are created in a context-sensitive way, i. e. the matching data definition table will be used automatically.

Copy filter

 Copies the selected filter.

Delete filter

 The marked filter in the list will be deleted. You can only delete filters that are not currently used.

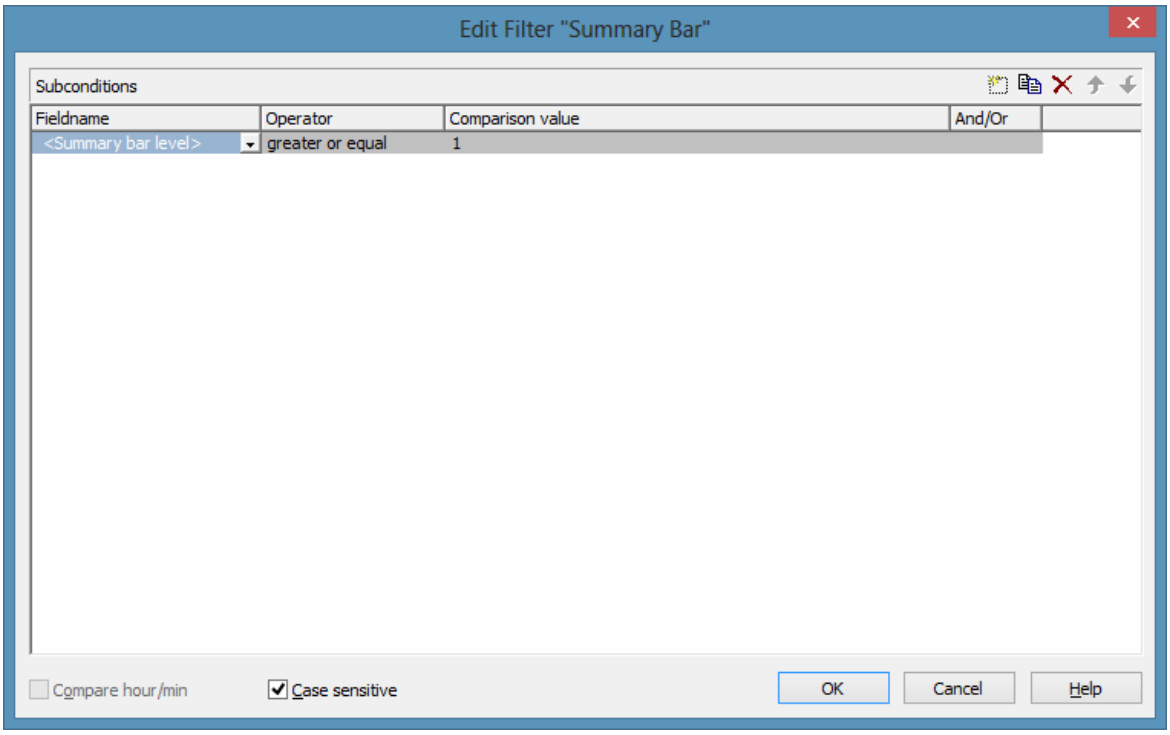
Edit filter

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

Promote / demote filter

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

4.12 The "Edit Filter" Dialog Box



You can get to this dialog box either

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

Add subcondition

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

Copy subcondition

Copies the selected subcondition.

Delete subcondition

Deletes the selected subcondition.

Evaluate subcondition earlier/later

 If a filter consists of several subconditions, the subconditions are evaluated one after the other. The top subcondition in the table is evaluated first.

Click on the **Evaluate subcondition earlier/later** button to move the selected subcondition by one position upward or downward in the list.

Fieldname

This list contains all data fields available to be compared with the comparison value.

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).

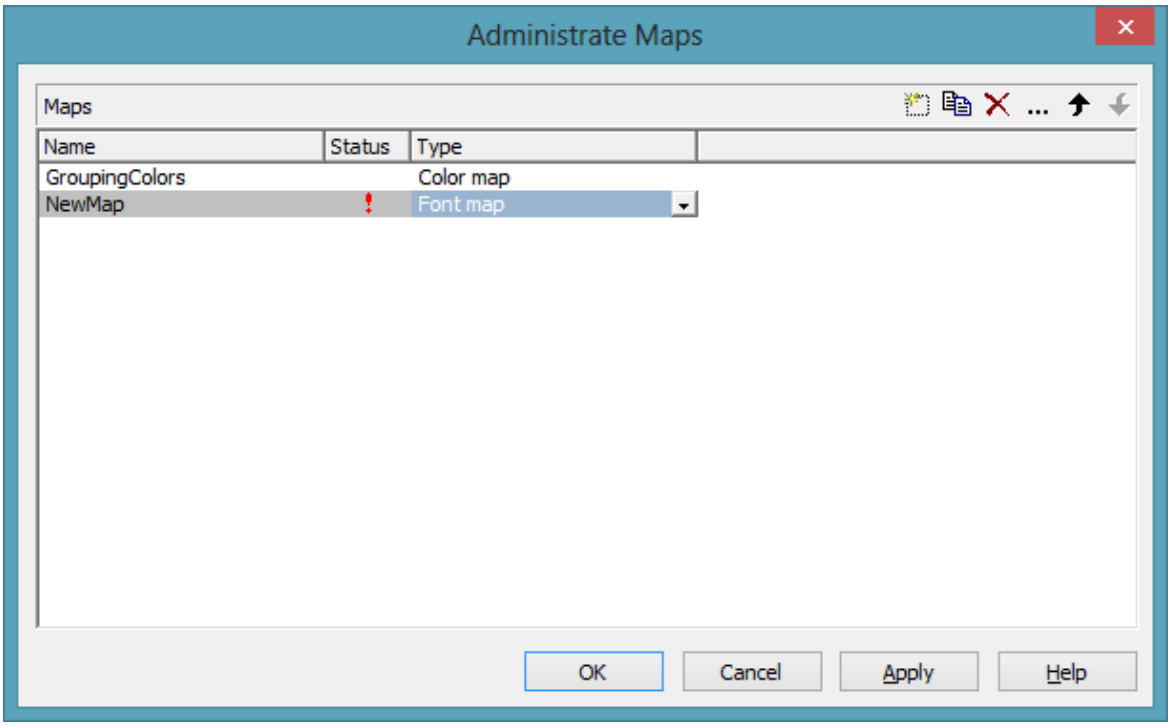
Compare hour/min

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



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.

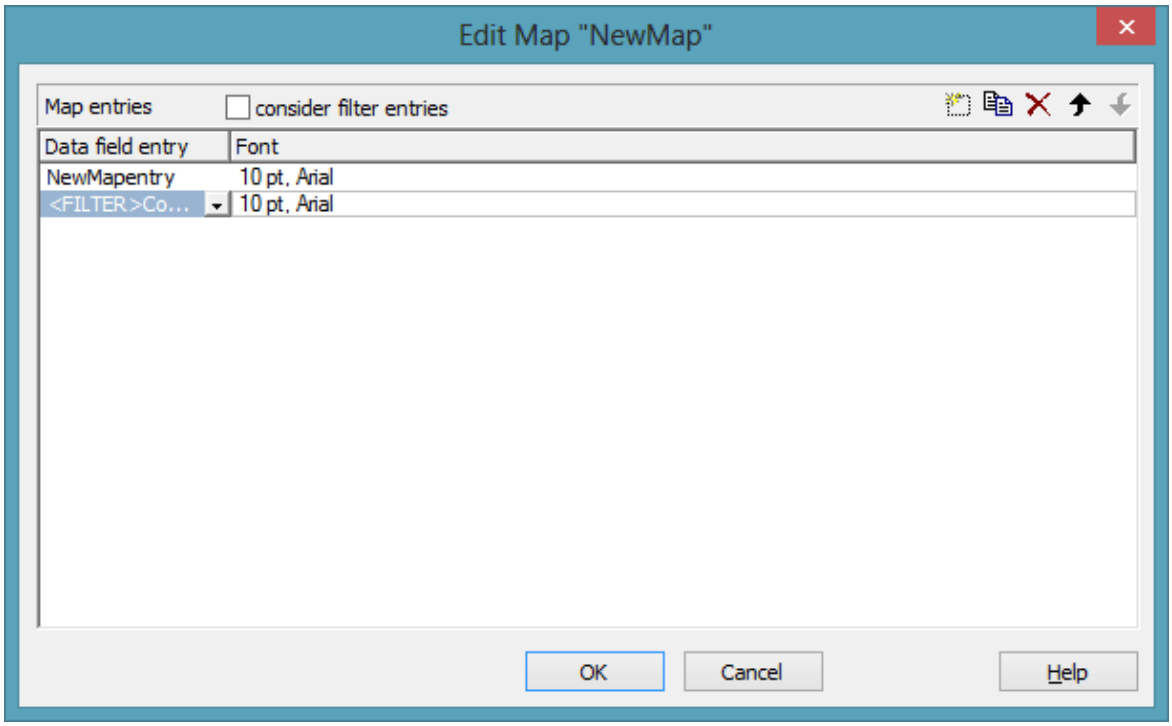
Edit map

 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



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.

Graphics File Name

Assign graphics files to the data field entries. To do so, click on the corresponding field. Then a dialog box opens that lets you select 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 VARCHART ActiveX control.

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 VARCHART ActiveX 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.

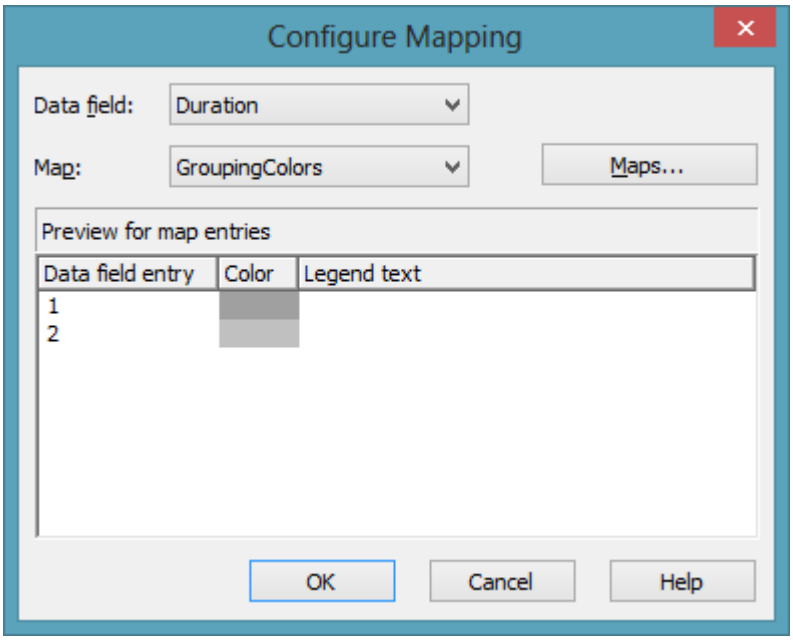
Delete map entry

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

Promote / demote map entry

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

4.15 The "Configure Mapping" Dialog Box



In this dialog box you can assign a map to a data field. You will get to it by clicking on the button  for the desired attribute in the dialog **Edit layer**.

Data field

Select the data field the entries of which control the desired attributes of the current object.

Data field

Select the data field whose entries control the color or pattern of the current object.

Map

(only activated if a data field has been specified) Select the map that depending on its type assigns the corresponding attributes to each data field entry.

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.

Maps

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

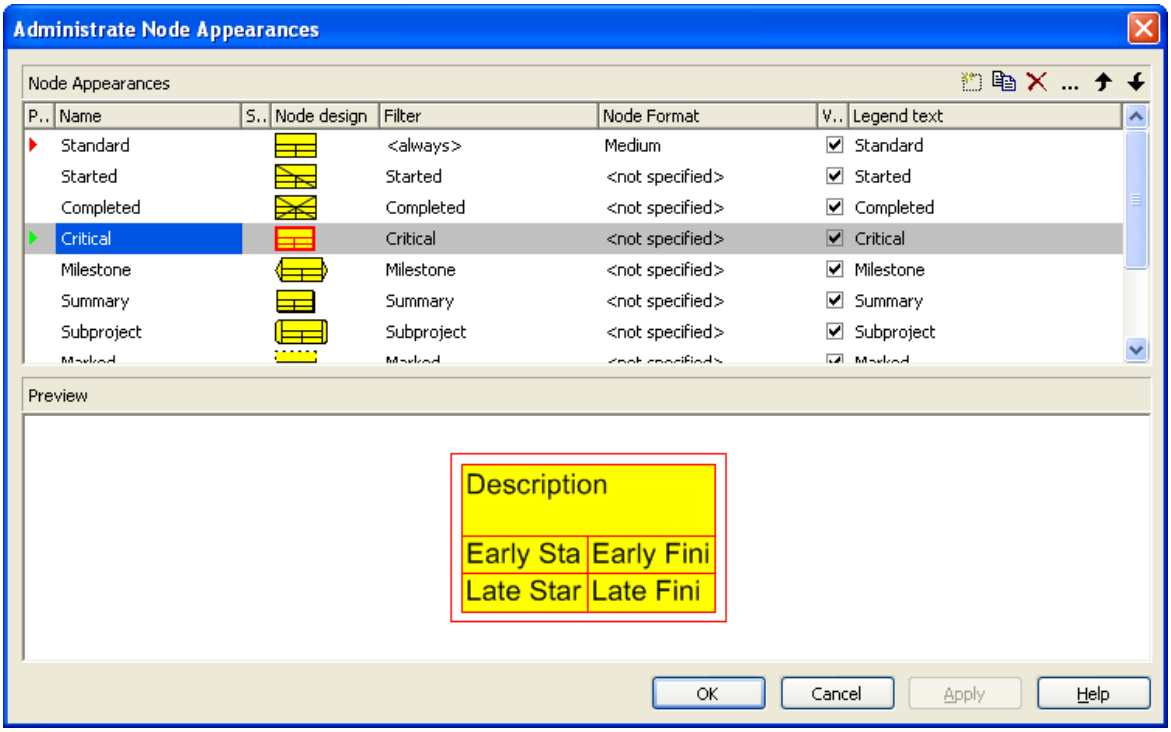
Preview for map entries

The preview shows the selected map: the data field entries and the attributes assigned to them.

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



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

The look of a node composes from filters that dynamically assign one or more node appearance objects to the node. The attributes of the node appearance objects collected by the filters superimpose to result in a final node appearance.

Preview

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

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

Name

There is a list of the names of the existing node appearances. All names can be edited.

Status

In this column each node appearance that has been added () and/or modified () since the dialog box was opened is marked by a symbol.

Node design

Contains 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** entry to reach the **Edit Node Appearance** dialog box.

Filter

The filter belonging to a node appearance regulates which activities are assigned that node appearance.

For most node appearance you can select the filter of your choice. Only for the node appearances "Standard" and "Interface" the filter is fixed ("<always>" or "<InterfaceNode>").

To assign a filter to a node appearance, mark the **Filter** field. A button for a combo box listing all available filters and an **Edit** button will appear (not applicable for the node appearances with fixed filters). Either select a filter for the node appearance in the combo box, or click on the **Edit** button to reach 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, first mark the **Node format** field. A button for a combo box listing all available node formats and an **Edit** button will appear. Either select a format in the combo box, or click on the **Edit** button to reach the **Administrate Node Formats** dialog box where you can edit, copy, add or delete a 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 each node appearance.

Add node appearance

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

Copy node appearance

 Copies the selected node appearance.

Delete node appearance

 This button lets you delete a node appearance you do not need 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 a node is assigned more than one node appearance, the node appearances are processed 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.

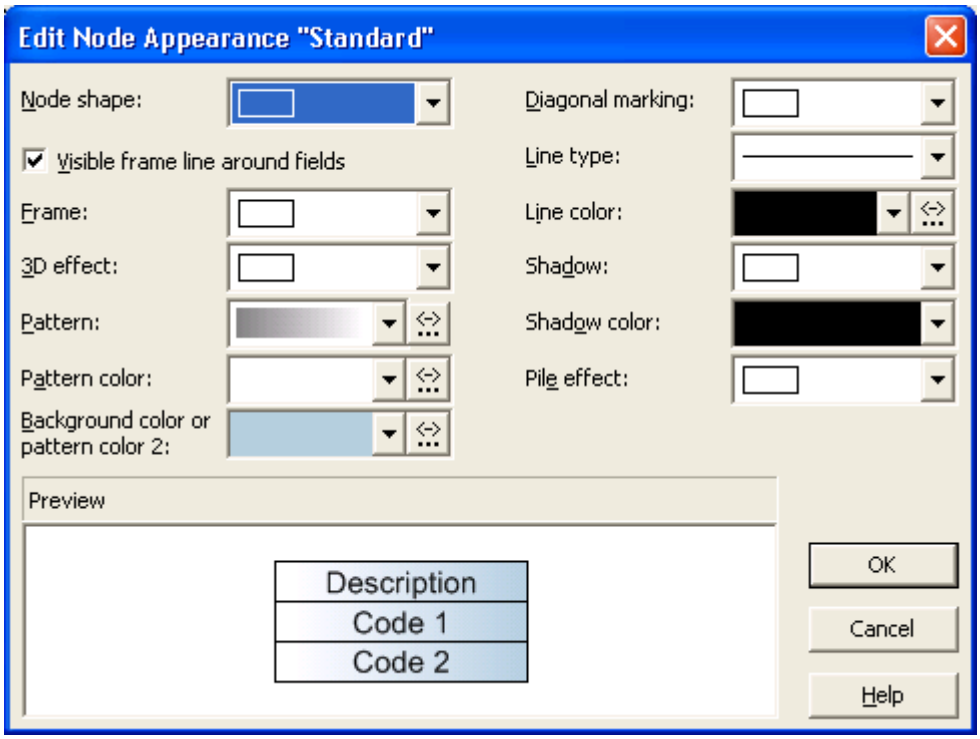
If several node appearances apply to a node, the attributes of each node appearance are replaced by the attributes of the node appearances that are processed later. Only the attributes whose value is "not specified" do not replace the attributes of their predecessors.

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



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.Frame-AroundFieldsVisible**.

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 indicates the default pattern.

 by the arrow button you can open the list of patterns and select a pattern.

 by the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent patterns.

 If a mapping has been configured, the arrow on the button will be displayed as filled.

Please note: If the background color of a field of a node format which was applied to the node appearance was not set to **transparent**, the selected pattern with its colors can not be seen!

Pattern color

This field lets you set the default color of the pattern.

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

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

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

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

Background color or Pattern color 2

This field lets you select a background color 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.

Please note: If the background color of a field of a node format which was applied to the node appearance was not set to **transparent**, the pattern selected above will remain invisible!

Diagonal marking

This field lets you specify whether 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 has been configured, the arrow on the button will be displayed as filled.

Shadow

This field lets you add a shadow to the nodes.

Shadow color

Select the color for the shadow or the pile effect.

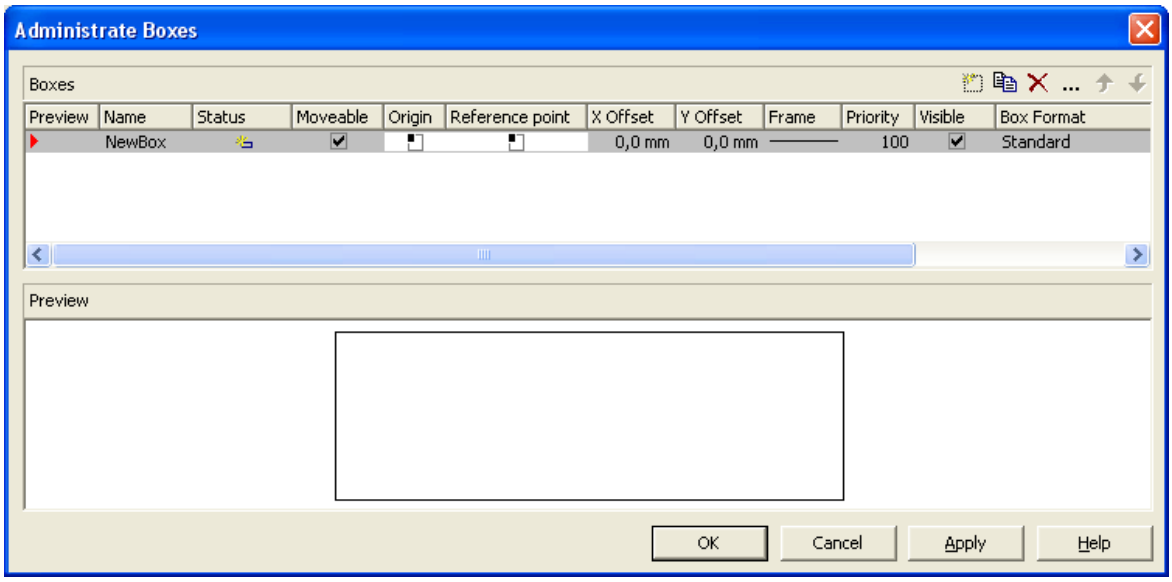
Pile effect

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



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

Preview

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

Name

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

Status

In the **Status** column each box that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Update behavior

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

Moveable

By moving a box, its offset will be modified. Activate this check box if the box is to be moveable in the diagram at run time. Deactivate the check box if

you have positioned a box correctly and do not want it to be moved at run time.

Origin

By the properties **Origin**, **Reference point**, **X Offset** and **Y Offset** you can position a box in the diagram area. The relative position of the boxes is independent of the current diagram size.

Specify the origin, i. e. the point of the diagram from which the offset to the reference point of the box will be measured. Possible values: top left, top centered, top right, centered left, centered centered, centered right, bottom left, bottom centered, bottom right.

Reference point

Set the reference point of the box, i. e. the point of the box from which the offset to the origin will be measured. Possible values: top left, top centered, top right, centered left, centered centered, centered right, bottom left, bottom centered, bottom right.

X Offset

Set the distance (in mm) between origin and reference point in x direction.

Y Offset

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

Frame

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

Priority

Specify the relative drawing priority of the box in comparison with the other objects in the diagram (nodes, grids, etc.). The priority of nodes is 0. If the priority of a box is higher than the priority of nodes, the boxes overlay the nodes so that an interactive access to the nodes won´t be possible.

Visible

Activate this check box if the box is to be visible at run time.

Box format

The current box format of the box is displayed here. If you click this field, two buttons will appear:

 From the combobox you can select a box format.

 by the **Edit** button you reach the **Administrate Box Formats** dialog box.

Add box

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

Copy box

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

Delete box

 The marked box in the list will be deleted.

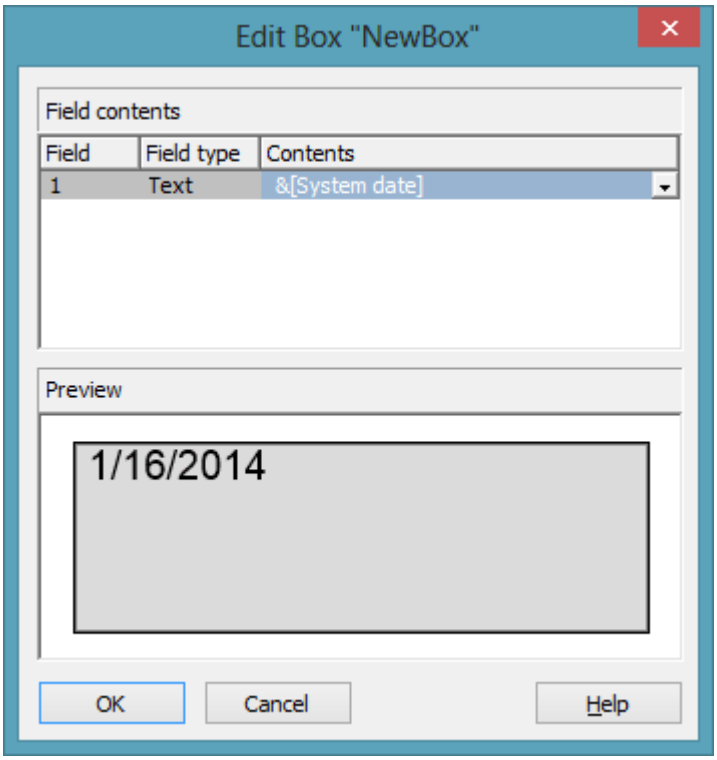
Edit box

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

Promote / demote box

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

4.19 The "Edit Box" Dialog Box



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

Field

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

Field Type

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

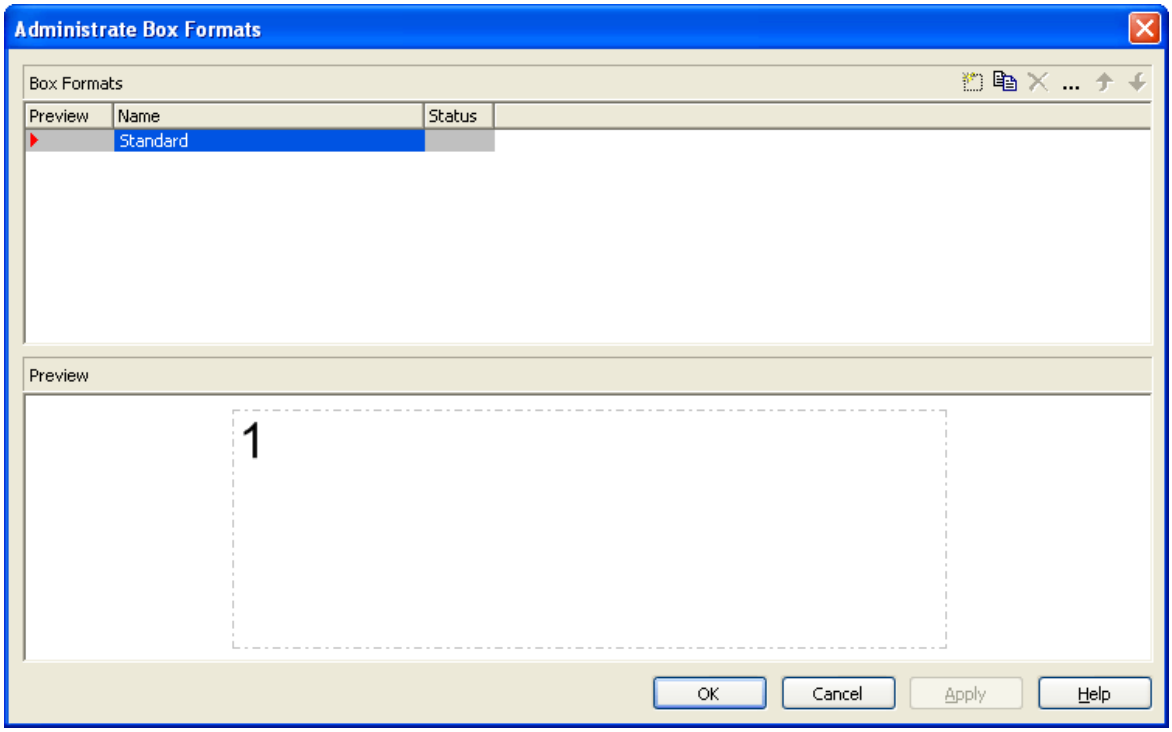
Contents

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

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

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

4.20 The "Administrate Box/Node Formats" Dialog Box



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

Preview

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

Name

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

Status

In the **Status** column each format that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Add box/node format

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

Copy box/node format

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

Delete box/node format

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

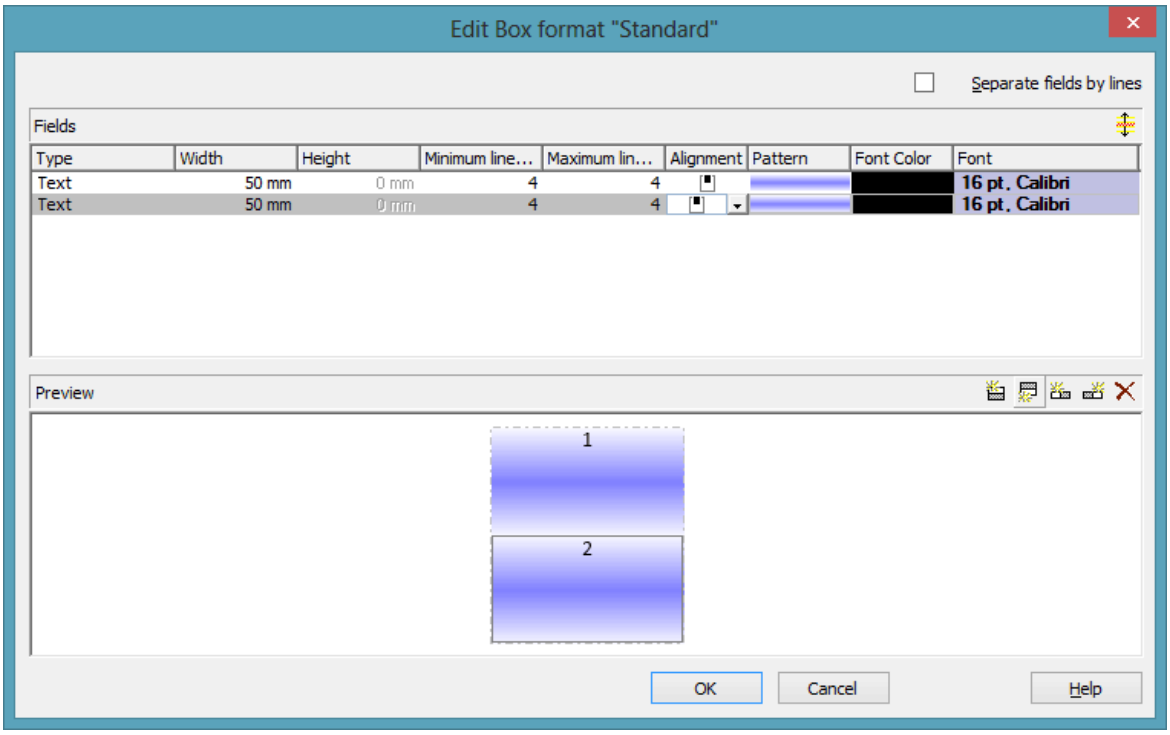
Edit box/node format

 The **Edit Box Format** or **Node Box Format** respectively dialog box will appear.

Promote / demote box format

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

4.21 The "Edit Box Format" Dialog Box



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

Separate fields by lines

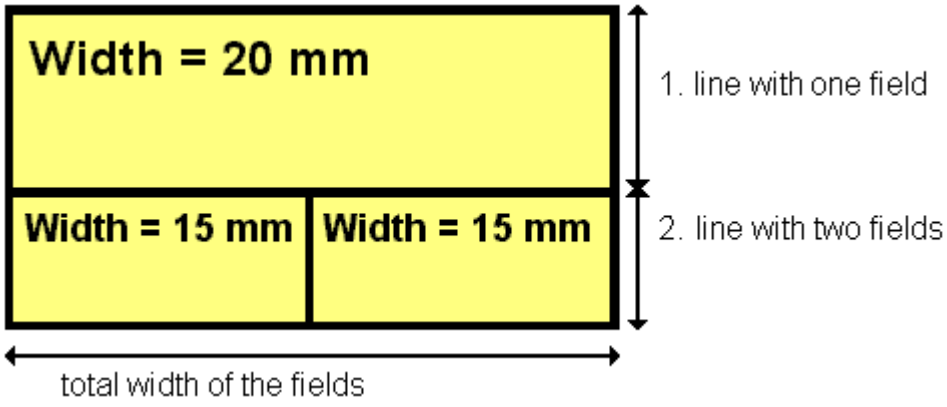
Activate this check box if the box fields are to be separated by lines.

Type

Select the field type: text or graphics.

Width

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



Height

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

Minimum/Maximum line count

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

Alignment

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

Pattern

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

Font Color

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

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

Font

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

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

Apply selected property to all fields

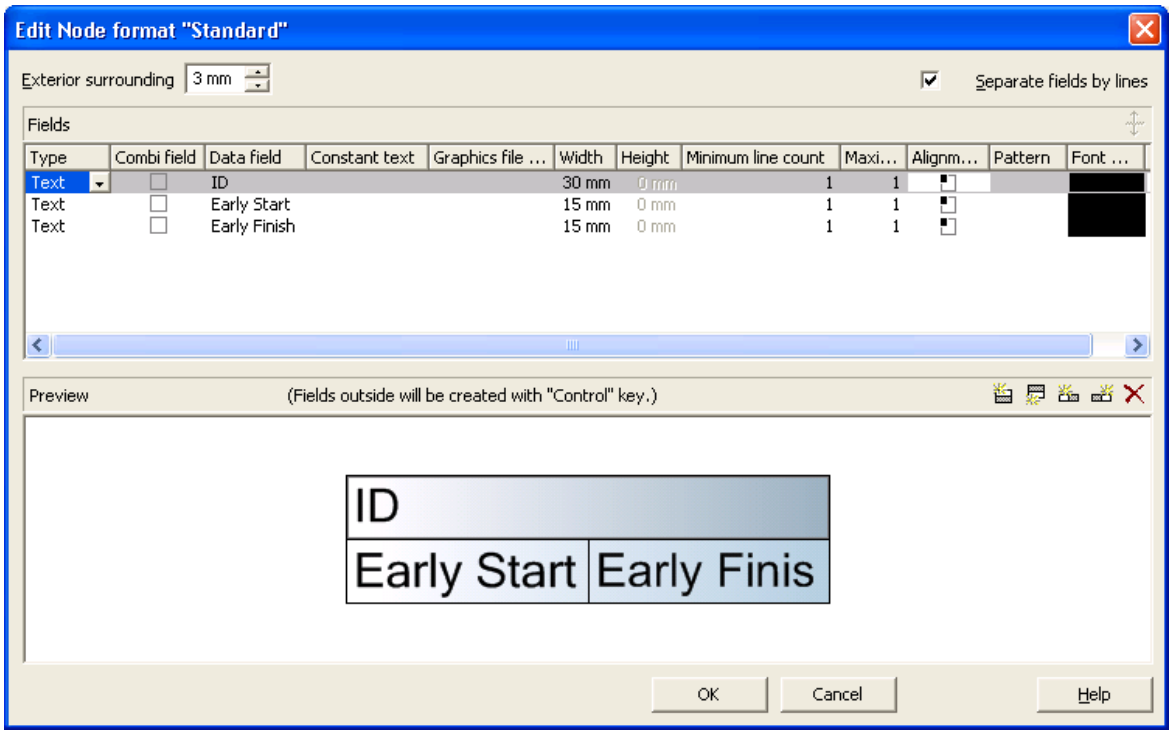
 Applies the marked property to all fields.

Preview

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

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

4.22 The "Edit Node Format" Dialog Box



This dialog box will open after clicking on the **Edit format** button of the **Administrate Node Formats** dialog box.

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.

Combi field

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

- **Type**: Text, **Combi field**: no: only text will be displayed (as specified for **Data field** or for **Constant text**)
- **Type**: Graphics, **Combi field**: no: only a graphics will be displayed (as specified for **Graphics file name**)
- **Type**: Text, **Combi field**: yes: text (as specified for **Data field** or for **Constant text**) and a graphics (as specified for **Graphics file name**) will be displayed
- **Type**: Graphics, **Combi field**: yes: only a graphics will be displayed (as specified for **Graphics file name**). Text (as specified for **Data field**) is visible only in a tooltip. If possible, it will be displayed as hyperlink.

Data field

Select the data field whose content is to be displayed in the current field. 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 ActiveX property **FilePath** first. If it won´t be found there, the file will be searched in the current directory of the application and in the installation directory of VARCHART ActiveX.

 Click this button if you want to use a map to display graphics in 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.

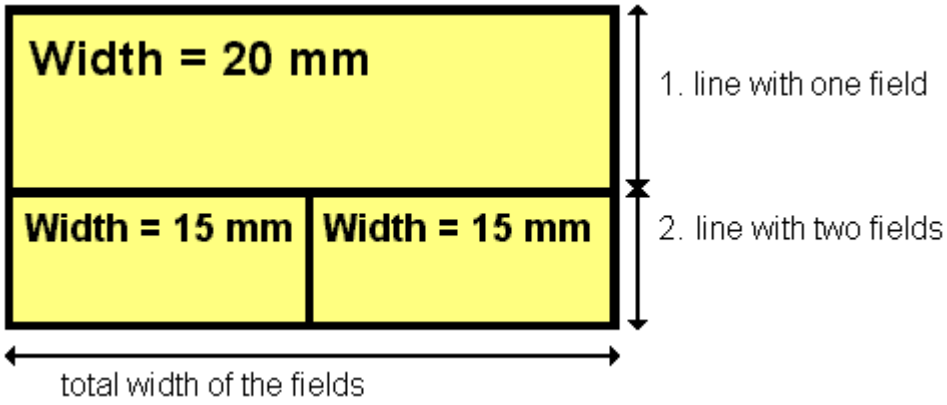
If a mapping has been configured, the arrow on the second button will be displayed in bold ( ).

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 reach the **Configure Mapping** dialog box. Here you can configure data-dependent font colors.

If a mapping has been configured, the arrow on the button will be displayed in bold ().

Font

Indicates the font style for the current field. If you click on the field, 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

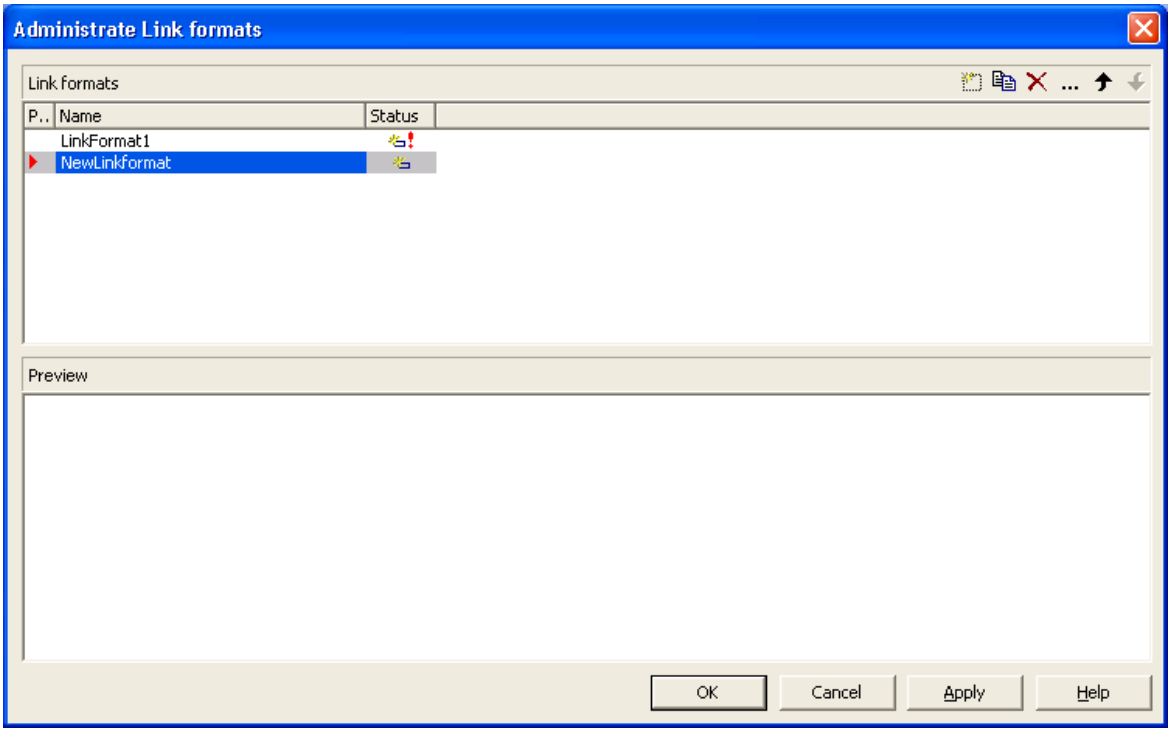
The current node format is displayed in the preview window. If you click on a field in the preview window you can modify its attributes in the **Fields** table.

 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



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 by a symbol.

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.

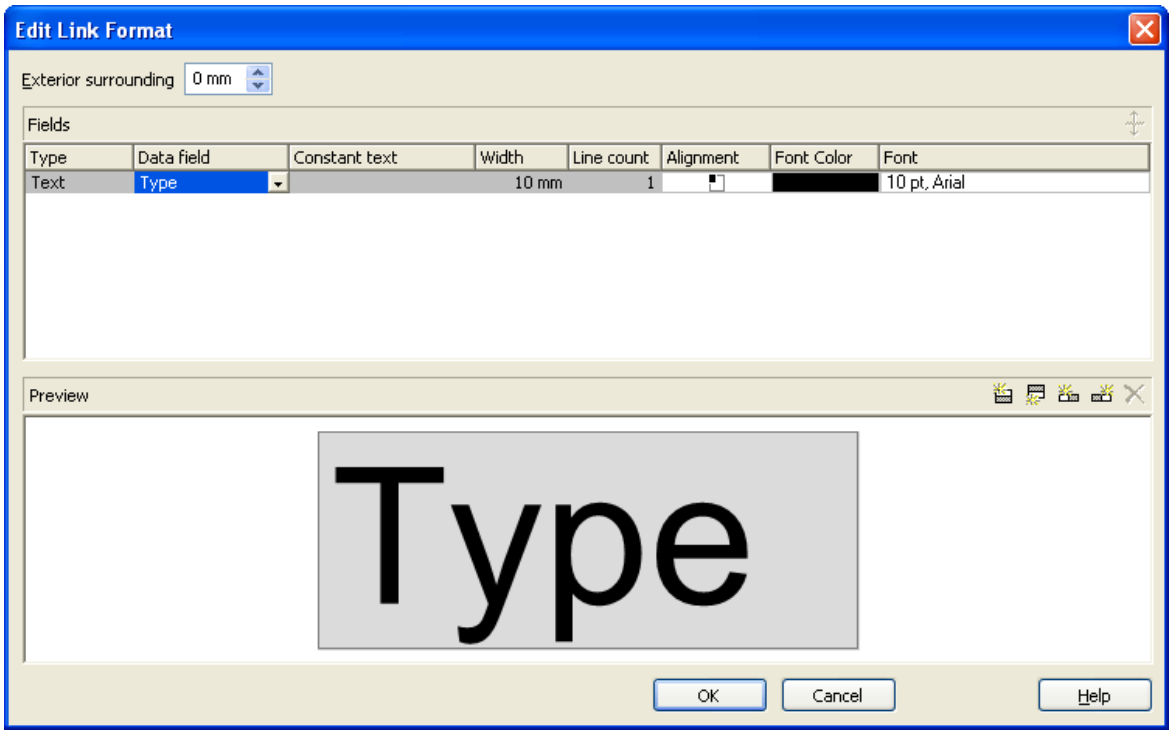
Delete link format

 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



This dialog box will open after clicking on ... of the **Administrate Link Appearance** dialog.

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 that you can open the Windows **Font** dialog.

Apply selected property to all fields

 Applies the marked property to all fields.

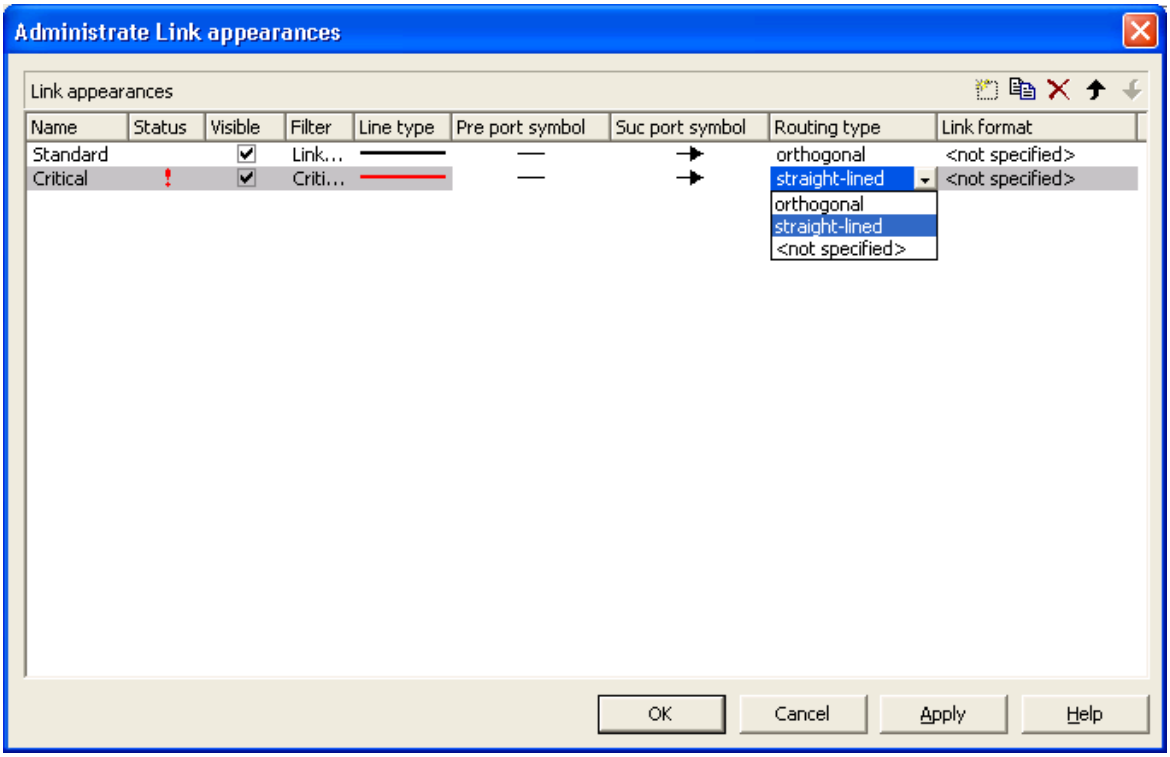
Preview

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



You can get to this dialog by clicking the **Link appearances** button on the **Objects** property page.

Name

This column displays the names of the link apperances available. The names can be edited.

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

Status

In the **Status** column each link appearance that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Visible

This check box lets you specify whether the links between the nodes should be displayed. This feature can be also set by the property **VcLink-Appearance.Visible**.

Filter

This column displays the filter used for a link appearance. From the select box you can select an appropriate filter.

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

Line type

Clicking on an entry in this column will cause an **Edit** button to occur, by which you can get to the **Line attributes** dialog box. There you can set type, thickness and color of the line.

This feature can also be set by the property **VcLink Appearance.LineType**.

Pre port symbol

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

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

Suc port symbol

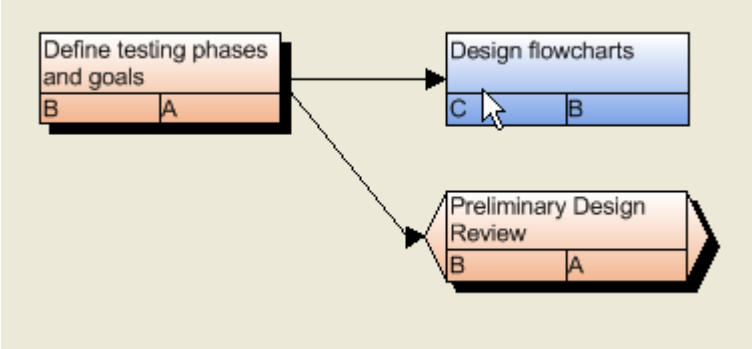
Select a port symbol for a link that visually accentuates the junction of the link and the successor node.

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

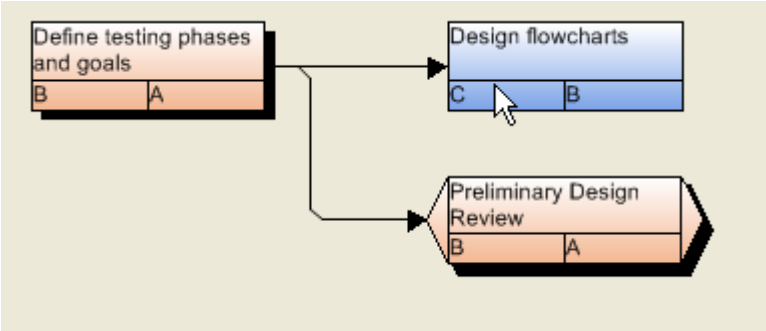
Routing type

This field allows to select a routing type. As the first row of the table containing the link appearance types is reserved for the default link appearance, the item <not specified> is selectable only from the second row on. If <not specified> has been selected, a routing type is used which is further up the list of the LinkAppearance objects.

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



Straight link type



Orthogonal link type

Link format

Click on  to select a link format or click on **sch-bearbeiten.gif 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.-FormatName>.**

Add link appearance

 **A new link appearance will be created. You can modify its default name by double-clicking and editing it.**

Copy link appearance

 **Copies the selected link appearance.**

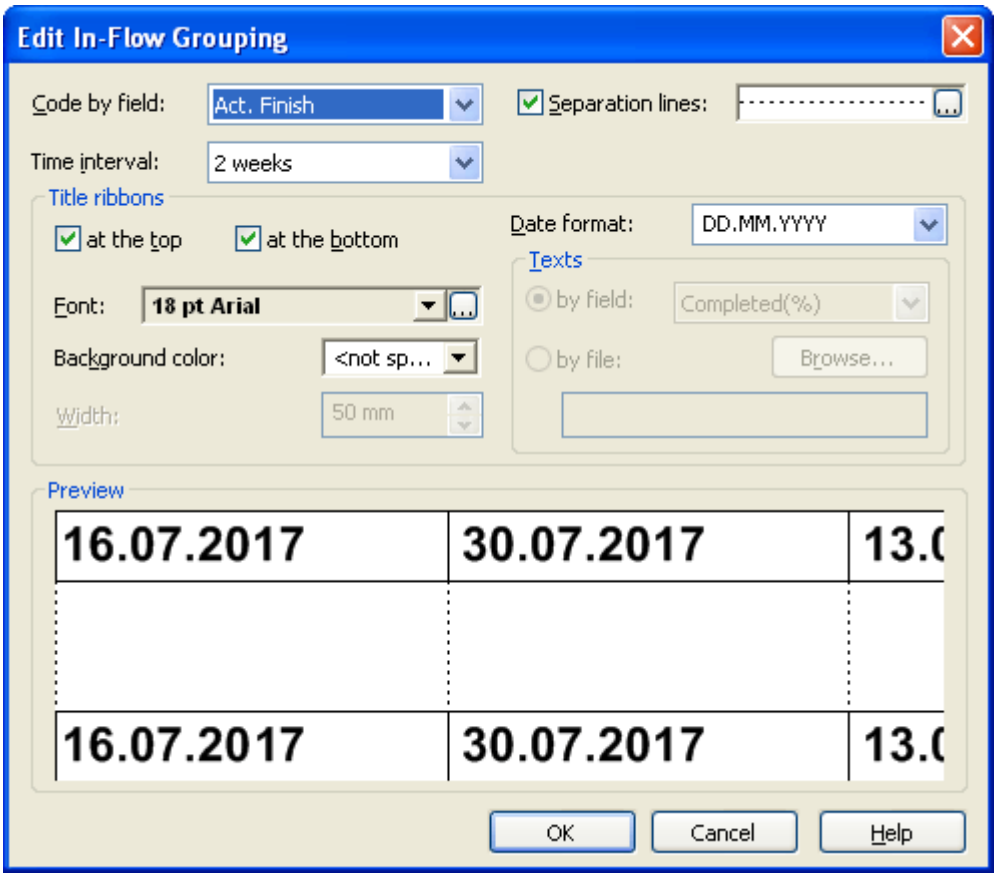
Delete link appearance

 **The marked link appearance in the list will be deleted. You can only delete link appearances that are not currently used.**

Promote / demote link appearance

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

4.26 The "Edit In-Flow Grouping" Dialog Box



You can get to this dialog box by the Nodes property page. In this dialog box you can define the criteria for in-flow grouping and for 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 reach 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

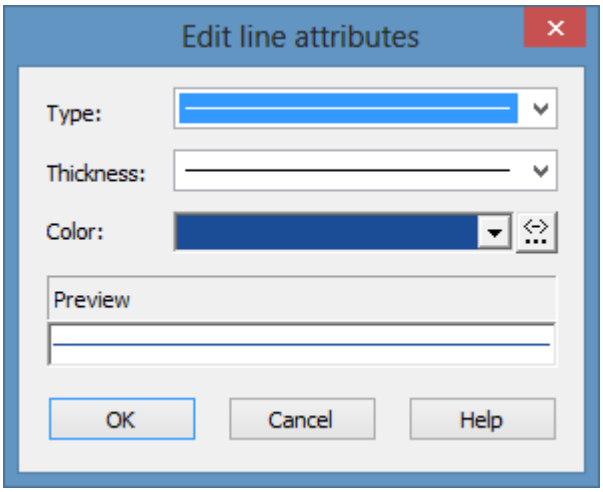
Select this option if you have selected a date field for **Code by field** and then specify the date format for the annotation of the ribbons.

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 and for box frames.

Type

Select the line type (dashed, dotted etc.).

Thickness

Define the line thickness.

Color

Select the line color.

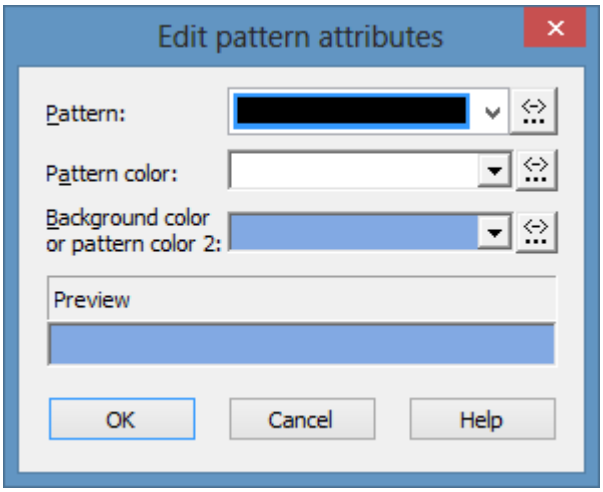
This button will open the **Configure Mapping** dialog box where you can specify the line color data-dependent.

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 box and node formats.

Pattern

Here you can select a fill pattern.

Pattern color

Select the foreground color of the fill pattern.

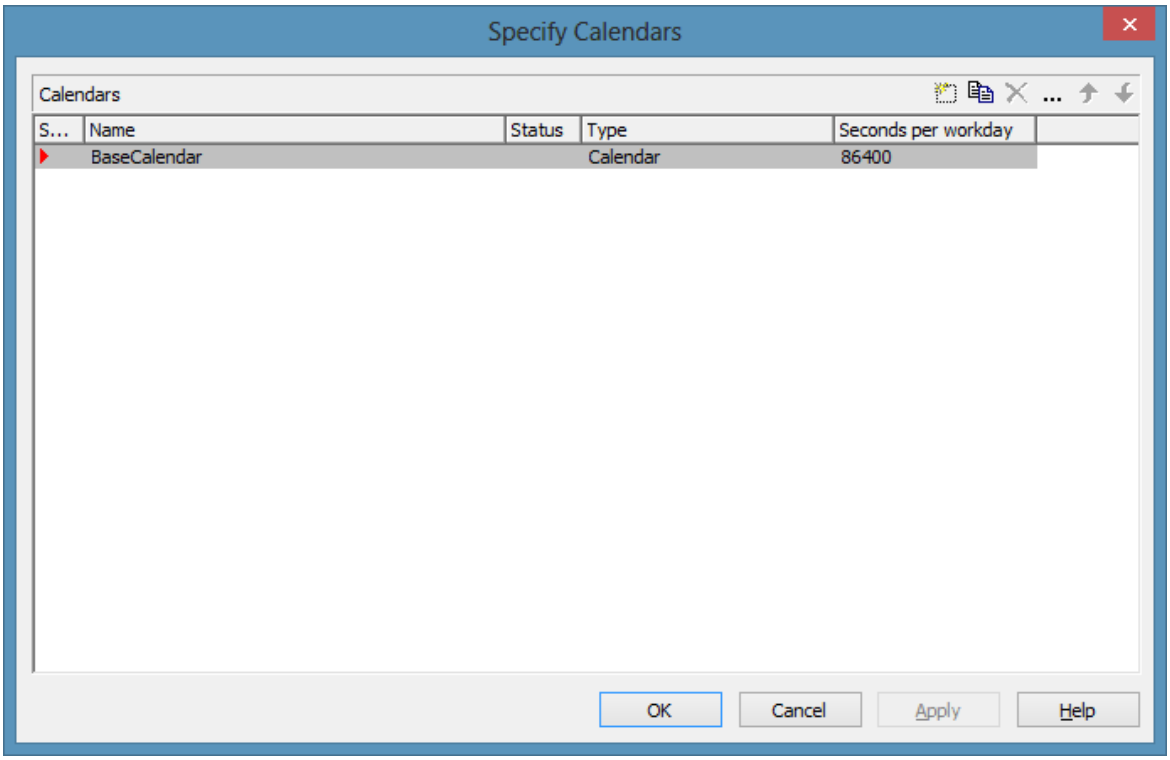
Background color or pattern color 2

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



You can get to this dialog box by the **Objects** property page. You can define a separate calendar for each line of the table.

Selected

The calendar marked by a small arrowhead in the **Selected** column is used for the calendar grid.

Name

Lists the names of all calendars defined.

Status

In the **Status** column each calendar that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Type

Specify the calendar type. Besides ordinary calendars shifts calendars are available, too.

Seconds per Workday

Specify how much seconds the workday has got.

Add calendar

 Click on this button to add a calendar.

Copy calendar

 The marked calendar is copied.

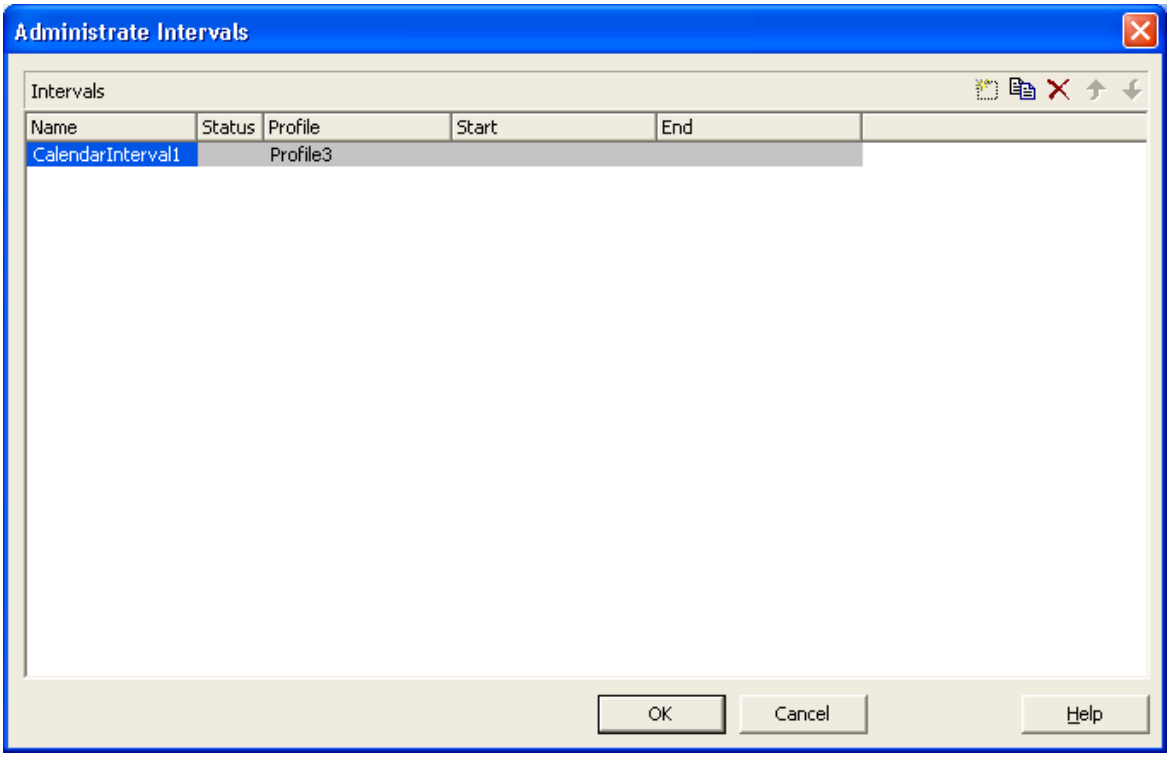
Delete calendar

 The marked calendar is deleted.

Edit calendar

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

4.30 The "Administrate Intervals" Dialog Box (Calendar)



In this dialog box you can create and modify intervals.

Name

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

Status

In this column each interval that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Profile

Here you can select a profile for your interval by clicking . If you want to edit the profile click on ... beside its name to open the **Administrate Calendar profiles** dialog.

Start/End

In this field you can set the beginning or end of of an interval. The date can be easily entered or modified by using the spin control.

Add interval

 A new interval will be created. You can modify the marked name by double-clicking and editing it.

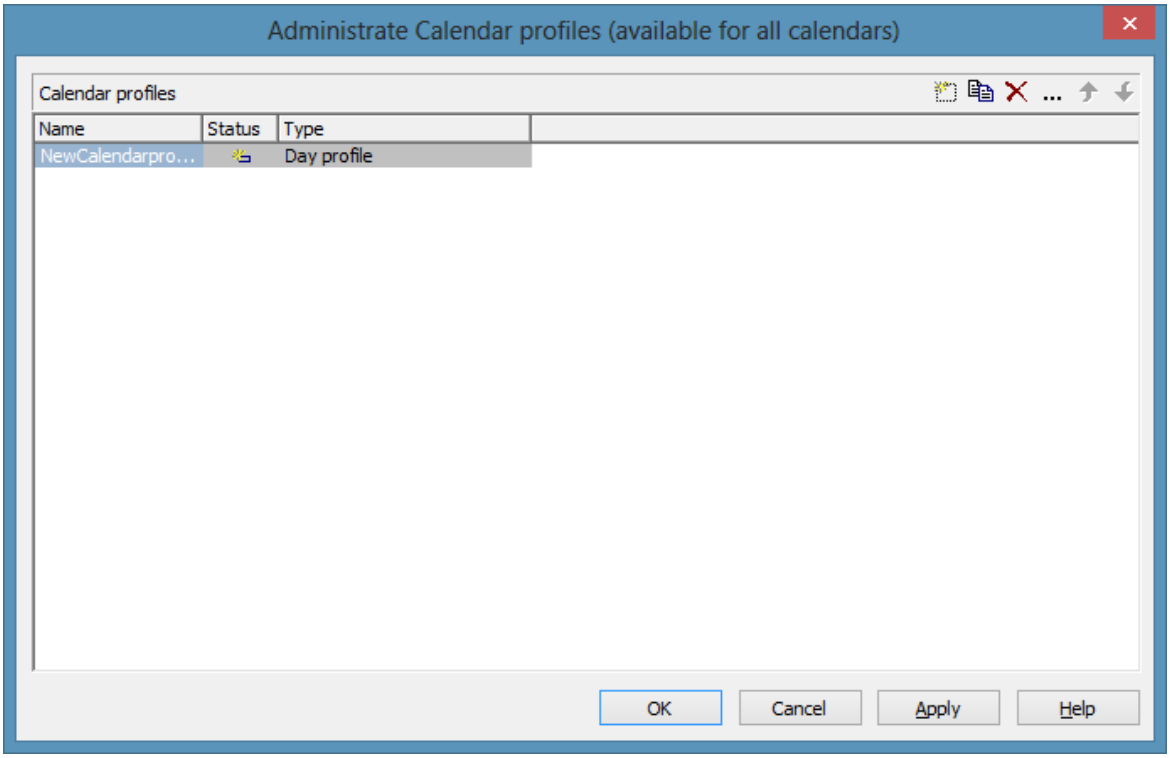
Copy interval

 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



In this dialog you can create and modify calendar profiles.

Name

Lists the names of all calendar profiles. All names can be edited.

Status

In this column each calendar profile that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Type

By clicking you can select the calendar profile type. You can choose between <Day profile>, <Week profile>, <Year profile> and <Variable profile>.

Add calendar profile

 A new calendar profile will be created. You can modify the marked name by double-clicking and editing it.

Copy calendar profile

 Click on this button to copy the marked calendar profile.

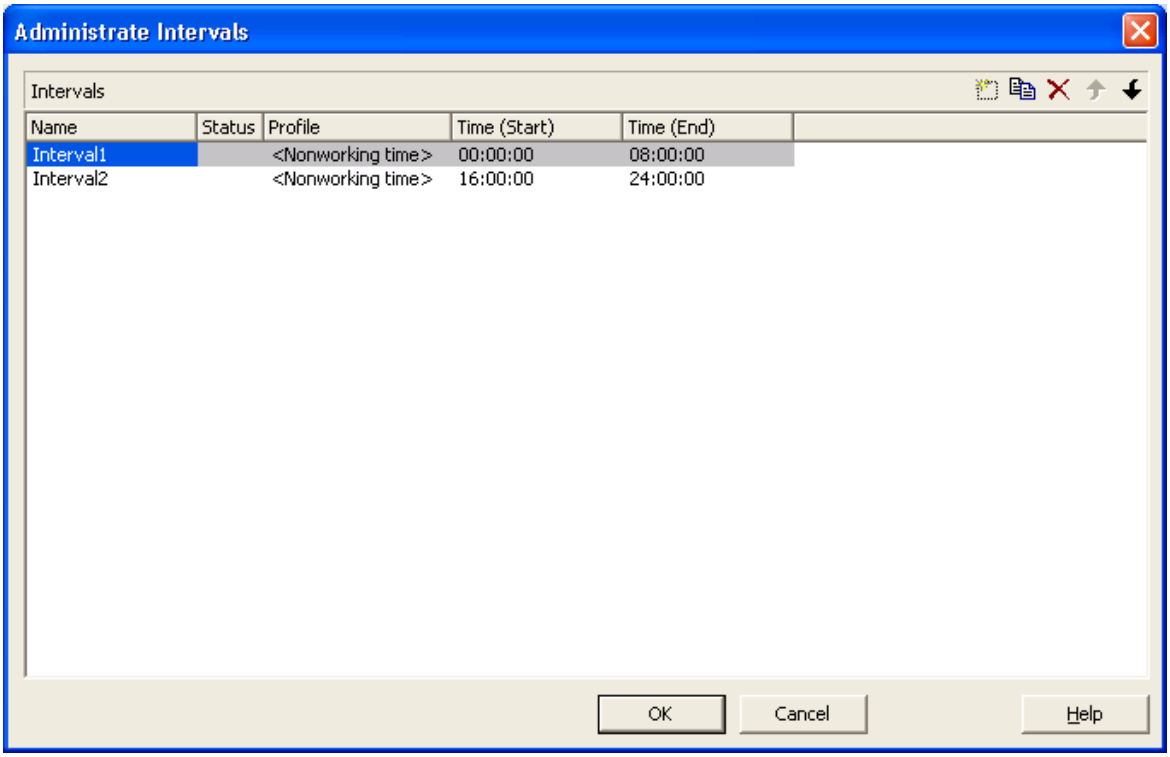
Delete calendar profile

 Click on this button to delete the calendar profile.

Edit calendar profile

 You will reach the **Administrate Intervals** (Calendar profiles) dialog box.

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



You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a day profile.

Name

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

Status

In this column each interval that has been added ( ) and/or modified ( ) since the dialog box was opened is marked by a symbol.

Profile

Here you can select a profile for your interval by clicking .

Time Start/Time End

In this field you can set the start or end time of an interval by clicking on the arrow buttons.

Add interval

 A new interval will be created. You can modify the marked name by double-clicking and editing it.

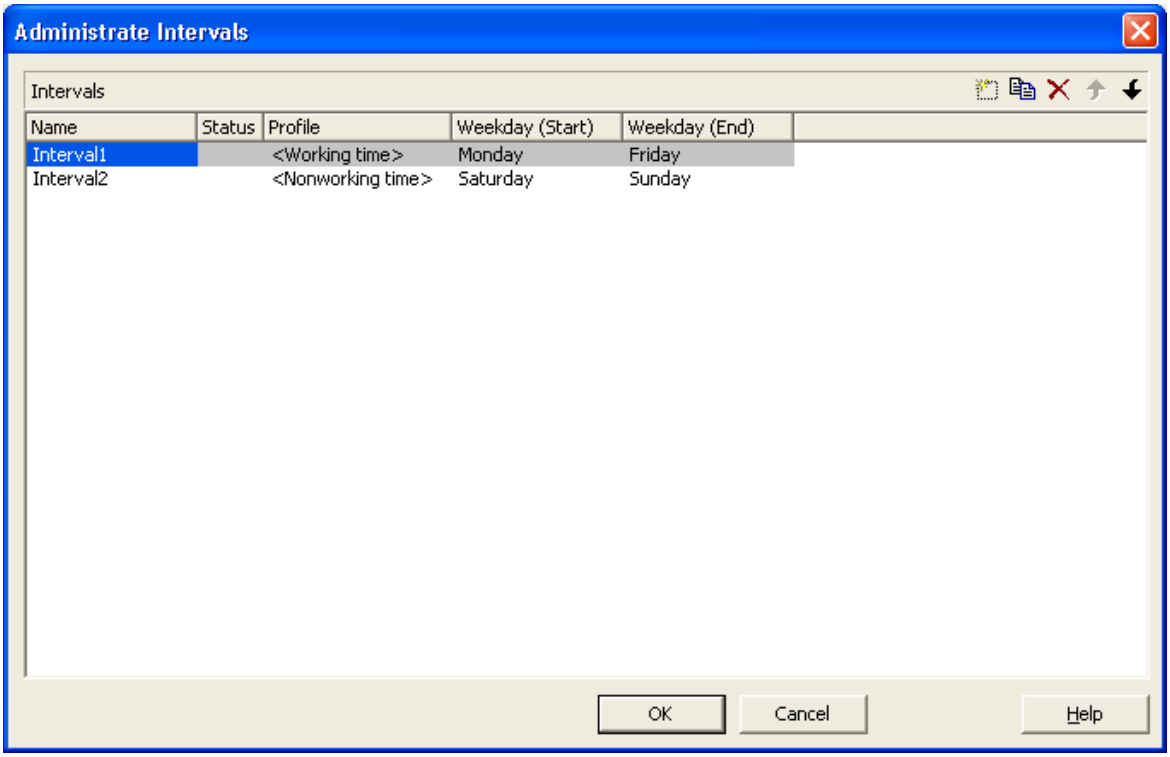
Copy interval

 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>)

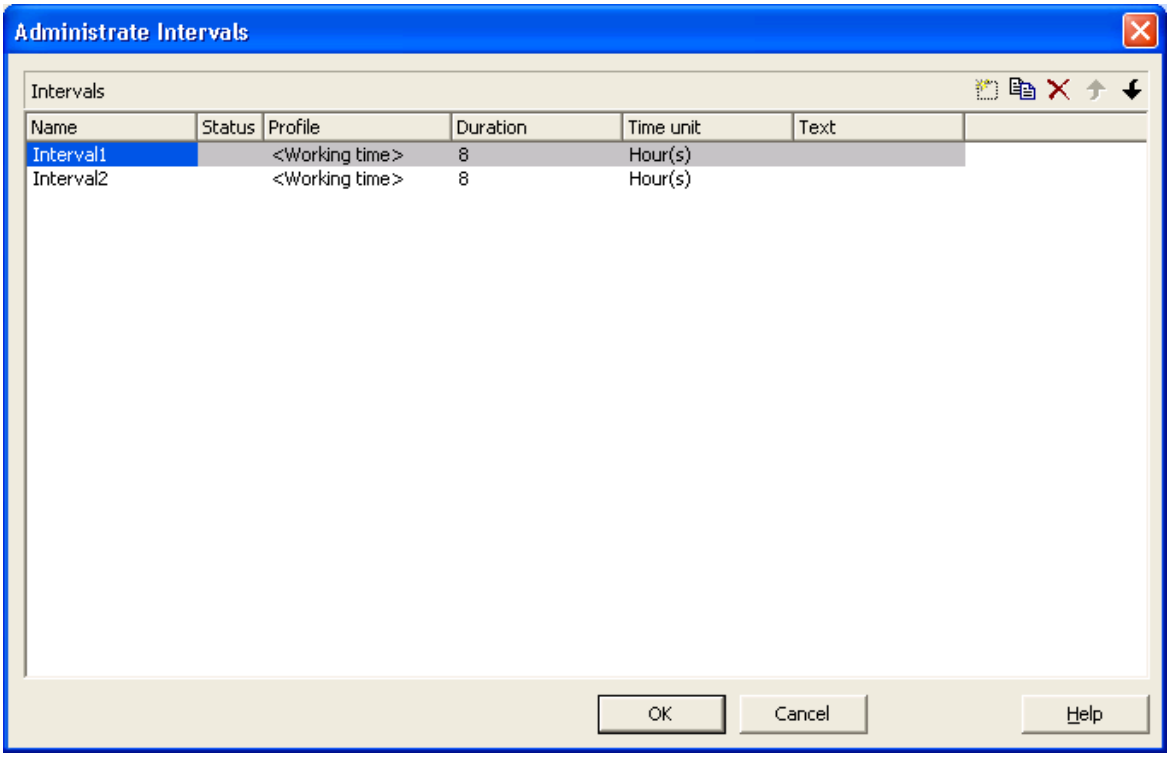


You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a week profile.

Weekday Start/Weekday End

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

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



You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a variable profile.

Duration

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

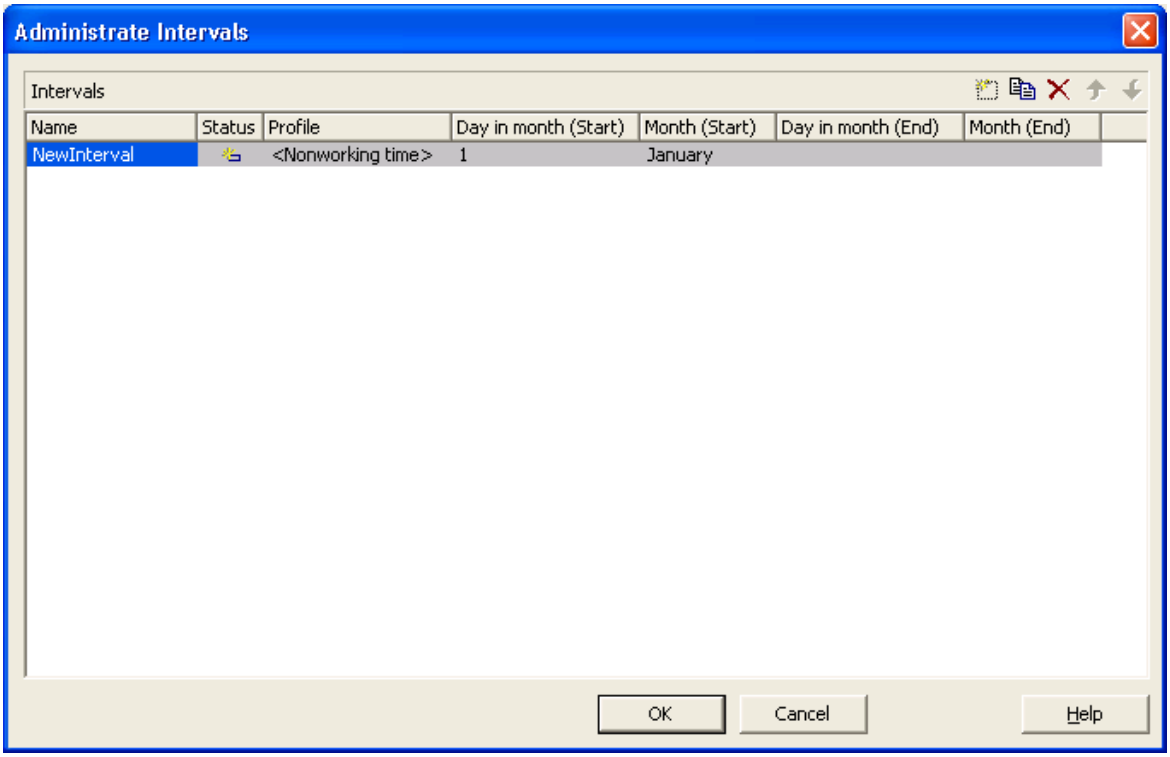
Time unit

Here you can specify the time unit of the interval. This feature can also be set by the property **VcInterval.TimeUnit**

Text

Here you can specify the text of the time ribbon This feature can also be set by the property **VcInterval.Text**

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



You can get to this dialog if you activate the dialog box "Admininstrate Calendar Profiles" on the "Objects" property page, and then click on the "Edit" button of the calendar profile. The different types of profiles offer different setting options. This dialog serves to create and modify intervals of a year profile.

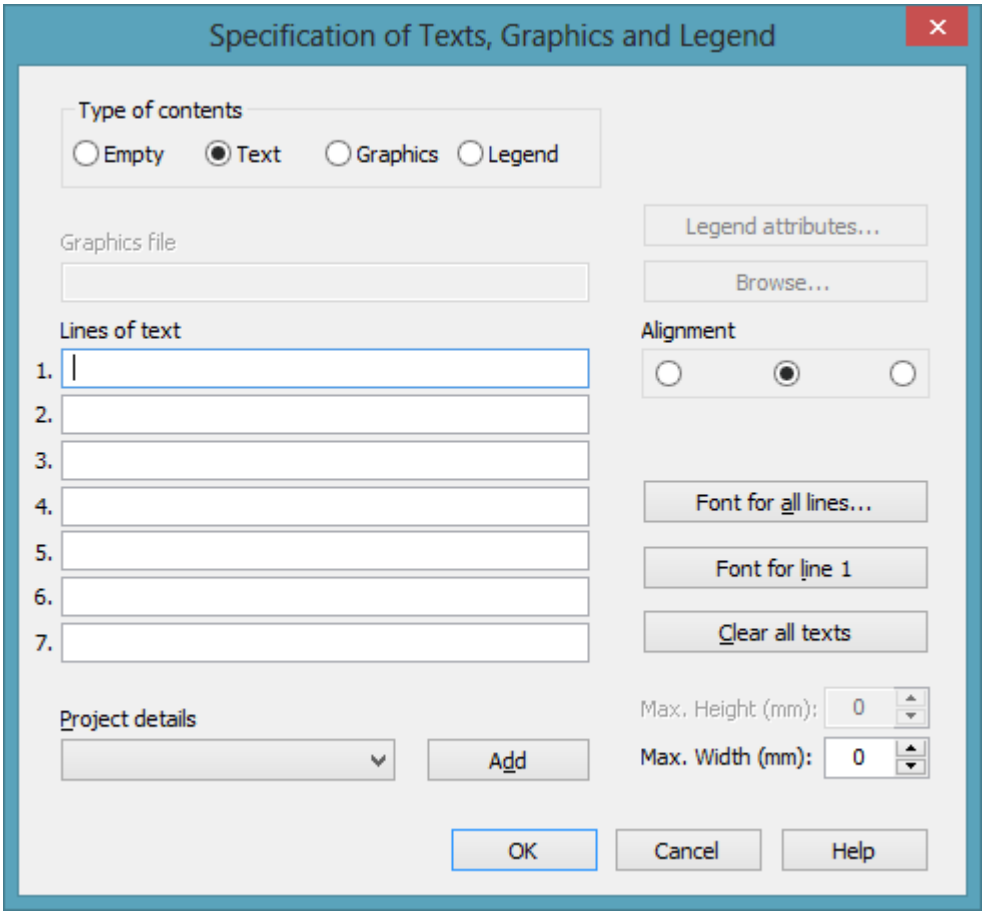
Day in month (Start)/Day in month (End)

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

Month (Start)/Month (End)

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

4.36 The "Specification of Texts, Graphics and Legend" Dialog Box



You can get to this dialog box if you click in the **Border Area** property page on one of the nine buttons above/below the drawing.

Type of contents

Specify the type of information that you want to display at the chosen location:

Empty: If you do not want to output anything at the chosen location, click on this flag.

Text: The text of the six text lines will be displayed at the chosen location.

Graphics: The graphics selected (by the **Browse** button) will be displayed at the chosen location. Graphics are always displayed in alignment centered.

Legend: A legend will be displayed at the chosen location. It describes the layers used in the current diagram.

Following your selection, the sections of the dialog box that are not required are deactivated (all entries are maintained).

Legend attributes

*Only activated when the check box **Legend** has been ticked.* You will open the **Legend attributes** dialog box where you can specify further attributes for the legend.

Graphics file

*Only activated when the check box **Graphics** has been ticked.* Select the graphics file you want to display by clicking on the **Browse** button or type the file name manually in the field. If the selected graphics file is not stored in the installation directory of the VARCHART ActiveX, you must also specify the drive and the directory.

Browse

*Only activated when the check box **Graphics** has been ticked.* Click on this button to reach the **Choose Graphics File** dialog box and select the drive, the directory and the name of the appropriate graphics file.

Lines of text

*Only activated when the check box **Text** has been ticked.* Specify the text (max. 6 lines) you want to display at the chosen diagram position and/or specify substitutes (e.g. &[System date]) to represent project info. If all six lines are empty, the area will not be displayed in the diagram.

Project details

*Only activated when the check box **Text** has been ticked.*

Here you can add several project details (number of pages, page number, system date) to your chart by selecting the appropriate place holder from the list and by clicking on the **Add** button.

The place holders will be replaced by the required data and will continuously be kept up-to-date in the print preview and the printout.

Add

*Only activated when the check box **Text** has been ticked.* When you have selected a project detail from the list, click on **Add** to confirm your choice. The project detail will be inserted in the line where the cursor is currently positioned.

Alignment of text

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

Font for all lines

*Only activated when the check box **Text** has been ticked.* You will reach the **Font** dialog box where you can specify the font attributes for all six lines. If you use this option to specify the font for all lines, the settings for the font for line 1...6 will be overwritten.

Font for line 1...6

*Only activated when the check box **Text** has been ticked.* To assign a different font to each of the six lines, click on this button. Depending on the line in which the cursor is currently positioned, the notation of this button will change to 1, 2, 3, 4, 5 or 6. You will reach the **Font** dialog box where you can specify the font attributes for each separate line.

Clear all texts

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

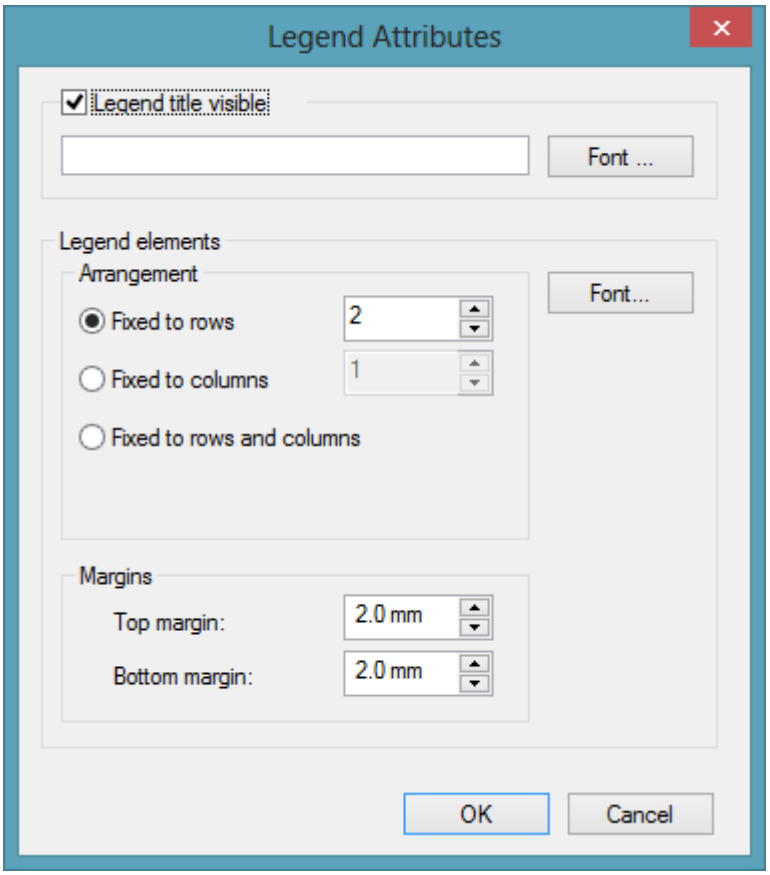
Max. Height (mm)

*Only activated when the check box **Graphics** has been ticked.* If you have specified several fields for text, graphics or legend, you can specify the max. height for the current field to prevent field contexts to be cropped.

Max. Width (mm)

*Only activated when the check box **Text** or **Graphics** has been ticked.* If you have specified several fields for text, graphics or legend, you can specify the max. width for the current field to prevent field contexts to be cropped.

4.37 The "Legend Attributes Dialog Box"



You can get to this dialog at runtime by clicking the corresponding item of the legend's contextmenu or at designtime by clicking the corresponding button in the dialog **Specification of Texts, Graphics and Legend**. The button can only be clicked after having selected **Legend** as **Type of contents**.

Legend title visible

Tick this check box if the legend title shall be displayed and enter a text. By clicking on **Font** you open the corresponding Windows dialog box which lets you specify the font attributes of the legend title.

Arrangement

- Fixed to Rows: Specify the number of rows to be displayed in the legend.
- Fixed to Columns: Specify the number of columns to be displayed in the legend.
- Fixed to Rows andColumns: Specify the number of rows and columns to be displayed in the legend. If the number entered here is lower than the existing layers, the surplus layers are not displayed.

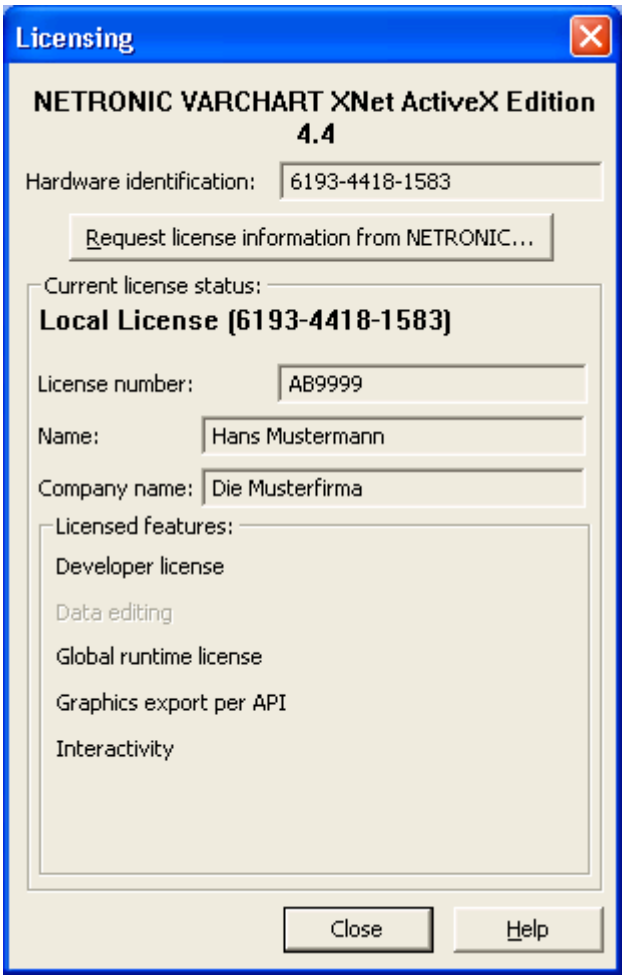
Margins

- Top margin: enter a value for the top margin of the element
- Bottom margin: enter a value for the bottom margin of the element.

Font

By clicking this button you open the Windows **Font** dialog box where you can specify the font attributes for the legend.

4.38 The "Licensing" Dialog Box



You can get to this dialog box by the **General** property page.

Before licensing, the program is automatically licensed as a trial version. Compared to the full version, the trial version is subject to restrictions: The trial period for testing the product is limited to 30 days. After this period, all diagrams will show a "Demo" watermark.

Hardware identification

(cannot be edited) The number that is indicated here is calculated by your hardware configuration. NETRONIC needs it for the licensing procedure. When you modify your hardware, you have to renew your licence. Please don´t hesitate to contact the technical support team of NETRONIC.

Request license information from NETRONIC

For licensing, click on this button. Then the **Request License Information** dialog will open.

License number/Name/Company name

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

Current license status

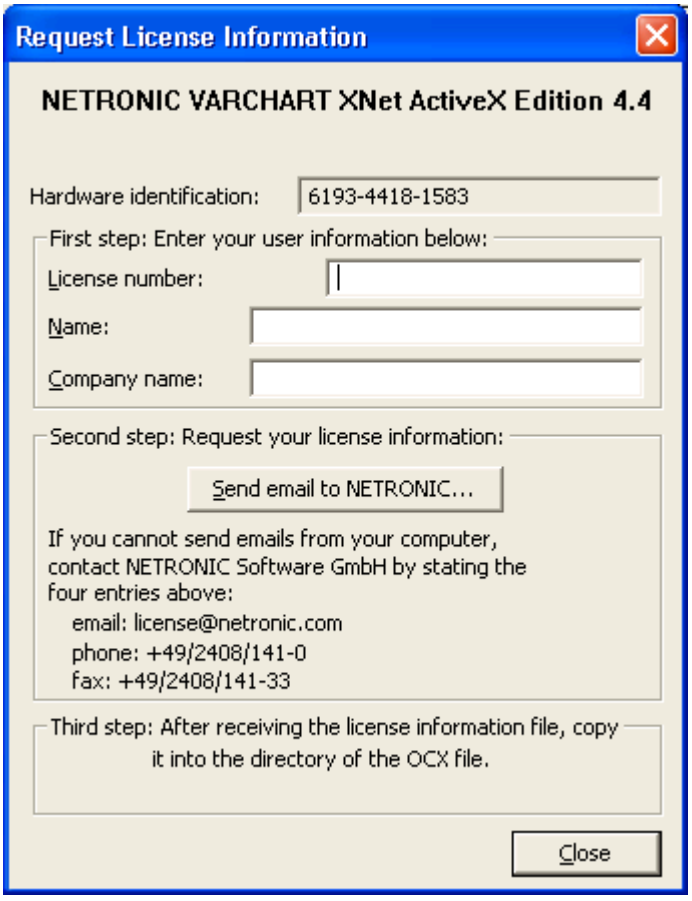
Indicates the modules that have been licenced. If the licencing procedure was successful, the licenced modules are activated.

- **Developer license**
- **Global runtime license** (the VARCHART ActiveX control runs in the runtime mode on each computer.)
- **Single-place runtime licenses** (the VARCHART ActiveX control has to be licensed individually on each computer to run on.)
- **Graphics export per API**
- **Interactivity**

Close

Quits the dialog box.

4.39 The "Request License Information" Dialog Box



Enter your license number, your name and the name of your company and click on **Send email to NETRONIC**. An email to NETRONIC will be generated automatically. As soon as we have received it, we will generate your license information file (**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.

- Navigation 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:

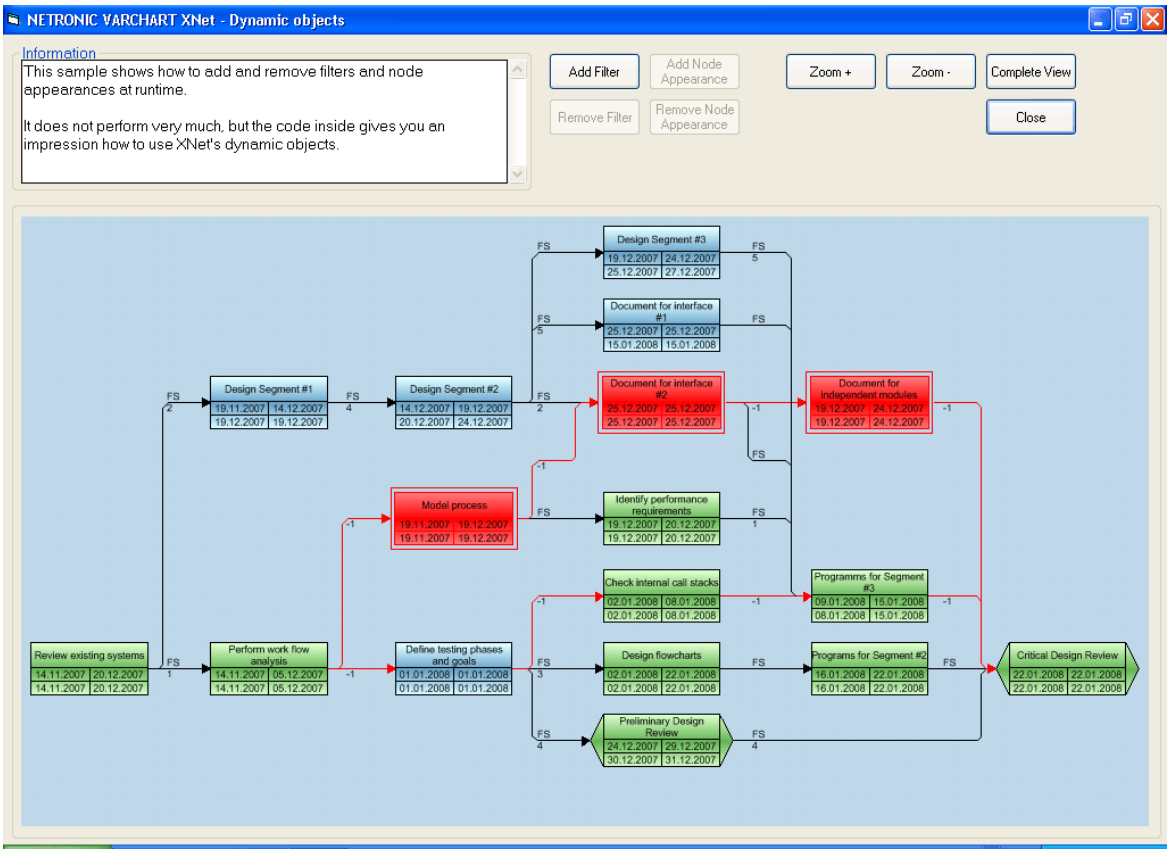
- **Ctrl** + **Num -**: zoom out
- **Ctrl** + **Num +**: zoom in

You can also use the mouse for zooming:

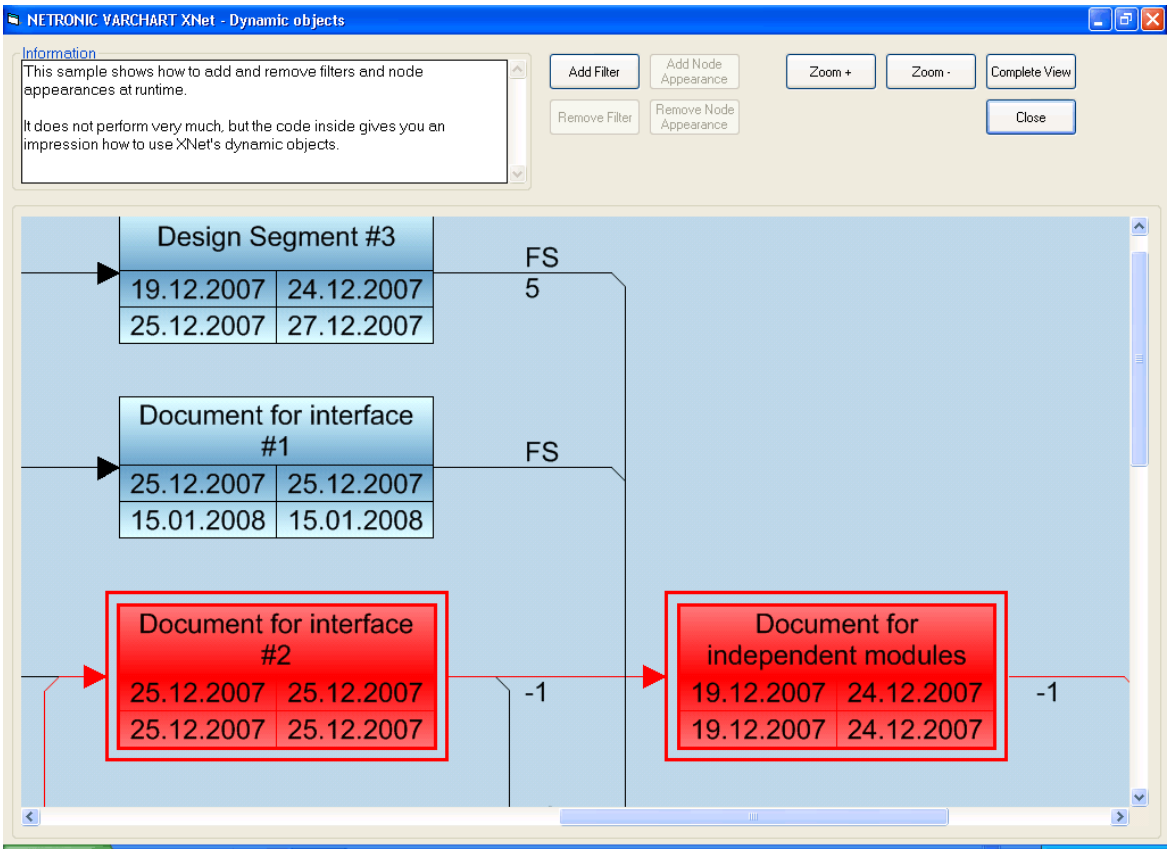
- Turn the mouse wheel while holding down the Ctrl key. For that purpose the usage of the mouse wheel for zooming has to be permitted. This can be done by ticking the **AllowZoomingByMouseWheel** box on the **General** property page or by setting the property **VcNet1.ZoomingPerMouseWheelAllowed** to **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".



Before zooming

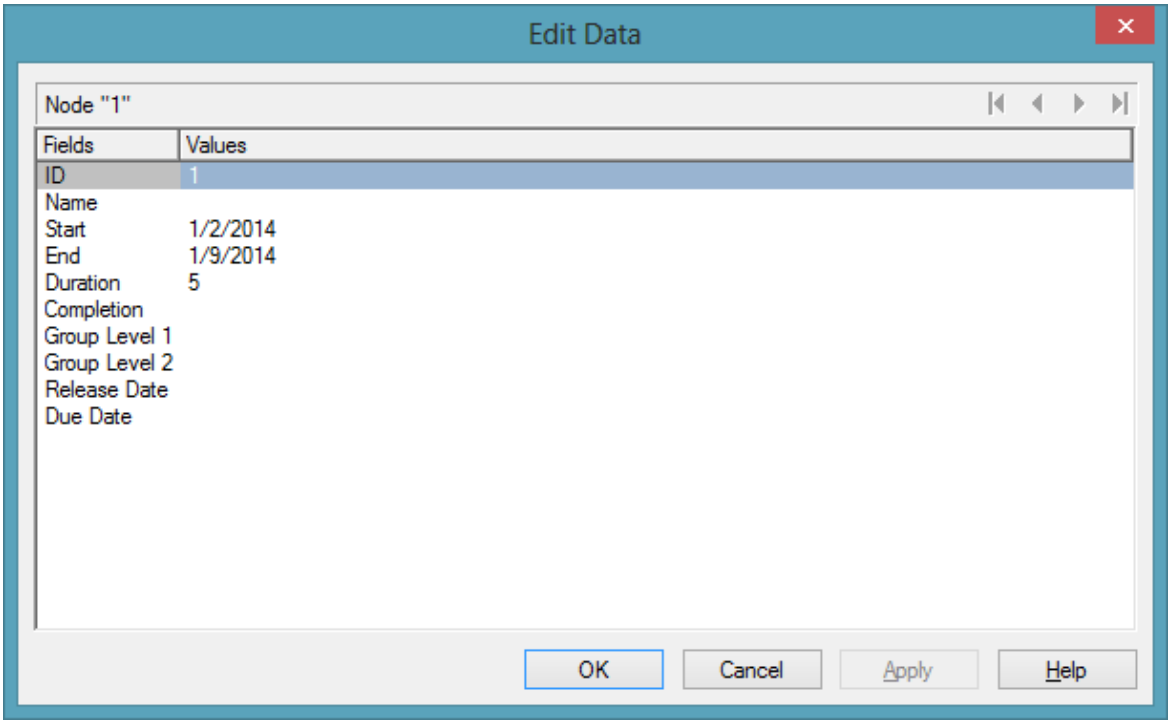


After zooming

5.4 Editing Node Data

In the dialog "Edit data" you can edit all node data. You open this dialog by either clicking on the **Edit** item of the corresponding context menu or by double-clicking on the node.

To edit several nodes, you mark the desired nodes and then click the **Edit** item of the context menu of one of the marked nodes to pop up the **Edit Data** dialog. Now you can edit the data of the marked nodes one after another



By double-clicking on a node, the event **OnNodeLDblClick** is triggered.

Modifiying a node interactively, e.g. by the **Edit Data** dialog, triggers the event **OnNodeModify**. By the **modificationType** parameter you get further information of the kind of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will be revoked.

Fields

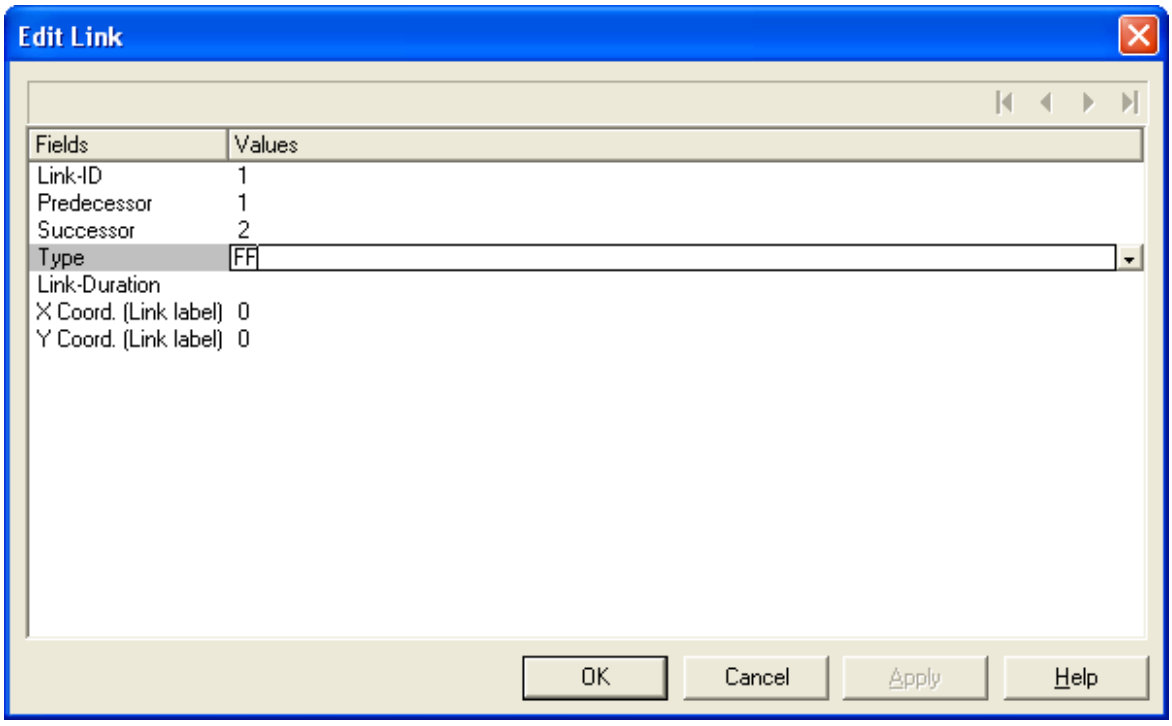
This column displays the data fields that define the marked node. The data fields available are the ones defined by the data definition in the **Administrate data tables** dialog. Only data fields that are **not** defined as **hidden** are displayed.

Values

This column lets you edit the values of the nodes marked, but only if they have been defined to be **Editable> in the Administrate Data Tables** dialog. If you edit a data field of the **Date/Time** type, a Date dialog will appear that you can select a date from.

The **Date Output Format** is defined on the **General** property page. When editing a field of the type **Integer** you can modify the value by a spin control that delivers the desired values via up and down arrows.

5.5 Edit Links



You can get to this dialog by double-clicking on a marked link (event **OnLinkLDblClick**). Here you can view and edit the data of the marked link. The ID of the link is indicated at the first position of the list.

Data fields

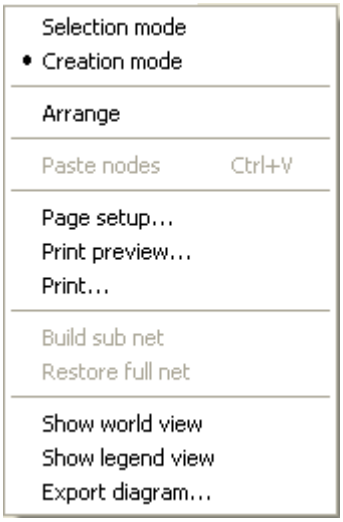
This column displays the data fields that define the marked link. The data fields available are the ones defined by the data definition. Only data fields that are not defined as **hidden** are displayed.

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.

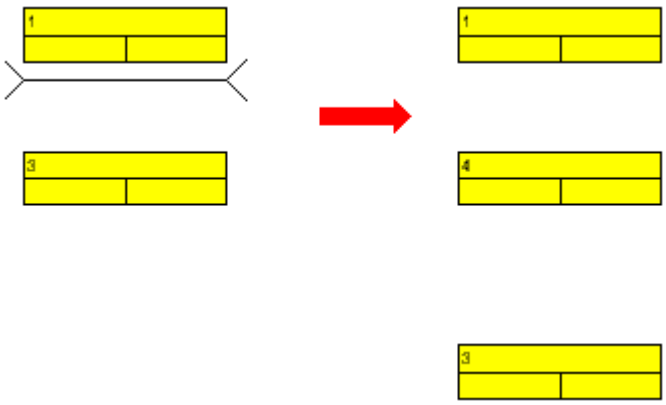
5.6 Creating Nodes and Links

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.

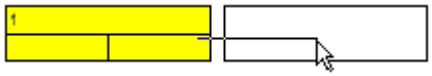


In Creation mode the cursor will transform into a rectangular frame. You can create a node by clicking on the left mouse key in an empty area of the diagram.

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.

5.7 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.

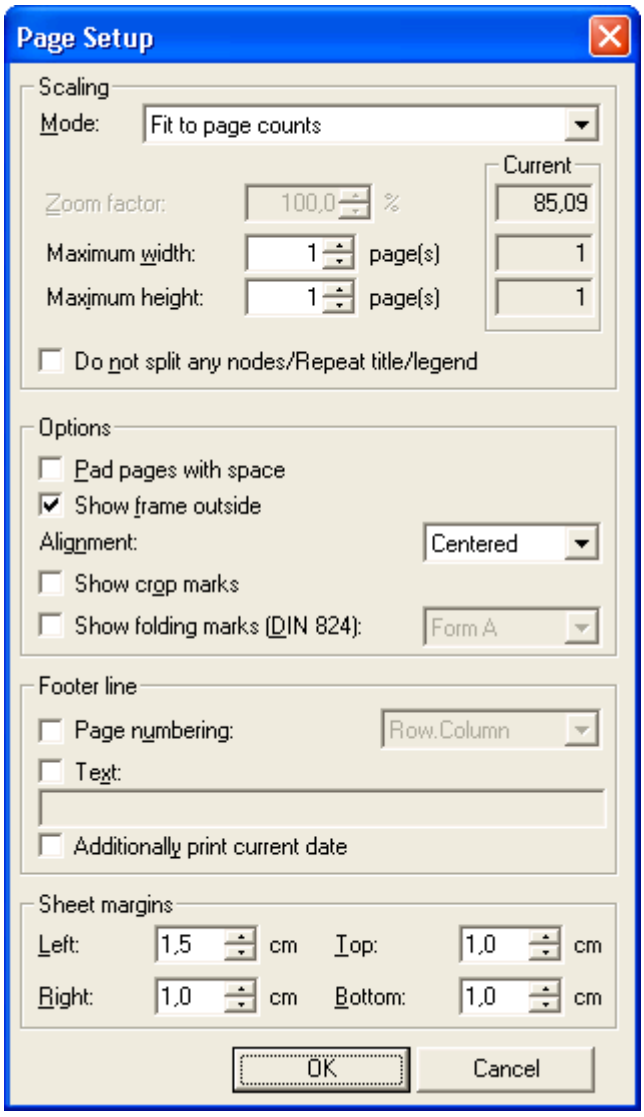
Beside, you can mark several nodes by dragging a framing rectangle around the nodes via the left mouse key.

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.

5.8 Setting up Pages

All settings concerning the page layout can be made in the corresponding dialog which can be opened either by clicking the **Page setup** item of the diagram contextmenu or by clicking the corresponding button in the **Print preview**.



Mode

By selecting a scaling mode from the drop down list and setting the corresponding values **Zoom factor** and **Maximum width/height** you specify a zoom factor for your output. After having clicked the **Apply** button, the values which result from your settings are shown under **Current**.

Zoom factor

100% is equivalent to the original size; a smaller value correspondingly reduces the size of the diagram, a greater value increases it.

Fit to page counts

By selecting this option you can specify the maximum number of pages, both heightwise and widthwise, into which the diagram may be split for the output (**Maximum width, Maximum height**. If necessary, one of the two values may be ignored in order to print the diagram as large as possible while preventing it from being distorted.

Do not split any nodes/Repeat title/legnd

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 folded drawing can be punched and filed away without flexi filing fastener
- **Form B:** slightly smaller so that a flexi filing fastener can be applied and together with the fastener the drawing corresponds to the width of DIN A4.
- **Form C:** the folded drawing is not to be punched but to be put in a sheet protector

The available folding marks can be displayed for every format, whereas the DIN 824 only mentions the formats DIN A0 to A3 explicitly.

Page numbers

If you tick this check box, a page number will be displayed in the bottom left-hand corner of each page. The following options are available:

- **Row.Column**: Useful for charts stretching across more than one page both heigthwise and widthwise. The vertical position of the page is displayed before the dot, the horizontal position after it.
- **Column:Row**: Useful for charts stretching across more than one page both heigthwise and widthwise. The horizontal position of the page is displayed before the dot, the vertical position after it.
- **Page/Count**: The current page number is displayed before the slash and after it the total number of pages: 1/6, 2/6 etc.

Text

Please tick this check box to set a text into the bottom left-hand corner of each page. If there is a page number, the additional text will be placed right of it.

For numbering the pages you may enter in **Additional text** the following place holders which will be replaced with the appropriate contents on the printout:

{PAGE}	= consecutive numbering of pages
{NUMPAGES}	= total number of pages
{ROW}	= line position of the section in the complete chart

{COLUMN} = column position of the section in the complete chart

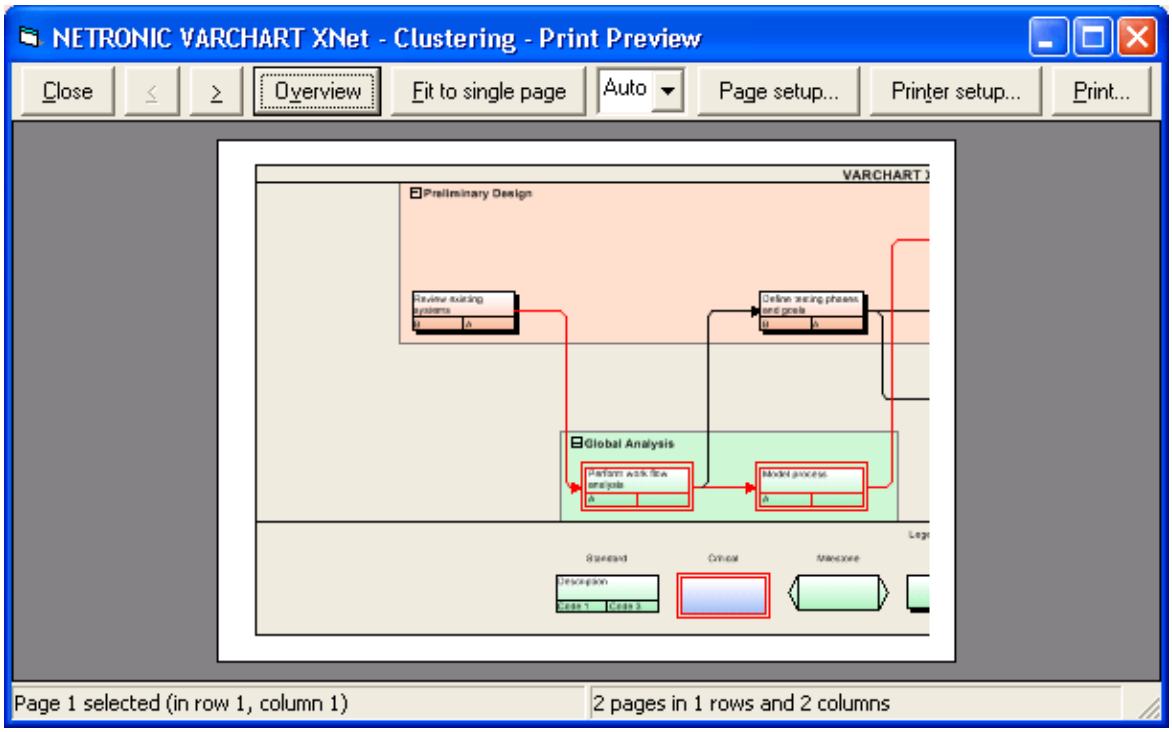
Additionally print current date

If you tick this check box, the printing date of will be displayed in the bottom left corner. If there is a page number or an additional text, the print date will be placed right of them.

Sheet margins

The fields **Top, Botttom, Left** and **Right** let you set the margin between the diagram and the edge of the paper sheet (unit: cm).

5.9 Print Preview



Before printing, you can view the diagram in the print preview where it will be displayed as defined by the settings of the **Page Setup** dialog and as it will be printed.

You can view single pages or an overview of all pages or you can zoom and print a certain section of your diagram interactively.

The status bar shows the total number of pages and their horizontal and vertical spreading. In the **Single Page** mode, also the number of the current page is shown.

Close

By clicking on this button, you will leave the page preview and return to your diagram.

<

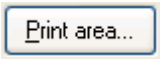
*Only activated when the **Single** button has been pressed*. If the diagram consists of more than one page, you can click this button to view the previous page. You traverse the pages horizontally starting from the bottom right and finishing at the top left page.

>

*Only activated when the **Single** button has been pressed*. If the diagram consists of more than one page, you can press this button to view the next page. You traverse the pages horizontally starting from the top left and finishing at the bottom right page.

Show Single Page/Overview

If the diagram consists of more than one page you can either view the pages one by one or in the overview. The overview shows all pages, their size depending on the total number of pages. The **Single Page** mode inititally shows the first page in full size, the buttons  and  allowing to browse through the pages. By double-clicking a page you can easily switch between the two modes **Single Page** and **Overview**.

If you want to zoom a certain section of your diagram, switch to the **Single Page** mode and with the mouse draw a rectangle around the desired section while holding down the left mouse button. As soon as you release the button, the selected section will be enlarged and can be printed by clicking the 

button that appears in place of the **Print** button. Please note that the zooming factor will not influence the scaling factor set in the **Page Setup** dialog.

Fit To Single Page

This button lets you scale down a multiple-page diagram to one page. The **Fit To Single Page** mode also allows to zoom a certain section as described under **Show Single Page/Overview**

Zoom factor

You can modify the size of the diagram by selecting a zoom factor from the list or by defining an individual one. This is only possible in the "Show Single Page" mode. To modify the zoom factor you can also use the scroll-wheel while holding down the <CTRL> key. The zoom factor it will not modify the size of the output. Depending on the selected zoom factor, vertical and/or horizontal scroll bars will be displayed. You can also use the mouse wheel to scroll vertically, holding down <Shift> to scroll horizontally.

The zoom factor **Auto** is the pre-set default and will always enlarge or downsize the sheet to the full size of the screen.

Page Setup

When clicking on this button, you will get to the dialog **Page Setup** to modify page settings.

Printer Setup

*Only visible if the check box **Use PrintDlgEx dialog** on the **General** property page has not been ticked*.

When clicking on this button, you will get to the Windows dialog **Printer Setup**, where you can modify printer settings.

Print/Print Area

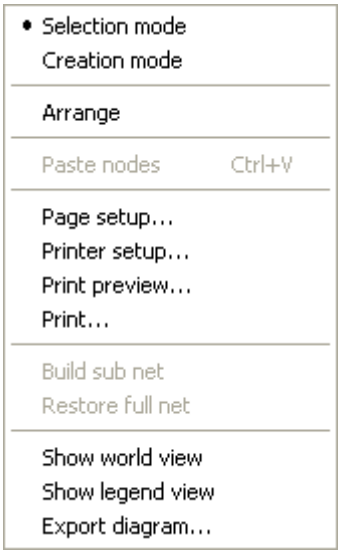
Click on this button to reach the Windows **Print** dialog box to start the print procedure.

If you have zoomed a section in the page preview, the button's label will change to **Print Area** and when you click it, the **Selection** radio button in the Windows **Print** dialog box will already be selected. If you click on **OK** the section displayed on the screen will be printed.

Please note that the zooming factor will not influence the scaling factor set in the **Page Setup** dialog.

5.10 The Context Menu of the Diagram

If you press the right mouse key after placing the cursor in an empty place of the diagram, the below context menu will appear:



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 **Allow creation of nodes and links** has been ticked.

The cursor 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 **Edit new nodes** 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 a node to a different one, a link will be created between them. If on the **General** property page the **Edit new links** 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 **vcCreateNodes-AndLinks**.

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 cursor.

Page Setup

The dialog **Page Setup** appears.

Printer Setup

*Only selectable if the check box **Use PrintDlgEx dialog** on the* <!eGeneral *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 selecting this menu item, you will get to the Windows dialog box **Save as**, that lets you save the diagram as a graphics file.

This dialog box also can be invoked by the VcNet method **ShowExport-GraphicsDialog**.

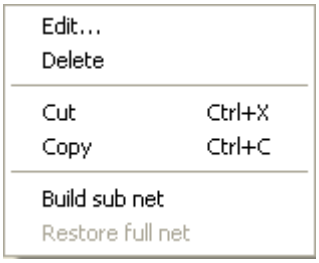
When exporting, the size of the exported diagram will be calculated this way:

- 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 coordinates between 0 and 10,000 while the shorter side uses correspondingly smaller values to keep the aspect ratio.
- EMF/EMF+: The total resolution is adopted, using coordinates scaled by 1/100 mm.

For further details on the different formats please read the chapter "Important Concepts: Graphics Formats".

5.11 The Context Menu of Nodes

If a node or several nodes have been marked and you press the right mouse key, the below context menu will appear:



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.

5.12 The Context Menu of Links

If you click the right mouse key on a link, the following menu will appear:

Edit

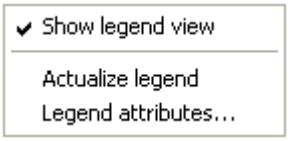
This menu item will pop up the dialog **Edit Link** where you can edit the data of the selected link.

Delete

To delete the marked link click on the **Delete** menu item.

5.13 Context Menu of the Legend

A right mouse button click on the legend will open the below menu:



Show legend view

This menu item lets you switch on or off the legend view.

Actualize legend

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

Legend attributes

With this item you open the corresponding dialog where you can specify the settings concerning legend title, legend elements and margins. For further information about this dialog please see chapter 4.44 "The Legend Attributes Dialog Box".

6 Frequently Asked Questions

6.1 How can I Activate the License File?

6.2 What can I do if Problems Occur during Licensing?

When you license a module for the first time or when you continue an expired license, please open the **Licensing** dialog box which you reach via the **General** property page. Click on the **Request** button. Then the **Request License Information** dialog will open.

Enter your license number, your name and the name of your company and click on **Send email to NETRONIC**. An email to NETRONIC will be generated automatically. As soon as we have received it, we will generate your license information file (vcnet.lic) and send 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. Please open a property page and make the system store it by making some change. This will activate the new license.

If during licensing of the VARCHART ActiveX control you receive an error message "REGSVR32 Error Return: 0X0000007e", the file *vcwin32u.dll* does not exist or is not stored in a directory indicated in the PATH. If the file does not exist, please contact the support of NETRONIC Software GmbH.

6.3 How can I Make the VARCHART ActiveX Control Use a Modified .INI File?

Some of the VARCHART ActiveX control's settings cannot be modified on the property pages. Still, you can adjust them via the *.ini file:

1. Open the **General** property page. The **Configuration file** field shows the current configuration file (for example *project.ini*).
2. Click on the **Browse** button. The dialog **Load/Save** will open. Please enter a file name into the **Temporary data file** field to be used as a temporary dummy configuration file, such as *dummy.ini*. Click on **Save**.
3. Now click on the **OK** or **Apply** button of the **General** property page. The configuration file *dummy.ini* will automatically be generated and applied.
4. Now you can edit your *.ini file (e.g. *project.ini*) in a text file editor and save your changes.
5. Then reset the true configuration file by selecting the former file (*project.ini*) on the **General** property page in the **Configuration file** field and click on **OK**. Your modified *.ini file is being used from now on.

6.4 What Borland Delphi Users Need to do on Upgrading a New VARCHART XNet Version.

After the upgrade or update of the VARCHART XNet to a higher version it is necessary to install the new version to the Delphi Package Borland User Components. Please proceed as described below:

1. Start Borland Delphi.
2. Click onto **Components** and **ActiveX import**.
3. Select *NETRONIC VARCHART XNet* from the ActiveX Controls list and click onto the **Remove** button to remove the registration. Quit the dialog by **Cancel**.
4. Now open the **Components > Install packages** dialog. Select the package *Borland User Components*. (This package is stored in the file *dclusr*0.bpl*. The '*' in the file name depends on your Delphi version: 5, 6 or 7.)
5. Click on **Edit**. The file *dclusrX0.dpk* will open.
6. Select the files *VcNetLib_TLB.pas* and *VcNetLib_TLB.dcr* succeedingly and remove them from the project by clicking the right mouse button.
7. Compile the package and close the dialog. This way the changes will be saved in the project *dclusrX0*.
8. Now re-open the dialog **Components > ActiveX import**.
9. Click on **Add**, select *vcnet.ocx*, and click on **Open**. *NETRONIC VARCHART XNet* will re-appear again in the list of the registrated ActiveX controls.
10. Click on **Install...** to recompile the package *dclusrX0.bpl*.
11. Quit the dialog to save the project to *dclusrX0*.

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 **Allow new nodes** on the **General** property page.

Check if the VARCHART VcNet property **AllowNewNodesAndLinks** 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, causes may be of different kind:

1. Please verify if on the property page **General** the check box **Allow creation of nodes and links** 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 if 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 **Allow creation of nodes and links** on the **General** property page.
2. You can set the return status of the event **OnNodeCreate** 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

```
'switching off the context menu of diagram
Private Sub VcNet1_OnDiagramRClick(ByVal x As Long, ByVal y As Long, _
                                   returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

'switching off the context menu of links
Private Sub VcNet1_OnLinkRClickCltn(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_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                   ByVal location As _
                                   VcNetLib.LocationEnum, _
                                   ByVal x As Long, _
                                   ByVal y As Long, _
                                   returnStatus As Variant)
   returnStatus = vcRetStatNoPopup
End Sub
```

6.9 What can I do if Problems Occur during Printing?

If printing of your diagram is impossible or if you cannot set up the printer, please verify whether the file *vcprct32.dll* exists. Also, please verify if the file can be located by the PATH settings, and if the Windows default printer has been set up.

If the file *vcprct32.dll* does not exist, please contact the support of NETRONIC Software GmbH.

6.10 How can I Improve the Performance?

> SuspendUpdate

Projects that include a large number of nodes may take too long if updating actions are repeated for each node. Not every automatic update procedure is necessary; in those cases you can suspend single updates, work off a sequence of code and then do a final update. Suspending and re-activating updates both can be done by the method **SuspendUpdate**, which is set to **True** at the beginning of the code sequence and to **False** at its end. Using this method can im improve the overall performance considerably.

Example Code

```
VcNet1.SuspendUpdate (True)

   If updateFlag Then
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.98" Then
            node.DataField(13) = "X"
            node.UpdateNode
            counter = counter + 1
         End If
      Next node
   Else
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.98" Then
            node.DataField(13) = ""
            node.UpdateNode
            counter = counter + 1
         End If
      Next node
   End If

VcNet1.SuspendUpdate (False)
```

You can also accelerate the updating procedure of links via the **Suspend-Update** 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.

6.11 Error Messages

> Error messages at runtime caused by the developer

Error Reason	Message
License failure	This is an unlicensed version of *. Please contact NETRONIC for a licensed version.
	The licensing failed. Please contact NETRONIC.
	The expiry date is exceeded. Please contact NETRONIC.
	Your identification has changed from * to *. Please contact NETRONIC!
	The ActiveX Control * used in this program has no runtime license!
ActiveX installation incomplete or older versions of a DLL in the system path	DLL * not found
	Loading the interface with identifier * failed
	The interface DLL (version *) is too old. This program needs version * or above.
Program installation incomplete or absolute path is erroneous	Group titles file not found
	The file * is not a valid graphics file.
	Graphics file not specified or not existent.
Error at assignment of a new INI file	The configuration file * was not found, program creates it using the default configuration.
INI file has errors	The highlight/table/layer * uses the non-existent filter *. The filter entry is corrected to <always>.
	The highlight/table * uses the non-existent node annotation *. The node annotation entry is corrected to *.
	Layer name * is not unique. Please check the configuration file.
	Highlight * non-existent
	The name * for link appearance is not unique. Please check the configuration file(s).
	Your configuration file * is corrupt. [*] must be unique.

> Error messages at runtime caused by the end user or by the developer

Error Reason	Message
Cycles detected in the method **ScheduleProject**	Project has cycled links!

7 API Reference

7.1 Object types

- DataObject
- DataObjectFiles
- VcBorderArea
- VcBorderBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcCalendar
- VcCalendarCollection
- VcCalendarProfile
- VcCalendarProfileCollection
- VcDataDefinition
- VcDataDefinitionTable
- VcDataRecord
- VcDataRecordCollection
- VcDataTable
- VcDataTableCollection
- VcDataTableField
- VcDataTableFieldCollection
- VcDefinitionField
- 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 DataObject

DataObject

The OLE Drag & Drop technique allows to move selected nodes from an activeX source control to a target control. The container to transfer the corresponding data is the object **DataObject**. The object provides appropriate properties for the transfer: **Files**, **Clear**, **GetData**, **GetFormat** and **SetData**.

You can also exchange data with other controls capable of OLE-Drag&Drop. When doing so, please keep in mind that VARCHART-ActiveX controls store and interpret data in the CSV text format.

To make OLE Drag & Drop work, in the properties window the properties **OLEDragMode** and **OLEDropMode** need to be activated. On the **Nodes** property page by the option **Move all selected nodes** you can select whether just a single node or several marked nodes can be moved.

Please find detailed information in the chapter **Important Concepts** in the section **OLE-Drag&Drop**.

Properties

- Files

Methods

- Clear
- GetData
- GetFormat
- SetData

Properties

Files

Read Only Property of DataObject

This property returns a DataObjectFiles collection, which in turn contains a list of all file names used by a DataObject object (such as the names of files that a user drags to or from the Windows File Explorer.) This property can only be used if the DataObject contains Data of format **15** (**list of files**, please see property **GetFormat**).

	Data Type	Explanation
Property value	DataObjectFiles	List of available files

Methods

Clear

Method of DataObject

This method deletes the contents of the DataObject object. This method is available to drag operations only, i. e. **OLEStartDrag**, **OLESetData**, **OLEGiveFeedback** and **OLECompleteDrag**.

	Data Type	Explanation
Return value	Void	

GetData

Method of DataObject

This method returns data from a DataObject in the shape of the data type **Variant** and is available only to DataObject objects of the events **OLEDragOver** and **OLEDragDrop**.

It is possible for the **GetData** method to use data formats other than those listed below, including user-defined formats registered with Windows by the **RegisterClipboardFormat()** API function. However, there are a few caveats:

The **GetData** method always returns data in a byte array if it is in a format that it cannot recognize.

The byte array returned by **GetData** may be larger than the actual data, with arbitrary bytes at the end of the array. The reason for this is that VARCHART ActiveX does not know the format of the data, but merely has knowledge of the size of memory allocated for the data ba the operating system. The allocated size of memory often is larger than the one actually required for the data. Therefore, there may be an excess of bytes at the end of the allocated memory segment. As a result, you are supposed to use

appropriate functions to interpret the data in a meaningful way (in Visual Basic e.g. truncating a string at a particular length by the **Left** function if the data is in a text format).

Note: Not all applications support the formats **2** (bitmap) or **9** (color palette), so it is recommended that you use **8** (device-independent bitmap) whenever possible.

	Data Type	**Explanation**
Parameter:		
⇨ format	Integer	Identification number of the format (plus examples from Visual Basic and C): 1 - text in ANSI-code (.txt files) VB: vcCFText; C: CF_TEXT 2 - bitmap (.bmp-files) VB: vbCFBitmap; C: CF_BITMAP 3 - metafile (.wmf-files) VB: vbCFMETAFILE; C: CF_MetaFile 8 - device-independent Bitmap (DIB) VB: vbCFDIB; C: CF_DIB 9 - color palette VB: vbCFPalette; C: CF_PALETTE 13 - text in unicode code (.txt-Dateien) VB: 13; C: CF_UNICODETEXT 14 - enhanced Metafile (.emf-files) VB: vbCFEMetaFile; C: CF_EMETAFILE 15 - list of files VB: vbCFFiles; C: CF_FILES -16639 - rich text format (.rtf files) VB: vbCFRTF; C: CF_RTF
Return value	Variant	Data retrieved

GetFormat

Method of DataObject

This method returns a boolean value indicating whether data in the Data-Object object match a specified format. It is available only to DataObject objects of the events **OLEDragOver** and **OLEDragDrop**.

	Data Type	Explanation
Parameter:		
⇨ format	Integer	Identification number of the format (plus examples from Visual Basic and C): 1 - text in ANSI code (.txt files) VB: vcCFText; C: CF_TEXT 2 - bitmap (.bmp-files) VB: vbCFBitmap; C: CF_BITMAP 3 - metafile (.wmf-files) VB: vbCFMETAFILE; C: CF_MetaFile 8 - device-independent Bitmap (DIB) VB: vbCFDIB; C: CF_DIB 9 - color palette VB: vbCFPalette; C: CF_PALETTE 13 - text in unicode code (.txt-Dateien) VB: 13; C: CF_UNICODETEXT 14 - enhanced Metafile (.emf-files) VB: vbCFEMetaFile; C: CF_EMETAFILE 15 - list of files VB: vbCFFiles; C: CF_FILES -16639 - rich text format (.rtf files) VB: vbCFRTF; C: CF_RTF
Return value	Boolean	The **GetFormat** method returns **True** if an item in the DataObject object matches the specified format. Otherwise, it returns **False**.

SetData

Method of DataObject

This method inserts data into a DataObject using the specified data format. It is available only to DataObject objects of the events **OLEStartDrag**, **OLE-SetData**, **OLEGiveFeedback** and **OLECompleteDrag**.

It is possible for the **SetData** method to use data formats other than those listed below **format**, including user-defined formats registered with Windows by the **RegisterClipboardFormat()** API function. However, there are a few caveats:

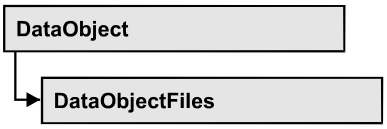
The **SetData** method requires the data to be in the form of a byte array if the data format specified could not be recognized.

Not all applications support **2** (bitmap) or **9** (palette), so it is recommended that you use **8** (device-independent bitmap) whenever possible.

	Data Type	Explanation
Parameter:		
⇨ data	Variant	Data to be set or **Empty** if you wish to transmit the format to be set on request by the event **OLESetData**.

⇨ format	Integer	Identification number of the format (plus examples from Visual Basic and C): 1 - text in ANSI code (.txt files) VB: vcCFText ; C: CF_TEXT 2 - bitmap (.bmp-files) VB: vbCFBitmap; C: CF_BITMAP 3 - metafile (.wmf-files) VB: vbCFMETAFILE; C: CF_MetaFile 8 - device-independent Bitmap (DIB) VB: vbCFDIB; C: CF_DIB 9 - color palette VB: vbCFPalette; C: CF_PALETTE 13 - text in unicode code (.txt-Dateien) VB: 13; C: CF_UNICODETEXT 14 - enhanced Metafile (.emf-files) VB: vbCFEMetaFile; C: CF_EMETAFILE 15 - list of files VB: vbCFFiles; C: CF_FILES -16639 - rich text format (.rtf files) VB: vbCFRTF; C: CF_RTF
Return value	Void	

7.3 DataObjectFiles



This object keeps a list of all file names, that are stored in a DataObject, if it contains data of format **15** (list of files). By **For Each Item in DataObject-Files** you can access all file names in a loop.

Properties

- _NewEnum
- Count
- Item

Methods

- Add
- Clear
- Remove

Properties

_NewEnum

Read Only Property of DataObjectFiles

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data object files. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Private Sub VcNet1_OLEDragOver(ByVal data As VcNetLib.DataObject, effect As
Long, ByVal button As Integer, ByVal Shift As Integer, ByVal x As Long, ByVal y
As Long, ByVal state As VcNetLib.OLEDragStateEnum)

Dim fileName as String
```

```
For Each fileName In DataObject.DataObjectFiles
   Debug.Print fileName
Next

End Sub
```

Count

Read Only Property of DataObjectFiles

This property returns the number of file names available in the list.

	Data Type	Explanation
Property value	Long	Number of files

Item

Property of DataObjectFiles

By this property you can assign or retrieve a file name by the index passed. Because this is the default property of the object, in many programming environments (e.g. Visaul Basic) the property name can be dropped. Example: DataObjectFiles(0) will return the first file name.

	Data Type	Explanation
Parameter:		
⇨ index	Long	Index of the file name {0...Count-1}
Property value	String	File name

Methods

Add

Method of DataObjectFiles

This method lets you add the file name specified to the list of file names. If an index (Integer, values: 0 to .Count-1) is specified, the file name will be inserted at the specified position. Otherwise it will be inserted at the end of the list.

	Data Type	Explanation
Parameter:		
⇨ index	Variant	Index of the position in the list that the file name is to be inserted at (optional)
⇨ fileName	String	Name of the file
Return value	Void	

Clear

Method of DataObjectFiles

This method lets you delete all file names available in the list.

	Data Type	Explanation
Return value	Void	

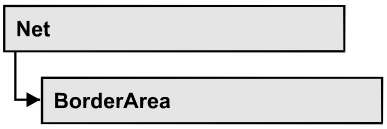
Remove

Method of DataObjectFiles

This method lets you remove the file name with the specified index (values: 0 to .Count-1).

	Data Type	Explanation
Parameter:		
⇨ index	Long	Index of the position in the list that the file name is to be removed from.
Return value	Void	

7.4 VcBorderArea



An object of the type **VcBorderArea** designates the title or legend area of the graphics.

Methods

- BorderBox

Methods

BorderBox

Method of VcBorderArea

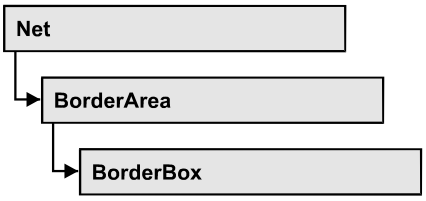
This method gives access to a BorderBox object.

	Data Type	Explanation
Parameter:		
boxPosition	BorderBoxPositionEnum	Box position
	Possible Values:	
	vcBBXPBottomBottomCentered 8	second line in the bottom area, centered
	vcBBXPBottomBottomLeft 7	second line in the bottom area, left
	vcBBXPBottomBottomRight 9	second line in the bottom area, right
	vcBBXPBottomTopCentered 5	first line in the bottom area, centered
	vcBBXPBottomTopLeft 4	first line in the bottom area, left
	vcBBXPBottomTopRight 6	first line in the bottom area, right
	vcBBXPLegend 51	legend
	vcBBXPTopCentered 2	top centered
	vcBBXPTopLeft 1	top left
	vcBBXPTopRight 3	top right
Return value	VcBorderBox	Box of the title and legend area

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

7.5 VcBorderBox



An object of the type **VcBorderBox** designates one of the boxes in the title or legend area of the graphics.

Properties

- Alignment
- GraphicsFileName
- LegendElementsArrangement
- LegendElementsBottomMargin
- LegendElementsMaximumColumnCount
- LegendElementsMaximumRowCount
- LegendElementsTopMargin
- LegendFont
- LegendTitle
- LegendTitleFont
- LegendTitleVisible
- Text
- TextFont
- Type

Properties

Alignment

Property of VcBorderBox

This property lets you set or retrieve the alignment of this BorderBox object.

	Data Type	Explanation
Property value	BorderBoxAlignmentEnum	Alignment of the border box
	Possible Values:	
	vcBBXACentered -1	Center
	vcBBXALeft -3	Left

vcBBXARight -2	Right

GraphicsFileName

Property of VcBorderBox

This property lets you set or retrieve the name of the graphics file used in the VcBorderBox object. *Available formats:*

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

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	Data Type	Explanation
Property value	String	Name of the graphics file

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxTR As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxTR = borderArea.BorderBox(vcBBXPTopRight)
```

```
bBoxTR.Type = vcBBXTGraphics
bBoxTR.GraphicsFilename = "Asterix.jpg"
```

LegendElementsArrangement

Property of VcBorderBox

This property lets you set or retrieve the arrangement of the elements in the legend.

	Data Type	Explanation
Property value	LegendElementsArrangementEnum **Possible Values:** vcLEAFixedToColumns 1 vcLEAFixedToRows 0 vcLEAFixedToRowsAndColumns 2	Type of arrangement of the legend elements The legend elements are merely aligned along columns. The legend elements are merely aligned along rows. The legend elements are aligned along rows and columns.

LegendElementsBottomMargin

Property of VcBorderBox

This property lets you set or retrieve the width between the legend elements and the bottom of the border box (unit: mm).

	Data Type	Explanation
Property value	Integer	Width of bottom margin

LegendElementsMaximumColumnCount

Property of VcBorderBox

This property lets you set or retrieve the number of columns to which the elements in the legend should disperse.

	Data Type	Explanation
Property value	Integer	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	Integer	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	Integer	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	StdFont	Font attributes of the legend

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTLegend
logThis (bBoxBBL.LegendFont.Name)
```

LegendTitle

Property of VcBorderBox

This property lets you set or retrieve the legend title.

	Data Type	Explanation
Property value	String	Legend title

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

LegendTitleFont

Property of VcBorderBox

This property lets you set or retrieve the font attributes of the legend title.

	Data Type	Explanation
Property value	StdFont	Font attributes of the legend title

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTLegend
logThis (bBoxBBL.LegendTitleFont.Name)
```

LegendTitleVisible

Property of VcBorderBox

This property lets you set or retrieve whether the legend title is visible.

	Data Type	Explanation
Property value	Boolean	Legend title visible (True)/ not visible (False)

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitleVisible = False
```

Text

Property of VcBorderBox

This property lets you set or retrieve the text of a head line (above or below the diagram). For numbering the pages or displaying the system date you may enter the below wild cards which will be replaced by the appropriate contents on the printout:

{COLUMN} = page number wide (of a two-dimensional page layout)

{NUMPAGES} = total number of pages

{PAGE} = consecutive numbering of pages

{ROW} = page number high (of a two-dimensional page layout)

{SYSTEMDATE} = system date

	Data Type	Explanation
Parameter:		
rowIndex	Integer	row index {0...6}
Property value	String	text in text boxes

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTText
bBoxBBL.Text(index) = "Department A"
```

TextFont

Property of VcBorderBox

This property lets you set or retrieve the font attributes of a title line (above or below the diagram).

This property is an indexed property, which in C# is referred to by one of the methods **set_TextFont (rowIndex, pvn)** and **get_TextFont (row-Index)**.

	Data Type	Explanation
Parameter:		
rowIndex	Integer	Row index {0...6}

Property value	StdFont	font attributes of the text

Example Code

```
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"
```

Code Sample in 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

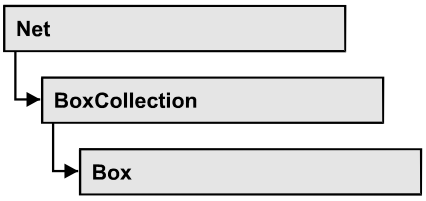
This property lets you set or retrieve the type of the BorderBox object.

	Data Type	Explanation
Property value	BorderBoxTypeEnum	box type
	Possible Values:	
	vcBBXTGraphics 3	graphics
	vcBBXTLegend 4	legend
	vcBBXTNothing 0	nothing
	vcBBXTText 1	text
	vcBBXTTextWithGraphics 2	text and graphics

Example Code

```
Dim borderArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTGraphics
```

7.6 VcBox



An object of the type **VcBox** designates a box to display texts or graphics.

Properties

- FieldText
- FormatName
- LineColor
- LineThickness
- LineType
- MarkBox
- Moveable
- Name
- Origin
- Priority
- ReferencePoint
- Specification
- UpdateBehaviorName
- Visible

Methods

- GetActualExtent
- GetTopLeftPixel
- GetXYOffset
- GetXYOffsetAsVariant
- 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.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Integer	Field index
Property value	String	Field content

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.boxCollection
Set box = boxCltn.FirstBox
box.FieldText(0) = "User: "
```

FormatName

Property of VcBox

This property lets you set or retrieve the name of the box format.

	Data Type	Explanation
Property value	VcBoxFormat	BoxFormat object or **Nothing**

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.FormatName = "Standard"
```

LineColor

Property of VcBox

This property lets you set or retrieve the color of the border line of the box.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.LineColor = RGB(255, 0, 0)
```

LineThickness

Property of VcBox

This property lets you set or retrieve the line thickness of the border line of the box.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.LineThickness = 2
```

LineType

Property of VcBox

This property lets you set or retrieve the type of the border line of the box.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3

vcLineType4 104	Line Type 4
vcLineType5 105	Line Type 5
vcLineType6 106	Line Type 6
vcLineType7 107	Line Type 7
vcLineType8 108	Line Type 8
vcLineType9 109	Line Type 9
vcNone 1	No line type
vcNotSet -1	No line type assigned
vcSolid 2	Line solid

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.LineType = vcDotted
```

MarkBox

Property of VcBox

By this property you can set or retrieve whether a box is marked.

	Data Type	Explanation
Property value	Boolean	**True**: box marked; **false**: box unmarked

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.MarkBox = True
```

Moveable

Property of VcBox

This property lets you set or retrieve whether the box can be moved interactively.

	Data Type	Explanation
Property value	Boolean	Moveable (True)/ not moveable (False) **Default value:** True

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Moveable = False
```

Name

Property of VcBox

This property lets you retrieve/set the name of a box. You can specify the name in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	String	Box name

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim boxName As String

Set boxCltn = VcNet1.boxCollection
Set box = boxCltn.FirstBox
boxName = box.Name
MsgBox boxName
```

Origin

Property of VcBox

This property lets you set or retrieve the point of origin of the box, i. e. the point of the diagram from which the offset to the reference point of the box will be measured.

By using the properties **Origin**, **ReferencePoint** and the method **GetXYOffset** you can position boxes individually in the diagram area. The relative position of a box does not depend on the diagram size.

	Data Type	Explanation
Property value	BoxOriginEnum **Possible Values:** vcBOBottomLeft 27	origin of the box bottom left

vcBOBottomRight 29	bottom right
vcBOCenterCenter 25	center center
vcBOCenterLeft 24	center left
vcBOCenterRight 26	center right
vcBOTopCenter 22	top center
vcBOTopLeft 21	top left
vcBOTopRight 23	top right

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Origin = vcBOTopCenter
```

Priority

Property of VcBox

This property lets you specify or enquire the priority of the box.

	Data Type	Explanation
Property value	Integer	Priority value

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.Priority = 3
```

ReferencePoint

Property of VcBox

This property lets you set or retrieve the reference point of the box, i. e. the point of the box from which the offset to the origin will be measured.

	Data Type	Explanation
Property value	BoxReferencePointEnum **Possible Values:** vcBRPBottomLeft 27 vcBRPBottomRight 29 vcBRPCenterCenter 25 vcBRPCenterLeft 24 vcBRPCenterRight 26 vcBRPTopCenter 22 vcBRPTopLeft 21 vcBRPTopRight 23	reference point of the box bottom left bottom right center center center left center right top center top left top right

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
box.ReferencePoint = vcBRPCenterRight
```

Specification

Read Only Property of VcBox

This property lets you retrieve the specification of a box. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a box by the method **VcBoxCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(0)
MsgBox box.Specification
```

UpdateBehaviorName

Property of VcBox

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior

Visible

Property of VcBox

This property lets you set or retrieve whether a box is visible. You also can specify this property in the **Administrate Boxes** dialog box.

	Data Type	Explanation
Property value	Boolean	box visible/invisible **Default value:** True

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.FirstBox
box.Visible = False
```

Methods

GetActualExtent

Method of VcBox

This method lets you retrieve the extent of the box (unit: 1/100 mm).

By regarding these values when setting the XY offset, you can modify the reference point of the anchoring line without changing the position of the box.

	Data Type	Explanation
Parameter:		
⇦ width	Integer	width of the box
⇦ height	Integer	height of the box
Return value	Boolean	Extent of the box is returned/not returned

GetTopLeftPixel

Method of VcBox

This method lets you convert to pixel and display the saved XY offset for the top left corner.

The x value can be further used with the method **VcGantt.GetDate** for instance to get a date.

	Data Type	Explanation
Parameter:		
⇦ x	Integer	X value of the offset
⇦ y	Integer	Y value of the offset
Return value	Boolean	Offset is returned/not returned

GetXYOffset

Method of VcBox

This method lets you enquire the distance between origin and reference point in x and y direction (unit: 1/100 mm).

Note: If you use VBScript, you can only use the analogous method **GetXYOffsetAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇦ xOffset	Integer	X value of the offset
⇦ yOffset	Integer	Y value of the offset
Return value	Boolean	Offset is returned/not returned

GetXYOffsetAsVariant

Method of VcBox

This method is identical with the method **GetXYOffset** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

IdentifyFormatField

Method of VcBox

This method lets you retrieve the index of the format field at the specified position. If there is a field at the position specified, **True** will be returned, if there isn't, the method will deliver **False**.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the position
⇨ y	Long	Y coordinate of the position
⇦ format	VcBoxFormat	Identified format
⇦ formatFieldIndex	Integer	Index of the format field
Return value	Boolean	A format field exists/does not exist at the position specified

SetXYOffset

Method of VcBox

This method lets you specify the distance between origin and reference point in x and y direction (unit: 1/100 mm).

You also can specify the offset in the **Administrate Boxes** dialog box.

Note: If you use VBScript, you can only use the analogous method **GetXYOffsetAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ xOffset	Integer	X value of the offset
⇨ yOffset	Integer	Y value of the offset
Return value	Boolean	Offset is set (True) / not set (False)

Example Code

```
Dim OffsetSet As Boolean
OffsetSet = VcNet1.boxCollection.FirstBox.SetXYOffset(100, 100)
```

SetXYOffsetByTopLeftPixel

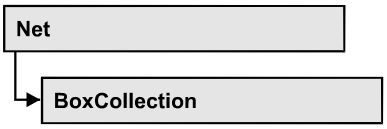
Method of VcBox

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	Integer	X value of the offset
⇨ y	Integer	Y value of the offset
Return value	Boolean	Offset is set (True) / not set (False)

7.7 VcBoxCollection



The VcBoxCollection object contains all boxes available. You can access all objects in an iterative loop by **For Each box In BoxCollection** or by the methods **First...** and **Next...**. You can access a single box by the method **BoxByName** and **BoxByIndex**. The number of boxes in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the boxes in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- BoxByIndex
- BoxByName
- Copy
- FirstBox
- NextBox
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcBoxCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all box objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim box As VcBox

For Each box In VcNet1.BoxCollection
   Debug.Print box.Name
Next
```

Count

Read Only Property of VcBoxCollection

This property lets you retrieve the number of boxes in the box collection.

	Data Type	Explanation
Property value	Long	Number of boxes

Example Code

```
Dim boxCltn As VcBoxCollection
Dim numberOfBoxes As Long

Set boxCltnn = VcNet1.BoxCollection
Dim numberOfBoxes = boxCltn.Count
```

Methods

Add

Method of VcBoxCollection

By this method you can create a box as a member of the BoxCollection. If the name was not used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. To make the new box visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ boxName	String	Box name
Return value	VcBox	New box object

Example Code

```
Set 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	String	Box specification
Return value	VcBox	New box object

BoxByIndex

Method of VcBoxCollection

This method lets you access a box by its index. If a box of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter: ⇨ index	Integer	Index of the box
Return value	VcBox	Box object returned

Example Code

```
Dim boxCltn As VcBoxCollection

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(2)
box.LineThickness = 2
```

BoxByName

Method of VcBoxCollection

By this method you can retrieve a box by its name. If a box of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ boxName	String	Box name
Return value	VcBox	Box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByName("Box 1")
```

Copy

Method of VcBoxCollection

By this method you can copy a box. If the box that is to be copied exists, and if the name for the new box does not yet exist, the new box object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. To make the copied box visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ boxName	String	Name of the box to be copied
⇨ newBoxName	String	Name of the new box
Return value	VcBox	Box object

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.Copy("BoxOne", "NewBox")
boxCltn.Update
```

FirstBox

Method of VcBoxCollection

This method can be used to access the initial value, i.e. the first box of a box collection, and then to continue in a forward iteration loop by the method **NextBox** for the boxes following. If there is no box in the BoxCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBox	First box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.FirstBox
```

NextBox

Method of VcBoxCollection

This method can be used in a forward iteration loop to retrieve subsequent boxes from a box collection after initializing the loop by the method **FirstBox**. If there is no box left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBox	Subsequent box

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.FirstBox

While Not box Is Nothing
    Listbox.AddItem box.Name
    Set box = boxCltn.NextBox
Wend
```

Remove

Method of VcBoxCollection

This method lets you delete a box. To make the deletion visible in the diagram, the box collection needs to be updated by the **Update** call.

	Data Type	Explanation
Parameter:		
⇨ boxName	String	Box name
Return value	Boolean	Box deleted (True)/not deleted (False)

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(2)
boxCltn.Remove (box.Name)
boxCltn.Update
```

Update

Method of VcBoxCollection

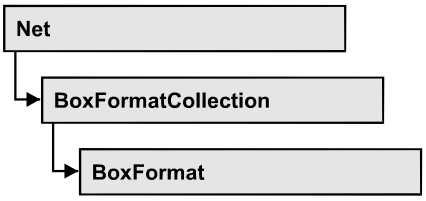
This method lets you update a box collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

Set boxCltn = VcNet1.BoxCollection
Set box = boxCltn.BoxByIndex(2)
boxCltn.Remove (box.Name)
boxCltn.Update
```

7.8 VcBoxFormat



An object of the type **VcBoxFormat** defines the formats of boxes.

Properties

- _NewEnum
- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification

Methods

- CopyFormatField
- RemoveFormatField

Properties

_NewEnum

Read Only Property of VcBoxFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all box format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim formatField As VcBoxFormatField
```

```
For Each formatField In format
    Debug.Print formatField.Index
Next
```

FieldsSeparatedByLines

Property of VcBoxFormat

This property lets you set or retrieve whether fields are to be separated by lines.

	Data Type	Explanation
Property value	Boolean	Box fields separated by lines (True)/ not separated by lines (False).

Example Code

```
Dim boxFormat As VcBoxFormat

Set boxFormat = VcNet1.BoxFormatCollection.FormatByIndex(2)
boxFormat.FieldsSeparatedByLines = True
```

FormatField

Read Only Property of VcBoxFormat

This property lets you access a VcBoxFormatField object by its index. The index has to be in the range 0 to .FormatFieldCount-1.

Note for users of a version earlier than 3.0: The index does **not** count from 1 to .FormatFieldCount as (as did the field properties up to 3.0).

	Data Type	Explanation
Parameter: index	Integer	Index of the box format field 0 ... .FormatFieldCount-1
Property value	VcBoxFormatField	Box format field

Example Code

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

Set boxFormat = VcNet1.BoxFormatCollection.FirstFormat
Set formatField = boxFormat.formatField(0)
MsgBox formatField.FormatName
```

FormatFieldCount

Read Only Property of VcBoxFormat

This property allows to determine the number of fields in a box format.

	Data Type	Explanation
Property value	Integer	Number of fields of the box format

Example Code

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

Set boxFormat = VcNet1.BoxFormatCollection.FirstFormat
MsgBox boxFormat.FormatFieldCount
```

Name

Property of VcBoxFormat

This property lets you retrieve/set the name of a box format. You can also specify the name in the **Administrate Box Formats** dialog box.

	Data Type	Explanation
Property value	String	Box format name

Example Code

```
Dim boxFormat As VcBoxFormat

For Each boxFormat In VcNet1.BoxFormatCollection
   List1.AddItem (boxFormat.Name)
Next
```

Specification

Read Only Property of VcBoxFormat

This property lets you retrieve the specification of a box Format. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a box format by the method **VcBoxFormatCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the box format

Methods

CopyFormatField

Method of VcBoxFormat

This method allows to copy a box format field. The new VcBoxFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	FormatFieldInnerPositionEnum **Possible Values:** vcInnerAbove 1 vcInnerBelow 3 vcInnerLeftOf 0 vcInnerRightOf 4	Position of the new box format field above below left of right of
⇨ refIndex	Integer	Index of the reference box format field
Return value	VcBoxFormatField	Box format field object

Example Code

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

Set boxFormat = VcNet1.BoxFormatCollection.FormatByIndex(2)
Set formatField = boxFormat.CopyFormatField(vcInnerRightOf, 0)
```

RemoveFormatField

Method of VcBoxFormat

This method lets you remove a layer format field by its index. After that, the program will update all layer format field indexes so that they are consecutively numbered again.

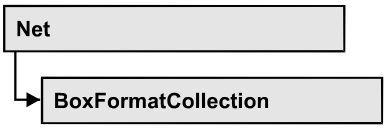
	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the box format field to be deleted

Example Code

```
Dim boxFormat As VcBoxFormat
Dim i As Integer

boxFormat = VcNet1.BoxFormatCollection.FirstFormat

For i = 0 To boxFormat.FormatFieldCount - 1
   boxFormat.RemoveFormatField (i)
```

```
Next
```

7.9 VcBoxFormatCollection



The VcBoxFormatCollection object contains all box formats available. You can access all objects in an iterative loop by **For Each boxFormat In BoxFormatCollection** or by the methods **First...** and **Next...**. You can access a single box format by the methods **BoxFormatByName** and **BoxFormat-ByIndex**. The number of box formats in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the box formats in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcBoxFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all box format objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim format As VcBoxFormat

For Each format In VcNet1.BoxCollection
   Debug.Print format.Name
Next
```

Count

Read Only Property of VcBoxFormatCollection

This property lets you retrieve the number of box formats in the box format collection.

	Data Type	Explanation
Property value	Long	Number of box formats

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim numberOfBoxformats As Long

Set boxFormatCltn = VcNet1.BoxFormatCollection
Dim numberOfBoxformats = boxFormatCltn.Count
```

Methods

Add

Method of VcBoxFormatCollection

By this method you can create a box format as a member of the BoxFormatCollection. If the name was not used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	Name of the box format
Return value	VcBoxFormat	New box format object

Example Code

```
Set 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	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	String	Name of the box format to be copied
⇨ newFormatName	String	Name of the new box format
Return value	VcBoxFormat	Box format object

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat")
```

FirstFormat

Method of VcBoxFormatCollection

This method can be used to access the initial value, i.e. the first box format of a box format collection and then to continue in a forward iteration loop by the method **NextFormat** for the box formats following. If there is no box format in the box format collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBoxFormat	First box format

Example Code

```
Dim format As VcBoxFormat

Set format = VcNet1.BoxFormatCollection.FirstFormat
```

FormatByIndex

Method of VcBoxFormatCollection

This method lets you access a box format by its index. If a box format of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the box format
Return value	VcBoxFormat	Box format object returned

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim format As VcBoxFormat

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set format = boxFormatCltn.FormatByIndex(2)
```

FormatByName

Method of VcBoxFormatCollection

By this method you can retrieve a box format by its name. If a box format of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ formatName	String	Name of the box format
Return value	VcBoxFormat	Box format

Example Code

```
Dim formatCltn As VcBoxFormatCollection
Dim format As VcBoxFormat

Set formatCltn = VcNet1.BoxFormatCollection
Set format = formatCltn.FormatByName("Standard")
```

NextFormat

Method of VcBoxFormatCollection

This method can be used in a forward iteration loop to retrieve subsequent box formats from a box format collection after initializing the loop by the method **FirstFormat**. If there is no format left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcBoxFormat	Subsequent box format

Example Code

```
Dim formatCltn As VcBoxFormatCollection
Dim format As VcBoxFormat

Set formatCltn = VcNet1.BoxFormatCollection
Set format = formatCltn.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCltn.NextFormat
Wend
```

Remove

Method of VcBoxFormatCollection

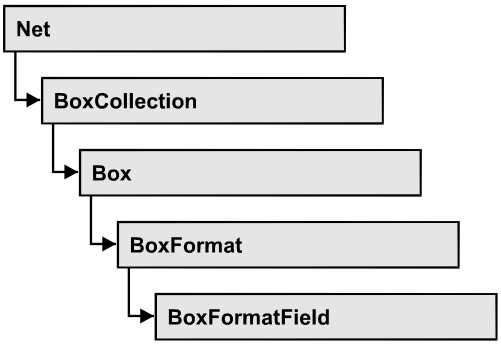
This method lets you delete a box format. If the box format is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	Box format name
Return value	Boolean	Box format deleted (True)/not deleted (False)

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormat = boxFormatCltn.FormatByIndex(1)
boxFormatCltn.Remove (boxFormat.Name)
```

7.10 VcBoxFormatField



An object of the type **VcBoxFormat** represents a field of a VcBoxFormat object. A box format field does not have a name as many other objects, but it has an index that defines its position in the box format.

Properties

- Alignment
- FormatName
- GraphicsHeight
- Index
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColorAsARGB
- PatternColorAsARGB
- PatternEx
- TextFont
- TextFontColor
- Type

Properties

Alignment

Property of VcBoxFormatField

This property lets you set or retrieve the alignment of the content of the box format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum **Possible Values:** vcFFABottom 28 vcFFABottomLeft 27 vcFFABottomRight 29 vcFFACenter 25 vcFFALeft 24 vcFFARight 26 vcFFATop 22 vcFFATopLeft 21 vcFFATopRight 23	Alignment of the field content bottom bottom left bottom right center left right top top left top right

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.Alignment = vcFFACenter
```

FormatName

Read Only Property of VcBoxFormatField

This property lets you retrieve the name of the box format to which this box format field belongs.

	Data Type	Explanation
Property value	String	Name of the box format

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
MsgBox boxFormatField.FormatName
```

GraphicsHeight

Property of VcBoxFormatField

This property lets you set or retrieve for the type **vcFFTGraphics** the height of the graphics in the box format field.

	Data Type	Explanation
Property value	Integer	Height of the graphics in mm 0 ... 99

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

Index

Read Only Property of VcBoxFormatField

This property lets you enquire the index of the box format field in the corresponding box format.

	Data Type	Explanation

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
MsgBox boxFormatField.Index
```

MaximumTextLineCount

Property of VcBoxFormatField

This property lets you set or retrieve the maximum number of lines in the box format field, if the box format field is of the type **vcFFTText**. Also see the property **MinimumTextLineCount**.

	Data Type	Explanation
Property value	Integer	Maximum number of lines 0 ... 9

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
```

```
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTText
boxFormatField.MaximumTextLineCount = 5
```

MinimumTextLineCount

Property of VcBoxFormatField

This property lets you set or retrieve the minimum number of lines in the box format field, if it is of the type **vcFFTText**. If there is more text than can be taken by the lines, the format field will be enlarged dynamically up to the maximum number of lines. When assigning a value by this property, please also remember to set the **MaximumTextLineCount** value anew, since otherwise the minimum value might overwrite the maximum value.

	Data Type	Explanation
Property value	Integer	Minimum number of lines 0 ... 9

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTText
boxFormatField.MinimumTextLineCount = 3
```

MinimumWidth

Property of VcBoxFormatField

This property lets you set or retrieve the minimum width of the box field in mm. The field width may be enlarged, if above or below the field fields exist that have greater minimum widths.

	Data Type	Explanation
Property value	Integer	Minimum width of the box format field 0 ... 9

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.MinimumWidth = 100
```

PatternBackgroundColorAsARGB

Property of VcBoxFormatField

This property lets you set or retrieve the background color of the box format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	Long	Background color of the box format **Default value:** -1

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.BackColor = RGB(0, 255, 0)
```

PatternColorAsARGB

Property of VcBoxFormatField

This property lets you set or retrieve the pattern color of the box format field. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	Long	Pattern color of the box format field

Example Code

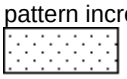
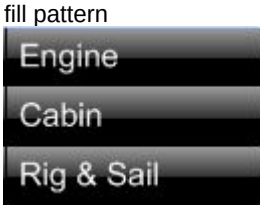
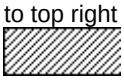
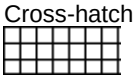
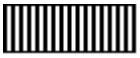
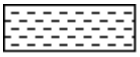
```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField
```

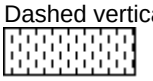
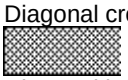
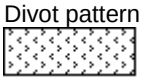
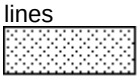
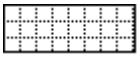
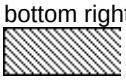
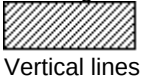
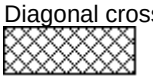
```
Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.formatField(0)
boxFormatField.PatternColor = RGB(0, 255, 0)
```

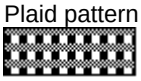
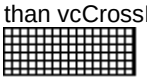
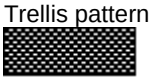
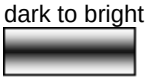
PatternEx

Property of VcBoxFormatField

This property lets you set or retrieve the pattern of the field background of the box format field.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type **Default value:** As defined in the dialog
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 

vcDashedVerticalPattern 2027	Dashed vertical lines 
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right 
vcHorizontalBrickPattern 2033	Horizontal brick pattern 
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 
vcSmallCheckerBoardPattern 2043	Checkerboard pattern 
vcSmallConfettiPattern 2028	Confetti pattern 
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern 
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares 
vcSpherePattern 2041	Checkerboard of spheres 
vcTrellisPattern 2040	Trellis pattern 
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright 
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
vcVerticalGradientPattern 62	Vertical color gradient 
vcVerticalPattern 2	Vertical lines 
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
vcWavePattern 2031	Horizontal wave pattern 
vcWeavePattern 2034	Interwoven stripe pattern 
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern 
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern 
vcZigZagPattern 2030	Horizontal zig-zag lines 

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Pattern = 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	StdFont	Font type of the box format

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFont.Bold = True
```

TextFontColor

Property of VcBoxFormatField

This property lets you set or retrieve the font color of the box format field, if it is of the type **vcFFTText**.

	Data Type	Explanation
Property value	OLE_COLOR	Font color of the box format **Default value:** -1

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFontColor = RGB(0, 255, 0)
```

Type

Property of VcBoxFormatField

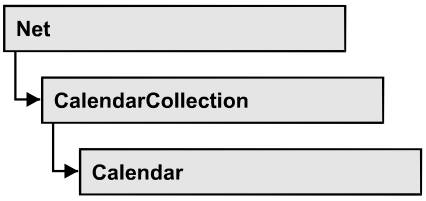
This property lets you enquire the type of the box format field.

	Data Type	Explanation
Property value	FormatFieldTypeEnum **Possible Values:** vcFFTGraphics 64 vcFFTText 36	Type of the box format field graphics text

Example Code

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

Set boxFormatCltn = VcNet1.BoxFormatCollection
Set boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = vcFFTGraphics
boxFormatField.GraphicsHeight = 200
```

7.11 VcCalendar



A calendar serves to define work and non work periods. It is composed of a continuous sequence of work and nonwork periods, that commonly are made of Workday and Workweek objects, but may also consist of intervals. A calendar just created by default contains an interval that covers the whole project. 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	String	Name of the calendar

Example Code

```
Dim calendar As VcCalendar
Dim calendarName As String

Set calendar = VcNet1.CalendarCollection.FirstCalendar
calendarName = calendar.Name
```

SecondsPerWorkday

Property of VcCalendar

This property lets you set/retrieve the number of seconds of a workday. This feature can be also set in the **Specify Calendars** dialog.

	Data Type	Explanation
Property value	Long	Seconds of a workday

Specification

Read Only Property of VcCalendar

This property lets you retrieve the specification of a calendar. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create a calendar by the method **VcCalendar-Collection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the calendar

Methods

AddDuration

Method of VcCalendar

This method lets you assign a duration (work time) to a date of the calendar, considering the settings of the calendar. If e.g. you have defined workfree weekends to your calendar, a duration of three days added to a Friday will result in the Wednesday following.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date the duration is to be inserted at
⇨ Duration	Long	Number of time units (e.g.days)
Return value	Date/Time	Date the duration was inserted at

Example Code

```
Dim calendar As VcCalendar
Dim newDate As Date

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
newDate = calendar.AddDuration("16.06.2014", 3)
```

CalcDuration

Method of VcCalendar

This method lets you retrieve the number of work time elements (e.g. work days) available between two defined dates. The unit (e.g. days) of the value returned is the one defined in the **Time Unit** field on the **General** property page.

	Data Type	Explanation
Parameter:		
⇨ fromDate	Date/Time	Start date of the duration that the number of work time elements is to be retrieved of
⇨ toDate	Date/Time	End date of the duration that the number of work time elements is to be retrieved of
Return value	Long	Number of time units (e.g. days) of the duration

Example Code

```
Dim calendar As VcCalendar
Dim duration As Long

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
duration = calendar.CalcDuration("01.01.2013", "31.12.2014")
```

Clear

Method of VcCalendar

Removes the profiles and intervals formerly defined in this VcCalendar object, thus completely clearing it (=> 100% working time). The changes will only be displayed after an update.

	Data Type	Explanation

GetEndOfPreviousWorktime

Method of VcCalendar

This method lets you retrieve the end of the work time that precedes the reference date. The reference date has to belong to a non-working period.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date that the previous work time refers to
Return value	Date/Time	Final date of the previous work time

Example Code

```
Dim calendar As VcCalendar
Dim endOfWork As Date

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
endOfWork = calendar.GetEndOfPreviousWorktime("18.06.2014")
```

GetNextIntervalBorder

Method of VcCalendar

This method lets you retrieve the beginning of the interval succeeding. If the reference date is in a non work time, the date returned will be the beginning of the succeeding work time, and vice versa.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date that the following interval border refers to
Return value	Date/Time	Start date of the interval border following

Example Code

```
Dim calendar As VcCalendar
Dim nextIntervalBorder As Date

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
nextIntervalBorder = calendar.GetNextIntervalBorder("18.06.2014")
```

GetPreviousIntervalBorder

Method of VcCalendar

This method lets you retrieve the end of the preceding interval. If the reference date is in a non work time, the date returned will be the end of the preceding work time, and vice versa.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date that of the preceding interval border refers to
Return value	Date/Time	End date of the interval border preceding

Example Code

```
Dim calendar As VcCalendar
Dim previousIntervalBorder As Date

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
previousIntervalBorder = calendar.GetPreviousIntervalBorder("18.06.2014")
```

GetStartOfInterval

Method of VcCalendar

This method lets you retrieve the beginning of the interval that the reference date is located in.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Reference date of the interval, that the start date is to be retrieved of
Return value	Date/Time	Start date of the interval

Example Code

```
Dim calendar As VcCalendar
Dim startOfInterval As Date

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
startOfInterval = calendar.GetStartOfInterval("18.06.2014")
```

GetStartOfNextWorktime

Method of VcCalendar

This method lets you retrieve the beginning of the work time that succeeds the reference date.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Reference date, that the start date of the work time following is to be retrieved of
Return value	Date/Time	Start date of the work time following

Example Code

```
Dim calendar As VcCalendar
Dim startOfNextWorktime As Date

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
startOfNextWorktime = calendar.GetStartOfNextWorktime("18.06.2014")
```

IsWorktime

Method of VcCalendar

This method lets you enquire whether or not the date passed is in a work time.

	Data Type	Explanation
Parameter:		
⇨ Date	Date/Time	Date to be checked for being a work time
Return value	Boolean	Date passed does /does not belong to a work time

Example Code

```
Dim calendar As VcCalendar
Dim isWorktime As Boolean

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
isWorktime = calendar.isWorktime("18.06.2014")
```

Update

Method of VcCalendar

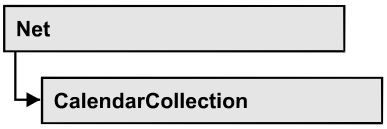
This method lets you update a calendar after having modified it. It ensures other objects that use calendar (e.g. a calendarGrid) to be updated as well.

	Data Type	Explanation
Return value	Void	

Example Code

```
Dim calendar As VcCalendar

Set calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
calendar.Update
```

7.12 VcCalendarCollection



An object of the type VcCalendarCollection automatically contains all available calendars. You can access all objects in an iterative loop by **For Each calendar In CalendarCollection** or by the methods **First...** and **Next...**. You can access a single calendar by the methods **CalendarByName** and **CalendarByIndex**. The number of calendars in the collection object can be retrieved by the property **Count**. By the property **Active** you can set or retrieve the calendar which controls the calendar grid.

Properties

- _NewEnum
- Active
- Count

Methods

- Add
- AddBySpecification
- CalendarByIndex
- CalendarByName
- Copy
- FirstCalendar
- NextCalendar
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcCalendarCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all calender objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim calendar As VcCalendar

For Each calendar In VcNet1.CalendarCollection
   Debug.Print calendar.Name
Next
```

Active

Property of VcCalendarCollection

This property lets you set or retrieve the default calendar that is used by nodes, if no other calendar was assigned.

	Data Type	Explanation
Property value	VcCalendar	Calendar currently used

Example Code

```
Dim workday As VcWorkday
Dim freeday As VcWorkday
Dim workweek As VcWorkweek
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

Set workday = VcNet1.WorkdayCollection.CreateWorkday("Work day")
workday.AddNonWorkInterval "00:00:00", "00:00:00"
workday.AddWorkInterval "08:00:00", "16:30:00"

Set freeday = VcNet1.WorkdayCollection.CreateWorkday("Workfree day")
freeday.AddNonWorkInterval "00:00:00", "00:00:00"

Set calendarCltn = VcNet1.calendarcollection
Set calendar = calendarCltn.AddCalendar("New Calendar")

Set workweek = VcNet1.WorkweekCollection.CreateWorkweek("Work week")
workweek.AddWorkday workday, vcMonday, vcFriday
workweek.AddWorkday freeday, vcSaturday, vcSunday

calendar.AddWorkweek workweek, "01.01.13", "31.12.14"

calendar.Update

Set calendarCltn.Active = calendar
```

Count

Read Only Property of VcCalendarCollection

This property lets you retrieve the number of calendars in the calendar collection.

	Data Type	Explanation
Property value	Long	Number of calendars

Methods

Add

Method of VcCalendarCollection

By this method you can create a calendar as a member of the CalendarCollection. If the name has not been used before, the new calendar object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter: ⇨ calendarName	String	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	String	Calendar specification
Return value	VcCalendar	New calendar object

CalendarByIndex

Method of VcCalendarCollection

This method lets you access a calendar by its index. If no calendar of the specified index does exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	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	String	Name of the calendar
Return value	VcCalendar	Calendar

Example Code

```
Dim calendarCltn As VcCalendarCollection

Set calendarCltn = VcNet1.CalendarCollection
calendarCltn.Active = calendarCollection.CalendarByName("Calendar_1")
```

Copy

Method of VcCalendarCollection

By this method you can copy a calendar. If the calendar that is to be copied exists, and if the name for the new calendar does not yet exist, the new calendar object is returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ calendarName	String	Name of the calendar to be copied
⇨ newCalendarName	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, and then to continue in a forward iteration loop by the method **NextCalendar** for the calendars following. If there is no calendar in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendar	First calendar

NextCalendar

Method of VcCalendarCollection

This method can be used in a forward iteration loop to retrieve subsequent calendars from a calendar collection after initializing the loop by the method **FirstCalendar**. If there is no calendar left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcCalendar	Subsequent calendar

Example Code

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

Set calendarCltn = VcNet1.CalendarCollection
Set calendar = calendarCltn.FirstCalendar

While Not calendar Is Nothing
    List1.AddItem (calendar.Name)
    Set calendar = calendarCltn.NextCalendar
Wend
```

Remove

Method of VcCalendarCollection

This method lets you delete a calendar. If the calendar is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Return value	Boolean	Calendar deleted (True)/not deleted (False)

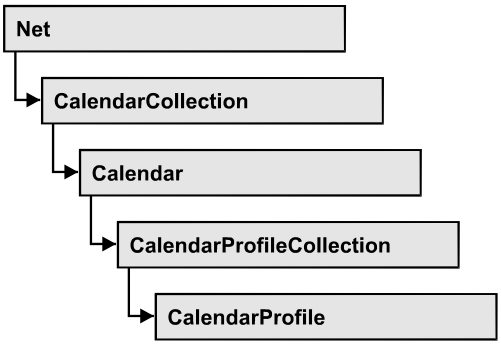
Update

Method of VcCalendarCollection

This method lets you update a calendar collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.13 VcCalendarProfile



An object of the type **VcCalendarProfile** designates a calendar profile.

Properties

- IntervalCollection
- Name
- Specification
- Type

Methods

- PutInOrderAfter

Properties

IntervalCollection

Read Only Property of VcCalendarProfile

This property gives access to the IntervalCollection object that contains all intervals available.

	Data Type	Explanation
Property value	VcIntervalCollection	IntervalCollection object

Name

Read Only Property of VcCalendarProfile

This property lets you set or retrieve the name of a calendar profile.

	Data Type	Explanation
Property value	String	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	String	Specification of the calendar profile

Type

Property of VcCalendarProfile

This property lets you set or retrieve the calendar profile type. If you change the type, all properties of this calendar profile will be deleted.

	Data Type	Explanation
Property value	CalendarProfileTypeEnum	Type of the calendar profile

Methods

PutInOrderAfter

Method of VcCalendarProfile

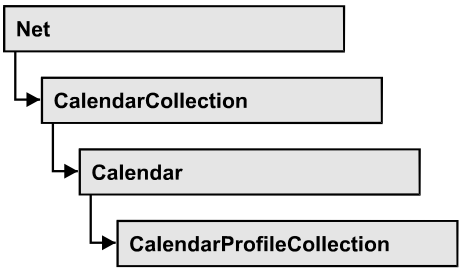
This method lets you set the calendar profile behind the calendar profile specified by name, within the CalendarProfileCollection. If you set the name to "", the calendar profile will be put in the first position. The order of the calendar profiles within the collection determines the order by which they apply to the calendars.

	Data Type	Explanation
Parameter:		
refNameParam	String	Name of the calendar profile behind which the current calendar profileis to be put.

Example Code

```
Dim calProfCltn As VcCalendarProfileCollection
Dim calProf1 As VcCalendarProfile
Dim calProf2 As VcCalendarProfile

calProfCltn = VcGantt1.CalendarProfileCollection()
calProf1 = calProfCltn.Add("calProf1")
calProf2 = calProfCltn.Add("calProf2")
calProf1.PutInOrderAfter("calProf2")
calProfCltn.Update()
```

7.14 VcCalendarProfileCollection



An object of the type VcCalendarProfileCollection automatically contains all available calendar profiles. You can access all objects in an iterative loop by **For Each calendarProfile In CalendarProfileCollection** or by the methods **First...** and **Next...**. You can access a single calendar profile using the methods **CalendarProfileByName** and **CalendarProfileByIndex**. The number of calendar profiles in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the calendar profiles in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- CalendarProfileByIndex
- CalendarProfileByName
- Copy
- FirstCalendarProfile
- NextCalendarProfile
- Remove
- SelectCalendarProfiles
- Update

Properties

_NewEnum

Property of VcCalendarProfileCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all calendar profile objects contained. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language constructs.

	Data Type	Explanation
Property value	Object	Reference object

Count

Read Only Property of VcCalendarProfileCollection

This property lets you retrieve the number of calendar profiles in the calendar profile collection.

	Data Type	Explanation
Property value	Long	Number of CalendarProfile objects

Methods

Add

Method of VcCalendarProfileCollection

By this method you can create a calendar profile as a member of the CalendarProfileCollection. If the name has not been used before, the new filter object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ profileName	String	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	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	Integer	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	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	String	Name of the calendar profile to be copied
⇨ newprofileName	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	String	Calendar profile name
Return value	Boolean	Calendar profile deleted (True)/not deleted (False)

SelectCalendarProfiles

Method of VcCalendarProfileCollection

This method lets you specify the calendar profiles that the calendar profile collection is to contain.

	Data Type	Explanation
Parameter:		
⇨ selectionType	CalendarProfileTypeEnum	Type of calendar profile to be selected
Return value	Long	Number of calendar profiles selected

Example Code

```
Dim calendarProfileCltn As VcCalendarProfileCollection

Set calendarProfileCltn = VcNet1.CalendarProfileCollection
calendarProfileCltn.SelectCalendarProfile (vcSelected)
```

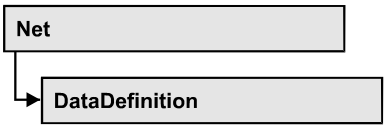
Update

Method of VcCalendarProfileCollection

This method lets you update a calendar profile collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.15 VcDataDefinition



The data of nodes and links can be defined in the dialog **Administrate Data Tables** which can be reached by selecting **Data tables...** on the **Objects** property page. It grants access to the names and types of the available fields. The data definition of a VcNet object contains two data definition tables: vcMaindata and vcRelations.

Properties

- DefinitionTable

Properties

DefinitionTable

Read Only Property of VcDataDefinition

This property allows the access to the two tables of the data definition object.

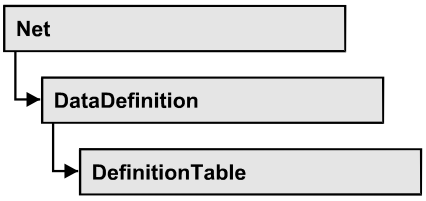
- **vcMaindata:** definitions for nodes
- **vcRelations**: definitions for links

	Data Type	Explanation
Parameter:		
⇨ tableType	DataTableEnum	Type of data definition table
	Possible Values: vcMaindata 0 vcRelations 1	 Table type **vcMaindata** (for nodes) Table type **vcRelations** (for links)
Property value	VcDataDefinitionTable	Data definition table

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable

Set dataDefinition = VcNet1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
```

7.16 VcDataDefinitionTable



A VcDataDefinitionTable object is an element of a data definition. It represents a table of data definition fields. You can access these fields individually by the methods **FieldByIndex** or **FieldByName** or retrieve them in an iterative loop by the methods **FirstField** and **NextField**. By the **Count** property you can enquire the number of the fields of the table. You can set data field definitions on the property page **Administrate Data Tables**.

Properties

- _NewEnum
- Count

Methods

- CreateDataField
- FieldByIndex
- FieldByName
- FirstField
- NextField

Properties

_NewEnum

Read Only Property of VcDataDefinitionTable

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data definition field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim datdeftable As VcDataDefinitionTable

For Each datdeftable In VcNet1.VcDataDefinition
   Debug.Print datdeftable.Count
Next
```

Count

Read Only Property of VcDataDefinitionTable

This property lets you retrieve the number of fields in the data table. You can add fields by the **Administrate Data Tables** dialog or at run time by the method **CreateDataField**.

	Data Type	Explanation
Property value	Long	Number of fields

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim numberOfFields As Long

Set dataDefinition = VcNet1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

numberOfFields = dataDefinitionTable.Count
```

Methods

CreateDataField

Method of VcDataDefinitionTable

This method lets you add a new data field at run time to the end of the data table. The data field of the new data field is Integer. You can change the data type by the property **Type** of **VcDefinitionField**.

	Data Type	Explanation
Parameter:		
⇨ newfieldName	String	Name of the new field
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable =  _ VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.CreateDataField("Description")
dataDefinitionField.Type = vcDefFieldAlphanumericType
VcNet1.DataTableCollection.Update
```

FieldByIndex

Method of VcDataDefinitionTable

By this method you can access a field of the data definition table by index. A field can be referred to by its name or by its index. The index of the first field is 1. You can set data field definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Integer	Field index
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Set dataDefinitionTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
For I = 1 To dataDefinitionTable.Count
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.FieldByIndex(I)
Next
```

FieldByName

Method of VcDataDefinitionTable

By this method you can get a field of the data table by its name. If a field of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic). A field can be referred to by its name or by its index. You can set data definitions in the **Administrate Data Tables** dialog.

	Data Type	Explanation
Parameter:		
⇨ fieldName	String	Field name
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcNet1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FieldByName("Code 1")
```

FirstField

Method of VcDataDefinitionTable

This method can be used to access the first field of a data table and to continue in a forward iteration loop by the method **NextField** for the fields following. If there is no field in the data table, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDefinitionField	First Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FirstField
```

NextField

Method of VcDataDefinitionTable

This method can be used in a forward iteration loop to retrieve subsequent fields from a data table after initializing the loop by the method **FirstField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

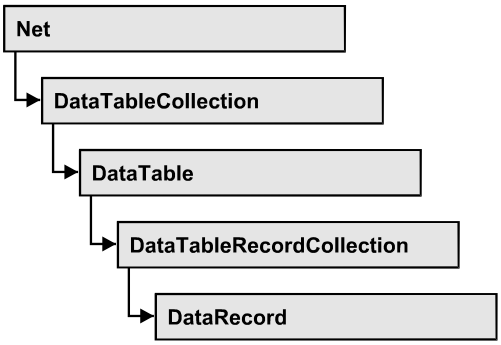
	Data Type	Explanation
Return value	VcDefinitionField	Subsequent data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
While Not dataDefinitionField Is Nothing
```

```
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
Wend

or

Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinitionTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
For I = 1 To dataDefinitionTable.Count
      List1.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
 Next
```

7.17 VcDataRecord



A data record is the logical base of an object in a diagram, for example of a node. Objects have specific features, that are described in the fields of the record. For the fields of a data record, descriptions exist that are stored to data table fields. Data records and data table fields are collected in corresponding collection objects, which form a data table.

Properties

- AllData
- DataField
- DataTableName
- ID

Methods

- DeleteDataRecord
- IdentifyObject
- RelatedDataRecord
- UpdateDataRecord

Properties

AllData

Property of VcDataRecord

This property lets you set or retrieve the complete data of a data record. When setting the property, a CSV string (using semicolons as separators) or the data type "variant" are allowed, that contains all data fields of the record in an array. On retrieving the property, a string will be returned.

	Data Type	Explanation
Property value	Variant	All data of the data record

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRecValue() As Variant
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata1")
Set dataRecCltn = dataTable.DataRecordCollection
ReDim dataRecValue(dataTable.DataTableFieldCollection.Count)
dataRecValue(0) = 1
dataRecValue(1) = "Node One"

'Variant
Set dataRecord = dataRecCltn.Add(dataRecValue)
'CSV
dataRecord.AllData = "1;Node One;"

dataRecord.UpdateDataRecord
```

DataField

Property of VcDataRecord

This property lets you assign or retrieve data to/from a field of a data record. After the data field was modified by the **DataField** property, the graphical display in the diagram needs to be updated by the **UpdateDataRecord** method.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field
Property value	Variant	Content of the data field

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.UpdateDataRecord
```

DataTableName

Read Only Property of VcDataRecord

This property lets you retrieve the name of the data table that this data record belongs to.

	Data Type	Explanation
Property value	String	Name of the associated table

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)

MsgBox dataRecord.DataTableName
```

ID

Read Only Property of VcDataRecord

By this property you can retrieve the ID of a data record.

	Data Type	Explanation
Property value	String	Data record ID

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord
Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)
MsgBox dataRecord.ID
```

Methods

DeleteDataRecord

Method of VcDataRecord

This method lets you delete a data record.

	Data Type	Explanation
Return value	Boolean	Data record was (true) / was not (false) deleted successfully

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecCltn.DataRecordByID(1)

dataRecord.DeleteDataRecord
```

IdentifyObject

Method of VcDataRecord

This method lets you identify the object having been established via this VcDataRecord object.

The return value will be **true** if a data-based object could be identified, i.e. if a data-based object could be created for the graphic from the record.

	Data Type	Explanation
Parameter:		
⇨ establishedObject Param	Object	Identified object
establishedObjectTypeParam	VcObjectTypeEnum	Object type
	Possible Values: vcObjTypeBox 15 vcObjTypeGroup 7 vcObjTypeLinkCollection 3 vcObjTypeNode 2 vcObjTypeNodeInLegend 17 vcObjTypeNone 0	 object type **box** object type **group** object type **link collection** object type **node** object type **node in legend area** no object
Return value	Boolean	data-based object has been/has not been established

RelatedDataRecord

Method of VcDataRecord

This property lets you relate a data record to another one or retrieve a related data record. When using extended data tables, the data records of a table can be related to the data records of another table by primary keys.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field
Return value	VcDataRecord	Related data record

Example Code

```
Private Sub VcNet1_OnNodeLClick(ByVal node As VcNetLib.VcNode, ByVal location As
VcNetLib.LocationEnum, ByVal x As Long, ByVal y As Long, returnStatus As
Variant)

    Dim dataTable As VcDataTable
    Dim dataRecordCltn As VcDataRecordCollection
    Dim firstDataRecord As VcDataRecord
    Dim secondDataRecord As VcDataRecord

    Set dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
    Set dataRecordCltn = dataTable.DataRecordCollection

    Set firstDataRecord = dataRecordCltn.DataRecordByID(node.DataField(0))
    Set secondDataRecord = firstDataRecord.RelatedDataRecord(2)

    MsgBox secondDataRecord.AllData

End Sub
```

UpdateDataRecord

Method of VcDataRecord

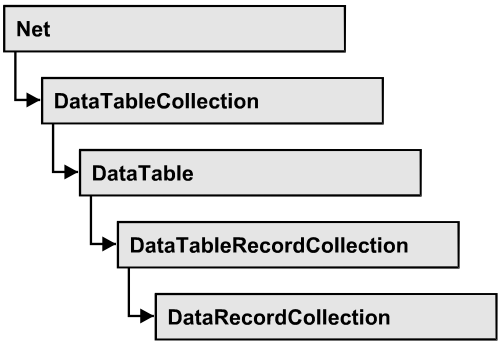
If data fields of a data record were modified by the **DataField** property, the diagram needs to be updated by the **UpdateDataRecord** method.

	Data Type	Explanation
Return value	Boolean	Data record was (true) / was not (false) updated successfully

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.UpdateDataRecord
```

7.18 VcDataRecordCollection



An object of the type VcDataRecordCollection automatically contains all data records of a table. The property **Count** retrieves the number of records present in the collection; the Enumerator object and the methods **FirstDataRecord** and **NextDataRecord** allow to access data records by iteration while by **DataRecordByID** single data records can be accessed. **Add** and **Remove** are basic administering methods, and **Update** lets you refresh the graphical display of objects by data of the records recently modified.

Properties

- _NewEnum
- Count

Methods

- Add
- DataRecordByID
- FirstDataRecord
- GetNewUniqueID
- NextDataRecord
- Remove
- Update

Properties

_NewEnum

Property of VcDataRecordCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data records. In Visual Basic this property is not indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection

For Each dataRecord In dataRecordCltn
   Debug.Print dataRecord.AllData
Next dataRecord
```

Count

Read Only Property of VcDataRecordCollection

This property lets you retrieve the number of data records in the DataRecord-Collection object.

	Data Type	Explanation
Property value	Long	Number of data records in the collection object

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
MsgBox "Number of DataRecords: " & dataRecordCltn.Count
```

Methods

Add

Method of VcDataRecordCollection

By this method you can create a data record as a member of the DataRecord-Collection. If the recordDescription did not fail to have a new data record created, the data record will be returned; otherwise a **VcPrimaryKeyNot-UniqueException** will be thrown.

After adding the data record, the method **VcNet.EndLoading** needs to be invoked to make the modification take effect.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	Object	Content of the data record (as an array or a string)
Return value	VcDataRecord	Data record created

Example Code

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

'...

Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim dataRecVal() As Variant

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

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(Main_ID) = 1
dataRecVal(Main_Name) = "Node 1"
dataRecVal(Main_Start) = DateSerial(2014, 1, 8)
dataRecVal(Main_Duration) = 8
Set dataRec1 = dataRecCltn.Add(dataRecVal)
VcNet1.EndLoading()

' equivalent
' dataRec1 = dataRecCltn.Add("1;Node 1;01.08.14;;8")
```

DataRecordByID

Method of VcDataRecordCollection

This method lets you access a data record by its identification. If a data record of the specified ID does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

If the identification consists of several fields (composite primary key), this multipart ID has to be specified as follows:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ dataRecordID	String	ID of data record
Return value	VcDataRecord	Data record object

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.DataRecordByID(0)
```

FirstDataRecord

Method of VcDataRecordCollection

This method can be used to access the initial value, i.e. the first data record of a data record collection, and to continue in a forward iteration loop by the method **NextDataRecord** for the data records following. If there is no data record in the data record collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataRecord	First data record

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
Set dataRecord = dataRecordCltn.FirstDataRecord
```

GetNewUniqueID

Method of VcDataRecordCollection

By this method you can have a unique ID generated for a data record. This method is useful if you wish to add a data record for example by the method **Add** but do not wish to create the ID manually.

	Data Type	Explanation
Return value	Long	New data record ID

NextDataRecord

Method of VcDataRecordCollection

This method can be used in a forward iteration loop to retrieve subsequent data records from a data record collection after initializing the loop by the method **FirstDataRecord**. If there is no data record left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataRecord	Subsequent data record

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection

VcNet1.SuspendUpdate True

Set dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.UpdateDataRecord
   Set dataRecord = dataRecordCltn.NextDataRecord
Wend

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	Object	Content of the data record (as an array or a string)
Return value	Boolean	True

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection

VcNet1.SuspendUpdate True

Set dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.UpdateDataRecord
   Set dataRecord = dataRecordCltn.NextDataRecord
Wend

VcNet1.SuspendUpdate False
VcNet1.EndLoading()
```

Update

Method of VcDataRecordCollection

This method updates a data record in the the data record collection if it previously was created by the **Add()** method. If the data record to be updated does not exist, it will then be created by the **Update** method. Also see **VcDataRecordCollection.Add()**.

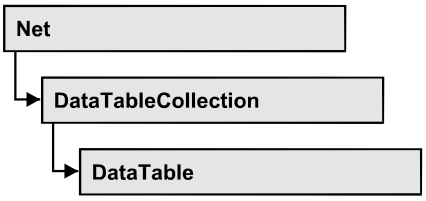
After updating the data record, the method **VcNet.EndLoading** needs to be invoked to make the modification take effect.

	Data Type	Explanation
Parameter:		
⇨ dataRecordContent	Object	Content of the data record (as an array or a string)
Return value	Boolean	Update successful (True) / not successful (False)

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
Set dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Update("1;1.8.2017;;8")
VcNet1.EndLoading()
```

7.19 VcDataTable



A data table comprises **data records**, including their data fields and their contents, and it comprises the descriptions of the record fields, which are called **data table fields**. Data records and data table fields can be processed and iterated over by collection objects.

Data tables on their hand can be processed by a collection object of their own.

Properties

- DataRecordCollection
- DataTableFieldCollection
- Description
- MultiplePrimaryKeysAllowed
- Name

Properties

DataRecordCollection

Read Only Property of VcDataTable

This property returns the DataRecordCollection object of the data table. The collection contains all existing data records of a table. It is empty on the start of the program.

	Data Type	Explanation
Property value	VcDataRecordCollection	DataRecordCollection object

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcNet1.DataTableCollection.FirstDataTable()
MsgBox dataTable.DataRecordCollection.Count
```

DataTableFieldCollection

Read Only Property of VcDataTable

This property returns the DataTableFieldCollection object of the data table. The collection contains the definitions of the fields of a data record of the table. On the start of the program, it holds the data fields that were defined at design time. More data fields can be added at run time by the method **Add** of the object **DataTableFieldCollection**. The definition of data table fields needs to be terminated before data records are filled in the table.

	Data Type	Explanation
Property value	VcTableFieldCollection	DataTableFieldCollection object

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
MsgBox dataTable.DataTableFieldCollection.Count
```

Description

Property of VcDataTable

This property lets you set or retrieve the description of the data table. Names of objects, for example of the table, that contain some information on the object, often are long and cannot be displayed fully in previews; so their benefit is limited. To use the opportunity of short names without having to abandon the information of a long name, you can store additional information to this field. Its contents will be displayed in the data table dialog.

	Data Type	Explanation
Property value	String	Description of the data table **Default value:** Empty string

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataTable.Description = "This table contains data for nodes"
```

MultiplePrimaryKeysAllowed

Property of VcDataTable

This property lets you set or retrieve whether using a composed primary keys is permited.

	Data Type	Explanation
Property value	Boolean	Use of composite primary keys allowed (true)/not allowed (false) **Default value:** False

Name

Property of VcDataTable

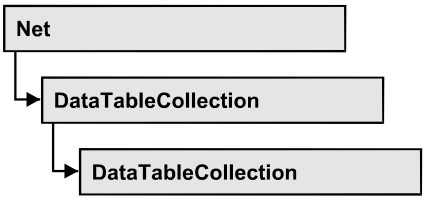
This property lets you set or retrieve the name of the data table. The name of a data table has to set by obligation; beside, it has to be unique. An empty character string is not allowed. Upper and lower case characters are accepted as different. By the method **DataTableByName** of the object **DataTable-Collection** you can retrieve a reference to the data table object.

	Data Type	Explanation
Property value	String	Name of the data table **Default value:** Empty string

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
MsgBox dataTable.Name
```

7.20 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, which is equivalent to an update.

Properties

- _NewEnum
- Count

Methods

- Add
- Copy
- DataTableByIndex
- DataTableByName
- FirstDataTable
- NextDataTable
- Update

Properties

_NewEnum

Property of VcDataTableCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data tables. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
For Each dataTable In dataTableCltn
   List1.AddItem (dataTable.Name)
Next
```

Count

Property of VcDataTableCollection

This property lets you retrieve the number of data tables in the DataTable-Collection object.

	Data Type	Explanation
Property value	Long	Number of data tables in the collection object

Example Code

```
Dim dataTableCltn As VcDataTableCollection

Set dataTableCltn = VcNet1.DataTableCollection
MsgBox (dataTableCltn.Count)
```

Methods

Add

Method of VcDataTableCollection

By this method you can create a data table as a member of the DataTable-Collection. If the name was not used before,an object of the type **VcData-Table** will be returned; otherwise **Nothing** in Visual Basic or **0** in other languages. Only if the DummyObjec3 property **ExtendedDataTables** is set to **True**, tables can be added. In total, 90 data tables can be added at maximum.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String	Name of the new data table
Return value	VcDataTable	Data table generated

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.Add("Resources")
dataTableCltn.Update
```

Copy

Method of VcDataTableCollection

This method lets you copy a data table. Probably existing data records are not copied, just the definition fields. Only if the VcNet property **ExtendedData-Tables** was set to **True**, data tables can be copied. If the data table could be copied, a new object of the type **VcDataTable** will be returned; otherwise **Nothing** in Visual Basic or **0** in other languages. The table names are case sensitive.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String	Name of the data table to be copied (source table)
⇨ newDataTableName	String	Name of the data table to be generated (target table)
Return value	VcDataTable	Data table object generated

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.Copy("Resources", "NewResources")
dataTableCltn.Update
```

DataTableByIndex

Method of VcDataTableCollection

This method lets you access a data table by its index. The index of the first table is 0. If a data table of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic or **0** in other languages).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the data table
Return value	VcDataTable	Data table object returned

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.DataTableByIndex(2)
MsgBox (dataTable.Name)
```

DataTableByName

Method of VcDataTableCollection

This method lets you access a data table by its name. If a data table of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic or **0** in other languages).

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String	Name of the data table
Return value	VcDataTable	Data table object returned

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.DataTableByName("Resources")
MsgBox (dataTable.Description)
```

FirstDataTable

Method of VcDataTableCollection

This method can be used to access the initial value, i.e. the first data table of a data table collection, and to continue in a forward iteration loop by the method **NextDataTable** for the data tables following. If there is no data table in the data table collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTable	First data table

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.FirstDataTable
```

NextDataTable

Method of VcDataTableCollection

This method can be used in a forward iteration loop to retrieve subsequent data tables from a data table collection after initializing the loop by the method **FirstDataTable**. If there is no data table left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTable	Subsequent data table

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable
Dim i As Integer

Set dataTableCltn = VcNet1.DataTableCollection
Set dataTable = dataTableCltn.FirstDataTable
For i = 0 To dataTableCltn.Count
   List1.AddItem (dataTable.Name)
   Set dataTable = dataTableCltn.NextDataTable
Next i
```

Update

Method of VcDataTableCollection

This method lets you update recent modifications of the data structures. It makes the modifications on data table definitions and on data table fields become operative in the VARCHART component and avoids individual updates after several modifications.

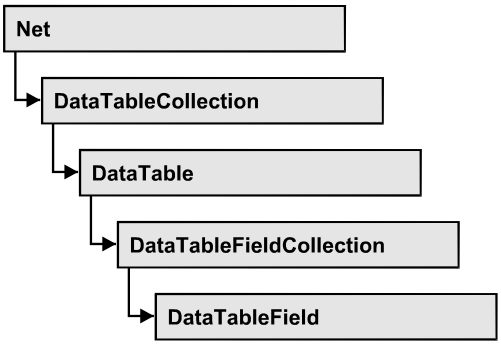
	Data Type	Explanation
Return value	Boolean	Update successful (True) / not successful (False)

Example Code

```
Dim dataTableCltn As VcDataTableCollection
```

```
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTable.DataTableFieldCollection.Add ("Id")
dataTableCltn.Update
```

7.21 VcDataTableField



An object of the type **VcDataTableField** defines the properties of a data field in a data record. Part of the definition of a data table field are its name, its data type and whether it represents the primary key, by which a data record can be uniquely identified. For example, by referring to the primary key, other data tables can relate to a data table. To create a relation, a table needs to specify the primary key of a different table by the property **RelationshipFieldIndex**.

The DataTableField objects of a data table are administered by the object **DataTableFieldCollection**.

Properties

- DataTableName
- DateFormat
- Editable
- Hidden
- Index
- Name
- PrimaryKey
- RelationshipFieldIndex
- Type

Properties

DataTableName

Read Only Property of VcDataTableField

This property lets you retrieve the name of the associated data table.

	Data Type	Explanation
Property value	String	Name of the data table

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
MsgBox dataTable.DataTableFieldCollection.FirstDataTableField.DataTableName
```

DateFormat

Property of VcDataTableField

This property lets you set or retrieve the date format of the record field that is specified by the property **RelationshipFieldIndex**. The date format is used when reading or storing CSV files and when the format type **String** is used when adding a data record by the method **Add**. This property only works if the data type of the field was set to **vcDataTableFieldDateTime**.

Note: Remember to set the property **Type** before setting the property **DateFormat**.

	Data Type	Explanation
Property value	String	Date format {DMYhms:;./} **Default value:** DD.MM.YYYY hh:mm:ss

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = vcDataTableFieldDateTimeType
'DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY"
```

Editable

Property of VcDataTableField

This property lets you set or retrieve whether the record field should be editable at run time in the chart table and in the dialog **EditNode**.

	Data Type	Explanation
Property value	Boolean	Field editable (True) / not editable (False) **Default value:** True

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
Set 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	Boolean	Field hidden (True) / not hidden (False) **Default value:** False

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
Set 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	Integer	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	String	Name of the field **Default value:** Empty string

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
Set 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	Boolean	The field serves (True) / does not serve (False) as a primary key. **Default value:** False

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
Dim isPrimaryKey As Boolean

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
Set 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	Long	Index of the record field to which the data definition of the data table field refers. **Default value:** -1

Example Code

```
Dim dataTableTask As VcDataTable
Dim dataTaskFieldId As VcDataTableField
Dim dataTaskFieldName As VcDataTableField
Dim dataTableOperation As VcDataTable
Dim dataOperationFieldId As VcDataTableField
Dim dataOperationFieldName As VcDataTableField
Dim dataOperationFieldTaskId As VcDataTableField

'Create table Task
dataTableTask = VcNet1.DataTableCollection.Add("Task")
dataTaskFieldId = dataTableTask.DataTableFieldCollection.Add("Id")
dataTaskFieldId.PrimaryKey = True
dataTaskFieldName = dataTableTask.DataTableFieldCollection.Add("Name")
dataTaskFieldName.Type = vcDataTableFieldAlphanumericType

'Create table Operation
dataTableOperation = VcNet1.DataTableCollection.Add("Operation")
dataOperationFieldId = dataTableOperation.DataTableFieldCollection.Add("Id")
dataOperationFieldId.PrimaryKey = True
dataOperationFieldName = dataTableOperation.DataTableFieldCollection.Add("Name")
dataOperationFieldName.Type = vcDataTableFieldAlphanumericType
dataOperationFieldTaskId =
dataTableOperation.DataTableFieldCollection.Add("TaskId")
dataOperationFieldTaskId.Type = vcDataTableFieldIntegerType

'Link tables Task and Operations
dataOperationFieldTaskId.RelationshipFieldIndex =
VcNet1.DetectFieldIndex("Task", "Id")
```

Type

Property of VcDataTableField

This property lets you set or retrieve the data type of the field.

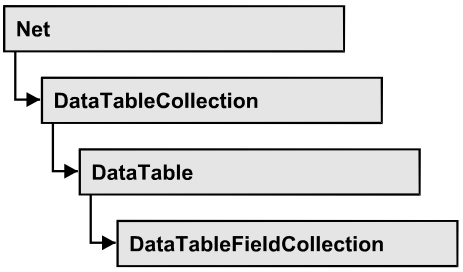
Note: Setting the property **Type** may change the property **DateFormat**. By setting this property to **vcDataTableAlphanumeric** or to **vcDataTable-FieldInteger** the date format probably set will change to "".

	Data Type	Explanation
Property value	DataTableFieldTypeEnum	Data type of the field, can contain 512 characters maximum **Default value:** VcDataTableFieldIntegerType

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
Set dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = vcDataTableFieldDateTimeType
VcNet1.DataTableCollection.Update
```

7.22 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 **FirstData-Field** and **NextDataField** allow to access data fields by iteration while by **DataFieldByName** and **DataFieldByIndex** single data fields can be accessed. **Add** and **Copy** represent basic administering methods.

Properties

- _NewEnum
- Count

Methods

- Add
- Copy
- DataTableFieldByIndex
- DataTableFieldByName
- FirstDataTableField
- NextDataTableField

Properties

_NewEnum

Property of VcDataTableFieldCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all data table fields objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
For Each dataTableField In dataTable.DataTableFieldCollection
   List1.AddItem (dataTableField.Name)
Next
```

Count

Read Only Property of VcDataTableFieldCollection

This property lets you retrieve the number of data table fields in the DataTableFieldCollection object.

	Data Type	Explanation
Property value	Long	Number of data table fields in the collection object

Example Code

```
Dim dataTable As VcDataTable

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
MsgBox ("Number of data fields: " &  dataTable.DataTableFieldCollection.Count)
```

Methods

Add

Method of VcDataTableFieldCollection

By this method you can create a data table field as a member of the DataTableFieldCollection. If the name was not used before, the new data field will be returned; otherwise "Nothing" (Visual Basic) or "0" (other languages) will be returned. You can add at maximum 9,999 fields to a table.

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	String	Name of the data table field to be generated
Return value	VcDataTableField	Data table field generated

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set 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	String	Name of the data table field to be copied (source field)
⇨ newDataTableFieldName	String	Name of the data table field to be generated (target field)
Return value	VcDataTableField	Data table field generated

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set 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	Integer	Index of data table field
Return value	VcDataTableField	Data table field returned

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
```

```
Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByIndex(1)
MsgBox (dataTableField.Name)
```

DataTableFieldByName

Method of VcDataTableFieldCollection

This method lets you access a data table field by its name. If a field of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ dataTableFieldName	String	Name of data table field
Return value	VcDataTableField	Data table field returned

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.DataTableFieldBy("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

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataTableField = dataTable.DataTableFieldCollection.FirstDataTableField
```

NextDataTableField

Method of VcDataTableFieldCollection

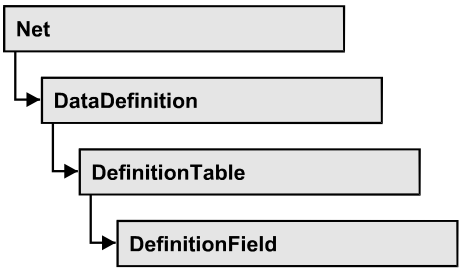
This method can be used in a forward iteration loop to retrieve subsequent data table fields from a data table field collection after initializing the loop by the method **FirstDataTableField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcDataTableField	Subsequent data table field

Example Code

```
Dim dataTable As VcDataTable
Dim dataTableFieldCltn As VcDataTableFieldCollection
Dim dataTableField As VcDataTableField
Dim i As Integer

Set dataTable = VcNet1.DataTableCollection.FirstDataTable
Set dataTableFieldCltn = dataTable.DataTableFieldCollection
Set dataTableField = dataTableFieldCltn.FirstDataTableField
For i = 0 To dataTableFieldCltn.Count
   List1.AddItem (dataTableField.Name)
   Set dataTableField = dataTableFieldCltn.NextDataTableField
Next i
```

7.23 VcDefinitionField



An object of the type VcDefinitionField defines a field of the data definition table. The definition basically consists of a name and a data type.

Properties

- DateFormat
- Editable
- Hidden
- ID
- Name
- Type

Properties

DateFormat

Property of VcDefinitionField

This property lets you set or retrieve the date format of the field of a data definition table. This property only works if the data type of the field was set to **vcDataTableFieldDateTime**. The dateFormat setting is used when reading or storing CSV files and when the format type **String** is used when adding a data record by the methods **InsertNodeRecord** or **InsertLink-Record** of the VcNet object. The format of the date output in the chart is controlled by the VcNet property **DateOutputFormat**.

Note: You should set the property Type first before setting the property DateFormat.

	Data Type	Explanation
Property value	String	Date format {DMYhms:;./} **Default value:** bei vcDefFieldDateTime DD.MM.YYYY hh:mm:ss

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
dataDefField.Type = vcDefFieldDateTimeType
'DateFormat = "DD.MM.YYYY"
dataDefField.DateFormat = "01.12.2014"
```

Editable

Property of VcDefinitionField

This property lets you set or retrieve whether the data field should be editable at run time in the chart table and in the dialog **EditNode**.

	Data Type	Explanation
Property value	Boolean	Definition field editable/not editable **Default value:** True

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
dataDefField.Editable = False
```

Hidden

Property of VcDefinitionField

This property lets you require/set whether a data field is hidden at run time.

	Data Type	Explanation
Property value	Boolean	Definition field hidden/not hidden **Default value:** False

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField
```

```
Set dataDefTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
dataDefField.Hidden = True
```

ID

Read Only Property of VcDefinitionField

This property lets you retrieve the index of the field of a data definition table.

	Data Type	Explanation
Property value	Integer	Index of the definition field

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.FieldByName("Start")
MsgBox dataDefField.ID
```

Name

Property of VcDefinitionField

This property lets you set or retrieve the name of the field of a data definition table.

	Data Type	Explanation
Property value	String	Name of the definition field

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.CreateDataField("Start")
```

Type

Property of VcDefinitionField

This property lets you set or retrieve the type of the field of a data definition table.

Note: By setting the property **Type** the property **DateFormat** will change!

vcDefFieldAlphanumericType: DateFormat = ""

vcDefFieldDateTimeType: DateFormat = "DD.MM.YYYY hh:mm:ss"

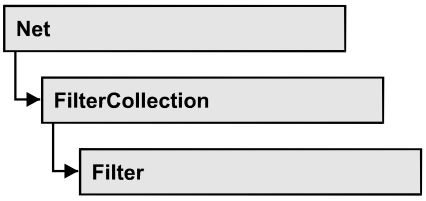
vcDefFieldIntegerType: DateFormat = ""

	Data Type	Explanation
Property value	DefinitionFieldTypeEnum	type of the definition field **Default value:** vcDefFieldIntegerType
	Possible Values: vcDefFieldAlphanumericType 1 vcDefFieldDateTimeType 4 vcDefFieldIntegerType 2	 Data type **alphanumeric**: "" Data type **date** : DD.MM.YYYY Data type **integer** (32 bits): ""

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDefinitionField

Set dataDefTable = VcNet1.DataDefinition.DefinitionTable(vcMaindata)
Set dataDefField = dataDefTable.CreateDataField("Start")
dataDefField.Type = vcDefFieldDateTimeType
```

7.24 VcFilter



An object of the type VcFilter contains subconditions (VcFilterSubCondition), e.g. permitted values to be compared to the data fields of a node or a link, so that the filter conditions may or may not apply to an object. Filters are used p.e. to assign a format to an activity. Only if the filter is valid after the subconditions have been modified, the modified subconditions will become valid. Otherwise the former filter subconditions will remain valid. This can be controlled via the methods VcFilter.IsValid and VcFilterSubCondition.IsValid.

Properties

- _NewEnum
- DataDefinitionTable
- DatesWithHourAndMinute
- Name
- Specification
- StringsCaseSensitive
- SubCondition
- SubConditionCount

Methods

- AddSubCondition
- CopySubCondition
- Evaluate
- IsValid
- RemoveSubCondition

Properties

_NewEnum

Read Only Property of VcFilter

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all filter condition objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim fiSuCo As VcFilterSubCondition

For Each fiSuCo In filter
    Debug.Print fiSuCo.Index
Next
```

DataDefinitionTable

Property of VcFilter

This property lets you enquire whether the filter is a filter for nodes (vcMainData) or for links (vcRelations). This property can be modified only if the filter does not contain subconditions.

	Data Type	Explanation
Property value	DataTableEnum	Type of data definition table
	Possible Values:	
	vcMaindata 0	Table type **vcMaindata** (for nodes)
	vcRelations 1	Table type **vcRelations** (for links)

DatesWithHourAndMinute

Property of VcFilter

This property lets you enquire/set whether the comparison of subconditions that contain dates checks the information on hours and minutes. The setting

can only be modified when there is at least one subcondition containing a date comparison. Otherwise the property value is always False.

	Data Type	Explanation
Property value	Boolean	hours and minutes are compared (True)/ not compared (False)

Name

Property of VcFilter

This property lets you enquire/set the name of the filter.

	Data Type	Explanation
Property value	String	Name of the filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcNet1.FilterCollection

For Each filter In filterCltn
    ListBox.AddItem filter.name
Next filter
```

Specification

Read Only Property of VcFilter

This property lets you retrieve the specification of a filter. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a filter by the method **VcFilterCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the filter

StringsCaseSensitive

Property of VcFilter

This property lets you enquire/set whether subconditions that contain strings are case-sensitive.

	Data Type	Explanation
Property value	Boolean	case-sensitive (True)/not case-sensitive (False)

SubCondition

Property of VcFilter

This property lets you access a VcFilterSubCondition object by its index.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the filter subcondition {0 ... VcFilter.SubConditionCount-1}
Property value	VcFilterSubCondition	filter subcondition object

SubConditionCount

Read Only Property of VcFilter

This property lets you enquire the number of filter subconditions.

	Data Type	Explanation
Property value	Integer	number of filter subconditions

Methods

AddSubCondition

Method of VcFilter

This method lets you create a new filter condition in the collection of the filter conditions. Its position is specified by the index. The corresponding VcFilterSubCondition object will be returned.

Default properties of this object:

- DataFieldIndex: -1
- Operator: vcInvalidOp
- ComparisonValueAsString: "<INVALID>"
- ConnectionOperator: vcInvalidConnOp.

	Data Type	Explanation
Parameter:		
⇨ atIndex	Integer	Index of the new filter subcondition {0 to VcFilter.SubConditionCount and –1 for "at the end of the Collection" (identical with the value VcFilter.SubConditionCount)}
Return value	VcFilterSubCondition	Filter subcondition object

CopySubCondition

Method of VcFilter

This method lets you copy a filter subcondition by its index. The new filter subcondition will be inserted into the collection at the position specified by the index. It will be returned as a VcFilterSubCondition object.

	Data Type	Explanation
Parameter:		
⇨ fromIndex	Integer	Index of the filter subcondition to be copied

⇨ atIndex	Integer	Index of the new filter subcondition {0 to VcFilter.SubConditionCount and –1 for "at the end of the Collection" (identical with the value VcFilter.SubConditionCount)}
Return value	VcFilterSubCondition	Filter subcondition object

Evaluate

Method of VcFilter

This methods lets you check whether the specified filter applies for a certain data record or not. You should only pass objects that are internally linked with data records of the data tables. Those are **VcNode, VcLink, VcGroup, VcDataRecord**. If an object is passed that is not listed, an exception will be triggered.

	Data Type	**Explanation**
Parameter:		
⇨ dataObjectParam	Variant	Data record object
Return value	Boolean	Filter applies for data record (True)/does not apply (False)

IsValid

Method of VcFilter

This property checks whether all filter subconditions are correct. The correctness of all subconditions is the condition that changed filter subconditions become valid. Otherwise the former subconditons will remain valid.

	Data Type	**Explanation**
Return value	Boolean	Filter subconditions correct (True)/ not correct (False)

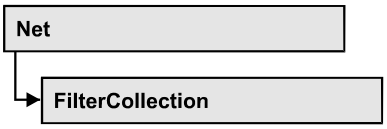
RemoveSubCondition

Method of VcFilter

This method lets you delete a filter subcondition by its index.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the filter subcondition to be removed

7.25 VcFilterCollection



An object of the type VcFilterCollection automatically contains all available filters. You can access all objects in an iterative loop by **For Each filter In FilterCollection** or by the methods **First...** and **Next...**. You can access a single filter using the methods **FilterByName** and **FilterByIndex**. The number of filters in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the filters in the corresponding way.

Properties

- _NewEnum
- Count
- MarkedNodesFilter

Methods

- Add
- AddBySpecification
- Copy
- FilterByIndex
- FilterByName
- FirstFilter
- NextFilter
- Remove

Properties

_NewEnum

Read Only Property of VcFilterCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all filter objects contained. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET

languages the method GetEnumerator is offered instead. Some development environments replace this property by own language constructs.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim filter As VcFilter

For Each filter In VcNet1.FilterCollection
   Debug.Print filter.Name
Next
```

Count

Read Only Property of VcFilterCollection

This property lets you retrieve the number of filters in the filter collection.

	Data Type	Explanation
Property value	Long	Number of filters

Example Code

```
Dim filterCltn As VcFilterCollection
Dim numberOfFilters As Long

Set filterCltn = VcNet1.FilterCollection
numberOfFilters = filterCltn.Count
```

MarkedNodesFilter

Read Only Property of VcFilterCollection

This property lets you retrieve a constant pseudo-filter that can be used only for **ActiveNodeFilter** for filtering the nodes currently marked (sub-diagram).

	Data Type	Explanation
Property value	VcFilter	Pseudo filter

Example Code

```
Set 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 has not been used before, the new filter object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

The new filter automatically refers to the data definition table vcMainData (see VcFilter.DataDefinitionTable). You can select vcRelations instead, as long as the filter does not contain any subconditions.

	Data Type	Explanation
Parameter:		
⇨ newName	String	Filter name
Return value	VcFilter	New filter object

Example Code

```
Set 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	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	String	Name of the filter to be copied
⇨ newName	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 of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the 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	String	Filter name
Return value	VcFilter	Filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcNet1.FilterCollection
Set filter = filterCltn.FilterByName("Department A")
```

FirstFilter

Method of VcFilterCollection

This method can be used to access the initial value, i.e. the first filter of a filter collection, and then to continue in a forward iteration loop by the method **NextFilter** for the filters following. If there is no filter in the FilterCollection object, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcFilter	First filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcNet1.FilterCollection
Set filter = filterCltn.FirstFilter
```

NextFilter

Method of VcFilterCollection

This method can be used in a forward iteration loop to retrieve subsequent filters from a curve collection after initializing the loop by the method **FirstFilter**. If there is no filter left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcFilter	Subsequent filter

Example Code

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

Set filterCltn = VcNet1.FilterCollection
Set filter = filterCltn.FirstFilter

While Not filter Is Nothing
    Listbox.AddItem filter.Name
    Set filter = filterCltn.NextFilter
Wend
```

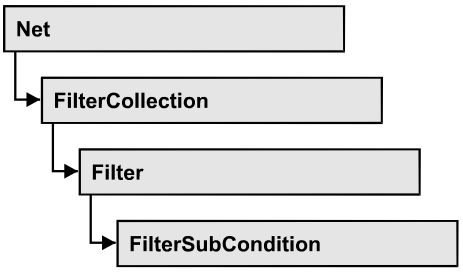
Remove

Method of VcFilterCollection

This method lets you delete a filter. If the filter is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	**Explanation**
Parameter:		
⇨ name	String	Filter name
Return value	Boolean	Filter deleted (True)/not deleted (False)

7.26 VcFilterSubCondition



An object of the type VcFilterSubCondition contains a single filter subcondition. It does not have a name, but only an index that specifies its position in the filter.

In the **Edit Filter** dialog each line corresponds to a subcondition. The properties specified at design time in that dialog can be modified via the API at runtime.

Properties

- ComparisonValueAsString
- ConnectionOperator
- DataFieldIndex
- FilterName
- Index
- Operator

Methods

- IsValid

Properties

ComparisonValueAsString

Property of VcFilterSubCondition

This property lets you enquire/set the comparison value. This string must have the following format:

- String: included by double quotation marks. Example in VB: """Aachen"""; Example in C/C++: "\"Aachen\""

- Date: included by # signs. Example: "#18.06.2015; 12:34:56;#" (as this is the control's default format that is independent of the operating system and its local settings the date format is always "DD.MM.YYYY;hh:mm:ss;". A special date comparison value is "<TODAY>".

- Date field: included by square brackets. Example: "[ID]"

- Number: entered directly. Example: "52076"

- List: for a vc...In operator: included by {} brackets. All values included must have the same type (string, date or number). They may have one of the formats mentionned above. Example: "{"NETRONIC", [Name]}"

- Invalid (e.g. after creating a subcondition): "<INVALID>"

The type of the comparison value has to match the type of the data field and the operator type.

	Data Type	Explanation
Property value	String	Comparison value

ConnectionOperator

Property of VcFilterSubCondition

This property lets you enquire/set the operator for the connetion with the following subcondition. **vcAnd** binds stronger than **vcOr**.

	Data Type	Explanation
Property value	ConnectionOperatorEnum **Possible Values:** vcAnd 1 vcInvalidConnOp 0 vcOr 2	Operator to connect to the subsequent condition And operator invalid operator Or operator

DataFieldIndex

Property of VcFilterSubCondition

This property lets you enquire/set 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	Long	Index of the data field to be compared

FilterName

Read Only Property of VcFilterSubCondition

This property lets you enquire the name of the filter to which this subcondition belongs to.

	Data Type	Explanation
Property value	String	Name of the filter

Index

Read Only Property of VcFilterSubCondition

This property lets you enquire the index of this subcondition in the corresponding filter.

	Data Type	Explanation
Property value	Integer	Index of the subcondition in the filter

Operator

Property of VcFilterSubCondition

This property lets you set or retreive the comparison operator. The operators that are available in the API correspond to the operators in the **Edit Filter** dialog. The operator type has to match the types of the data field and of the comparison value.

	Data Type	Explanation
Property value	OperatorEnum **Possible Values:** vcDateEarlier 27 vcDateEarlierOrEqual 28 vcDateEqual 25 vcDateIn 31 vcDateLater 29 vcDateLaterOrEqual 30 vcDateNotEqual 26 vcDateNotIn 32 vcIntEqual 9 vcIntGreater 13 vcIntGreaterOrEqual 14 vcIntIn 15 vcIntLess 11 vcIntLessOrEqual 12 vcIntNotEqual 10 vcIntNotIn 16 vcInvalidOp 0 vcStringBeginsWith 3 vcStringContains 5 vcStringEqual 1 vcStringIn 7 vcStringNotBeginsWith 4 vcStringNotContains 6 vcStringNotEqual 2 vcStringNotIn 8	comparison operator date earlier than date earlier than or equal date equal date in date later than date later than or equal date not equal date not in integer equal integer greater integer greater or equal integer in integer smaller than integer smaller than or equal integer not equal integer not in invalid operator string begins with string contains string equal string contains string does not begin with string does not contain string is not equal string is not in

Methods

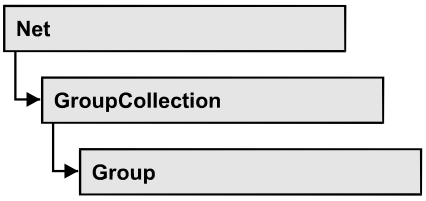
IsValid

Method of VcFilterSubCondition

This property checks whether the filter subcondition is correct.

	Data Type	Explanation
Return value	Boolean	Filter subcondition correct (True)/ not correct (False)

7.27 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

- BackColor
- Collapsed
- LineColor
- LineThickness
- LineType
- Name
- NodeCollection
- Title
- TitleLineCount
- X
- Y

Methods

- SetXY

Properties

BackColor

Property of VcGroup

This property lets you set or retrieve a background color to a group. The default color is white.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcNet1.GroupCollection
Set group = groupCltn.FirstGroup

group.BackColor = RGB(128, 128, 128)
```

Collapsed

Property of VcGroup

This property lets you set or retrieve, whether (True) or not (False) a group is collapsed. This property can only be used in the clustering mode (GroupMode = vcGMClustering). The property also can be set interactively when the property VcNet.GroupInteractionsAllowed was set.

	Data Type	Explanation
Property value	Boolean	Group collapsed/expanded

Example Code

```
Private Sub VcNet1_OnGroupLClick(ByVal group As VcNetLib.VcGroup,  _
                                 ByVal x As Long, ByVal y As Long,  _
                                 returnStatus As Variant)

   If group.Collapsed = False Then
      group.Collapsed = True
   Else
      group.Collapsed = False
   End If

End Sub
```

LineColor

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
Parameter: ⇨ Rückgabewert	Color	RGB color values ({0...255},{0...255},{0...255})
Property value	System.Drawing.Color	RGB color values ({0...255},{0...255},{0...255})

LineThickness

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	Integer	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined in the dialog

LineType

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	LineTypeEnum	Line type
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5
	vcLineType6 106	Line Type 6
	vcLineType7 107	Line Type 7
	vcLineType8 108	Line Type 8
	vcLineType9 109	Line Type 9
	vcNone 1	No line type
	vcNotSet -1	No line type assigned
	vcSolid 2	Line solid

Name

Read Only Property of VcGroup

This property lets you retrieve the name of a group (= the value of the grouping field GroupField).

	Data Type	Explanation
Property value	String	Group name

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String

Set groupCltn = VcNet1.GroupCollection
Set group = groupCltn.FirstGroup

groupName  = group.Name
```

NodeCollection

Read Only Property of VcGroup

This property gives access to each node of a group.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code

```
Dim groupCollection As VcGroupCollection
Dim group As VcGroup
Dim nodeCollection As VcNodeCollection
Dim numberOfNodes As Integer

Set groupCollection = VcNet1.GroupCollection
Set group = groupCollection.FirstGroup
Set nodeCollection = group.NodeCollection

nodeCollection.SelectNodes (vcAll)
numberOfNodes = nodeCollection.Count

Dim groupCollection As VcGroupCollection
Dim group As VcGroup
Dim nodeCollection As VcNodeCollection

Dim name As String
Dim number As Long

Set groupCollection = VcNet1.GroupCollection
Set group = groupCollection.FirstGroup
Set nodeCollection = group.NodeCollection

name = group.Name
number = nodeCollection.Count
```

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	String	Title of the Group

Example Code

```
Dim groupCollection As VcGroupCollection
Dim group As VcGroup
Dim nodeCollection As VcNodeCollection
Dim groupTitle As String

Set groupCollection = VcNet1.GroupCollection
Set group = groupCollection.FirstGroup

groupTitle =  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	Integer 1 ... 5	number of lines of the title text **Default value:** 1

Example Code

```
Dim groupCollection As VcGroupCollection
Dim group As VcGroup
Dim nodeCollection As VcNodeCollection

Set groupCollection = VcNet1.GroupCollection
Set group = groupCollection.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	Long	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	Long	Y coordinate

Methods

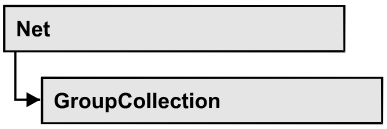
SetXY

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	Long	X coordinate in band numbers
⇨ y	Long	Y coordinate in band numbers
Return value	Boolean	values set successfully (True)/not set successfully (False)

7.28 VcGroupCollection



If nodes were grouped, an object of the type VcGroupCollection contains all available groups.You can access all objects in an iterative loop by **For Each group In GroupCollection** or by the methods **First...** and **Next...**. You can access a single group using the method **GroupByName**. The number of groups in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstGroup
- GroupByName
- NextGroup

Properties

_NewEnum

Read Only Property of VcGroupCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all group objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim group As VcGroup

For Each group In VcNet1.GroupCollection
```

```
   Debug.Print group.Name
Next
```

Count

Read Only Property of VcGroupCollection

This property lets you retrieve the number of groups in the group collection.

	Data Type	Explanation
Property value	Long	Number of nodes

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim numberOfGroups As Integer

Set groupCltn = VcNet1.GroupCollection
numberOfGroups = groupCltn.Count
```

Methods

FirstGroup

Method of VcGroupCollection

This method can be used to access the initial value, i.e. the first group of a group collection, and then to continue in a forward iteration loop by the method **NextGroup** for the groups following. If there is no group in the group collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcGroup	First group of the GroupCollection

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcNet1.GroupCollection
Set group = groupCltn.FirstGroup
```

GroupByName

Method of VcGroupCollection

By this method you can get a group by its name. Beforehand, the group needs to be selected 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	String	Name of group
Return value	VcGroup	Group

Example Code

```
Dim groupCollection As VcGroupCollection
Dim group As VcGroup

Set groupCollection = VcNet1.GroupCollection
Set group = groupCollection.GroupByName("Group A")
```

NextGroup

Method of VcGroupCollection

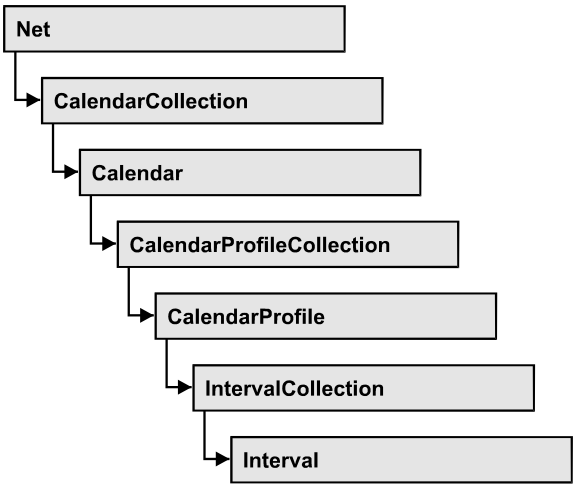
This method can be used in a forward iteration loop to retrieve subsequent groups from a group collection after initializing the loop by the method **FirstGroup**. If there is no group left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcGroup	Subsequent group

Example Code

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

Set groupCltn = VcNet1.GroupCollection
Set group = groupCltn.FirstGroup

While Not group Is Nothing
    List1.AddItem group.Name
    Set group = groupCltn.NextGroup
Wend
```

7.29 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	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	Integer	Day of last month

DayInStartMonth

Property of VcInterval

This property returns or sets the day in the start month of this interval (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Integer	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	Date	End date and time of interval

EndMonth

Property of VcInterval

This property returns or sets the end month of this interval object (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	MonthEnum **Possible Values:** vcApril 4 vcAugust 8 vcDecember 12 vcFebruary 2 vcJanuary 1 vcJuly 7 vcJune 6 vcMarch 3 vcMay 5 vcNovember 11 vcOktober 10 vcSeptember 9	End month of interval **April** **August** **December** **February** **January** **July** **June** **March** **May** **November** **October** **September**

EndTime

Property of VcInterval

This property returns or sets the end time of this interval object (for profiles of the type **vcDayProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Date	End time of interval

EndWeekday

Property of VcInterval

This property returns or sets the last weekday of this interval object (for profiles of the type **vcWeekProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	WeekdayEnum **Possible Values:** vcFriday 5 vcMonday 1 vcSaturday 6 vcSunday 7 vcThursday 4 vcTuesday 2 vcWednesday 3	Last weekday of interval Week day **Friday** Week day **Monday** Week day **Saturday** Week day **Sunday** Week day **Thursday** Week day **Tuesday** Week day **Wednesday**

Name

Read Only Property of VcInterval

This property lets you retrieve the name of the interval. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	String	Name of the interval

Specification

Read Only Property of VcInterval

This property lets you retrieve the specification of an interval. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored smoothly to text files or data bases. This allows for persistency. A specification can be used to create an interval by the method **VcInterval-Collection.AddBySpecification**.

	Data Type	Explanation
Property value	String	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	Date	Start date and time of interval

StartMonth

Property of VcInterval

This property returns or sets the start month of this interval object (for profiles of the type **vcYearProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	MonthEnum **Possible Values:** vcApril 4 vcAugust 8 vcDecember 12 vcFebruary 2 vcJanuary 1 vcJuly 7 vcJune 6 vcMarch 3 vcMay 5 vcNovember 11 vcOktober 10 vcSeptember 9	Start month of interval **April** **August** **December** **February** **January** **July** **June** **March** **May** **November** **October** **September**

StartTime

Property of VcInterval

This property returns or sets the start time of this interval object (for profiles of the type **vcDayProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	Date	Start time of interval

StartWeekday

Property of VcInterval

This property returns or sets the first weekday of this interval object (for profiles of the type **vcWeekProfile** only). This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	WeekdayEnum **Possible Values:** vcFriday 5 vcMonday 1 vcSaturday 6 vcSunday 7 vcThursday 4 vcTuesday 2 vcWednesday 3	Start weekday of interval Week day **Friday** Week day **Monday** Week day **Saturday** Week day **Sunday** Week day **Thursday** Week day **Tuesday** Week day **Wednesday**

Type

Property of VcInterval

This property lets you enquire the type of the interval. This feature can also be set in the **Administrate Intervals** dialog.

	Data Type	Explanation
Property value	IntervalTypeEnum	Type of the interval

Methods

PutInOrderAfter

Method of VcInterval

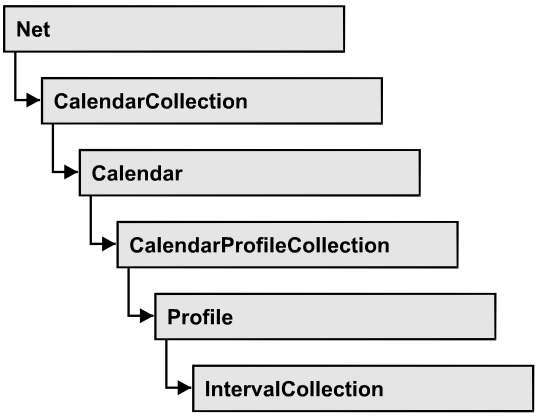
This method lets you set the interval behind an interval specified by name, within the IntervalCollection. If you set the name to "", the interval will be put in the first position. The order of the intervals within the collection determines the order by which they apply to the calendars.

	Data Type	Explanation
Parameter: refNameParam	String	Name of the interval behind which the current interval is to be put.
Return value	Void	

Example Code

```
Dim intvlCltn As VcIntervalCollection
Dim intvl1 As VcInterval
Dim intvl2 As VcInterval

intvlCltn = VcGantt1.IntervalCollection()
intvl1 = intvlCltn.Add("intvl1")
intvl2 = intvlCltn.Add("intvl2")
intvl1.PutInOrderAfter("intvl2")
intvlCltn.Update()
```

7.30 VcIntervalCollection



The VcIntervalCollection object contains all intervals available. You can access all objects in an iterative loop by **For Each Interval In BoxFormatCollection** or by the methods **First...** and **Next...**. You can access a single interval by the methods **IntervalByName** and **ntervalByIndex**. The number of intervals in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the intervals in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstInterval
- IntervalByIndex
- IntervalByName
- NextInterval
- Remove
- Update

Properties

_NewEnum

Property of VcIntervalCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all interval objects contained. In Visual Basic this property never is displayed, but it can be addressed by the command **For Each** *element* **In** *collection*. In .NET languages the method GetEnumerator is offered instead. Some development environments replace this property by own language constructs.

	Data Type	Explanation
Property value	Object	Reference object

Count

Read Only Property of VcIntervalCollection

This property lets you retrieve the number of intervals in the interval collection.

	Data Type	Explanation
Property value	Long	Number of Interval objects

Methods

Add

Method of VcIntervalCollection

By this method you can create an interval as a member of the IntervalCollection. If the name has not been used before, the new interval object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ intervalName	String	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 the interval can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ Specification	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	String	Name of the interval to be copied
⇨ newIntervalName	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	Integer	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	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	String	Interval name
Return value	Boolean	interval deleted (True)/not deleted (False)

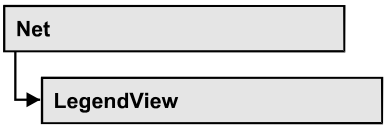
Update

Method of VcIntervalCollection

This method lets you update an interval collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

7.31 VcLegendView



An object of the type **VcWorldView** designates the legend view window.

Properties

- Border
- Height
- HeightActualValue
- Left
- LeftActualValue
- ParentHWnd
- ScrollBarMode
- Top
- TopActualValue
- Visible
- Width
- WidthActualValue
- WindowMode

Methods

- Update

Properties

Border

Property of VcLegendView

This property lets you set or retrieve whether the legend view has a frame (not in **vcPopupWindow** mode). he color of the frame is **Color.Black**. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean	Legend view with a border line (True)/without border line (False) **Default value:** True

Example Code

```
VcNet1.LegendView.Mode = vcNotFixed
VcNet1.LegendView.Border = True
```

Height

Property of VcLegendView

This property lets you retrieve the vertical extent of the legend view. In the modes **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Height of the legend view {0, ...} **Default value:** 100

Example Code

```
VcNet1.LegendView.Height = 100
```

HeightActualValue

Read Only Property of VcLegendView

This property lets you retrieve the vertical extent of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/in Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual height of the legend view {0, ...} **Default value:** 100

Example Code

```
VcNet1.LegendView.HeightActualValue = 300
```

Left

Property of VcLegendView

This property lets you retrieve the left position of the legend view. In the modes **vcLVNotFixed** and **vcLVPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Left position of the legend view **Default value:** 0

Example Code

```
VcNet1.LegendView.Left = 200
```

LeftActualValue

Read Only Property of VcLegendView

This property lets you retrieve the left position of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either height or width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual left position of the legend view **Default value:** 0

Example Code

```
VcNet1.LegendView.LeftActualValue = 150
```

ParentHWnd

Property of VcLegendView

In the **vcLVNotFixed** mode, this property lets you set the HWnd handle of the parent window, for example, if the legend view is to appear in a frame window implemented by your own. By default, the frame window is positioned on the HWnd handle of the parent window of the VARCHART ActiveX main window. This property can be used only at run time.

	Data Type	Explanation
Property value	OLE_HANDLE	Handle

Example Code

```
MsgBox (VcNet1.legendview.ParentHWnd)
```

ScrollBarMode

Property of VcLegendView

This property lets you set or retrieve the scroll bar mode of the legend view. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	LegendViewScrollBarModeEnum	Scrollbarmode **Default value:** NoScrollBar
	Possible Values:	
	vcAutomaticScrollBar 3	Display of a horizontal or vertical scrollbar if required.
	vcHorizontalScrollBar 1	Display of a horizontal scrollbar if required.
	vcNoScrollBar 0	The complete chart is displayed without scrollbars.
	vcVerticalScrollBar 2	Display of a vertical scrollbar if required.

Example Code

```
VcNet1.LegendView.ScrollBarMode = vcAutomaticScrollBar
```

Top

Property of VcLegendView

This property lets you retrieve the top position of the legend view. In the modes **vcNotFixed** und **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Top position of the legend view **Default value:** 0

Example Code

```
VcNet1.LegendView.Top = 20
```

TopActualValue

Read Only Property of VcLegendView

This property lets you retrieve the top position of the legend view which actually is displayed. In the modes b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height or the width is preset.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual top position of the legend view **Default value:** 0

Example Code

```
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	Boolean	Legend view visible (True)/not visible (False) **Default value:** False

Example Code

```
VcNet1.LegendView.Visible = True
```

Width

Property of VcLegendView

This property lets you retrieve the horizontal extent of the legend view. In the modes **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Horizontal extension of the legend view {0, ...} **Default value:** 100

Example Code

```
VcNet1.LegendView.Width = 200
```

WidthActualValue

Read Only Property of VcLegendView

This property lets you retrieve the horizontal extent of the legend view which actually is displayed. In the mode b!vcLVFixedAtBottom, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** the actual value may differ from the one that was set because in these modes either the height

or the width is preset. Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual horizontal extension of the legend view {0, ...} **Default value:** 100

Example Code

```
VcNet1.LegendView.WidthActualValue = 600
```

WindowMode

Property of VcLegendView

This property lets you enquire/set the legend view mode. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	LegendViewWindowModeEnum	Mode of the legend view **Default value:** vcPopupWindow
	Possible Values:	
	vcFixedAtBottom 4	The legend view is displayed on the bottom of the VARCHART ActiveX control window. Then the height can be specified, whereas the position and the width are fixed.
	vcFixedAtLeft 1	The legend view is displayed on the left side of the VARCHART ActiveX control window. Then the width can be specified, whereas the position and the height are fixed.
	vcFixedAtRight 2	The legend view is displayed on the right side of the VARCHART ActiveX control window. Then the width can be specified, whereas the position and the height are fixed.
	vcFixedAtTop 3	The legend view is displayed on the top of the VARCHART ActiveX control window. Then the height can be specified, whereas the position and the width are fixed.
	vcNotFixed 5	The legend view is a child window of the current parent window of the VARCHART ActiveX. It can be positioned at any position with any extension. The parent window can be modified via the property **VcWorldView.ParentHWnd**.
	vcPopupWindow 6	The legend view is a popup window with its own frame. The user can modify its position and extension, open it via the default context menu, and close it via the **Close** button in the frame.

Example Code

```
VcNet1.LegendView.WindowMode = vcNotFixed
```

Methods

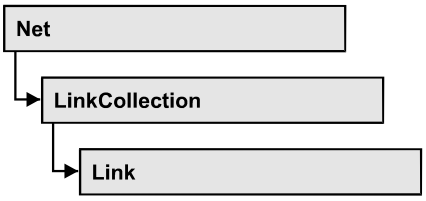
Update

Method of VcLegendView

This method lets you update the legend.

	Data Type	Explanation

7.32 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
- MarkLink
- PredecessorNode
- SuccessorNode

Methods

- DataRecord
- DeleteLink
- RelatedDataRecord
- UpdateLink

Properties

AllData

Property of VcLink

This property lets you set or retrieve all data fields of a link. When setting the data, you can specify a CSV string (using semicolons as separators) or a data field. When retrieving the data, a character string will be returned. (See also **InsertLinkRecord**.)

	Data Type	Explanation
Property value	data field/string	All data of the link

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim allDataOfLink As String

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink

allDataOfLink = link.AllData
```

DataField

Property of VcLink

This property lets you set or retrieve a specific data field of a link. The values which identify the predecessor and the successor nodes must not be changed.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the data field
Property value	Variant	Content of data field

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim message As String

Set linkCltn = VcNet1.LinkCollection
For Each link in linkCltn
         message = "Delete link from " & link.DataField(1)
                          & " to " &  link.DataField(2) & " ?"
         If MsgBox(message, vbOKCancel, "Delete Link") = vbOK Then
              link.DeleteLink
         End If
Next link
```

ID

Read Only Property of VcLink

By this property you can retrieve the ID of a link.

	Data Type	Explanation
Property value	String	Link ID

MarkLink

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

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink
Set node = link.PredecessorNode
nodeName = node.DataField(1)
```

SuccessorNode

Read Only Property of VcLink

This method lets you identify the successor node of a link.

	Data Type	Explanation
Property value	VcNode	Successor node

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink
Set node = link.SuccessorNode
nodeName = node.DataField(1)
```

Methods

DataRecord

Method of VcLink

This property lets you retrieve the link as a data record object. The properties of the data record object give access to the corresponding data table and the data table collection.

	Data Type	Explanation
Return value	VcDataRecord	Data record returned

DeleteLink

Method of VcLink

By this method you can delete a link.

	Data Type	Explanation
Return value	Boolean	Link was (True) / was not (False) successfully deleted

Example Code

```
Private Sub VcNet1_OnLinkRClick(ByVal link As VcNetLib.VcLink, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

   Dim message As String

   message = "Delete Link: " & link.AllData

   If MsgBox(message, vbOKCancel, "Delete Link") = vbOK Then
       link.DeleteLink
   End If

   returnStatus = vcRetStatNoPopup

End Sub
```

RelatedDataRecord

Method of VcLink

This property lets you retrieve a data record from a data table that is related to the link data table. The index passed by the parameter denotes the field in the data record that holds the key of the related data record.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field that holds the key
Return value	VcDataRecord	Related data record returned

UpdateLink

Method of VcLink

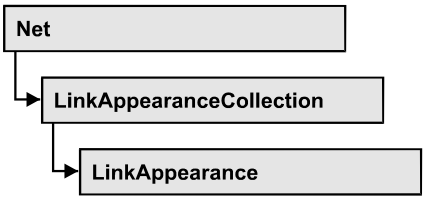
When a data field of a link was edited by the **DataField** property, you can update the diagram by the **UpdateLink** method.

	Data Type	Explanation
Return value	Boolean	Link was (True) / was not (False) updated successfully

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink

link.DataField(2) = "10"
link.UpdateLink
```

7.33 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.

Properties

- FilterName
- FormatName
- LineColor
- LineThickness
- LineType
- Name
- PrePortSymbol
- RoutingType
- Specification
- SuccPortSymbol
- Visible

Methods

- PutInOrderAfter

Properties

FilterName

Read Only Property of VcLinkAppearance

This property lets you enquire the filter that is used for a specific link appearance. This property also can be set on the **Link** property page.

	Data Type	Explanation
Property value	VcFilter	Filter object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim filterOfLinkApp As String

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
filterOfLinkApp = linkAppearance.Filter
```

FormatName

Property of VcLinkAppearance

This property lets you set or retrieve a format of the LinkAppearance object. If empty, the property will adopt the value of the property of a LinkAppearance object next in the descending hierarchy which matches the filter conditions and which is not empty (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	String	Name of a LinkFormat object or empty string

Example Code

```
Dim linkAppearanceCollection As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim format1 As VcLinkFormat

Set linkAppearanceCollection = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCollection.FirstLinkAppearance

Set format1 = linkAppearance.format
MsgBox (format1.name)
```

LineColor

Property of VcLinkAppearance

This property lets you set or retrieve the line color of a LinkAppearance object. This property can also be set on the **Link** property page in the **Line attributes** dialog.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
```

```
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")

linkAppearance.LineColor = RGB(0, 0, 255)
```

LineThickness

Property of VcLinkAppearance

This property lets you set or retrieve the line thickness of a LinkAppearance object.

If you set this property to values between 1 and 4, an absolute line thickness is defined in pixels. Irrespective of the zoom factor a line will always show the same line thickness in pixels. When printing though, the line thickness is adapted for the sake of legibility and becomes dependent of the zoom factor:

Value	Points	mm
1	1/2 point	0.09 mm
2	1 point	0.18 mm
3	3/2 points	0.26 mm
4	2 points	0.35 mm

A point equals 1/72 inch and represents the unit of the font size.

If you set this property to values between 5 and 1,000, the line thickness is defined in 1/100 mm, so the lines will be displayed in a true thickness in pixels that depends on the zoom factor.

	Data Type	Explanation
Property value	Long	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined on property page

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")

linkAppearance.LineThickness = 4
```

LineType

Property of VcLinkAppearance

This property lets you set or retrieve the line type of a LinkAppearance object. This property can also be set in the **Line Attributes** dialog box that can be invoked by the **Link** property page.

	Data Type	Explanation
Property value	LineTypeEnum	Line type **Default value:** vcSolid
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5
	vcLineType6 106	Line Type 6
	vcLineType7 107	Line Type 7
	vcLineType8 108	Line Type 8
	vcLineType9 109	Line Type 9
	vcNone 1	No line type
	vcNotSet -1	No line type assigned
	vcSolid 2	Line solid

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")

linkAppearance.LineType = 5
```

Name

Read Only Property of VcLinkAppearance

This property lets you retrieve the name of a LinkAppearance object.

	Data Type	Explanation
Property value	String	Name

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

nameLinkApp  = linkAppearance.name
```

PrePortSymbol

Property of VcLinkAppearance

This property lets you assign/retrieve a port symbol to/from a link, that visually accentuates the junction of the link and the predecessor node.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	LinkPredecessorSymbolEnum	Symbol on the predecessor node **Default value:** vcLPSNone
	Possible Values:	
	vcLPSArrow 64	Predecessor port symbol **arrow**
	vcLPSDoubleArrow 65	Predecessor port symbol **double arrow**
	vcLPSDoubleSemiCircle 97	Predecessor port symbol **double semi-circle**
	vcLPSFilledArrow 72	Predecessor port symbol **filled Arrow**

vcLPSFilledDoubleArrow 88	Predecessor port symbol **filled double arrow** 
vcLPSFilledDoubleSemiCircle 120	Predecessor port symbol **filled double semi-circle** 
vcLPSFilledSemiCircle 104	Predecessor port symbol **filled semi-circle** 
vcLPSNone 0	Predecessor port symbol **none**
vcLPSSemiCircle 96	Predecessor port symbol **semi-circle** 

Example Code

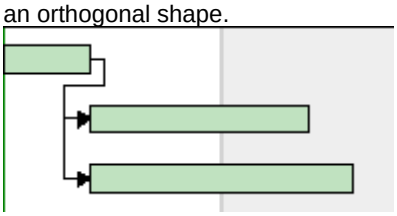
```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

linkAppearance.PrePortSymbol = vcLPSFilledDoubleSemiCircle
```

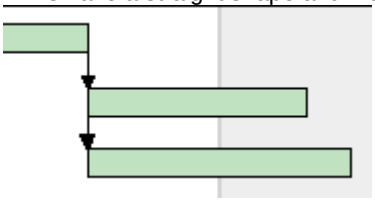
RoutingType

Property of VcLinkAppearance

This property lets you set or retrieve, whether the links of the diagram should be drawn horizontally and vertically only (and therefore show orthogonal shapes), or if they are allowed to lead directly to their aim, probably on an oblique route, allowing to cut through objects.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	LinkRoutingTypeEnum	Routing type **Default value:** vcLRTOrthogonal
	Possible Values:	
	vcLRTNotSet -1	A routing type is used which is further up the list of the LinkAppearance objects.
	vcLRTOrthogonal 1	Links run horizontally and vertically only and show an orthogonal shape. 

vcLRTStraightLined 4	Links have a straight shape and may run obliquely. 

Specification

Read Only Property of VcLinkAppearance

This property lets you retrieve the specification of a link appearance. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a link appearance by the method **VcLinkAppearanceCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the link appearance

SuccPortSymbol

Property of VcLinkAppearance

This property lets you assign/retrieve a port symbol to a link, that visually accentuates the intersection of the link and the successor node.

This property can also be set on the **Links** property page.

	Data Type	Explanation
Property value	LinkSuccessorSymbolEnum	Symbol on the succesor node **Default value:** vcLSSNone
	Possible Values:	
	vcLSSArrow 32	Successor port symbol **arrow** ⟶
	vcLSSDoubleArrow 33	Successor port symbol **double arrow** ⟶⟶
	vcLSSFilledArrow 40	Successor port symbol **filled arrow** ⟶
	vcLSSFilledDoubleArrow 56	Successor port symbol **filled double arrow** ⟶⟶
	vcLSSNone 0	Successor port symbol **none**

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

linkAppearance.SuccPortSymbol = vcLSSFilledDoubleArrow
```

Visible

Property of VcLinkAppearance

This property lets you set or retrieve whether the link is to be visible or not, taking no effect, however, on the phantom lines for links while dragging.

This property can also be set on the **Links** property page, but here also applying to the phantom lines.

	Data Type	Explanation
Property value	Boolean	Property active/not active **Default value:** True

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

linkAppearance.Visible = False
```

Methods

PutInOrderAfter

Method of VcLinkAppearance

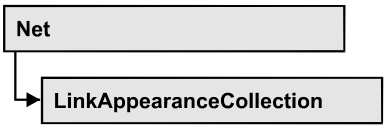
This method lets you set the link appearance behind a link appearance specified by name, within the LinkAppearanceCollection. If you set the name to "", the link appearence will be put in the first position. The order of the link appearances within the collection determines the order by which they apply to the links.

	Data Type	Explanation
Parameter:		
refLinkAppearanceName	String	Name of the link appearance behind which the current link appearance is to be put.
Return value	Void	

Example Code

```
Dim linkAppCltn As VcLinkAppearanceCollection
Dim linkApp1 As VcLinkAppearance
Dim linkApp2 As VcLinkAppearance

linkAppCltn = VcGantt1.LinkAppearanceCollection()
linkApp1 = linkAppCltn.Add("linkApp1")
linkApp2 = linkAppCltn.Add("linkApp2")
linkApp1.PutInOrderAfter("linkApp2")
linkAppCltn.Update()
```

7.34 VcLinkAppearanceCollection



An object of the type VcLinkAppearanceCollection automatically contains all available link appearances. You can access all objects in an iterative loop by **For Each linkAppearance In LinkAppearanceCollection** or by the methods **First...** and **Next...**. You can access a single line format by the methods **LinkAppearanceByName** and **LinkAppearandeByIndex**. The number of link appearances in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstLinkAppearance
- LinkAppearanceByIndex
- LinkAppearanceByName
- NextLinkAppearance
- Remove
- Update

Properties

_NewEnum

Read Only Property of VcLinkAppearanceCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all link appearance objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim linkApp As VcLinkAppearance

For Each linkApp In VcNet1.LinkAppearanceCollection
   Debug.Print linkApp.Name
Next
```

Count

Read Only Property of VcLinkAppearanceCollection

This property lets you retrieve the number of link appearances in the LinkAppearanceCollection object.

	Data Type	Explanation
Property value	Long	Number of link appearance objects

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim numberOfLinkAppearances As Integer

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection

numberOfLinkAppearances = linkAppearanceCltn.Count
```

Methods

Add

Method of VcLinkAppearanceCollection

By this method you can create a new link appearance as a member of the LinkAppearanceCollection. If the name was not used before, the new linke appearance object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned. All attributes of the new links appearance by default are set to transparent .

	Data Type	Explanation
Parameter:		
⇨ newName	String	Name of the link appearance
Return value	VcLinkAppearance	New linke appearance object

Example Code

```
Set 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 later session the link appearance can be created again from the specification and is identified by its name.

	Data Type	Explanation
Parameter:		
⇨ linkAppearanceSpecification	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" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	String	Name of the link appearance to be copied
⇨ newName	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

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance
```

LinkAppearanceByIndex

Method of VcLinkAppearanceCollection

This method lets you access a link appearance object by its index. If a link appearance object of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	Integer	Index of the link appearance object

LinkAppearanceByName

Method of VcLinkAppearanceCollection

This method retrieves a link appearance object by its name. If a link appearance object of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ linkAppearanceName	String	Name of the link appearance object
Return value	VcLinkAppearance	LinkAppearance object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
```

NextLinkAppearance

Method of VcLinkAppearanceCollection

This method can be used in a forward iteration loop to retrieve subsequent link appearances from a link appearance collection after initializing the loop by the method **FirstLinkAppearance**. If there is no link appearance left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLinkAppearance	Subsequent linkAppearance object

Example Code

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCltn = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCltn.FirstLinkAppearance

While Not linkAppearance Is Nothing
    linkAppearance.Visible = False
    Listbox.AddItem ("Name:" & linkAppearance.name)
    Set linkAppearance = linkAppearanceCltn.NextLinkAppearance
Wend
```

Remove

Method of VcLinkAppearanceCollection

This method lets you delete a link appearance. If the link appearance is used by a different object, it cannot be deleted. In the latter case **False** will be returned, otherwise **True**.

	Data Type	Explanation
Parameter:		
⇨ name	String	Name of the link appearance
Return value	Boolean	Link appearance deleted (True)/not deleted (False)

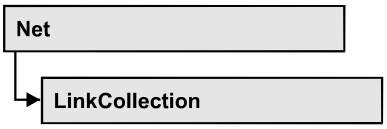
Update

Method of VcLinkAppearanceCollection

This method lets you update a collection of link appearances after having modified it.

	Data Type	Explanation
Return value	Boolean	Update successful (True) / not successful (False)

7.35 VcLinkCollection



An object of the type VcLinkCollection contains all available links. You can access all objects in an iterative loop by **For Each link In LinkCollection** or by the methods **First...** and **Next...**. The number of links in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstLink
- NextLink
- SelectLinks

Properties

_NewEnum

Read Only Property of VcLinkCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all link objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim link As VcLink

For Each link In VcNet1.LinkCollection
   Debug.Print link.Name
Next
```

Count

Read Only Property of VcLinkCollection

This property lets you retrieve the number of links in the link collection.

	Data Type	Explanation
Property value	Long	Number of links

Example Code

```
Dim linkCltn As VcLinkCollection
Dim numberLinks  As Integer

Set linkCltn = VcNet1.LinkCollection
numberLinks = linkCltn.Count
```

Methods

FirstLink

Method of VcLinkCollection

This method can be used to access the initial value, i.e. the first link of a link collection, and to continue in a forward iteration loop by the method **NextLink** for the links following. If there is no link in the link collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLink	First link

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink
```

NextLink

Method of VcLinkCollection

This method can be used in a forward iteration loop to retrieve subsequent links from a link collection after initializing the loop by the method **FirstLink**. If there is no link left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcLink	Subsequent link

Example Code

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim stringLink As String

Set linkCltn = VcNet1.LinkCollection
Set link = linkCltn.FirstLink

While Not link Is Nothing
     stringLink = link.AllData
     Listbox.AddItem (stringLink)
     Set link = linkCltn.NextLink
Wend
```

SelectLinks

Method of VcLinkCollection

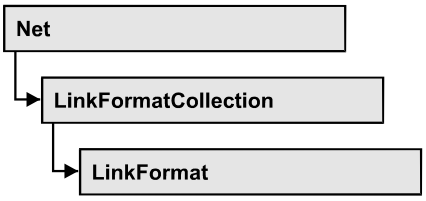
This method lets you specify the links that the link collection is to contain.

	Data Type	Explanation
Parameter:		
⇨ selectionType	SelectionTypeEnum	Links to be selected
	Possible Values:	
	vcAll 0	All objects in the diagram will be selected
	vcAllLinksCausingCycles 7	If this selection type is chosen, the link collection will contain all links that cause the existence of cycles. If these links are deleted, cycles will cede to exist in this chart.
	vcAllLinksInCycles 6	If this selection type is chosen, the link collection will contain all links that participate in forming cycles. Cycles are chains of nodes and links of which the beginning and end join.
	vcAllVisible 1	All visible objects will be selected
	vcMarked 2	All marked objects will be selected
Return value	Long	Number of links selected

Example Code

```
Dim linkCollection As VcLinkCollection

Set linkCollection = VcNet1.LinkCollection
linkCollection.SelectGroups (vcAllMarked)
```

7.36 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

- _NewEnum
- FormatField
- FormatFieldCount
- Name
- Specification

Methods

- CopyFormatField
- RemoveFormatField

Properties

_NewEnum

Read Only Property of VcLinkFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all link format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim formatField As VcLinkFormatField

For Each formatField In format
    Debug.Print formatField.Index
Next
```

FormatField

Read Only Property of VcLinkFormat

This property gives access to a VcLinkFormatField object by the index. The index has to be in the range from 0 to FormatFieldCount-1.

Note for users of a version earlier than 3.0: The index does **not** count from 1 to FormatFieldCount, as it does in more recent versions.

	Data Type	Explanation
Parameter:		
index	Integer	Index of the link format field 0 ... .FormatFieldCount-1
Property value	VcLinkFormatField	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	Integer	Number of fields of the link format

Example Code

```
Dim formatCollection As VcLinkFormatCollection
Dim format As VcLinkFormat
Dim nameofFormat As String

Set formatCollection = VcNet1.LinkFormatCollection
Set format = formatCollection.FormatByName("Standard")

numberofFormatField = format.FormatFieldCount
```

Name

Property of VcLinkFormat

This property lets you set or retrieve the name of the link format.

	Data Type	Explanation
Property value	String	Name of the link format

Example Code

```
Dim format As VcLinkFormat
Dim formatName As String

Set format = VcNet1.LinkFormatCollection.FirstFormat
formatName = format.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 node format by the method **VcLinkFormatCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the link format

Methods

CopyFormatField

Method of VcLinkFormat

This method allows to copy a link format field. The new VcLinkFormatField object is returned. It is given automatically the next index not used before.

	Data Type	Explanation
Parameter:		
⇨ position	FormatFieldPositionEnum	Position of the new link format field
	Possible Values:	
	vcAbove 1	above
	vcBelow 3	below

	vcLeftOf 0 vcOutsideAbove 9 vcOutsideBelow 11 vcOutsideLeftOf 8 vcOutsideRightOf 12 vcRightOf 4	left of outside, above outside, below outside, left of outside, right of right of
⇨ refIndex	Integer	Index of the reference link format field
Return value	VcLinkFormatField	Link format field object

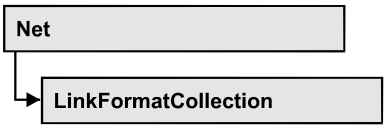
RemoveFormatField

Method of VcLinkFormat

This method lets you remove a link format field by its index. After that, the program will update all link format field indexes so that they are consecutively numbered again.

	Data Type	**Explanation**
Parameter:		
⇨ index	Integer	Index of the link format field to be deleted

7.37 VcLinkFormatCollection



An object of the type VcLinkFormatCollection contains all available link formats. You can access all objects in an iterative loop by **For Each link In NodeCollection** or by the methods **First...** and **Next...**. You can access a single link formats by using the methods **FormatByName**. The number of link formats in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcLinkFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all link format objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim format As VcLinkFormat

For Each format In VcNet1.LinkFormatCollection
   Debug.Print format.Name
Next
```

Count

Read Only Property of VcLinkFormatCollection

This property lets you retrieve the number of link formats in the node format collection.

	Data Type	Explanation
Property value	Long	Number of link formats

Example Code

```
Dim formatCltn As VcLinkFormatCollection
Dim numberOfFormats As Long

Set formatCltn = VcNet1.LinkFormatCollection
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 was not used before, the new VcLinkFormat object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

A link format by default has the below properties:

- It is a single field
- WidthOfExteriorSurrounding: 3 mm

A field has these 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	String	Name of the link format
Return value	VcLinkFormat	Link format object

Example Code

```
Set 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 VcLinkFormat 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	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 will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	String	Name of the link format to be copied
⇨ newName	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

```
Dim format As VcLinkFormat

Set format = VcNet1.LinkFormatCollection.FirstFormat
```

FormatByIndex

Method of VcLinkFormatCollection

This method lets you access a link format by its index. If a link format of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	Integer	Index of the link format

Example Code

```
Dim linkFormatCltn As VcLinkFormatCollection

Set linkFormatCltn = VcNet1.LinkFormatCollection
Set linkFormat = linkFormatCltn.LinkFormatByIndex(2)
linkFormat.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	String	Name of the link format
Return value	VcLinkFormat	Link format

Example Code

```
Dim formatCollection As VcLinkFormatCollection
Dim format As VcLinkFormat

Set formatCollection = VcNet1.LinkFormatCollection
Set format = formatCollection.FormatByName("Standard")
```

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	VLinkFormat	Subsequent link format

Example Code

```
Dim formatCollection As VcLinkFormatCollection
Dim format As VcLinkFormat

Set formatCollection = VcNet1.LinkFormatCollection
```

```
Set format = formatCollection.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCollection.NextFormat
Wend
```

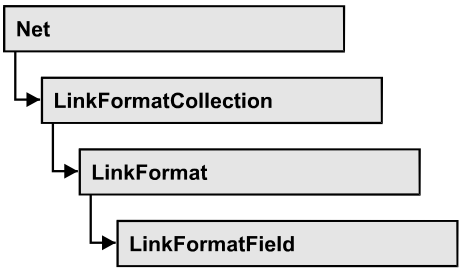
Remove

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	String	Link format name
Return value	Boolean	Link format deleted (True)/not deleted (False)

7.38 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

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	FormatFieldAlignmentEnum **Possible Values:** vcFFABottom 28 vcFFABottomLeft 27 vcFFABottomRight 29 vcFFACenter 25 vcFFALeft 24 vcFFARight 26	Alignment of the field content bottom bottom left bottom right center left right

vcFFATop 22	top
vcFFATopLeft 21	top left
vcFFATopRight 23	top right

ConstantText

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	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	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	Integer	Index of the table format field

MinimumWidth

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	Integer	Minimum width of the layer format field in mm 0 ... 99

TextDataFieldIndex

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	Long	Index of the data field

TextFont

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 **TextFontMap-Name**, the map will control the text font in dependence of the data.

	Data Type	Explanation
Property value	StdFont	Font type of the table format

TextFontColor

Property of 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	OLE_COLOR	Font color of the table format **Default value:** -1

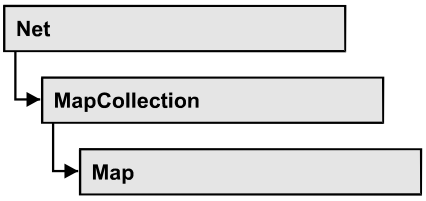
TextLineCount

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	Integer	Number of lines

7.39 VcMap



Maps define certain properties of nodes by data field entries, for example their background color which is based on the data of the node record.

In a map you can specify 150 map entries at maximum. By the call **For Each mapEntry In Map** you can retrieve all data field entries in an iterative loop.

Properties

- _NewEnum
- ConsiderFilterEntries
- Count
- Name
- Specification
- Type

Methods

- CreateEntry
- DeleteEntry
- FirstMapEntry
- GetMapEntry
- NextMapEntry

Properties

_NewEnum

Read Only Property of VcMap

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all map objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim map As VcMap

For Each map in VcNet1.Map
   Debug.Print.map.Name
Next
```

ConsiderFilterEntries

Read Only Property of VcMap

This property lets you set/retrieve whether filters are considered when a map is assigned to data field entries so that ranges of values can also be specified as keys.

	Data Type	Explanation

Count

Read Only Property of VcMap

This property lets you retrieve the number of map entries in a map.

	Data Type	Explanation
Property value	Long	Number of map entries

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim numberOfEntries As Long

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps vcAnyMap
Set map = mapCltn.MapByName("Map1")
numberOfEntries = map.count
```

Name

Read Only Property of VcMap

This property lets you retrieve the name of a map.

	Data Type	Explanation
Property value	String	Name

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapName As String

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.FirstMap
mapName = map.Name
```

Specification

Read Only Property of VcMap

This property lets you retrieve the specification of a map. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a map by the method **VcMapCollection.AddBySpecification**.

	Data Type	Explanation
Property value	String	Specification of the map

Type

Property of VcMap

This property lets you enquire/set the map type.

	Data Type	Explanation
Property value	MapTypeEnum **Possible Values:** vcAnyMap 0 vcColorMap 1 vcFontMap 8 vcGraphicsFileMap 7 vcMillimeterMap 9 vcNumberMap 10 vcPatternMap 3 vcTextMap 6	map type **any** (used only for selecting) **Colors** **Fonts** **Graphics file** **Millimeters** **Numbers** **Pattern** **Text**

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
```

```
Set mapCollection = VcNet1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.MapByName("Map1")
map.Type = vcPatternMap
```

Methods

CreateEntry

Method of VcMap

This method lets you create a new entry (a new row) for a map. To make the entry work, the method **MapCollection.Update()** should be invoked after creating.

	Data Type	Explanation
Return value	VcMapEntry	Map entry

Example Code

```
Set mapCltn = VcNet1.MapCollection
Set map = mapCltn.Add("MapColor")

map.Type = vcColorMap
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = RGB(0, 255, 0)
Set mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = RGB(255, 0, 0)
mapCltn.Update
```

DeleteEntry

Method of VcMap

This method lets you delete an entry (a row) of the map. To make the deletion work, the method **MapCollection.Update()** should be invoked after deleting.

	Data Type	Explanation
Parameter:		
⇨ mapEntry	VcMapEntry	Map entry
Return value	Boolean	Map entry was (True) / was not (False) deleted successfully

Example Code

```
Dim mapCltn As VcMapCollection
```

```
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps vcAnyMap
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

map.DeleteEntry mapEntry
mapCltn.Update
```

FirstMapEntry

Method of VcMap

This method can be used to access the initial value, i.e. the first entry of a map object and then to continue in a forward iteration loop by the method **NextMapEntry** for the entries following. If there is no entry in the map, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMapEntry	First map entry

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)

Set map = mapCltn.FirstMap
Set mapEntry = map.FirstMapEntry
```

GetMapEntry

Method of VcMap

This method returns the corresponding map entry for the given data field value.

	Data Type	Explanation
Return value	VcMapEntry	Map entry according to field value

NextMapEntry

Method of VcMap

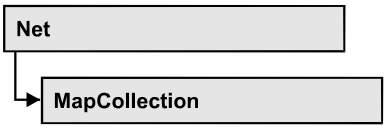
This method can be used in a forward iteration loop to retrieve subsequent entries (rows) from a map object after initializing the loop by the method **FirstMapEntry**. If there is no map entry left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMapEntry	Subsequent map entry

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)

Set map = mapCltn.FirstMap
Set mapEntry = map.FirstMapEntry

While Not mapEntry Is Nothing
   List1.AddItem (mapEntry.Legend)
   Set mapEntry = map.NextMapEntry
Wend
```

7.40 VcMapCollection



An object of the type VcMapCollection contain the maps, which were assigned to the collection by the method **SelectMaps**. You can access all objects in an iterative loop by **For Each map In MapCollection** or by the methods **First...** and **Next...**. You can access a single map using the methods **MapByName** and **MapByIndex**. The number of maps in the collection object can be retrieved by the property **Count**. The methods **Add**, **Copy** and **Remove** allow to handle the maps in the corresponding way.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstMap
- MapByIndex
- MapByName
- NextMap
- Remove
- SelectMaps
- Update

Properties

_NewEnum

Read Only Property of VcMapCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all map objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method

GetEnumerator is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim map As VcMap

For Each map In VcNet1.MapCollection
   Debug.Print map.Count
Next
```

Count

Read Only Property of VcMapCollection

This property lets you retrieve the number of maps in the MapCollection object.

	Data Type	Explanation
Property value	Long	Number of maps

Example Code

```
Dim mapCltn As VcMapCollection
Dim numberOfMaps As Long

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps vcAnyMap
numberOfMaps = mapCltn.Count
```

Methods

Add

Method of VcMapCollection

By this method you can create a map as a member of the MapCollection. If the name was not used before, the new map object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ mapName	String	Map name

Return value	VcMap	New map object

Example Code

```
Set 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	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	String	Name of the map to be copied
⇨ newMapName	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). Before using this method, a selection of maps needs to have been defined by the method **VcMapCollection.SelectMaps**.

	Data Type	Explanation
Return value	VcMap	First map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.FirstMap
```

MapByIndex

Method of VcMapCollection

This method lets you access a map by its index. If a map of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ index	Integer	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, a set of maps needs to be selected by the method **SelectMaps**. If a map of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ mapName	String	Name of the map
Return value	VcMap	Map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.MapByName("Map_1")
```

NextMap

Method of VcMapCollection

This method can be used in a forward iteration loop to retrieve subsequent maps from a map collection after initializing the loop by the method **FirstMap**. If there is no map left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMap	Subsequent map

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.FirstMap

While Not map Is Nothing
   List1.AddItem map.Name
   Set map = mapCltn.NextMap
Wend
```

Remove

Method of VcMapCollection

This method lets you delete a map. If the map is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ mapName	String	Map name

Return value	Boolean	Map deleted (True)/not deleted (False)

SelectMaps

Method of VcMapCollection

This method lets you specify the map types that your map collection is allowed to contain.

	Data Type	Explanation
Parameter:		
⇨ selectionType	MapTypeEnum	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	**Pattern**
	vcTextMap 6	**Text**
Return value	Long	Number of maps selected

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps vcAnyMap
```

Update

Method of VcMapCollection

This method has to be used when map modifications have been made. The method **UpdateMaps** updates all objects that are concerned by the maps you have edited. You should call this method at the end of the code that defines the maps and the map collection. Otherwise the update will be processed before all map definitions are processed.

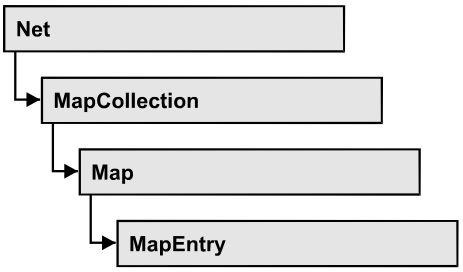
	Data Type	Explanation
Return value	Boolean	update successful (True)/ not successful (False)

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
```

```
Dim mapEntry As VcMapEntry

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps vcAnyMap
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

While Not mapEntry.DataFieldValue = "A"
   Set mapEntry = map.NextMapEntry
Wend

mapEntry.Color = RGB(0, 0, 0)

mapCltn.Update
```

7.41 VcMapEntry



An object of the type VcMapEntry is a map entry and therefore an element of a map. A map entry is defined by the combination of a data field content of the node´s record, a color or graphics file and a legend text.

In each map you can specify up to a maximum of 150 map entries. If you need further map entries, please specify a new map, e. g. as a copy of the current one.

Properties

- ColorAsARGB
- DataFieldValue
- FontBody
- FontName
- FontSize
- GraphicsFileName
- Pattern

Properties

ColorAsARGB

Property of VcMapEntry

for Color Maps: This property lets you set or retrieve the color value of a map entry. Color values have a transparency or alpha value, followed by a value for a red, a blue and a green partition (ARGB). The values range between 0..255. An alpha value of 0 equals complete transparency, whereas 255 represents a completely solid color. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},{0...255},{0...255},{0...255})

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim colorOfMapEntry As OLE_COLOR

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcColorMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

colorOfMapEntry = mapEntry.Color
```

DataFieldValue

Property of VcMapEntry

This property lets you set or retrieve the content of a data of each map entry.

	Data Type	Explanation
Property value	String	Content of the data field

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim dataFieldValue As String

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcAnyMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

FontBody

Property of VcMapEntry

for font maps: This property lets you set or retrieve the font body of the map entry.

	Data Type	Explanation
Property value	FontBodyEnum	Font body

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontBodyOfMapEntry As FontBodyEnum

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontBodyOfMapEntry = vcBold
```

FontName

Property of VcMapEntry

for font maps: This property lets you set or retrieve the font name of the map entry.

	Data Type	Explanation
Property value	String	Font type

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontNameOfMapEntry As String

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontNameOfMapEntry = "Arial"
```

FontSize

Property of VcMapEntry

for font maps: This property lets you set or retrieve the font name of he map entry.

	Data Type	Explanation
Property value	Long	Font size

Example Code

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontSizeOfMapEntry As Long

Set mapCltn = VcNet1.MapCollection
```

```
mapCltn.SelectMaps (vcFontMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontSizeOfMapEntry = 12
```

GraphicsFileName

Property of VcMapEntry

For graphics file maps: This property lets you set or retrieve the graphics file name of a map entry. *Available formats:*

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

EMF, EMF+, VMF and WMF are vector formats that allow to store a file independent of pixel resolution. All other formats are pixel-oriented and confined to a limited resolution.

The VMF format basically has been deprecated, but it will still be supported for some time to maintain compatibility with existing applications.

	Data Type	Explanation
Property value	String	Name of the graphics file

Example Code

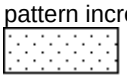
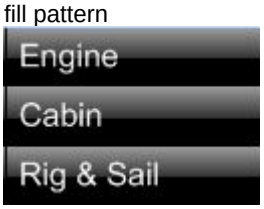
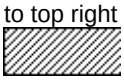
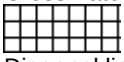
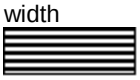
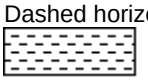
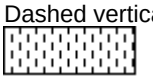
```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (vcGraphicsFileMap)
Set map = mapCltn.MapByName("Map1")
```

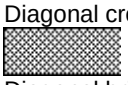
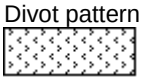
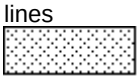
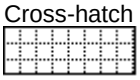
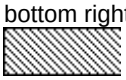
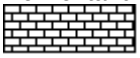
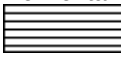
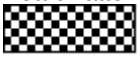
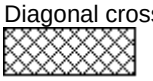
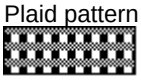
```
Set mapEntry = map.FirstMapEntry

mapEntry.GraphicsFileName = AppPath & "\picture1.bmp"
```

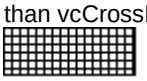
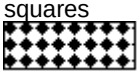
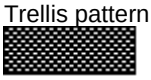
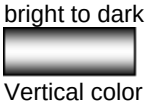
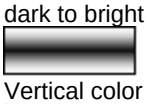
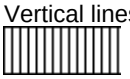
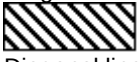
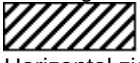
Pattern

Property of VcMapEntry

for pattern maps (vcPatternMap): This property lets you set or retrieve the pattern of a map entry.

	Data Type	Explanation
Property value	FillPatternEnum	Pattern type
	Possible Values: vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
	vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
	vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
	vcCrossPattern 6	Cross-hatch pattern 
	vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
	vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
	vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
	vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
	vcDashedHorizontalPattern 2026	Dashed horizontal lines 
	vcDashedVerticalPattern 2027	Dashed vertical lines 

vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right 
vcHorizontalBrickPattern 2033	Horizontal brick pattern 
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 

vcSmallCheckerBoardPattern 2043	Checkerboard pattern 
vcSmallConfettiPattern 2028	Confetti pattern 
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern 
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares 
vcSpherePattern 2041	Checkerboard of spheres 
vcTrellisPattern 2040	Trellis pattern 
vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright 
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
vcVerticalGradientPattern 62	Vertical color gradient 
vcVerticalPattern 2	Vertical lines 
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
vcWavePattern 2031	Horizontal wave pattern 
vcWeavePattern 2034	Interwoven stripe pattern 
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern 
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern 
vcZigZagPattern 2030	Horizontal zig-zag lines 

Example Code

```
Dim mapCltn As VcMapCollection
```

```
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim pattern As FillPatternEnum

Set mapCltn = VcNet1.mapCollection
mapCltn.SelectMaps (vcPatternMap)
Set map = mapCltn.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

pattern = vcBDiagonalPattern
```

7.42 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
- AllowMultipleBoxMarking
- AllowNewNodesAndLinks
- AssignCalendarToNodes
- BorderArea
- BoxCollection
- BoxFormatCollection
- CalendarCollection
- CalendarProfileCollection
- ConfigurationName
- CtrlCXVProcessing
- CurrentVersion
- DataDefinition
- DataTableCollection
- DateOutputFormat
- DiagramBackColor
- DialogFont
- DoubleOutputFormat
- EditNewLink
- EditNewNode
- Enabled
- EnableSupplyTextEntryEvent
- EventReturnStatus
- EventText
- ExtendedDataTables
- FilePath
- FilterCollection
- FontAntiAliasingEnabled
- GroupCollection
- GroupDescriptionName
- GroupField

- GroupHorizontalMargin
- Grouping
- GroupingTitlesFullyVisible
- GroupInteractionsAllowed
- GroupMode
- GroupMovingAllowed
- GroupSortField
- GroupSortMode
- GroupTitleField
- GroupVerticalMargin
- hWnd
- InFlowGroupDescriptionName
- InFlowGroupField
- InFlowGroupingEnabled
- InFlowGroupSeparationLineColor
- InFlowGroupSeparationLineType
- InFlowGroupTimeInterval
- InFlowGroupTitleField
- InFlowGroupTitlesBackColor
- InFlowGroupTitlesFont
- InFlowGroupTitlesVisibleAtBottomOrRight
- InFlowGroupTitlesVisibleAtTopOrLeft
- InFlowGroupTitleTimeFormat
- InFlowGroupVerticalCaptionWidth
- InPlaceEditingAllowed
- InteractionMode
- InterfaceNodesShown
- LegendView
- LinkAnnotationColumnNumberDataFieldIndex
- LinkAnnotationRowNumberDataFieldIndex
- LinkAppearanceCollection
- LinkCollection
- LinkFormatCollection
- LinkPredecessorDataFieldIndex
- LinksDataTableName
- LinkSuccessorDataFieldIndex
- LinkTypeDataFieldIndex
- MapCollection
- MinimumColumnWidth
- MinimumRowHeight

- MouseProcessingEnabled
- NodeAppearanceCollection
- NodeCalendarNameDataFieldIndex
- NodeChangeRankToPredecessorRankDataFieldIndex
- NodeCollection
- NodeColumnNumberDataFieldIndex
- NodeFormatCollection
- NodeRowNumberDataFieldIndex
- NodesDataTableName
- NodeTooltipTextField
- ObliqueTracksOnLinks
- OLEDragMode
- OLEDragWithOwnMouseCursor
- OLEDragWithPhantom
- OLEDropMode
- Orientation
- Printer
- RoundedLinkSlantsEnabled
- Scheduler
- ScrollOffsetX
- ScrollOffsetY
- ShortenedLinks
- ShowToolTip
- StraightLinkDrawing
- TimeUnit
- ToolTipChangeDuration
- ToolTipDuration
- ToolTipPointerDuration
- ToolTipShowAfterClick
- UngroupedNodesAllowed
- WaitCursorEnabled
- WorldView
- ZoomFactor
- ZoomingPerMouseWheelAllowed

Methods

- AboutBox
- Arrange
- Clear
- CopyNodesIntoClipboard

- CutNodesIntoClipboard
- DeleteLinkRecord
- DeleteNodeRecord
- DetectDataTableFieldName
- DetectDataTableName
- DetectFieldIndex
- DumpConfiguration
- EditLink
- EditNode
- EndLoading
- ExportGraphicsToFile
- GetAValueFromARGB
- GetBValueFromARGB
- GetGValueFromARGB
- GetLinkByID
- GetLinkByIDs
- GetNodeByID
- GetRValueFromARGB
- IdentifyFormatField
- IdentifyFormatFieldAsVariant
- IdentifyObjectAt
- IdentifyObjectAtAsVariant
- InsertLinkRecord
- InsertNodeRecord
- MakeARGB
- Open
- PageLayout
- PasteNodesFromClipboard
- PixelsToRaster
- PixelsToRasterAsVariant
- PrintDirectEx
- PrinterSetup
- PrintIt
- PrintPreview
- PrintToFile
- RasterToPixels
- RasterToPixelsAsVariant
- Reset
- SaveAsEx
- ScheduleProject

- ScrollToNodePosition
- ShowAlwaysCompleteView
- ShowExportGraphicsDialog
- SuspendUpdate
- UpdateLinkRecord
- UpdateNodeRecord
- Zoom
- ZoomOnMarkedNodes

Events

- Error
- ErrorAsVariant
- KeyDown
- KeyPress
- KeyUp
- OLECompleteDrag
- OLEDragDrop
- OLEDragOver
- OLEGiveFeedback
- OLESetData
- OLEStartDrag
- OnBoxLClick
- OnBoxLDblClick
- OnBoxModifyComplete
- OnBoxModifyCompleteEx
- OnBoxRClick
- OnDataRecordCreate
- OnDataRecordCreateComplete
- OnDataRecordDelete
- OnDataRecordDeleteComplete
- OnDataRecordModify
- OnDataRecordModifyComplete
- OnDataRecordNotFound
- OnDiagramLClick
- OnDiagramLDblClick
- OnDiagramRClick
- OnGiveFeedbackForNodeCreating
- OnGroupCreate
- OnGroupDelete
- OnGroupLClick

- OnGroupLDblClick
- OnGroupModify
- OnGroupModifyComplete
- OnGroupRClick
- OnHelpRequested
- OnLegendViewClosed
- OnLinkCreate
- OnLinkCreateComplete
- OnLinkDelete
- OnLinkDeleteComplete
- OnLinkLClickCltn
- OnLinkLDblClickCltn
- OnLinkModifyComplete
- OnLinkModifyEx
- OnLinkRClickCltn
- OnLinksMark
- OnLinksMarkComplete
- OnModifyComplete
- OnMouseDblClk
- OnMouseDown
- OnMouseMove
- OnMouseUp
- OnNodeCreate
- OnNodeCreateCompleteEx
- OnNodeDelete
- OnNodeDeleteCompleteEx
- OnNodeLClick
- OnNodeLDblClick
- OnNodeModifyComplete
- OnNodeModifyCompleteEx
- OnNodeModifyEx
- OnNodeRClick
- OnNodesMarkComplete
- OnNodesMarkEx
- OnSelectField
- OnShowInPlaceEditor
- OnStatusLineText
- OnSupplyTextEntry
- OnSupplyTextEntryAsVariant
- OnToolTipText

- OnToolTipTextAsVariant
- OnWorldViewClosed
- OnZoomFactorModifyComplete

Properties

ActiveNodeFilter

Property of VcNet

This property lets you set or retrieve a filter that collects the selection of nodes to be displayed.

	Data Type	Explanation
Property value	VcFilter	Filter object **Default value:** Nothing

Example Code

```
Dim filter As VcFilter
Dim filterName As String

Set filter = VcNet1.ActiveNodeFilter

If Not Filter Is Nothing Then
    filterName = filter.Name
End If

Set VcNet1.ActiveNodeFilter = VcNet1.FilterCollection. _
                    FilterByName("Filter_1")
```

AllowMultipleBoxMarking

Property of VcNet

This property lets you set or retrieve whether at run time several boxes can be marked simultaneously. If the property is not activated, the user has to keep the CTRL key pressed in order to mark several boxes. You can also set this property on the **General** property page

	Data Type	Explanation
Property value	Boolean	Multiple box marking enabled / not enabled **Default value:** True

Example Code

```
VcNet1.AllowMultipleBoxMarking = True
```

AllowNewNodesAndLinks

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	Boolean	Property active (True)/not active (False) **Default value:** True

Example Code

```
Dim boole As Boolean

boole = VcNet1.AllowNewNodesAndLinks
```

AssignCalendarToNodes

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	Boolean	A calendar is assigned (True) / is not assigned (False) **Default value:** False

Example Code

```
VcNet1.AssignCalendarToNodes = False
```

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

```
Dim borderArea As VcBorderArea

Set 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

```
Dim boxCltn As VcBoxCollection

Set 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 gives access to the calendar collection object that contains all calendars available.

	Data Type	Explanation
Property value	VcCalendarCollection	CalendarCollection object

Example Code

```
Dim calendarCltn As VcCalendarCollection

Set 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

Example Code

```
Dim calendarProfileCltn As VcCalendarProfileCollection
Dim calendarProfile As VcCalendarProfile

Set calendarProfileCltn = VcNet1.CalendarProfileCollection
```

ConfigurationName

Property of VcNet

This property enables a configuration file (*.ini) to be loaded, that all settings are adopted from, including the corresponding data interface.

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

- *local file:* The default configuration file *vcnet.ini* should be stored in the directory where the *vcnet.ocx* is registered. If you specify the file name without path, *vcnet.ini* will be expected to exist in the installation directory. If the specified file does not exist, the default configuration will be loaded, which does not necessarily exist at the site of end user.

- *URL:* A URL should be used as configuration file only if the configuration is specified during runtime by the API because only then the *ini* and *ifd* files will be loaded from the URL specified. (Otherwise, if you specify a URL as a configuration file during design time, the *ini* and *ifd* files will be downloaded, but they will be stored in the Structured Storage (VB: *frx* file). That store will be used during runtime instead of loading the files directly.) So when embedding VARCHART ActiveX into an HTML page, you can specify the *ini* and *ifd* files directly, not

needing other ways to temporarily create a local file which is considered insecure by browsers anyway.

Also see "Introduction: ActiveX Controls in Browser Environment"

Note: When loading a new configuration file, existing data will be lost and may have to be re-loaded again.

	Data Type	Explanation
Property value	String	Name of file including path, if necessary

Example Code

```
VcNet1.ConfigurationName = "c:\VARCHART\XNet\sample.ini"
' or:
VcNet1.ConfigurationName = "http://members.tripod.de/netronic_te/ _
                                                    xnet_sample.ini"
```

CtrlCXVProcessing

Property of VcNet

This property automatically translates the key combinations <Ctrl>+<C>, <Ctrl>+<X> and <Ctrl>+<V> into the clipboard commands **CopyNodesTo-Clipboard**, **CutNodesToClipboard** and **PasteNodesFromClipboard**, respectively. You can suppress this feature in order to avoid conflicts with shortcuts for menu items in e.g. Visual Basic applications. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Key combinations will/will not be translated into clipboard commands **Default value:** True

Example Code

```
VcNet1.CtrlCXVProcessing = False
```

CurrentVersion

Read Only Property of VcNet

This property lets you retrieve the number of the current version of the VARCHART XNet object. This is an easy way to identify the version on your customer's system at runtime, and to probably request the installation to be repaired, if a version is identified which is too old. The version number

can alternatively be found by the property page of the file vcnet.ocx in the section **version** or it can be read by the FILEVERSION resource of that file.

	Data Type	Explanation
Property value	String	Version number

Example Code

```
MsgBox VcNet1.CurrentVersion
```

DataDefinition

Read Only Property of VcNet

This property gives access to the current DataDefinition object to retrieve the names of the field types. The data definition of VcNet has got two data definition tables: **vcMaindata** and **vcRelations**. Each DataDefinitionTable contains a set of DefinitionFields, that you can access.

	Data Type	Explanation
Property value	VcDataDefinition	DataDefinition object

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcNet1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set definitionField = dataDefinitionTable.FirstField

While Not definitionField Is Nothing
       Listbox.AddItem definitionField.Name
       Set definitionField = dataDefinitionTable.NextField
Wend
```

DataTableCollection

Read Only Property of VcNet

This property gives access to the DataTableCollection object that contains the existing data tables.

	Data Type	Explanation
Property value	VcDataTableCollection	Data table collection object returned

Example Code

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable
```

```
Set dataTableCltn = VcNet1.DataTableCollection
For Each dataTable In dataTableCltn
    List1.AddItem (dataTable.Name)
Next
```

DateOutputFormat

Property of VcNet

This property lets you specify the date output format. To compose the date you can use the below codes:

D:	first letter of the day of the week (not adjustable)
TD:	Day of the Week (adjustable by using the event **OnSupplyTextEntry**)
DD:	two-digit figure for the day of the month: 01-31
DDD:	first three letters of the day of the week (not adjustable)
M:	first letter of the name of the month (not adjustable)
TM:	name of the month (adjustable by using the event **OnSupplyTextEntry**)
MM:	two-digit figure for the month: 01-12
MMM:	first three letters of the name of the month (not adjustable)
YY:	two-digit figure for the year
YYYY:	four-digit figure for the year
WW:	two-digit figure for the number of the calendar week: 01-53
TW:	text for "calendar week" (adjustable by using the event **OnSupplyTextEntry**)
Q:	one-digit figure for the quarter: 1-4
TQ:	name of quarter (adjustable by using the event **OnSupplyTextEntry**)
hh:	two-digit figure for the hour in 24 hours format: 00-23
HH:	two-digit figure for the hour in 12 hours format: 01-12
Th:	Text of "o' clock" (adjustable by using the event **OnSupplyTextEntry**)
TH:	"am" or "pm" (adjustable by using the event **OnSupplyTextEntry**)
mm	two-digit figure for the minute: 00-59

ss: two-digit figure for the second: 00-59

TS: short date format, as defined in the regional settings of the windows control panel

TL: long date format, as defined in the regional settings of the windows control panel

TT: time format, as defined in the regional settings of the windows control panel

Note: Characters which are not to be interpreted as part of the date should be preceded by a backslash '\'. '\\' for instance results in "\'. The special characters: **':, /, -**' and **blank** don't need '\' as prefix.

	Data Type	Explanation
Property value	String	Date format {DMYhms:;/}

Example Code

```
VcNet1.DateOutputFormat = "DD.MM.YY"
```

DiagramBackColor

Property of VcNet

This property lets you set or retrieve a background color to your network diagram. The default color is white.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255}) **Default value:** (255,255,255)

Example Code

```
VcNet1.DiagramBackColor = RGB(255, 204, 204)
```

DialogFont

Property of VcNet

This property lets you set or retrieve 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	String	Font name

Example Code

```
Dim font As New StdFont

font.Size = 18
font.name = "Courier"

VcNet1.DialogFont = font
```

DoubleOutputFormat

Property of VcNet

This property lets you set or retrieve the output format of numbers as a double value in the network diagram. The format is composed by the below characters:

- Text
- I
- D

plus the separators **comma** and **period**. **Text** represents a character string; **I** represents the figures before the decimal separator and **D** represents the figures after the decimal separator. The overall sequence is **Text I D Text**, where a comma and a period can be inserted in the places desired. An example be the number -284901,3458. By the format **I,DDDD ppm** it will be output as **-284901,3458 ppm**. By the format **$I,III.DD** it will be output as **$-284,901.35**.

	Data Type	Explanation
Property value	String	Character string which describes the double format, for example "I,DDDD ppm"

Example Code

```
VcNet1.DoubleOutputFormat = "I,DDDD ppm"
```

EditNewLink

Property of VcNet

This property specifies whether the **Edit Data** dialog box is to appear when a new link is created. The **AllowNewNodesAndLinks** property must be set to **True** to enable the user to create new links.

	Data Type	Explanation
Property value	Boolean	Property active/not active

Example Code

```
VcNet1.EditNewLink = False
```

EditNewNode

Property of VcNet

This property specifies 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	Boolean	The **Edit Data** dialog appears/does not appear.

Example Code

```
VcNet1.EditNewNode = False
```

Enabled

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
Property value	Boolean	VARCHART ActiveX control enabled/disabled

Example Code

```
VcNet1.Enabled = False
```

EnableSupplyTextEntryEvent

Property of VcNet

This property lets you activate the **OnSupplyTextEntry** 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	Boolean	Property active/not active

Example Code

```
VcNet1.EnableSupplyTextEntryEvent = True
```

EventReturnStatus

Property of VcNet

You will need this property only in a development environment which does not allow the setting of a return value in an event procedure as e.g. javascript.

With this property the default returnStatus is overwritten within the event method by the desired value.The setting is valid only for the event in which it was made.

	Data Type	Explanation
Property value	ReturnStatusEnum	Return value of the event **Default value:** vcRetStatOK
	Possible Values:	
	vcRetStatDefault 2	The default behavior remains unchanged.
	vcRetStatFalse 0	The default behavior will not be performed.
	vcRetStatNoPopup 4	The popup of the right-click mouse menu is inhibited.
	vcRetStatOK 1	The default behavior will be performed.

Example Code

```
Private Sub VcNet1_OnDiagramRClick(ByVal x As Long, ByVal y As Long,
returnStatus As Variant)

    VcNet1.EventReturnStatus = vcRetStatNoPopup

End Sub
```

EventText

Read Only Property of VcNet

You will need this property only in a development environment which does not allow the setting of the delivery parameter in an event procedure as e.g. javascript.

With this property you set the ToolTipText.The setting is only valid for the event in which it was made.

	Data Type	Explanation
Property value	String	Tool Tip

Example Code

```
Private Sub VcNet1_OnSupplyTextEntry(ByVal controlIndex As
VcNetLib.TextEntryIndexEnum, TextEntry As String, returnStatus As Variant)

        VcNet1.EventText = "Order189"
End Sub
```

ExtendedDataTables

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	Boolean	**True**: only two data tables (Maindata and Relations) **False**: up to 99 data tables **Default value:** False

Example Code

```
VcNet1.ExtendedDataTables = 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 ActiveX control.

This property should be set when the application is started during the initializing procedure of the VARCHART ActiveX control. We recommend to set the file path to the path of the application or to a subdirectory of the application. The advantage of this action is that the application can be stored in any directory.

	Data Type	Explanation
Property value	String	File path **Default value:** " "

Example Code

```
Dim graphicsPath As String

graphicsPath = App.Path & "\bitmaps"
VcNet1.FilePath = graphicsPath
```

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

```
Dim filterCollection As VcFilterCollection

Set filterCollection = VcNet1.FilterCollection
```

FontAntiAliasingEnabled

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 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	Boolean	Characters will / will not be anti-aliased **Default value:** True

GroupCollection

Read Only Property of VcNet

If activities were grouped in a chart, this property gives access to the GroupCollection object that contains all groups available.

	Data Type	Explanation
Property value	VcGroupCollection	GroupCollection object

Example Code

```
Dim groupCollection As VcGroupCollection

Set groupCollection = VcNet1.GroupCollection
```

GroupDescriptionName

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 for in the path set by the VARCHART ActiveX property **FilePath** first. If it cannot be found there, the file will be searched for 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	String	Name

Example Code

```
VcNet1.GroupDescriptionName = "C:\varchart\xnet\samples\net.des"
```

GroupField

Property of VcNet

This property lets you specify or enquire the field of the data definition table to be used as a criterion for grouping. By default, the groups created will be sorted in alphabetical or numerical order (also see GroupSortField). This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	Integer	Index of data definition field

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcNet1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Code 1")

VcNet1.GroupField = dataDefinitionField.ID
```

GroupHorizontalMargin

Property of VcNet

This property lets you specify the left/right margins of groups.

This property can also be set on the **Grouping** property page.

	Data Type	Explanation
Property value	Single 0 ... 9.9 mm	Width of the left/right group margins (mm) **Default value:** 0

Example Code

```
VcNet1.GroupHorizontalMargin = 1.1
```

Grouping

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	Boolean	Property active/not active

Example Code

```
VcNet1.Grouping = True
```

GroupingTitlesFullyVisible

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
Property value	Boolean	Visible (True)/ not visible (False) **Default value:** False

GroupInteractionsAllowed

Property of VcNet

This property specifies whether groups can be collapsed or expanded interactively (by the **Plus** or **Minus** symbol beside the group title).

The interactive collapsing or expanding triggers the **OnGroupModify** event.

You should not modify this property any more if groups are visible in the diagram.

This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	Boolean	Property active (True)/not active (False) **Default value:** True

Example Code

```
VcNet1.GroupInteractionsAllowed = False
```

GroupMode

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 Terms: Grouping".

	Data Type	Explanation
Property value	GroupModeEnum	Mode of visualization **Default value:** 0
	Possible Values: vcGMClustering 1 vcGMGrouping 0	 Clustering standard grouping mode

Example Code

```
VcNet1.GroupMode = vcGMClustering
```

GroupMovingAllowed

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	Boolean	Moving collapsed clusters allowed (True) / not allowed (False) **Default value:** True

Example Code

```
Dim boole As Boolean

boole = VcNet1.GroupMovingAllowed
```

GroupSortField

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 **GroupSortField**, you can specify any sorting order. This property also can be set on the **Grouping** property page.

	Data Type	Explanation
Property value	Integer	Index of data definition field

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcNet1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Code 3")

VcNet1.GroupSortField = dataDefinitionField.ID
```

GroupSortMode

Property of VcNet

This property lets you set or enquire 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	GroupSortModeEnum	Sort Mode of Groups **Default value:** vcAscending
	Possible Values:	
	vcAscending 65	Nodes are sorted in ascending order of the group field.
	vcDescending 68	Nodes are sorted in descending order of the group field.
	vcGroupDescriptionOrder 88	Nodes are sorted according to the group description order as described by the group title file (see GroupDescriptionName).
	vcNodeOrder 73	Nodes are sorted according to the sequence of their generation

Example Code

```
VcNet1.GroupSortMode = vcDescending
```

GroupTitleField

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	Integer 0 ... 9.9 mm	Index of data definition field

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcNet1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Code 2")

VcNet1.GroupTitleField = dataDefinitionField.ID
```

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	Single 0 ... 9.9 mm	Height of the upper/bottom group margins in mm **Default value:** 0

Example Code

```
VcNet1.GroupVerticalMargin = 0.9
```

hWnd

Read Only Property of VcNet

This property returns a handle. The Microsoft Windows operating environment identifies each form and control in an application by assigning it a handle, or **hWnd**. The **hWnd** property is used with Windows API calls. Many Windows operating environment functions require the **hWnd** of the active window as an argument.

Note: Because the value of this property can change while a program is running, never store the **hWnd** value in a variable.

	Data Type	Explanation
Property value	Long	Handle

Example Code

```
MsgBox (Me.hWnd)
```

InFlowGroupDescriptionName

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.

	Data Type	Explanation
Property value	String	File name which contains the title texts for in-flow groups

InFlowGroupField

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
Property value	Integer	Data field which determines the in-flow groups **Default value:** -1

Example Code

```
VcNet1.InFlowGroupField = 16
```

InFlowGroupingEnabled

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 by the **Nodes** property page.

	Data Type	Explanation
Property value	Boolean	In-flow grouping activated (True)/ deactivated (False) **Default value:** False

Example Code

```
VcNet1.InFlowGroupingEnabled = True
```

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 by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	OLE_COLOR	Color of separation lines of in-flow groups

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 by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	LineTypeEnum	Type of separation lines of in-flow groups
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5
	vcLineType6 106	Line Type 6
	vcLineType7 107	Line Type 7
	vcLineType8 108	Line Type 8
	vcLineType9 109	Line Type 9
	vcNone 1	No line type
	vcNotSet -1	No line type assigned
	vcSolid 2	Line solid

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 by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	TimeOrientationIntervalEnum **Possible Values:** vcDay 5 vcFifteenMinutes 1848 vcFifteenSeconds 1845 vcFourHours 1853 vcHalfYear 1242 vcHour 6 vcMinute 7 vcMonth 3 vcQuarter 2 vcSecond 8 vcSixHours 1854 vcThirtyMinutes 1849 vcThirtySeconds 1846 vcThreeHours 1852 vcTwelveHours 1855 vcTwoHours 1851 vcTwoWeeks 1238 vcTwoYears 1245 vcWeek 4 vcYear 1	In-flow grouping interval day 15 minutes 15 seconds 4 hours half year hour minute month quarter (3 month) second 6 hours 30 minutes 30 seconds 3 hours 12 hours 2 hours two weeks two years week year

Example Code

```
VcNet1.InFlowGroupTimeInterval = vcDay
```

InFlowGroupTitleField

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 by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	Integer	Data field which is taken for in-flow group title ribbons

Example Code

```
VcNet1.InFlowGroupTitleField = 1
```

InFlowGroupTitlesBackColor

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 by the **Edit In-Flow Grouping** dialog.

	Data Type	Explanation
Property value	OLE_COLOR	Background color of title ribbons of 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 by 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	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	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	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	Integer	Width of title ribbons for in-flow grouping **Default value:** 50

Example Code

```
VcNet1.InFlowGroupVerticalCaptionWidth = 30
```

InPlaceEditingAllowed

Property of VcNet

This property lets you set or retrieve whether inline editing in node fields and boxes is possible or not. 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	Boolean	Inline editing in node fields possible (True) / not possible (False) **Default value:** True

Example Code

```
VcNet1.InPlaceEditingAllowed = True
```

InteractionMode

Property of VcNet

This property activates/retrieves one of the available modes of interaction.

	Data Type	Explanation
Property value	InteractionModeEnum	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

```
VcNet1.InteractionMode = vcCreateNodesAndLinks
```

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	Boolean	Property active/not active **Default value:** True

Example Code

```
VcNet1.InterfaceNodesShown = False
```

LegendView

Read Only Property of VcNet

This property gives access to the LegendView object that lets you define the legend view of the diagram.

	Data Type	Explanation
Property value	VcLegendView	LegendView object

Example Code

```
Dim legendview As VcLegendView

Set legendview = VcNet1.LegendView
legendview.Visible = True
```

LinkAnnotationColumnNumberDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which stores the column number of a link annotation. Setting this property is only possible if node data was not loaded yet.

	Data Type	Explanation
Property value	Long	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 stores the row number of an link annotation. Setting this property is only possible if node data was not loaded yet.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the row number of a link annotation

LinkAppearanceCollection

Read Only Property of VcNet

This property gives access to the LinkAppearanceCollection object that contains all link appearances available.

	Data Type	Explanation
Property value	VcLinkAppearanceCollection	LinkAppearanceCollectionObject

Example Code

```
Dim linkAppearanceCollection As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

Set linkAppearanceCollection = VcNet1.LinkAppearanceCollection
Set linkAppearance = linkAppearanceCollection.FirstLinkAppearance
```

LinkCollection

Read Only Property of VcNet

This property gives access to the link collection object and to all defined links.

	Data Type	Explanation
Property value	VcLinkCollection	LinkCollection object

Example Code

```
Dim linkCollection As VcLinkCollection
Dim numberOfLinks  As Integer

Set linkCollection = VcNet1.LinkCollection
numberOfLinks = linkCollection.Count
```

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

```
Dim formatCollection As VcLinkFormatCollection

Set formatCollection = VcNet1.LinkFormatCollection
```

LinkPredecessorDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which holds the identification of the predecessor node of the link. You can only set this property if data was not yet loaded.

	Data Type	Explanation
Parameter:		
Rückgabewert	Long	Field index of the data table
Property value	Integer	Index of predecessor node {0...2}

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
Set dataTable = VcNet2.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
Set dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
Set 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	String	Name of the data table which provides the fields for the links

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
Set dataTable = VcNet2.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
Set dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading
```

LinkSuccessorDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which holds the identification of the successor node of the link. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Parameter: identifierIndex	Integer	Index of predecessor node {0...2}
Property value	Long	Field index of the data table

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
Set dataTable = VcNet2.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
Set dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading
```

LinkTypeDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which contains the link type. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Data field which contains the link type

Example Code

```
Dim dataTable As VcDataTable

'create Link DataTable
Set 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 gives access to the map collection that contains a defined number of maps. The maps contained are selected by the method **VcMapCollection.SelectMaps**.

	Data Type	Explanation
Property value	VcMapCollection	MapCollection object

Example Code

```
Dim mapCollection As VcMapCollection

Set mapCollection = VcNet1.MapCollection
mapCollection.SelectMaps 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	Long {1...1 000}	Minimum column width in mm **Default value:** 1

Example Code

```
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 further reduce the row height. 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	Long {1...1 000}	Minimum row height in mm **Default value:** 1

Example Code

```
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 **OnMouseDown** event and the **OnMouseUp** 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 OnMouse events.

	Data Type	Explanation
Property value	Boolean	Property active (True)/ not active (False) **Default value:** True

NodeAppearanceCollection

Read Only Property of VcNet

This property gives access to the NodeAppearanceCollection object and to all defined node appearances.

	Data Type	Explanation
Property value	VcNodeAppearanceCollection	NodeAppearanceCollection object

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.BackColor = RGB(100, 100, 100)
```

NodeCalendarNameDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field to store the name of the calendar if you wish to use an individual calendar for a node. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the name of a calendar for a 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.

	Data Type	Explanation
Property value	Integer	Index of the data field which holds the rank number of the predecessor node

NodeCollection

Read Only Property of VcNet

This property gives access to the NodeCollection, depending on the settings of SelectNodes: either to all, to the marked or to visible nodes.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code

```
Dim nodeCollection As VcNodeCollection
Dim node As VcNode
Dim numberNodes As Integer

Set nodeCollection = VcNet1.NodeCollection
numberNodes = nodeCollection.Count
```

NodeColumnNumberDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which stores the column number of an activity. Setting this property is only possible if node data was not loaded yet.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the column number of an activity

NodeFormatCollection

Read Only Property of VcNet

This property gives access to the NodeFormatCollection object that contains all node formats available.

	Data Type	Explanation
Property value	VcNodeFormatCollection	NodeFormatCollection object

Example Code

```
Dim formatCollection As VcNodeFormatCollection

Set formatCollection = VcNet1.NodeFormatCollection
```

NodeRowNumberDataFieldIndex

Property of VcNet

This property lets you set or retrieve the index of the data field which stores the row number of each activity. Setting this property is only possible if node data was not loaded yet.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the row number of an activity

Example Code

```
Private Sub Form_Load()

   VcNet1.NodeRowNumberDataFieldIndex = VcNet1.DetectFieldIndex("NodeDataTable",
"SortNumber")

   'Load data
   Call loadData

   VcNet1.UpdateRowNumberFields
   VcNet1.SaveAsEx "C:\ProjectData.txt", vcUnicodeEncoding
End Sub
```

NodesDataTableName

Property of VcNet

This property lets you set or retrieve the name of the data table which provides the fields for the nodes. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	String	Name of the data table which provides the fields for the nodes

Example Code

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Node DataTable
 Set dataTable = VcNet1.DataTableCollection.Add("NodeDataTable")
VcNet1.NodesDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
'Load Data
Set dataTable = VcNet1.DataTableCollection.DataTableByName("NodeDataTable")
```

```
Set dataRecord = dataTable.DataRecordCollection.Add("1;Node One;")
Set dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;")

Set dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
Set dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading
```

NodeTooltipTextField

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	Integer	Index of the node data field for tooltip texts **Default value:** 4

Example Code

```
VcNet1.NodeTooltipTextField = 1
```

ObliqueTracksOnLinks

Property of VcNet

This property lets you specify or enquire whether the link lines will be orthogonal or oblique, connecting the short horizontal line sections. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean {True, False}	Oblique link lines (True)/orthogonal link lines (False) **Default value:** False

Example Code

```
VcNet1.ObliqueTracksOnLinks = True
```

OLEDragMode

Property of VcNet

By this property you can set or retrieve, whether dragging a node beyond the limits of the current VARCHART XNet control is allowed. This property can also be set on the **Nodes** property page.

If the OLEDragMode was set to **vcOLEDragManual** you need to invoke the method **OLEDrag** to trigger dragging the node. If the property was set to **vcOLEDragAutomatic**, dragging a node beyond control limits will be started automatically.

On the start, the source component will assign the data it contains to the DataObject and will set the **effects** parameter before initiating the OLEStartDrag event, as well as other source-level OLE Drag & Drop events. This gives you control over the drag/drop operation and allows you to intercede by adding other data formats.

VARCHART XNet by default uses the clipboard format CF_TEXT (corresponding to the vbCFText format in Visual Basic), that can be retrieved easily.

During dragging, the user can decide whether to shift or to copy the object by using the Ctrl key.

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

	Data Type	**Explanation**
Property value	OLEDragModeEnum	Dragging mode for objects to leave the VARCHART XNet control **Default value:** vcOLEDragManual
	Possible Values: vcOLEDragAutomatic 1 vcOLEDragManual 0	 Method OLEDrag is invoked automatically Method OLEDrag needs to be invoked separately.

Example Code

```
VcNet1.OLEDragMode = vcOLEDragAutomatic
```

OLEDragWithOwnMouseCursor

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	Boolean	Cursor does/does not occur in the target control **Default value:** True

Example Code

```
VcNet1.OLEDragWithOwnMouseCursor = False
```

OLEDragWithPhantom

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	Boolean	Phantom does/does not occur **Default value:** True

Example Code

```
VcNet1.OLEDragWithPhantom = False
```

OLEDropMode

Property of VcNet

By this property you can set or retrieve, whether nodes and their links from a different VARCHART XNet control can be dropped in the current control.

Dropping will not be allowed if you set the property to **OLEDropNone**. If you set it to **vcOLEDropManual**, you will receive the event **OLEDragDrop** that enables you to process the data received by the object dropped, e.g. to generate a node or to read a file. If the source and the target component are identical, you will receive either the event **OnNodeModifyEx** or **OnNodeCreate** as with OLE Drag&Drop switched off. If you set the

property to **vcOLEDropAutomatic**, the dropping will automatically be processed by the control, displaying a node in the place of the dropping, if possible.

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

You also can set this property on the **Nodes** property page.

	Data Type	Explanation
Property value	OLEDropModeEnum	Dropping mode of the VARCHART ActiveX control to receiving objects from outside **Default value:** vcOLEDropNone
	Possible Values:	
	vcOLEDropAutomatic 2	The data of the object received are automatically processed and a node corresponding to the data received is displayed in the place of the dropping.
	vcOLEDropManual 1	The event **OLEDragDrop** is invoked for the programmer to process the data of the object received.
	vcOLEDropNone 0	Dropping of objects that do not originate from the current VARCHART ActiveX control is not allowed.

Example Code

```
VcNet1.OLEDropMode = vcOLEDropAutomatic
```

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	LayoutOrientationEnum	From top to bottm, from left to right
	Possible Values:	
	vcLeftToRight 0	Orientation of the ne chart **from left to right**
	vcTopToBottom 1	Orientation of the net chart **from left top to bottom**

Example Code

```
VcNet1.Orientation = vcLeftToRight
```

Printer

Property of VcNet

This method gives access to the printer object. This object lets you set or retrieve the properties of the current printer.

	Data Type	Explanation
Property value	VcPrinter	Printer object

Example Code

```
Dim printerZoomfactor As Integer
Dim printerCuttingMarks As String

printerZoomfactor = VcNet1.Printer.ZoomFactor
printerCuttingMarks = VcNet1.Printer.CuttingMarks
```

RoundedLinkSlantsEnabled

Read Only Property of VcNet

This property lets you set or retrieve whether the slants of links of the routing type **vcLRTOrthogonalDistinguishable** are to be displayed as quarter circles instead of straigt lines. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	System.Boolean Schrägen bei Verbindungen werden/werden nicht als Viertelkreise dargestellt	false

Example Code

```
VcNet1.RoundedLinkSlantsEnabled = True
```

Scheduler

Read Only Property of VcNet

This property returns the VcScheduler object.

	Data Type	Explanation
Property value	VcScheduler	Returns the VcScheduler object

ScrollOffsetX

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	Long	Scroll offset in x-direction

ScrollOffsetY

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	Long	Scroll offset in y-direction

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	Boolean	Property active / not active

Example Code

```
VcNet1.ShortenedLinks = False
VcNet1.Arrange
```

ShowToolTip

Property of VcNet

This property lets you activate/deactivate the event **OnToolTipText**. The event **OnToolTipText** lets you edit the tooltip texts. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Property active/not active **Default value:** False

Example Code

```
VcNet1.ShowToolTip = True
```

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 them. If set, this property disables the property **ObliqueTracksOnLinks**.

	Data Type	Explanation
Property value	Boolean	Straight link drawing enabled (True) / disabled (False) **Default value:** False

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 set on the **General** property page.

Note: If you want to change the time unit, you should do this before reading data because later modifications are not effective.

	Data Type	Explanation
Property value	TimeUnitEnum	Time unit **Default value:** vcDay

Example Code

```
Dim timeUnit As TimeUnitEnum

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, please set it to -1.

	Data Type	Explanation
Property value	Integer	Duration in milliseconds. Maximum value: 32767 msec **Default value:** -1

Example Code

```
VcNet1.ToolTipText = "Object"
VcNet1.ToolTipChangeDuration = 1000
```

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	Integer	Duration in milliseconds. Maximum value: 32767 msec **Default value:** -1

Example Code

```
VcNet1.ToolTipText = "Object"
VcNet1.ToolTipDuration = 1000
```

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 maximum value of 480 msec, please set it to -1.

	Data Type	Explanation
Property value	Integer	Duration in milliseconds **Default value:** -1

Example Code

```
VcNet1.ToolTipText = "Object"
VcNet1.ToolTipPointerDuration = 1000
```

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	Boolean	Tool tip window disappears (false) or remains (true) **Default value:** False

Example Code

```
VcNet1.ToolTipShowAfterClick = 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	Boolean	Property active (True)/not active (False) **Default value:** False

Example Code

```
VcNet1.UngroupedNodesAllowed  = True
```

WaitCursorEnabled

Read Only 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	Boolean	Wait cursor is set/is not set **Default value:** False

WorldView

Read Only Property of VcNet

This property gives access to the VcWorldView object, that defines the world view (complete view) of the diagram.

	Data Type	Explanation
Property value	VcWorldView	World View object

Example Code

```
Dim worldview As VcWorldView

Set worldview = 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	Integer {0...1000}	Zoom factor (%)

Example Code

```
VcNet1.ZoomFactor = 150
```

ZoomingPerMouseWheelAllowed

Property of VcNet

This property lets you set or retrieve whether zooming by mouse wheel is allowed to the user.

	Data Type	Explanation
Property value	Boolean	Zooming allowed (True)/not allowed (False)

Example Code

```
VcNet1.ZoomingPerMouseWheelAllowed = False
```

Methods

AboutBox

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 by a mouse click, copied by the <Ctrl>+<C> keys and inserted by the <Ctrl>+<V> keys into a mail.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcNet1.AboutBox
```

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

```
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	Boolean	Nodes were deleted successfully. {True}

Example Code

```
VcNet1.Clear
```

CopyNodesIntoClipboard

Method of VcNet

This method lets you copy the selected nodes from the network diagram to the clipboard. Please note that to copy links both their predecessors and successors have to be marked. Also see methods **CutNodesIntoClipboard** and **PasteNodesFromClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcNet1.CopyNodesIntoClipboard
```

CutNodesIntoClipboard

Method of VcNet

This method lets you cut the selected nodes and links from the diagram into the clipboard. Links are only included, however, if both their predecessors and successors are selected. Also see **CopyNodesIntoClipboard** and **PasteNodesFromClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcNet1.CutNodesIntoClipboard
```

DeleteLinkRecord

Method of VcNet

This method lets you delete a link by passing the link record. Also see method **DeleteLink** of object **VcLink**.

	Data Type	Explanation
Parameter:		
⇨ linkRecord	Variant	Link record
Return value	Boolean	Link record was (True) / was not (False) not deleted successfully

Example Code

```
VcNet1.DeleteLinkRecord "A100;A105;;"
```

DeleteNodeRecord

Method of VcNet

This method lets you delete a node. The node will be identified by the ID in the node record. The data field that is used for the identification of nodes is set on the **DataDefinition** property page.

	Data Type	Explanation
Parameter:		
⇨ nodeRecord	Variant	Node record
Return value	Boolean	Node record was (True) / was not (False) deleted successfully

Example Code

```
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	Long	Index of the data table field of which the name is to be retrieved
Return value	String	Name of the data table field returned

Example Code

```
'Find the name of a DataTableField
Dim fieldName As String

fieldName = 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	Long	Index of the data table of which the name is to be retrieved
Return value	String	Name of the data table

Example Code

```
'Find the name of a DataTable
Dim tableName As String

tableName = VcNet1.DetectDataTableName(0)
```

DetectFieldIndex

Method of VcNet

This property lets you retrieve the index of a data table field by its name and the name of the data table.

	Data Type	Explanation
Parameter:		
⇨ dataTableName	String	Name of the data table that holds the field of which the index is to be retrieved
⇨ dataTableFieldName	String	Name of the data table field of which the index is to be retrieved
Return value	String	Index of the data table field returned

Example Code

```
'Find the index of a DataTableField
Dim fieldIndex As Integer

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.

This method should only be used for diagnosis purposes.

	Data Type	Explanation
Parameter:		
⇨ FileName	String	File name (including a path, if necessary)
⇨ encoding	EncodingEnum	Mode of encoding
	Possible Values:	
	vcANSIEncoding 1	If a file was saved in ANSI encoding, it depends on the local settings of the Windows operating system. The file then contains characters which can be read correctly only if the language settings are the same as the ones that it was stored by.
	vcUnicodeEncoding 2	Saving a file in Unicode encoding makes it independent of whatever settings and hence should be the preferred mode if possible. If a file that was saved in Unicode encoding is to be loaded in Visual Basic 6 independently of the VARCHART component, it has to be treated in a special way.
Return value	Boolean	File was (True)/was not (False) stored successfully.

EditLink

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	Boolean	Link data were edited/edition was cancelled

Example Code

```
Private Sub VcNet1_OnLinkLClickCltn(ByVal linkCltn As VcNetLib.VcLinkCollection,
_
                                ByVal x As Long, ByVal y As Long, returnStatus As
Variant)
    If linkCltn.Count = 1 Then
        VcNet1.EditLink linkCltn.FirstLink
    End If
End Sub
```

EditNode

Method of VcNet

This property invokes the **Edit Data** dialog box for the node passed.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node of which data are to be edited
Return value	Boolean	Node data were edited/editing was cancelled.

Example Code

```
Private Sub VcNet1_OnNodeLClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, returnStatus As
Variant)
    VcNet1.EditNode node
End Sub
```

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	Boolean	Loading finished {True}

Example Code

```
VcNet1.EndLoading
```

ExportGraphicsToFile

Method of VcNet

This method lets you store a network diagram to a file without invoking a **Save as** dialog box. You can store the files to the formats:

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile or Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, prbably 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.

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	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	Integer	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	Integer	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	Boolean	File was (True) / was not (False) stored successfully.

Example Code

```
VcNet1.ExportGraphicsToFile"C:\temp\export", 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	Long	ARGB value, from which the alpha value is to be identified
Return value	Integer	Alpha value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = 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	Long	ARGB value, from which the "blue" value is to be identified
Return value	Integer	"Blue" value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = 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	Long	ARGB value, from which the "green" value is to be identified
Return value	Integer	"Green" value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = 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	Variant	Link identification
Return value	VcLink	Link

Example Code

```
Dim link As VcLink

Set link = VcNet1.GetLinkByID(" 5")
```

GetLinkByIDs

Method of VcNet

This method gives access to a link by the IDs of its predecessor node and its successor node. If the identification consists of more than one field (composite primary key), the multipart ID has to be noted as shown below:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ predecessorID	String	Identification of the predecessor node

⇨ successorID	String	Identification of the successor node
Return value	VcLink	Link

Example Code

```
Dim link As VcLink

Set link = VcNet1.GetLinkByIDs(" 2", " 3")
```

GetNodeByID

Method of VcNet

This method gives access to a node by its identification, which was specified on the **Administrate Data Tables** dialog. If the identification consists of several fields (composite primary key), this multipart ID has to be specified as follows:

ID=ID1|ID2|ID3

	Data Type	Explanation
Parameter:		
⇨ nodeID	Variant	Node identification
Return value	VcNode	Node

Example Code

```
Dim node As VcNode

Set node = 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	Long	ARGB value, from which the "red" value is to be identified

Return value	Integer	"Red" value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = 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**.

Note: If you use VBScript, you can only use the analogue method **IdentifyFormatFieldAsVariant** because of the parameters by Reference.

	Data Type	**Explanation**
Parameter:		
⇨ x	Long	X coordinate of the position
⇨ y	Long	Y coordinate of the position
⇨ node	VcNode	Reference Node
⇦ format	VcNodeFormat	Identified format
⇦ formatFieldIndex	Integer	Index of the format field
Return value	Boolean	A format field exists/does not exist at the position specified

Example Code

```
Private Sub VcNet1_OnNodeLClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
    Dim foundFlag As Boolean
    Dim format As VcNodeFormat
    Dim formatFieldIndex As Integer

    foundFlag = VcNet1.IdentifyFormatField(x, y, node, format, formatFieldIndex)
    If foundFlag Then
        MsgBox "You hit the field with the index "+CStr(formatFieldIndex)
```

```
    End If
End Sub
```

IdentifyFormatFieldAsVariant

Method of VcNet

This method is identical with the method **IdentifyFormatField** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

IdentifyObjectAt

Method of VcNet

This method lets you identify the object that is located at any position of the diagram. The object type will be returned.

Note: If you use VBScript, you can only use the analogous method **IdentifyObjectAtAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X coordinate of the cursor
⇨ y	Long	Y coordinate of the cursor
⇦ identifiedObject	Object	Object identified
⇦ identifiedObjectType	VcObjectTypeEnum	Type of the object identified
	Possible Values: vcObjTypeBox 15 vcObjTypeGroup 7 vcObjTypeLinkCollection 3 vcObjTypeNode 2 vcObjTypeNodeInLegend 17 vcObjTypeNone 0	 object type **box** object type **group** object type **link collection** object type **node** object type **node in legend area** no object
Return value	Boolean	Object identified/no object identified

Example Code

```
Private Sub VcNet1_OnMouseMove(ByVal button As Integer, ByVal Shift As Integer,
ByVal x As Long, ByVal y As Long)

    Dim identifiedObject As Object
    Dim identifiedObjectType As VcObjectTypeEnum
    Dim node As VcNode

    Call VcNet1.IdentifyObjectAt(x, y, identifiedObject, identifiedObjectType)
```

```
    Select Case identifiedObjectType
        Case VcObjectTypeEnum.vcObjTypeNodeInDiagram,
VcObjectTypeEnum.vcObjTypeNodeInTable
            Set node = identifiedObject
            Label1.Caption = node.DataField(1)
        Case Else
            Label1.Caption = ""
    End Select

End Sub
```

IdentifyObjectAtAsVariant

Method of VcNet

This method is identical with the method **IdentifyObjectAt** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

InsertLinkRecord

Method of 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 when 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

```
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
```

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 when 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

```
' 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"
VcNet1.InsertLinkRecord "A100;A105;FS;0"

VcNet1.EndLoading
```

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	Integer	Alpha value
⇨ red	Integer	"Red" value
⇨ green	Integer	"Green" value
⇨ blue	Integer	"Blue" value
Return value	Long	ARGB value returned

Example Code

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = FF
red = A0
green = 34
blue = AB
argb = VcNet1.MakeARGB(alpha,red,green,blue)
```

Open

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	String	File name
Return value	Boolean	No significance {True}

Example Code

```
VcNet1.Open "C:\ProjectData.net"
```

PageLayout

Method of VcNet

This method lets you invoke the **Page Setup** dialog.

	Data Type	Explanation
Return value	Boolean	No significance {True}

Example Code

```
VcNet1.PageLayout
```

PasteNodesFromClipboard

Method of VcNet

This method lets you paste the nodes and links from the clipboard into the diagram. Also see **CopyNodesIntoClipboard** und **CutNodesIntoClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcNet1.PasteNodesFromClipboard
```

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**.

Note: If you use VBScript, you can only use the analogue method **PixelsToRasterAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ x	Long	Y coordinate in pixels
⇨ y	Long	Y coordinate in pixels
⇦ xBandNo	Long	X coordinate in band numbers
⇦ yBandNo	Long	X coordinate in band numbers
Return value	Boolean	Converting was (True) / was not (False) performed successfully

Example Code

```
Private Sub VcNet1_OnNodeLClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
    Dim lineNo As Long
    Dim columnNo As Long

    'change a data field of the node to the line number
    VcNet1.PixelsToRaster x, y, columnNo, lineNo
    node.DataField(19) = bandY
End Sub
```

PixelsToRasterAsVariant

Method of VcNet

This method is identical with the method **PixelsToRaster** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

PrintDirectEx

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	PrintResultStatusEnum	**Possible values:**			
		vcPrintingSucceeded 0:	Printing was performed successfully.		
		vcNoPrinterInstalled	1:	No printer was found	neither the one specified by the call **VcPrinter.PrinterName** nor the one labeled as default printer by the Windows operating system.
		vcPrintingAbortedByUser 2:	Printing was aborted by the user.		
		vcPrintingAbortedByDriver 3:	Printing was aborted by the Windows printer driver.		
		vcUnprintablePageLayout 4.	Printing could not be performed since the page layout did not match the printer properties such as paper size or margins.		

Example Code

```
PrintStatusResultEnum status = VcNet1.PrintDirectEx()
If status <> vcPrintingSucceeded Then
   Debug.Print "Printing failed: " & status & vbCrLf
End If
```

PrinterSetup

Method of VcNet

This method lets you invoke the Windows **Print Setup** dialog.

	Data Type	Explanation
Return value	Boolean	No significance {True}

Example Code

```
VcNet1.PrinterSetup
```

PrintIt

Method of VcNet

This method triggers printing of the diagram, considering the parameters set in the **PageLayout** dialog.

	Data Type	Explanation
Return value	Boolean	No significance {True}

Example Code

```
VcNet1.PrintIt
```

PrintPreview

Method of VcNet

This method invokes the print preview.

	Data Type	Explanation
Return value	Boolean	No significance

Example Code

```
VcNet1.PrintPreview
```

PrintToFile

Method of VcNet

This method lets you print the diagram directly to a file. Whether the printing is successful, depends on the printer driver since many PDF printer drivers do not accept file names.

	Data Type	Explanation
Parameter:		
⇨ fileName	String	File name
Return value	Void	

Example Code

```
VcNet1.PrintToFile
```

RasterToPixels

Method of VcNet

This method turns band numbers into window coordinates, as they for example are returned by events. If the coordinates are beyond the chart limits, the function will return the value **False**. Also see **PixelsToRaster**.

Note: If you use VBScript, you can only use the analogue method **RasterToPixelsAsVariant** because of the parameters by Reference.

	Data Type	Explanation
Parameter:		
⇨ xBandNo	Long	Xcoordinate in band numbers
⇨ yBandNo	Long	Ycoordinate in band numbers
⇦ x	Long	X coordinate in pixels
⇦ y	Long	Y coordinate in pixels
Return value	Boolean	Converting was (True) / was not (False) performed successfully

Example Code

```
Private Sub VcNet1_OnDiagramLDblClick(ByVal x As Long, ByVal y As Long, _
                              returnStatus As Variant)

' change a line number into a node data field
Dim lineNo, columnNo As Long
VcNet1.RasterToPixels columnNo, lineNo, x, y
Call MsgBox(columnNo & lineNo, vbOK)

End Sub
```

RasterToPixelsAsVariant

Method of VcNet

This method is identical with the method **RasterToPixels** except for the parameters. It was necessary to implement this event because some languages

(e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

Reset

Method of VcNet

This methods lets you either delete objects (nodes, links, calendars etc.) from the diagram, the extent depending on the selected value of resetAction, or restore the settings of the property pages carried out at design time

	Data Type	Explanation
Parameter:		
⇦ resetAction	ResetActionEnum	Objects to be initialized or deleted
	Possible Values:	
	vcEmptyAllDataTables 4	The contents of all data tables are deleted but the data tables are kept.
	vcReloadConfiguration 2	Complete reinitialization. All settings and created objects are discarded.
Return value	Boolean	The objects in the diagram were deleted successfully. (True)

Example Code

```
VcNet1. Reset(vcReloadConfiguration) = True
```

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	String	File name
⇦ encoding	EncodingEnum	Mode of encoding
	Possible Values:	

	vcANSIEncoding 1 vcUnicodeEncoding 2	If a file was saved in ANSI encoding, it depends on the local settings of the Windows operating system. The file then contains characters which can be read correctly only if the language settings are the same as the ones that it was stored by. Saving a file in Unicode encoding makes it independent of whatever settings and hence should be the preferred mode if possible. If a file that was saved in Unicode encoding is to be loaded in Visual Basic 6 independently of the VARCHART component, it has to be treated in a special way.
Return value	Boolean	Storing was (True)/was not (False) performed successfully

Example Code

```
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	Boolean	Scheduling was (True) / was not (False) successfully performed

Example Code

```
VcNet1.ScheduleProject "21.06.14", 0
```

ScrollToNodePosition

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	Boolean	Scrolling was (True) / was not (False) performed successfully.

Example Code

```
Private Sub VcNet1_OnNodeLClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
    'scroll the diagram so that the node is completely on screen
    VcNet1.ScrollToNodePosition node
End Sub
```

ShowAlwaysCompleteView

Method of VcNet

This method allows you to always display the diagram completely. The zoom factor automatically is adapted to any changement 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

```
VcNet1.ShowAlwaysCompleteView
```

ShowExportGraphicsDialog

Method of VcNet

This method lets you invoke the **Save As** dialog for saving the diagram. Possible formats for saving:

- *.BMP (Microsoft Windows Bitmap)
- *.EMF (Enhanced Metafile or Enhanced Metafile Plus)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)

- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile, probably 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	Boolean	Chart was successfully (True) / was not successfully (False) exported

Example Code

```
VcNet1.ShowExportGraphicsDialog
```

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	Boolean	SuspendUpdate(True): Start of the SuspendUpdate method/ SuspendUpdate(False): end of the SuspendUpdate method

Example Code

```
VcNet1.SuspendUpdate (True)

   If updateFlag Then
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.14" Then
            node.DataField(13) = "X"
            node.UpdateNode
            counter = counter + 1
         End If
      Next node
   Else
      For Each node In nodeCltn
         If node.DataField(2) < "07.09.14" Then
            node.DataField(13) = ""
            node.UpdateNode
            counter = counter + 1
         End If
      Next node
   End If

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	Variant	Link record
Return value	VcLink	Link updated

Example Code

```
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	Variant	Node record
Return value	VcNode	Node updated

Example Code

```
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	Integer	Zoom factor {11...999}, other values will remain unconsidered
Return value	Boolean	Zooming was performed successfully {True}

Example Code

```
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

```
VcNet1.ZoomOnMarkedNodes
```

Events

Error

Event of VcNet

This event occurs when an unforeseen error is found in the code of VARCHART XNet. NETRONIC tries hard to avoid each error. This event helps to take down the errors that occur at the customers comfortably, e.g. in a file. The parameter profile is specified by the ActiveX default. Therefore some of the parameters that are passed are constant. The number always should be checked in the event, in order to prevent to suppress all error types in the future program development.

	Data Type	Explanation
Parameter:		
⇨ Description	String	Error description
⇨ Scode	Long	&h800a402f (constant)
⇨ Source	String	Name of the control (constant)
⇨ HelpFile	String	Help file: "" (constant)
⇨ HelpContext	Long	Help context: 0 (constant)
⇦ CancelDisplay	Boolean	If True, then no normal error with number 71 (which could be catched via On Error GoTo) will be output.

Example Code

```
Private Sub VcNet1_Error(Number As Integer, Description As String, _
          Scode As Long, Source As String, HelpFile As String, HelpContext _
          As Long, CancelDisplay As Boolean)

   Debug.Print CStr(Number) + " " + Description

End Sub
```

ErrorAsVariant

Event of VcNet

This method is identical with the method **Error** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

KeyDown

Event of VcNet

This event occurs when the user presses a key while VARCHART XNet has the focus. Key events allow to trigger VARCHART ActiveX functions by using the keyboard. (For the interpretation of ANSI symbols please use the **KeyPress** event.)

	Data Type	Explanation
Parameter:		
⇨ KeyCode	Integer	Key code, e.g. vbKeyF1 (F1 key) or vbKeyHome (POS1 key)
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".

Example Code

```
Private Sub VcNet1_KeyDown(KeyCode As Integer, Shift As Integer)
    MsgBox "key pressed"
End Sub
```

KeyPress

Event of VcNet

This event occurs when the user presses and releases an ANSI key while VARCHART XNet has the focus. Key events allow to trigger VARCHART ActiveX functions by using the keyboard.

	Data Type	Explanation
Parameter:		
⇨ KeyAscii	Integer	An integer that returns the numerical key code of an default ANSI key. KeyAscii is returned as reference. If the parameter is changed, a different symbol will be passed to the object. If KeyAscii is set to 0, pressing a key will have no effect, i.e. no symbol will be passed to the object.

Example Code

```
Private Sub VcNet1_KeyPress(KeyAscii As Integer)
    MsgBox "Key pressed and released."
End Sub
```

KeyUp

Event of VcNet

This event occurs when the user releases a key while VARCHART XNet has the focus. Key events allow to trigger VARCHART ActiveX functions by using the keyboard. (For the interpretation of ANSI symbols please use the **KeyPress** event.)

	Data Type	Explanation
Parameter:		
⇨ KeyCode	Integer	Key code, e.g. vbKeyF1 (F1 key) or vbKeyHome (POS1 key)
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".

Example Code

```
Private Sub VcNet1_KeyUp(KeyCode As Integer, Shift As Integer)
    MsgBox "key released"
End Sub
```

OLECompleteDrag

Event of VcNet

This event occurs when a source component is dropped onto a target component, informing the source component that a drag action was either performed or canceled.

	Data Type	Explanation
Parameter:		
⇨ effect	vcOLEDropEffectEnum	A long integer value that identifies the action performed when dropping the data in the drop target. It is initially set by the source object that identifies all OLE-Drag & Drop operations supported by the source. The target component should check these effects and finally set it when the user drops the selection on the component.

OLEDragDrop

Event of VcNet

Occurs when during OLE Drag & Dropping a source component is dropped onto a target component and if the **OLEDropMode** property of the target component is set to **vcOLEDropManual** and source and target component are not identical. If they are identical you will receive either the event **OnNodeModifyEx** or **OnNodeCreate**.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	An OLE Drag & Drop-DataObject by which the data is imported.
⇨ effect	vcOLEDropEffectEnum	A long integer that describes the action performed when dropping the data in the drop target.
⇨ button	Integer	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇨ y	Long	A number that specifies the current vertical position of the mouse cursor.
⇨ x	Long	X coordinate of the mouse cursor

OLEDragOver

Event of VcNet

This event occurs when the data is dragged over a drop target and the **OLEDropMode** property of the drop target was set to **vcOLEDropManual**.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	An OLE Drag & Drop-DataObject by which the data is imported.
⇔ effect	OLEDropEffectEnum	A long integer that describes the action performed when dropping the data in the drop target.
	Possible Values:	
	vcDropEffectCopy 1	Drop results in a copy of data from the source to the target. The original data is unaltered by the drag operation.
	vcDropEffectMove 2	Drop results in data being moved from the source to the target. The source should remove the data from itself after the move.
	vcDropEffectNone 0	Target cannot accept the data.
⇨ button	Integer	Indicates the mouse button pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇨ x	Long	A number that specifies the current horizontal position of the mouse cursor.
⇨ y	Long	A number that specifies the current vertical position of the mouse cursor-
⇨ state	OLEDragStateEnum	A constant that corresponds to the transition state of the control being dragged in relation to a target form or control.
	Possible Values:	
	vcEnter 0	Object of the source control reaches the target.
	vcLeave 1	Object of source control is dragged out of the target.
	vcOver 2	Object of the source control has moved from one position in the target to another.

OLEGiveFeedback

Event of VcNet

Occurs after every OLEDragOver event. OLEGiveFeedback allows the source component to provide visual feedback to the user, such as changing the mouse cursor to indicate what will happen if the user drops the object, or provide visual feedback on the selection (in the source component) to indicate what will happen.

	Data Type	Explanation
Parameter:		
⇨ effect	vcOLEDropEffectEnum	A long integer that describes the action performed when dropping the data in the drop target. It is initially set by the source object identifying all effects it supports. The target component should check these effects and finally determine it if the user drops the selection on the component.
⇦ defaultCursors	Boolean	Determines whether the control uses the default mouse cursor provided by the component (true), or uses a user-defined mouse cursor (false).

Example Code

```
Private Sub VcTree1_OLEGiveFeedback(ByVal Effect As Long, _
                                    DefaultCursors As Boolean)
    If Effect <> vcOLEDropEffectNone Then
        'activate own mouse cursor
        MousePointer = vbCustom
        MouseIcon = LoadPicture("h_point.cur")
        DefaultCursors = False
    End If
End Sub
```

OLESetData

Event of VcNet

Occurs on a source component when a target component performs the **GetData** method on the source´s DataObject object, but a format for data has not been defined.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	A DataObject to place the requested data in. The component calls the SetData method to load the requested format.

⇨ dataFormat	Integer	An integer specifying the format of the data that the target component is requesting. The source component uses this value to determine what to load into the DataObject object. There is a table listed with the **GetData** method that describes the values corresponding to the formats allowed.

OLEStartDrag

Event of VcNet

This event occurs when the **OLEDrag** method is performed, or when the VARCHART XNet control initiates an OLE Drag & Drop operation when the **OLEDragMode** property is set to **vcOLEAutomatic**.

This event specifies the data formats and drop effects that the source component supports. It can also be used to insert data into the DataObject object.

The source component should use the logical **Or** operator against the supported values and place the result in the **allowedEffect** parameter. The target component can use this value to determine the appropriate action (and what the appropriate user feedback should be).

You should defer putting data into the **DataObject** until the target component requests it. This allows the source component to save time by not loading multiple data formats. When the target performs the **GetData** method on the DataObject, the source's **OLESetData** event will occur if the requested data are not contained in the **DataObject**. At this point, the data can be loaded into the **DataObject**, which will in turn provide the data to the target.

If the user does not load any formats into the **DataObject**, then the drag&drop operation is canceled.

	Data Type	**Explanation**
Parameter:		
⇨ data	DataObject	An object of the type **DataObject**, that contains formats provided by the source and, optionally, the data for those formats. If no data are contained in the DataObject, they are provided when the control invokes the GetData method.
⇔ allowedEffect	vcOLEDropEffectEnum	A long integer containing the effects that the source component supports. The programmer should provide the values for this parameter in this event.

Example Code

```
Private Sub VcTree1_OLEStartDrag(ByVal data As VcNetLib.DataObject, _
                                 allowedEffects As Long)
    allowedEffects = vbDropEffectCopy

    ' make sure that dragging is allowed only from one XTree control
    ' into another one
    data.SetData Empty, myOLEDragFormat
End Sub
```

OnBoxLClick

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
Parameter:		
⇨ box	VcBox	Box hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnBoxLClick(ByVal box As VcNetLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)

    Text1.Text = box.FieldText(1)

End Sub
```

OnBoxLDblClick

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
Parameter:		
⇨ box	VcBox	Box hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnBoxLDblClick(ByVal box As VcNetLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)
    box.FieldText(0) = Text1.Text
End Sub
```

OnBoxModifyComplete

Event of VcNet

This event occurs when the modification of the box is finished.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box modified

Example Code

```
Private Sub VcNet1_OnBoxModifyComplete(ByVal box As _
                  VcNetLib.VcBox)

    MsgBox "The box has been modified."

End Sub
```

OnBoxModifyCompleteEx

Event of VcNet

This event occurs when the modification of the box is finished. The modified VcBox object and the modification type are passed as parameters.

	Data Type	Explanation
Parameter:		
⇨ modificationType	BoxModificationTypeEnum	Modification type
	Possible Values:	
	vcBMTAnything 1	any modification
	vcBMTNothing 0	no modification
	vcBMTTextModified 4	text modified
	vcBMTXYOffsetModified 2	Offset modified

Example Code

```
Private Sub VcNet1_OnBoxModifyCompleteEx(ByVal box As _
                  VcNetLib.VcBox)

    MsgBox "The box has been modified."

End Sub
```

OnBoxRClick

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
Parameter:		
⇨ box	VcBox	Box hit
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnBoxRClick(ByVal box As VcNetLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)

    ' Start own popup menu at the current mouse cursor position
    PopupMenu mnuBoxPopup

End Sub
```

OnDataRecordCreate

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.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **VcDataRecordCreateComplete**.

By setting the return status the create operation can be inhibited.

If a link or a node was created, you can in addition react to the analogous link or node event and verify additional graphical data (s. **OnNodeCreate** and **OnLinkCreate**).

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Data record created

⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The data record will be created.
	vcRetStatOK 1	The data record will not be created.

Example Code

```
Private Sub VcNet1_OnDataRecordCreate(ByVal node As VcNetLib.VcDataRecord, _
                                      returnStatus As Variant)

    'Show own "Edit" dialog for the new data record
    '(EditNewDataRecord attribute must be set to off!)
    On Error GoTo CancelError
    frmEditDialog.Show Modal, Me

    addDataRecord dataRecord.AllData

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnDataRecordCreateComplete

Event of 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. events **OnNodeCreate-Complete** and **OnLinkCreateComplete**).

	Data Type	**Explanation**
Parameter:		
⇨ node	VcNode	Data record created
⇨ creationType	CreationTypeEnum	Creation type
	Possible Values:	
	vcDataRecordCreated 6	Data record created by interaction
	vcDataRecordCreatedByResourceScheduling 5	Data record automatically created by resource scheduling
	vcLinkCreated 2	Link created by linking two nodes
	vcNodeCreated 1	node created via mouse-click

	vcNodesAndLinksCloned 4	selected nodes were copied via dragging the mouse and pressing the the Ctrl button
	vcNodeWithLinkCreated 3	nodes and links created simultanously
⇨ isLastNodeInSeries	Boolean	**True**:The data record created is the only one or the last one of a data record collection.
		False:The data record created is not the only one or the last one of a data record collection.

Example Code

```
Private Sub VcNet1_OnDataRecordCreateComplete(ByVal dataRecord As _
                                VcNetLib.VcDataRecord, ByVal creationType As _
                                VcNetLib.CreationTypeEnum, _
                                ByVal isLastDataRecordInSeries As Boolean)
   addDataRecord dataRecord.AllData
End Sub
```

OnDataRecordDelete

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 inhibit the deletion on a negative result by setting the return status.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Data record deleted
⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The data record will not be deleted.
	vcRetStatOK 1	The data record will be deleted.

Example Code

```
Private Sub VcNet1_OnDataRecordDelete(ByVal node As VcNetLib.VcNode, _
                                returnStatus As Variant)
  'deny the deletion of the last data record in the chart
    If VcNet1.DataRecordCollection.Count = 1 Then
        returnStatus = vcRetStatFalse
        MsgBox ("The last data record cannot be deleted.")
    End If
End Sub
```

OnDataRecordDeleteComplete

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. **OnNodeDelete-Complete** and **OnLinkDeleteComplete**).

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Data record deleted
⇨ isLastNodeInSeries	Boolean	**True**:The data record deleted is the only one or the last one of a data record collection. **False**:The data record deleted is not the only one or the last one of a data record collection.

OnDataRecordModify

Event of VcNet

This event occurs after an interactive modification of an object that is based on a data record. The modified VcDataRecord object and the modification type are returned.

The data passed by this event can be read, but must not be modified. For modifying them please use the event **OnDataRecordModifyComplete**.

By setting the return status the modification can be inhibited.

	Data Type	Explanation
Parameter:		
⇨ dataRecord	VcBox	Box modified
⇨ modificationType	ModificationTypeEnum	Modification type
	Possible Values: vcAnything 1 vcChangedGroup 16 vcMoved 8 vcNothing 0	 modification type not determined group of the node changed Object was moved no modification

⇔ returnStatus	Variant	Return status
	Possible Values:	
	vcRetStatFalse 0	The modification will be revoked.
	vcRetStatOK 1	The modification will be accepted.

OnDataRecordModifyComplete

Event of VcNet

This event occurs when the modification of the data record is finished.

	Data Type	Explanation
Parameter:		
⇨ dataRecord	VcDataRecord	Data record modified

Example Code

```
Private Sub VcNet1_OnDataRecordModifyComplete(ByVal box As _
                  VcNetLib.VcBox)

    MsgBox "The data record has been modified."

End Sub
```

OnDataRecordNotFound

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:		
⇦ index	Long	Index of the field that contains the key of the depending data record

OnDiagramLClick

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
Parameter:		
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnDiagramLClick(ByVal x As Long, _
                                   ByVal y As Long, returnStatus As Variant)
    Dim zoomfactor As Integer

    zoomfactor = VcNet1.Zoomfactor + 10
    VcNet1.Zoomfactor = zoomfactor
End Sub
```

OnDiagramLDblClick

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
Parameter:		
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnDiagramLDblClick(ByVal x As Long, _
                                   ByVal y As Long, returnStatus As Variant)
    Dim zoomfactor As Integer

    zoomfactor = VcNet1.Zoomfactor - 10
    VcNet1.Zoomfactor = zoomfactor
End Sub
```

OnDiagramRClick

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
Parameter:		
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnDiagramRClick(ByVal x As Long, ByVal y As Long, _
                                   returnStatus As Variant)

    'start a popup menu at the current mouse cursor position
    PopupMenu mnuDiagramPopup

    'revoke the VARCHART XNet context menu
    returnStatus = vcRetStatNoPopup

End Sub
```

OnGiveFeedbackForNodeCreating

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 **1** of **creationAllowed** is not changed, creating nodes at this cursor position is allowed . If **creationAllowed** is set to **0**, 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
Parameter:		
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ creationAllowed	Long	Return status

Example Code

```
Private Sub VcNet1_OnGiveFeedbackForNodeCreating(ByVal X As Long, ByVal Y As
Long, creationAllowed As Long)
   Dim obj As Object
   Dim objType As VcObjectTypeEnum
   VcNet1.IdentifyObjectAt X, Y, obj, objType
   If objType = vcObjTypeNone Then
      creationAllowed = False
   End If
End Sub
```

OnGroupCreate

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
Parameter:		
⇨ group	VcGroup	Group created
⇔ returnStatus	Variant	Return status: at the moment without function

OnGroupDelete

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
Parameter:		
⇨ group	VcGroup	Group hit
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnGroupDelete(ByVal group As VcNetLib.VcGroup, _
                                 returnStatus As Variant)
    MsgBox ("The last node of the group is deleted.")
End Sub
```

OnGroupLClick

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
Parameter:		
⇨ group	VcGroup	Group hit
⇨ x	Long	X value

⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnGroupLClick(ByVal group As VcNetLib.VcGroup, _
                                 ByVal x As Long, ByVal y As Long, _
                                 returnStatus As Variant)
    'change the color of the group to a light yellow
    group.BackColor = &HC0FFFF
End Sub
```

OnGroupLDblClick

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
Parameter:		
⇨ group	VcGroup	Group hit
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnGroupLDblClick(ByVal group As VcNetLib.VcGroup, _
                                    ByVal x As Long, ByVal y As Long, _
                                    returnStatus As Variant)
    MsgBox group.Name
End Sub
```

OnGroupModify

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
Parameter:		
⇨ Group	VcGroup	Group modified
⇨ modificationType	GroupModificationTypeEnum	Type of modification
	Possible Values: vcGMTCollapsing 2 vcGMTExpanding 4 vcGMTMoved 32	 Group collapsed Group expanded Object was moved
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnGroupModify(ByVal Group As VcNetLib.VcGroup, _
                            ByVal modificationType As _
                            VcNetLib.GroupModificationTypeEnum, _
                            returnStatus As Variant)

  Select Case modificationType
       Case vcGMTCollapsing
            MsgBox "Group is collapsed."
       Case vcGMTExpanding
            MsgBox "Group is expanded."
   End Select

End Sub
```

OnGroupModifyComplete

Event of VcNet

This event occurs when the interactive collapsing or expanding of a clustered group is finished.

	Data Type	Explanation
Parameter:		
⇨ Group	VcGroup	Group modified
⇨ modificationType	GroupModificationTypeEnum	Type of modification
	Possible Values: vcGMTCollapsing 2 vcGMTExpanding 4 vcGMTMoved 32	 Group collapsed Group expanded Object was moved

Example Code

```
Private Sub VcNet1_OnGroupModifyComplete(ByVal group As VcNetLib.VcGroup, _
                            ByVal modificationType As _
                            VcNetLib.GroupModificationTypeEnum)

   MsgBox "The group has been modified successfully."

End Sub
```

OnGroupRClick

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
Parameter:		
⇨ group	VcGroup	Group hit
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	

Example Code

```
Private Sub VcNet1_OnGroupRClick(ByVal group As VcNetLib.VcGroup, _
                                 ByVal x As Long, ByVal y As Long, _
                                 returnStatus As Variant)

    ' start a popup menu at the current mouse cursor position
    PopupMenu mnuGroupPopup

    returnStatus = vcRetStatNoPopup

End Sub
```

OnHelpRequested

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 information specific to the dialog and to the application.

	Data Type	Explanation
Parameter:		
⇦ dialogType	DialogTypeEnum	Dialog for which help was requested
	Possible Values:	
	vcEditDataRecordDialog 5400	Help was requested for the **Edit Data Record** dialog.
	vcPageSetupDialog 4097	Help was requested for the **Page Set Up** dialog.
	vcPrintPreviewDialog 4096	Help was requested for the **Print Preview** dialog.

OnLegendViewClosed

Event of VcNet

This event occurs when the legend view popup window is closed.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		

Example Code

```
Private Sub VcNet1_OnLegendViewClosed()
    MsgBox "Do you want to close the legend view window?", vbOKCancel
End Sub
```

OnLinkCreate

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. If you set the returnStatus to **vcRetStatFalse**, the link will be deleted.

This event should be used only for reading data of the current link, but not for modifying them. For modifying data please use **OnLinkCreateComplete**.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link created
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkCreate(ByVal link As VcNetLib.VcLink, _
                                returnStatus As Variant)
    'show own edit dialog for the new link
    '   (EditNewLinks attribute must be set to off!)
    On Error GoTo CancelError
    frmEditLinkDialog.setLink link
    frmEditLinkDialog.Show Modal, Me
    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse
End Sub
```

OnLinkCreateComplete

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
Parameter:		
⇨ link	VcLink	link created
⇨ creationType	CreationTypeEnum	creation type
	Possible Values:	
	vcDataRecordCreated 6	Data record created by interaction
	vcDataRecordCreatedByResourceScheduling 5	Data record automatically created by resource scheduling
	vcLinkCreated 2	Link created by linking two nodes
	vcNodeCreated 1	node created via mouse-click
	vcNodesAndLinksCloned 4	selected nodes were copied via dragging the mouse and pressing the the Ctrl button
	vcNodeWithLinkCreated 3	nodes and links created simultanously
⇨ isLastLinkInSeries	Boolean	The created link is/is not the only link or the last link of a link collection.

Example Code

```
Private Sub VcNet1_OnLinkCreateComplete(ByVal link As VcNetLib.VcLink, _
                            ByVal creationType As VcNetLib.CreationTypeEnum, _
                            ByVal isLastLinkInSeries As Boolean)

    'create a record in the underlying database of the application
    addLinkRecordToDatabase link.AllData

End Sub
```

OnLinkDelete

Event of VcNet

This event occurs when a user deletes a link by the context menu. The link object to be deleted is returned, so that you can still check for - whatever - conditions and prohibit the deletion on a negative result. If you wish to inhibit the deletion, the returnStatus needs to be set to **vcRetStatFalse**; on **vcRetStatOK** the link will be deleted; on **vcRetStatDefault** the pre-defined default behavior will remain unchanged and the link will also be deleted; on

vcRetStatPopup the popup menu will be invoked to offer further options for interaction to the user.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link deleted
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkDelete(ByVal link As VcNetLib.VcLink, _
                                returnStatus As Variant)
    'deny deletion of link with a certain predecessor
    If link.PredecessorNode.DataField(3) = "X" Then
        returnStatus = vcRetStatFalse
    End If
End Sub
```

OnLinkDeleteComplete

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
Parameter:		
⇨ link	VcLink	Link deleted
⇨ isLastLinkInSeries	Boolean	The deleted link is/is not the only one or/nor the last one of a link collection.

OnLinkLClickCltn

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
Parameter:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	Long	X value
⇨ y	Long	Y value

⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkLClickCltn(ByVal linkCltn As _
                                        VcNetLib.VcLinkCollection, _
                                    ByVal x As Long, ByVal y As Long, _
                                    returnStatus As Variant)
    Dim link As VcLink

    ' set certain data field of all links
    For Each link In linkCltn
        link.DataField(2) = "A"
    Next
End Sub
```

OnLinkLDblClickCltn

Event of 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
Parameter:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkLDblClickCltn(ByVal linkCltn As _
                               VcNetLib.VcLinkCollection, _
                               ByVal x As Long, ByVal y As Long, _
                               returnStatus As Variant)
    'edit link in own dialog
    If linkCltn.Count = 1 Then
        On Error GoTo CancelError
        frmEditLinkDialog.setLink linkCltn.FirstLink
        frmEditLinkDialog.Show Modal, Me
    End If

CancelError:
    'deny the "Edit Link Data" dialog
    returnStatus = vcRetStatFalse
End Sub
```

OnLinkModifyComplete

Event of VcNet

This event occurs when the modification of the link specified is 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 a validation can be made.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	link created
⇨ isLastLinkInSeries	Boolean	The created link is/is not the only link or the last link of a link collection.

Example Code

```
Private Sub VcNet1_OnLinkModifyComplete(ByVal link As _
                              VcNetLib.VcLink, ByVal isLastLinkInSeries _
                              As Boolean)
    ' modify a record in the underlying database of the application
    modifyDataRecord link.AllData
End Sub
```

OnLinkModifyEx

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 **OnLinkModifyComplete**.

	Data Type	Explanation
Parameter:		
⇨ link	VcLink	Link after modification
⇨ oldlink	VcLink	Link before modification
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkModify(ByVal link As VcNetLib.VcLink, _
                              returnStatus As Variant)
    'deny any modification
    returnStatus = vcRetStatFalse
```

```
End Sub
```

OnLinkRClickCltn

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
Parameter:		
⇨ linkCltn	VcLinkCollection	LinkCollection object hit
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkRClickCltn(ByVal linkCltn As _
                                        VcNetLib.VcLinkCollection, _
                                    ByVal x As Long, ByVal y As Long, _
                                    returnStatus As Variant)
    ' start a popup menu at the current mouse cursor position
    PopupMenu mnuLinkPopup

    ' revoke the VARCHART XNet context menu
    returnStatus = vcRetStatNoPopup
End Sub
```

OnLinksMark

Event of VcNet

This event occurs after the user selected links to be marked or unmarked. The parameters hold information on the mouse button and control key pressed. If you set the return status to **vcRetStatFalse**, the link will not be marked or unmarked.

This event should be used only for reading data from the current link, but not for modifying them. For modifying the data please use **OnLinksMark-Complete**.

	Data Type	Explanation
Parameter:		
⇨ linkCollection	VcLinkCollection	LinkCollection that contains the links selected by the user. If the user clicked in the diagram, the collection is empty.
⇨ button	Integer	Indicates in which way the buttons were marked: **0**: by keyboard, **1**: left mouse button pressed, **2**: right mouse button pressed, **4**: mouse button pressed
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and the Alt keys were pressed, the value of **shift** would equal "6".
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnLinkMark()
  If MsgBox("Mark this link?", vbYesNo, "Marking links") = _
                               vbNo Then returnStatus = vcRetStatFalse
End Sub
```

OnLinksMarkComplete

Event of VcNet

This event occurs when marking or unmarking links was finished.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

Example Code

```
Private Sub VcNet1_OnLinkMarkComplete()
    MsgBox "Links have been successfully marked."
End Sub
```

OnModifyComplete

Event of VcNet

This event occurs when data were modified interactively in the chart, i.e. after the below events:

- OnBoxModifyComplete

- OnLinkCreateComplete
- OnLinkDeleteComplete
- OnNodeCreateCompleteEx
- OnNodeDelete
- OnNodeModifyComplete

This event allows you to set a mark in the application that reminds to save the data before closing the program.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

OnMouseDblClk

Event of VcNet

This event occurs when the user double-clicks a mouse button.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OnMouseDown

Event of VcNet

This event occurs when the user clicks a mouse button.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OnMouseMove

Event of VcNet

This event occurs when the user moves the mouse.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇨ x	Long	X coordinate of the mouse cursor

⇨ y	Long	Y coordinate of the mouse cursor

OnMouseUp

Event of VcNet

This event occurs when the user loosens the pressed left mouse button.

Please also regard the **MouseProcessingEnabled** property.

	Data Type	Explanation
Parameter:		
⇨ button	Integer	indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ Shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇨ x	Long	X coordinate of the mouse cursor
⇨ y	Long	Y coordinate of the mouse cursor

OnNodeCreate

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**. If you set the returnStatus to **vcRetStatFalse**, the node will be deleted.

This event should be used only for reading data of the current node, but not for modifying them. For modifying data please use **OnNodeCreate-CompleteEx**.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node to be created

⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeCreate(ByVal node As VcNetLib.VcNode, _
                                returnStatus As Variant)
    'show own edit dialog for the new node
    '  (EditNewNodes attribute must be set to off!)
    On Error GoTo CancelError
    frmEditDialog.Show Modal, Me

    'create a record in the underlying database of the application
    addDataRecord node.AllData

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse
End Sub
```

OnNodeCreateCompleteEx

Event of 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
Parameter:		
⇨ node	VcNode	Node created
⇨ creationType	CreationTypeEnum	Creation type
	Possible Values:	
	vcDataRecordCreated 6	Data record created by interaction
	vcDataRecordCreatedByResourceScheduling 5	Data record automatically created by resource scheduling
	vcLinkCreated 2	Link created by linking two nodes
	vcNodeCreated 1	node created via mouse-click
	vcNodesAndLinksCloned 4	selected nodes were copied via dragging the mouse and pressing the the Ctrl button
	vcNodeWithLinkCreated 3	nodes and links created simultanously
⇨ isLastNodeInSeries	Boolean	The created node is/is not the only node or the last node of a node collection.

Example Code

```
Private Sub VcNet1_OnNodeCreateCompleteEx(ByVal node As _
                                VcNetLib.VcNode, ByVal creationType As _
```

```
                                VcNetLib.CreationTypeEnum, _
                                ByVal isLastNodeInSeries As Boolean)
   'create a record in the underlying database of the application
   addDataRecord node.AllData
End Sub
```

OnNodeDelete

Event of VcNet

This event occurs when the user deletes a node by the context menu. The node object to be deleted is returned, so that you can still check for - whatever - conditions and prohibit the deletion on a negative result. If you wish to inhibit the deletion, the returnStatus needs to be set to **vcRetStat-False**; on **vcRetStatOK** the node will be deleted; on **vcRetStatDefault** the pre-defined default behavior will remain unchanged and the node will also be deleted; on **vcRetStatPopup** the popup menu will be invoked to offer further options for interaction to the user.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node object
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeDelete(ByVal node As VcNetLib.VcNode, _
                                returnStatus As Variant)
   ' deny the deletion of the last node in the chart
    If VcNet1.NodeCollection.Count = 1 Then
        returnStatus = vcRetStatFalse
        MsgBox ("The last node in the chart cannot be deleted.")
    End If
End Sub
```

OnNodeDeleteCompleteEx

Event of VcNet

This event occurs when deleting a node interactively is completed. The node object and the information whether the deleted node was the last one of a batch are returned for data validation.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node deleted
⇨ isLastNodeInSeries	Boolean	The deleted node is (True) / is not (False) the last node of batch

OnNodeLClick

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
Parameter:		
⇨ node	VcNode	Node object
⇨ location	LocationEnum	Placed in the chart
	Possible Values: vcInDiagram 1	Located in the node area
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeLClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
    'change data field of the node
    node.DataField(10) = 1 - CInt(node.DataField(10))
End Sub
```

OnNodeLDblClick

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
Parameter:		
⇨ node	VcNode	Node object
⇨ location	LocationEnum	Placed in the chart
	Possible Values: vcInDiagram 1	Located in the node area

⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeLDblClick(ByVal node As VcNetLib.VcNode, _
                                   ByVal location As VcNetLib.LocationEnum, _
                                   ByVal x As Long, ByVal y As Long, _
                                   returnStatus As Variant)

    'show own "Edit Node" dialog
    On Error GoTo CancelError
    frmEditDialog.setNode node
    frmEditDialog.Show Modal, Me

    returnStatus = vcRetStatFalse

    Exit Sub

CancelError:
    returnStatus = vcRetStatFalse

End Sub
```

OnNodeModifyComplete

Event of VcNet

This event occurs when the modification of the node specified is finished.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	node created
⇨ isLastNodeInSeries	Boolean	The created node is/is not the only node or the last node of a node collection.

Example Code

```
Private Sub VcNet1_OnNodeModifyComplete(ByVal node As VcNetLib.VcNode, ByVal
isLastNodeInSeries As Boolean)
    'modify a record in the underlying database of the application
    modifyDataRecord node.AllData
End Sub
```

OnNodeModifyCompleteEx

Event of VcNet

This event occurs after the user has modified the node hierarchy.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	node modified
⇨ isLastNodeInSeries	Boolean	The modified node is/is not the only node or the last node of a node collection.
⇨ modificationType	ModificationTypeEnum	type of modification
	Possible Values:	
	vcAnything 1	modification type not determined
	vcChangedGroup 16	group of the node changed
	vcMoved 8	Object was moved
	vcNothing 0	no modification

OnNodeModifyEx

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 returnStatus to **vcRetStatFalse**, the modification will be inhibited.

This event should be used only for reading data of the current node, but not for modifying them. For modifying data please use **OnNodeModify-Complete**.

	Data Type	Explanation
Parameter:		
⇨ oldNode	VcNode	Node before the modification
⇨ node	VcNode	Node to be modified
⇨ modificationType	ModificationTypeEnum	Type of modification
	Possible Values:	
	vcAnything 1	modification type not determined
	vcChangedGroup 16	group of the node changed
	vcMoved 8	Object was moved
	vcNothing 0	no modification
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeModifyEx(ByVal oldNode As _
                                  VcNetLib.VcNode, ByVal node As _
                                  VcNetLib.VcNode, ByVal modificationType As _
                                  VcNetLib.ModificationTypeEnum, returnStatus _
```

```
                                    As Variant)

    ' Revoke the modification if the node would change the group
    If modificationType And vcChangedGroup Then
        MsgBox "The node cannot be moved into another group."
        returnStatus = vcRetStatFalse
    End If

End Sub
```

OnNodeRClick

Event of 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 **OnNodesMark-Complete**.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node object
⇨ location	LocationEnum	Placed in the chart
	Possible Values: vcInDiagram 1	Located in the node area
⇨ x	Long	X value
⇨ y	Long	Y value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    ' start a popup menu at the current mouse cursor position
    PopupMenu mnuNodePopup

    returnStatus = vcRetStatNoPopup

End Sub
```

OnNodesMarkComplete

Event of VcNet

This event occurs when the operation of marking or unmarking nodes is finished.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

Example Code

```
Private Sub VcNet1_OnNodesMarkComplete()
    MsgBox "Nodes have been successfully marked."
End Sub
```

OnNodesMarkEx

Event of VcNet

This event occurs after the user selected one or more nodes for marking or unmarking. The nodes are contained by the nodeCollection. The parameters **button** and **shift** hold information on the control key and mouse button pressed. If you set the return status to **vcRetStatFalse**, marking or unmarking will not take place.

	Data Type	Explanation
Parameter:		
⇨ nodeCollection	VcNodeCollection	NodeCollection that contains the nodes selected by the user. If the user has clicked in the diagram, the collection is empty.
⇨ button	Integer	Indicates in which way the buttons were marked: **0**: via keyboard, **1**: left mouse button pressed, **2**: right mouse button pressed, **4**: mouse button pressed
⇨ shift	Integer	Number that indicates which one of the **Shift**, **Ctrl**, and **Alt** keys was pressed. **1** corresponds to the Shift key, **2** to the Ctrl key and **4** to the Alt key. Some, all, or none of the numbers may have been set, indicating that some, all, or none of the keys are depressed, respectively. When more than one key is in depressed state, their values add up. For example, if both the Ctrl and Alt keys are depressed, the value of **shift** would be "6".
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodesMarkEx(ByVal NodeCollection As _
                                 VcNetLib.VcNodeCollection, _
                                 ByVal button As Integer, _
                                 ByVal shift As Integer, _
```

```
                                returnStatus As Variant)

    If MsgBox("Mark this node?", vbYesNo, "Marking nodes") = _
                                vbNo Then returnStatus = vcRetStatFalse

End Sub
```

OnSelectField

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:		
editObject	Object in	
editObjectType	VcObjectTypeEnum in	
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeGroup 7	object type **group**
	vcObjTypeLinkCollection 3	object type **link collection**
	vcObjTypeNode 2	object type **node**
	vcObjTypeNodeInLegend 17	object type **node in legend area**
	vcObjTypeNone 0	no object
fieldIndex	Long in	
objRectComplete	VcRect in	
objRectVisible	VcRect in	
fldRectComplete	VcRect in	
fldRectVisible	VcRect in	
returnStatus	Variant	

OnShowInPlaceEditor

Event of VcNet

This event occurs when the implemented editor is started.

The event will be triggered only if the property **InPlaceEditingAllowed** was set to **True**.

By setting the return status to **False** this event can be inhibited so that your own editor can be started at the coordinates passed.

	Data Type	Explanation
Parameter:		
⇨ editObject	Object	Object edited
⇨ editObjectType	VcObjectTypeEnum	Object type
	Possible Values: vcObjTypeBox 15 vcObjTypeGroup 7 vcObjTypeLinkCollection 3 vcObjTypeNode 2 vcObjTypeNodeInLegend 17 vcObjTypeNone 0	 object type **box** object type **group** object type **link collection** object type **node** object type **node in legend area** no object
⇨ fieldIndex	Long	Field index
⇨ objRectComplete	VcRect	Complete rectangle of the object hit
⇨ objRectVisible	VcRect	Visible rectangle of the object hit
⇨ fldRectComplete	VcRect	Complete rectangle of the field hit
⇨ fldRectVisible	VcRect	Visible rectangle of the field hit
returnStatus	Variant	

Example Code

```
Private Sub VcNet1_OnShowInPlaceEditor(ByVal editObject As Object, _
                  ByVal editObjectType As VcNetLib.VcObjectTypeEnum, _
                  ByVal fieldIndex As Long, ByVal objRectComplete As _
                  VcNetLib.VcRect, ByVal objRectVisible As _
                  VcNetLib.VcRect, ByVal fldRectComplete As _
                  VcNetLib.VcRect, ByVal fldRectVisible As _
                  VcNetLib.VcRect, returnStatus As Variant)

   Dim oldScaleMode As Long

   If editObjectType = vcObjTypeNode Then
      returnStatus = vcRetStatFalse

      Set myEditObject = editObject
      myEditObjectType = editObjectType
      myEditObjectFieldIndex = fieldIndex
      oldScaleMode = Me.ScaleMode
      Me.ScaleMode = vbPixels

      Select Case fieldIndex
         Case 1   'Name
            Text1.Left = fldRectVisible.Left + VcNet1.Left
            Text1.Top = fldRectVisible.Top + VcNet1.Top
            Text1.Width = fldRectVisible.Width
            Text1.Height = fldRectVisible.Height
            Text1.Text = editObject.DataField(fieldIndex)
            Text1.Visible = True
            Text1.SetFocus

         Case 2, 3     'Start or End
            MonthView1.Left = fldRectVisible.Left + VcNet1.Left
            MonthView1.Top = fldRectVisible.Top + VcNet1.Top
            MonthView1.Value = editObject.DataField(fieldIndex)
            MonthView1.Visible = True
            MonthView1.SetFocus

         Case 13   'Employee
            Combo1.Left = fldRectVisible.Left + VcNet1.Left
            Combo1.Top = fldRectVisible.Top + VcNet1.Top
```

```
            Combo1.Width = fldRectVisible.Width
            Combo1.Text = editObject.DataField(fieldIndex)
            Combo1.Visible = True
            Combo1.SetFocus

      End Select

      Me.ScaleMode = oldScaleMode

   End If

End Sub
```

OnStatusLineText

Event of 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
Parameter:		
⇨ text	String	text

Example Code

```
Private Sub VcNet1_OnStatusLineText(ByVal Text As String)
    'show text on status bar
    txtStatusBar.Text = Text
End Sub
```

OnSupplyTextEntry

Event of VcNet

This event only occurs when the VcNet property **EnableSupplyText-EntryEvent** was set to True. It occurs when a text is to be displayed. You can use this event for editing the texts of context menus, dialog boxes, info boxes, error messages and the names of days and months.

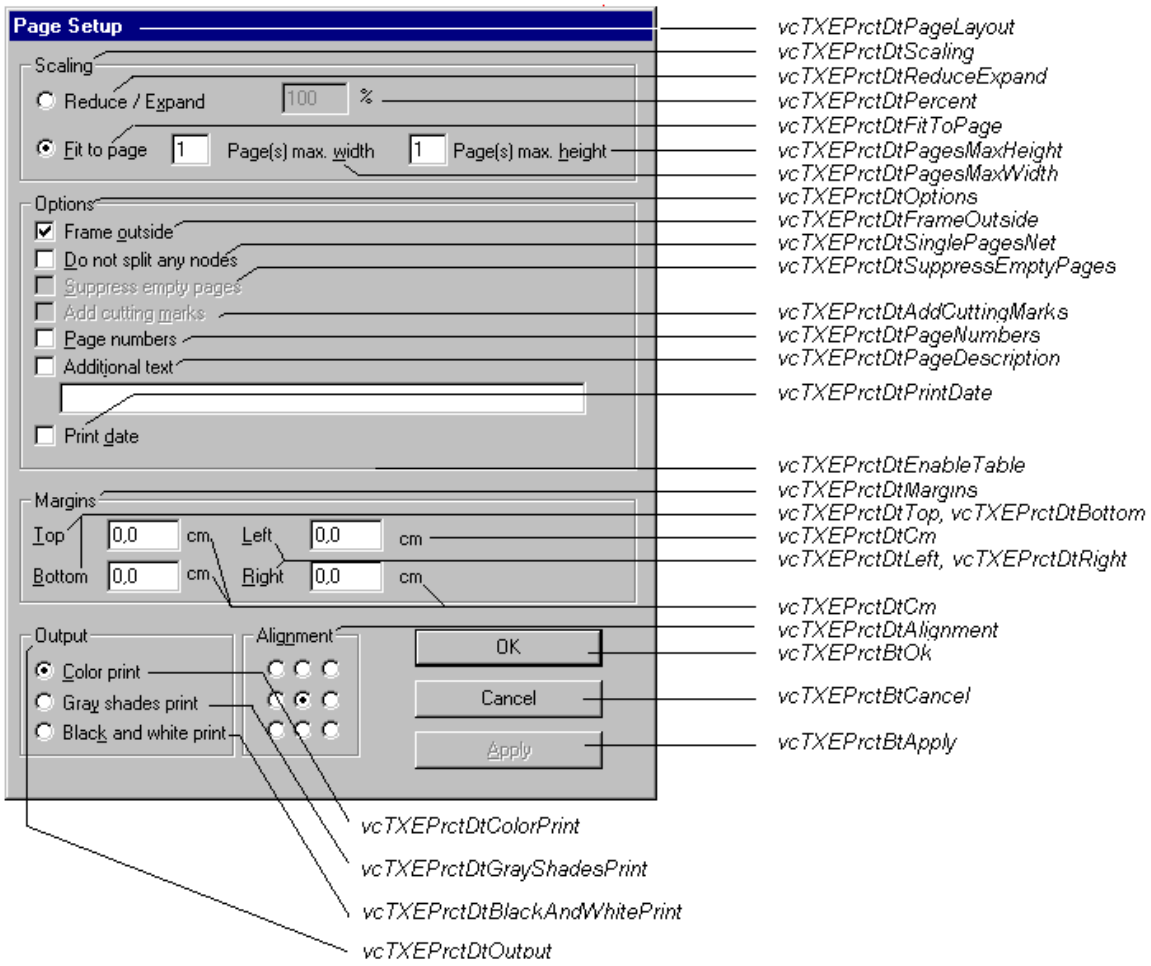
	Data Type	Explanation
Parameter:		
⇨ controlIndex	TextEntryIndexEnum	Text to be replaced
	Possible Values:	
	vcTXECtxmenArrange 2150	Text in context menu: **Arrnge nodes**
	vcTXECtxmenArrowMode 2116	Text in context menu:**Pointer mode**
	vcTXECtxmenCopyNodes 2152	Text in context menu: **Copy nodes**
	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**
vcTXECtxmenShowLegendView 2158	Text in context menu: **Show legend view**
vcTXECtxmenShowWorldView 2157	Text in context menu: **Show world view**
vcTXECtxmenSubDiagram 2155	Text in context menu: **Build sub net**
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

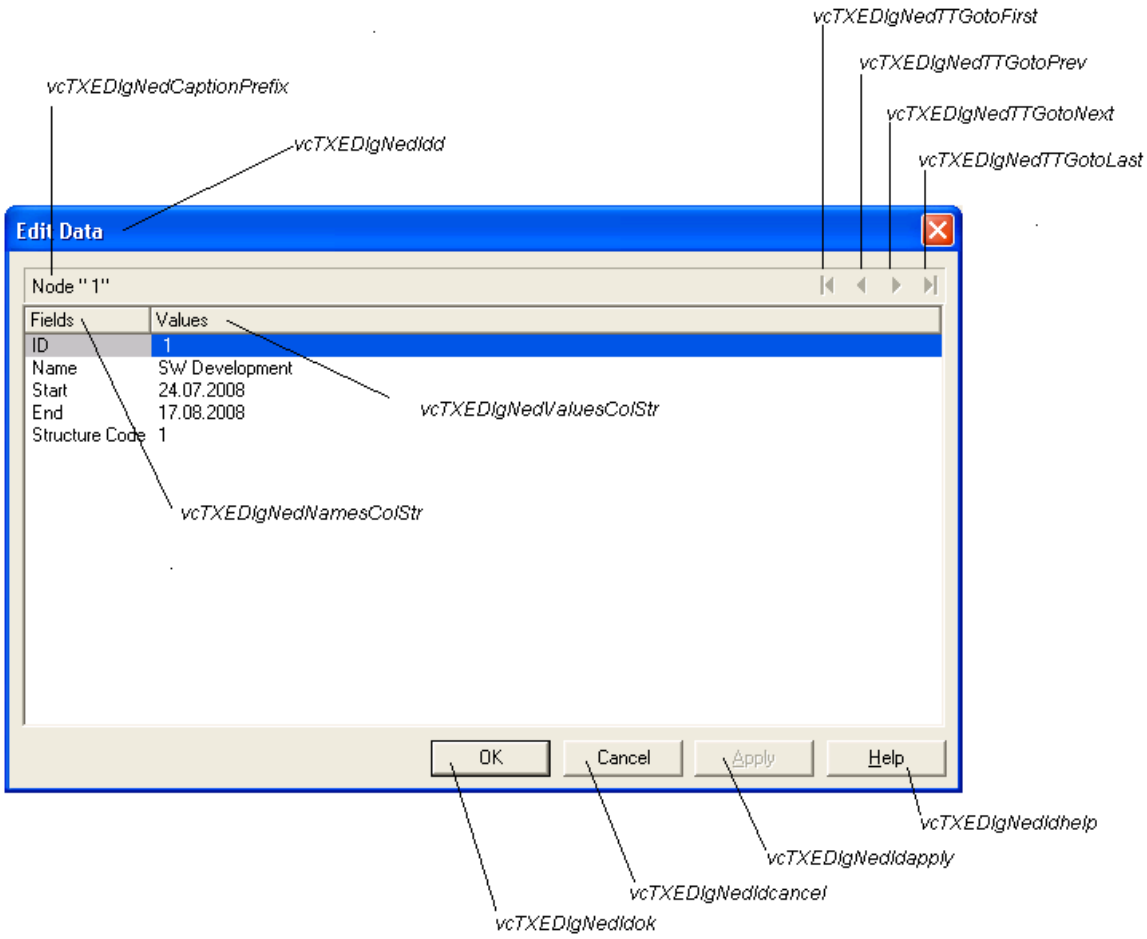
vcTXEDlgLegLegendTitleVisible 2043	Text in the **Legend Attributes** dialog: **Legend title visible**
vcTXEDlgLegMargins 2050	Text in the **Legend Attributes** dialog: **Margins**
vcTXEDlgLegTopMargin 2051	Text in the **Legend Attributes** dialog: **Top margin:**
vcTXEDlgNedCaptionPrefix 2024	**Edit 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."
vcTXEPrctBtAll 2306	Button text in **Print Preview** dialog: **Overview**
vcTXEPrctBtApply 2318	Button text in **Page Setup** dialog: **Apply**
vcTXEPrctBtCancel 2302	Button text in **Print Busy** box: **Cancel**
vcTXEPrctBtClose 2303	Button text in **Print Preview** dialog: **Close**
vcTXEPrctBtFitToPage 2308	Button text in **Print Preview** dialog: **Fit To Page**
vcTXEPrctBtNext 2305	Button text in **Print Preview** dialog: **Next**
vcTXEPrctBtOk 2301	Button text in **Page Setup** dialog: **OK**
vcTXEPrctBtPageLayout 2311	Button text in **Print Preview** dialog: **Page Setup**
vcTXEPrctBtPreviewZoomFactorItems 2321	Entries in the combobox **Zoom factor** of the **Print Preview** dialog: **!Auto|75%|100%|150%|200%**
vcTXEPrctBtPrevious 2304	Button text in **Print Preview** dialog: **Previous**
vcTXEPrctBtPrint 2313	Button text in **Print Preview** dialog: **Print**
vcTXEPrctBtPrinterSetup 2312	Button text in **Print Preview** dialog: **Printer setup**
vcTXEPrctBtSingle 2307	Button text in **Print Preview** dialog: **Single**
vcTXEPrctBtZoomPrint 2319	Button text in **Print Preview** dialog: **Print Area…**
vcTXEPrctDtAddCuttingMarks 2514	Text in the **Page Setup** dialog: **Show crop marks**
vcTXEPrctDtAlignment 2526	Text in the **Page Setup** dialog: **Alignment**
vcTXEPrctDtAlignmentItems 2583	Text in the **Page Setup** dialog: **Top left|Top|Top right|Left|Centered|Right|Bottom left|Bottom|Bottom right**
vcTXEPrctDtBottom 2521	Text in the **Page Setup** dialog: **Bottom**
vcTXEPrctDtCm 2530	Text in the **Page Setup** dialog: **cm**
vcTXEPrctDtCurrentValues 2581	Text in the **Page Setup** dialog: **Current**
vcTXEPrctDtEnableBoth 2561	not active
vcTXEPrctDtEnableDiagram 2559	Text in **Page Setup** dialog: **Show diagram**
vcTXEPrctDtEnableTable 2558	not active

vcTXEPrctDtExportPage 2568	
vcTXEPrctDtFitToPage 2508	Text in the **Page Setup** dialog: **Fit to page counts**
vcTXEPrctDtFoldingMarksItems 2577	Text in the **Page Setup** dialog: **Form A|Form B|Form C**
vcTXEPrctDtFoldingMarksText 2576	Text in the **Page Setup** dialog: **Show &folding marks (DIN 824):**
vcTXEPrctDtFooterGroup 2584	Text in the **Page Setup** dialog: **Footer line**
vcTXEPrctDtFrameOutside 2515	Text in the **Page Setup** dialog: **Show frame outside**
vcTXEPrctDtInch 2588	Text in the **Page Setup** dialog: **in**
vcTXEPrctDtLeft 2520	Text in the **Page Setup** dialog: **Left**
vcTXEPrctDtMargins 2529	Text in the **Page Setup** dialog: **Minimum sizes for sheet margins**
vcTXEPrctDtMaxPages 2580	Text in the **Page Setup** dialog: **pages**
vcTXEPrctDtOff 2557	Text **Off** dialog
vcTXEPrctDtOptions 2528	Text in the **Page Setup** dialog: **Options**
vcTXEPrctDtPageDescription 2562	Text in **Page Setup** dialog: **Text**
vcTXEPrctDtPageLayout 2532	**Page Setup** dialog: Text in Title Bar
vcTXEPrctDtPageNumberingItems 2582	Text in the **Page Setup** dialog: **Row.Column|Column.Row|Page/Count**
vcTXEPrctDtPageNumbers 2518	Text in the **Page Setup** dialog: **Page numbering**
vcTXEPrctDtPagePadding 2585	Text in the **Page Setup** dialog: **&Pad pages with space**
vcTXEPrctDtPagePreview 2533	**Print Preview** dialog: Text in Title Bar
vcTXEPrctDtPagesMaxHeight 2511	Text in the **Page Setup** dialog: **Maximum height**
vcTXEPrctDtPagesMaxWidth 2510	Text in the **Page Setup** dialog: **Maximum. width**
vcTXEPrctDtPercent 2509	Text in the **Page Setup** dialog: **%**
vcTXEPrctDtPrint 2506	Text in Print Busy Box: **Print**
vcTXEPrctDtPrintDate 2564	Text in **Page Setup** dialog: **Additionally print current &date**
vcTXEPrctDtPrintingPage 2556	Text in Print Busy Box: **Printing page %1 of %2 on**
vcTXEPrctDtProjectName 2502	Text in Print Busy Box: **project name**
vcTXEPrctDtReduceExpand 2507	Text in the **Page Setup** dialog: **Zoom factor**
vcTXEPrctDtRight 2522	Text in the **Page Setup** dialog: **Right**
vcTXEPrctDtScaling 2527	Text in the **Page Setup** dialog: **Scaling**
vcTXEPrctDtScalingMode 2578	Text in the **Page Setup** dialog: **&Mode:**
vcTXEPrctDtStatusBarCurrentValues 2586	Text in the **Status bar** of the **Page Setup** dialog: **Page %1 selected (in row %2, column %3)**
vcTXEPrctDtStatusBarSelectedPage 2587	Text in the **Status bar** of the **Page Setup** dialog: **Page %1 selected (in row %2, column %3)**
vcTXEPrctDtTableColumnRange 2575	Text in the **Page Layout** dialog: **Show table columns (e.g. 1-5;7)**
vcTXEPrctDtTop 2519	Text in the **Page Setup** dialog: **Top**
vcTXEPrctDtZoomFactor 2579	Text in the **Page Setup** dialog: **&Zoom factor:**
vcTXEPrctMtAdjustBottomAndTopMargin 2437	Message text: **The bottom margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the top margin will be adjusted to %2 cm.**
vcTXEPrctMtAdjustLeftAndRightMargin 2434	Message text: **The left margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the right margin will be reduced to %2 cm.**
vcTXEPrctMtAdjustRightAndLeftMargin 2435	Message text: **The right margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the left margin will be adjusted to %2 cm.**

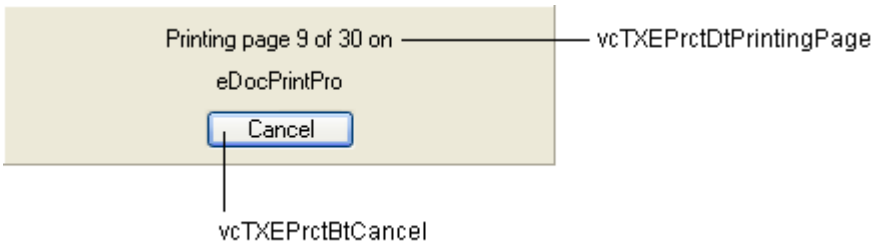
	vcTXEPrctMtAdjustTopAndBottomMargin 2436	Message text: **The top margin is out of range and therefore will be reduced to %1 cm.\r\nIn addition, the bottom margin will be reduced to %2 cm.**
	vcTXEPrctMtBottomMargin 2409	Message text: **Bottom margin is out of range and therefore will be reduced to %s cm.**
	vcTXEPrctMtIncompatibleVcVersion 2414	Message text: **VcVersion incompatible**
	vcTXEPrctMtLeftMargin 2406	Message text: **Left margin is out of range and therefore will be reduced to %s cm.**
	vcTXEPrctMtPrinterNotInstalled 2411	Message text: **Printer not installed**
	vcTXEPrctMtPrintingNotPossible 2402	Message text: **Printing not possible at time**
	vcTXEPrctMtRightMargin 2408	Message text: **Right margin is out of range and therefore will be reduced to %s cm.**
	vcTXEPrctMtSelectPaperSize 2413	Message text: **Selected paper size too small**
	vcTXEPrctMtTopMargin 2407	Message text: **Top margin is out of range and therefore will be reduced to %s cm.**
	vcTXEPrctMtValueOutOfRange 2404	Message text: **Value out of range %1 to %2**
	vcTXEPrctMtWillBeAdjustedTo 2410	Message text: **Will be adjusted to...**
	vcTXERelTypeLongFF 3001	Text in the **Edit links** dialog: **Finish-to-finish (FF)**
	vcTXERelTypeLongFS 3000	Text in the **Edit links** dialog: **Finish-to-start (FS)**
	vcTXERelTypeLongSF 3003	Text in the **Edit links** dialog: **Start-to-finish (SF)**
	vcTXERelTypeLongSS 3002	Text in the **Edit links** dialog: **Start-to-start (SS)**
⇨ textEntry	String	Text replacing the default
⇔ returnStatus	Variant	Return status



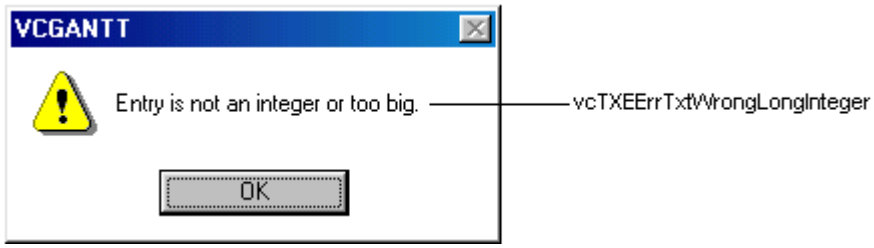
*Constants of the button texts of the **Page Setup** dialog*



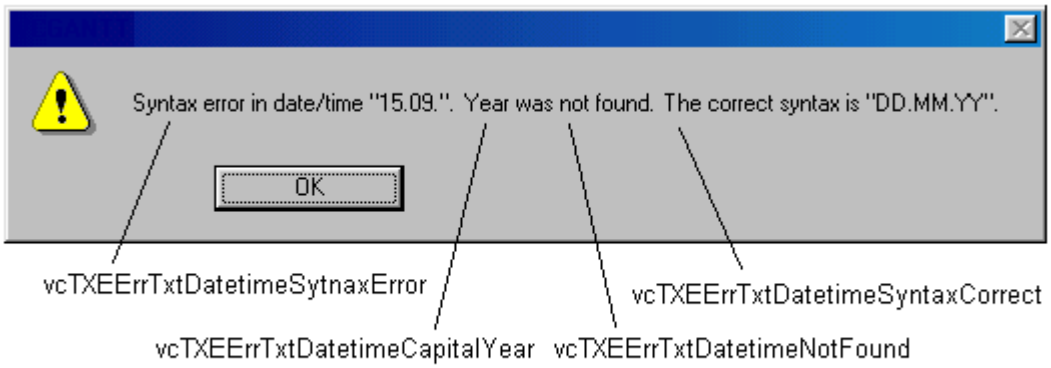
*Constants of the dialogs **Edit data** and **Edit link**, here illustrated by the **Edit data** dialog*



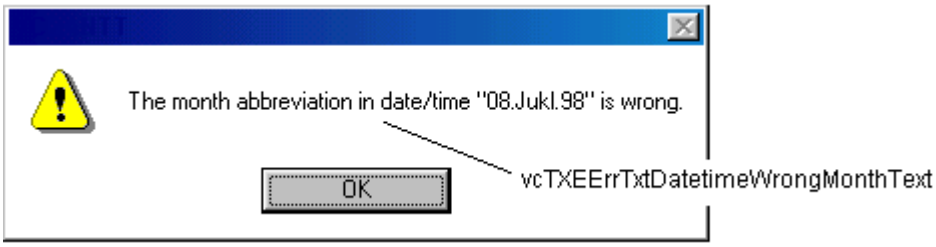
*Constants of the info box **Printing***



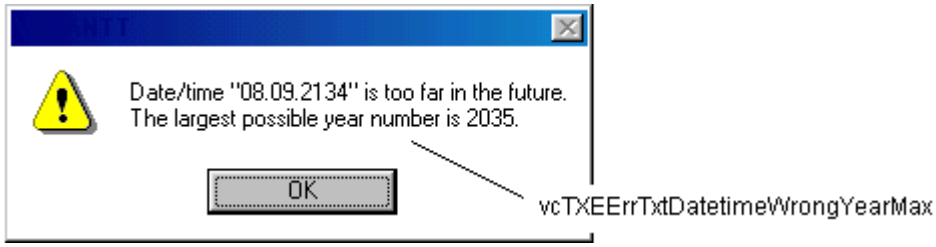
*Constants of the error message **Entry is not an integer value.***



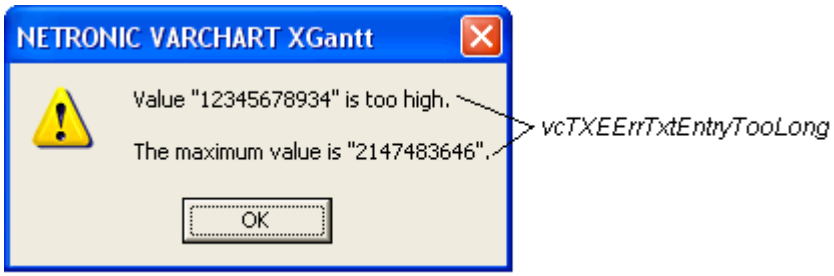
*Constants of the error message **Syntax error***



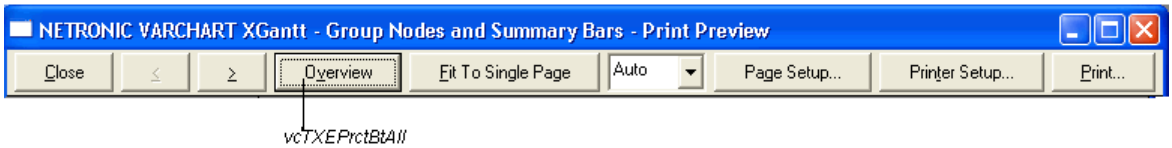
*Constants of the error message **Date error, wrong month***



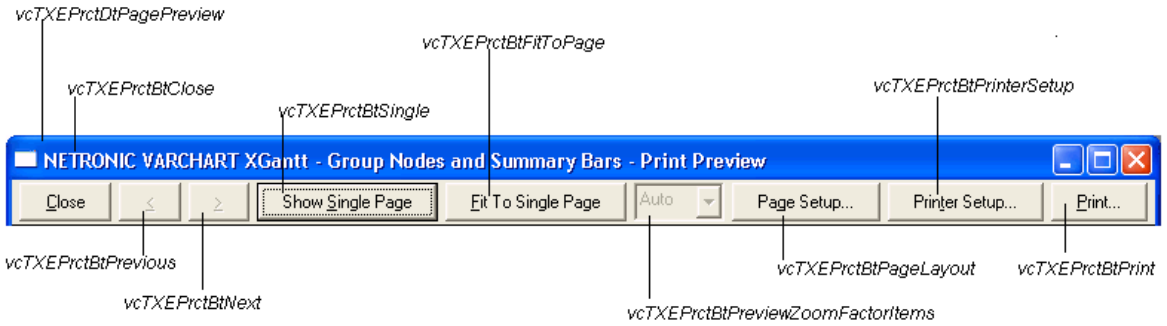
*Constants of the error message **Date error, maximum year exceeded***



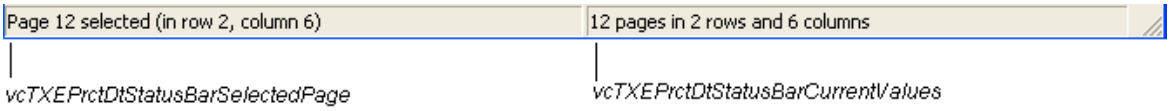
*Constants of the error message **Entry too large.***



*Constants of the button texts of the **Print Preview** dialog*



*Constants of the button texts of the **Print Preview, Overview** dialog*



*Constants of the status bar in the dialog **Print Preview***

Example Code

```
Private Sub VcNet1_OnSupplyTextEntry(ByVal controlIndex As _
                                        VcNetLib.TextEntryIndexEnum, _
                                     textEntry As String, _
                                     returnStatus As Variant)
    'change the texts of the context meun items
    Select Case controlIndex
        Case vcTXECtxmenArrange
            textEntry = "Position all nodes automatically"
    End Select
End Sub
```

OnSupplyTextEntryAsVariant

Event of VcNet

This event is identical with the event **OnSupplyTextEntry** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

OnToolTipText

Event of VcNet

This event only occurs when the VcNet property **ShowToolTip** 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 **vcRetStat-False** or by leaving the text string empty you can suppress the display of the tooltip.

	Data Type	Explanation
Parameter:		
⇨ hitObject	Object	Object
⇨ hitObjectType	VcObjectTypeEnum	object type
	Possible Values: vcObjTypeBox 15 vcObjTypeGroup 7 vcObjTypeLinkCollection 3 vcObjTypeNode 2 vcObjTypeNodeInLegend 17 vcObjTypeNone 0	 object type **box** object type **group** object type **link collection** object type **node** object type **node in legend area** no object
⇨ x	Long	X value
⇨ y	Long	Y value
⇨ ToolTipText	String	Text to be displayed, can contain 1024 characters maximum
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnToolTipText(ByVal hitObject As Object, _
                                 ByVal hitObjectType As _
                                 VcNetLib.VcObjectTypeEnum, _
                                 ByVal x As Long, ByVal y As Long, _
                                 toolTipText As String, _
                                 returnStatus As Variant)
    If hitObjectType = vcObjTypeNode Then
        toolTipText = "Moving over node!"
    End If
End Sub
```

OnToolTipTextAsVariant

Event of VcNet

This event is identical with the event **OnToolTipText** except for the parameters. It was necessary to implement this event because some languages (e.g. VBScript) can use parameters by Reference (indicated by ⇦) only if the type of these parameters is VARIANT.

OnWorldViewClosed

Event of VcNet

This event occurs when the worldview popup window is closed.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		

Example Code

```
Private Sub VcNet1_OnWorldViewClosed()
    MsgBox "Do you want to close the worldview window?", vbOKCancel
End Sub
```

OnZoomFactorModifyComplete

Event of VcNet

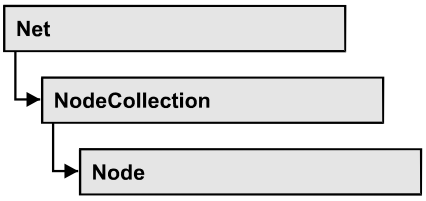
This events occurs if the user modified the size of the rectangle in the world view or if he zoomed marked objects. You can zoom smoothly by keeping the **Ctrl** key pressed while turning the mouse wheel, or in discrete steps while using the **Plus** or **Minus** keys in the number pad.

	Data Type	Explanation
Parameter: ⇦ (no parameter)		

Example Code

```
Private Sub VcNet1_OnZoomFactorModifyComplete()
     MsgBox "Zoomfactor: " & Net1.ZoomFactor
End Sub
```

7.43 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
- MarkNode
- OutgoingLinks

Methods

- DataRecord
- DeleteNode
- RelatedDataRecord
- UpdateNode

Properties

AllData

Property of VcNode

This record lets you set or retrieve all data of a node at once. When setting the property, a CSV string (using semicolons as separators) or a variant that contains all data fields of the node in an array are allowed. When retrieving the property, a string will be returned. (See also **InsertNodeRecord**.)

	Data Type	Explanation
Property value	String/data field	All data of the data set

Example Code

```
Private Sub VcNet1_OnNodeModify(ByVal node As VcNetLib.VcNode,  _
                                ByVal modificationType As _
                                VcNetLib.ModificationTypeEnum, _
                                returnStatus As Variant)

    Dim allDataOfNode As String

    returnStatus = vcRetStatFalse

    allDataOfNode = node.AllData
    MsgBox allDataOfNode

End Sub
```

DataField

Property of VcNode

This property lets you assign/retrieve data to/from the data field of a node. If the data field was modified by the **DataField** property, the diagram needs to be updated by the **UpdateNode** method.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field
Property value	Variant	Content of the data field

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

   If MsgBox("Delete Node: " & node.dataField(0), vbYesNo, "Delete Node") = _
                                vbYes Then node.DeleteNode

   returnStatus = vcRetStatNoPopup

End Sub
```

ID

Read Only Property of VcNode

By this property you can retrieve the ID of a node.

	Data Type	Explanation
Property value	String	Node ID

IncomingLinks

Read Only Property of VcNode

This property gives access to all incoming links of a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim incomingLinks As VcLinkCollection
    Dim link As VcLink
    Dim predecessorNode As VcNode

    Set incomingLinks = node.IncomingLinks

    For Each link In incomingLinks
        Set predecessorNode = link.PredecessorNode
        predecessorNode.MarkNode = True
    Next link

    returnStatus = vcRetStatNoPopup

End Sub
```

MarkNode

Property of VcNode

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	Boolean	Node marked/not marked

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
```

```
    Dim nodeMarked As Boolean

    nodeMarked = node.MarkNode
    MsgBox (nodeMarked)
    returnStatus = vcRetStatNoPopup

End Sub
```

OutgoingLinks

Read Only Property of VcNode

This property gives access to the set of links that leave a node.

	Data Type	Explanation
Property value	VcLinkCollection	Link collection

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As VcNetLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim outgoingLinks As VcLinkCollection
    Dim link As VcLink
    Dim successorNode As VcNode

    Set outgoingLinks = node.outgoingLinks

    For Each link In outgoingLinks
        Set successorNode = link.successorNode
        successorNode.MarkNode = True
    Next link

    returnStatus = vcRetStatNoPopup

End Sub
```

Methods

DataRecord

Method of VcNode

This property lets you retrieve the node as a data record object. The properties of the data record object give access to the corresponding data table and the data table collection.

	Data Type	Explanation
Return value	VcDataRecord	Data record returned

DeleteNode

Method of VcNode

This method lets you delete a node.

	Data Type	Explanation
Return value	Boolean	Node was (true) / was not (false) deleted successfully

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcNetLib.VcNode, _
                                ByVal location As _
                                VcNetLib.LocationEnum, ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    If MsgBox("Delete Node: " & node.DataField(0), vbYesNo, "Delete Node") = _
              vbYes Then node.DeleteNode
    returnStatus = vcRetStatNoPopup

End Sub
```

RelatedDataRecord

Method of VcNode

This property lets you retrieve a data record from a data table that is related to the node data table. The index passed by the parameter denotes the field in the data record that holds the key of the related data record.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of data field that holds the key
Return value	VcDataRecord	Related data record returned

UpdateNode

Method of VcNode

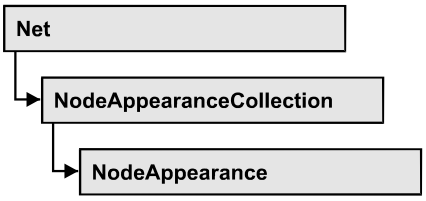
If data fields of a node have been modified by the **DataField** property, the diagram needs to be updated by the **UpdateNode** method.

	Data Type	Explanation
Return value	Boolean	Node was (true) / was not (false) updated successfully

Example Code

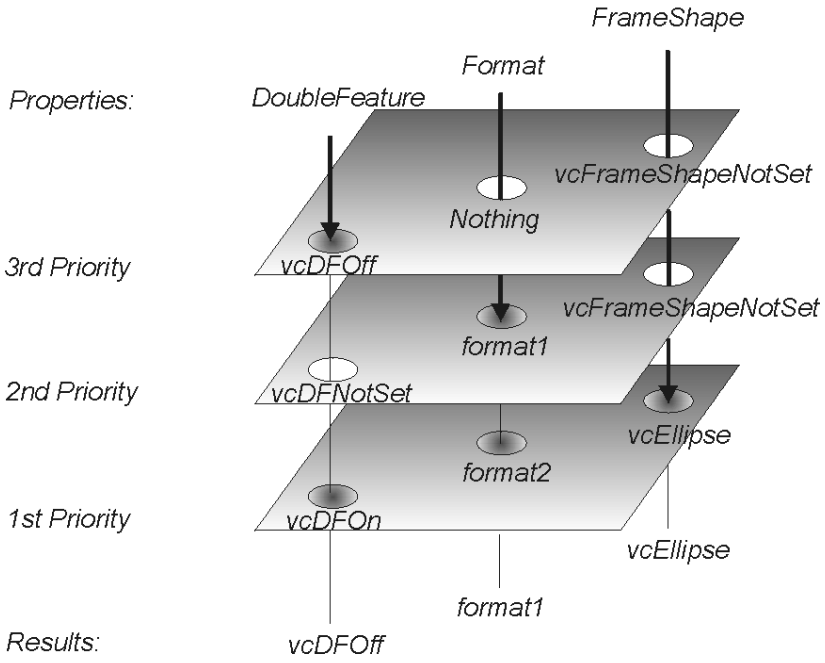
```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcNet1.NodeCollection
Set node = nodeCltn.FirstNode

node.DataField(12) = "Group A"
node.UpdateNode
```

7.44 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

- BackColorAsARGB
- BackColorDataFieldIndex
- BackColorMapName
- DoubleFeature
- FilterName
- FormatName
- FrameAroundFieldsVisible

- FrameShape
- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- Pattern
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- Piles
- Shadow
- ShadowColorAsARGB
- Specification
- StrikeThrough
- StrikeThroughColor
- ThreeDEffect
- VisibleInLegend

Methods

- PutInOrderAfter

Properties

BackColorAsARGB

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. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

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 VcNode-Appearance object).

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},0...255},{0...255},{0...255})

Example Code

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance

nodeAppearance.BackColor = RGB(100, 100, 100)
```

BackColorDataFieldIndex

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	Integer	Data field index

BackColorMapName

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	String	Name of the color map

DoubleFeature

Property of VcNodeAppearance

This property lets you set or retrieve a double lining around the node. When 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	AppearanceDoubleFeatureEnum **Possible Values:** vcDFNotSet -1 vcDFOff 0 vcDFOn 1	Different types of double frames Flag of DoubleFeature not set Flag of DoubleFeature set off Flag of DoubleFeature set on

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.DoubleFeature = 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	String	Name of the filter

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim filtername As String

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance
```

```
filtername = nodeAppearance.filtername
```

FormatName

Property of VcNodeAppearance

This property lets you set or retrieve a format to/from the NodeAppearance object. When empty, the property will adopt the value of the property of a NodeAppearance object next in the descending hierarch which matches the filter conditions and is not empty (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	String	Name of a NodeFormat object or empty string

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim format1 As VcNodeFormat

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

Set format1 = nodeAppearance.format
MsgBox (format1.name)
```

FrameAroundFieldsVisible

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**.

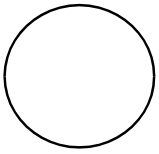
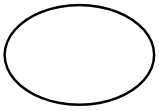
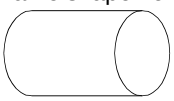
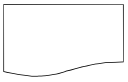
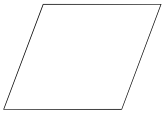
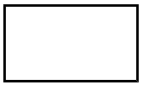
This feature can also be set in the dialog **Edit Node Appearance**.

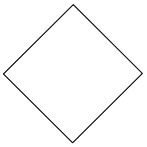
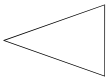
	Data Type	Explanation
Property value	AppearanceFrameAroundFieldsVisibleEnum	Frame around fields **Default value:** -1
	Possible Values: vcFFVNotSet -1 vcFFVOff 0 vcFFVOn 1	 frame line around fields not set Flag of FrameAroundFields set off Flag of FrameAroundFields set on

FrameShape

Property of VcNodeAppearance

This property lets you assign/retrieve the frame shape to/of 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 

vcRhombus 21	frame shape rhombus 
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

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.FrameShape = 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	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/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	Color	RGB color values or **-1** ({0...255},{0...255},{0...255})

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.LineColor = RGB(256, 0, 100)
```

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	Integer	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	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.

When set to **-1**, this 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	Long	Line thickness LineType {1...4}: line thickness in pixels LineType {5...1000}: line thickness in 1/100 mm **Default value:** As defined on property page

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.NodeAppearanceByName("Standard")

nodeAppearance.LineThickness = 3
```

LineType

Property of VcNodeAppearance

This property lets you assign/retrieve the line type to/of 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 was not set to the value **vcNotSet** (see sketch at VcNodeAppearance object).

	Data Type	Explanation
Property value	LineTypeEnum	Line type
	Possible Values:	
	vcDashed 4	Line dashed
	vcDashedDotted 5	Line dashed-dotted
	vcDotted 3	Line dotted
	vcLineType0 100	Line Type 0
	vcLineType1 101	Line Type 1
	vcLineType10 110	Line Type 10
	vcLineType11 111	Line Type 11
	vcLineType12 112	Line Type 12
	vcLineType13 113	Line Type 13
	vcLineType14 114	Line Type 14
	vcLineType15 115	Line Type 15
	vcLineType16 116	Line Type 16
	vcLineType17 117	Line Type 17
	vcLineType18 118	Line Type 18
	vcLineType2 102	Line Type 2
	vcLineType3 103	Line Type 3
	vcLineType4 104	Line Type 4
	vcLineType5 105	Line Type 5
	vcLineType6 106	Line Type 6
	vcLineType7 107	Line Type 7
	vcLineType8 108	Line Type 8
	vcLineType9 109	Line Type 9
	vcNone 1	No line type
	vcNotSet -1	No line type assigned
	vcSolid 2	Line solid

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.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	String	Name

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim nodeAppName As String

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppName = nodeAppearance.name
```

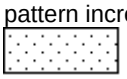
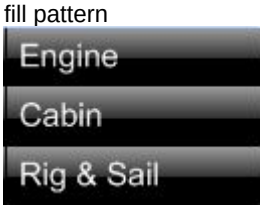
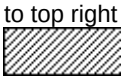
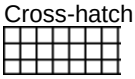
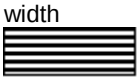
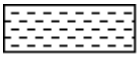
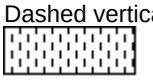
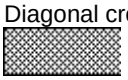
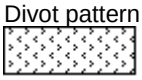
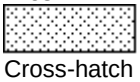
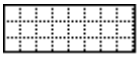
Pattern

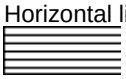
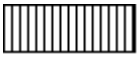
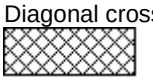
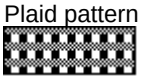
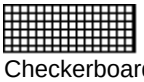
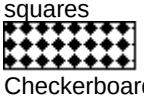
Property of VcNodeAppearance

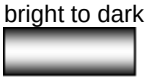
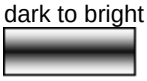
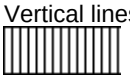
This property lets you set or retrieve the pattern of the node. If in the property **PatternMapName** a map is specified, this map will control the pattern in dependance on the data. If set to **-1**, the property will give way to the property of a nodeAppearance object that matches the filter conditions, 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	FillPatternEnum **Possible Values:**	Pattern type **Default value:** As defined in the dialog

vc05PercentPattern... vc90PercentPattern 01 - 11	Dots in foreground color on background color, the density of the foreground pattern increasing with the percentage 
vcAeroGlassPattern 40	Vertical color gradient in the color of the fill pattern 
vcBDiagonalPattern 5	Diagonal lines slanting from bottom left to top right 
vcCrossPattern 6	Cross-hatch pattern 
vcDarkDownwardDiagonalPattern 2014	Diagonal lines slanting from top left to bottom right; spaced 50% closer than vcFDiagonalPattern and of twice the line width 
vcDarkHorizontalPattern 2023	Horizontal lines spaced 50% closer than vcHorizontalPattern and of twice the line width 
vcDarkUpwardDiagonalPattern 2015	Diagonal lines slanting from bottom left to top right, spaced 50% closer than vcBDiagonalPattern and of twice the line width 
vcDarkVerticalPattern 2022	Vertical lines spaced 50% closer than vcVerticalPattern and of of twice the line width 
vcDashedHorizontalPattern 2026	Dashed horizontal lines 
vcDashedVerticalPattern 2027	Dashed vertical lines 
vcDiagCrossPattern 7	Diagonal cross-hatch pattern, small 
vcDiagonalBrickPattern 2032	Diagonal brick pattern 
vcDivotPattern 2036	Divot pattern 
vcDottedDiamondPattern 2038	Diagonal cross-hatch pattern of dotted lines 
vcDottedGridPattern 2037	Cross-hatch pattern of dotted lines 
vcFDiagonalPattern 4	Diagonal lines slanting from top left to bottom right 

vcHorizontalBrickPattern 2033	Horizontal brick pattern 
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 
vcSmallCheckerBoardPattern 2043	Checkerboard pattern 
vcSmallConfettiPattern 2028	Confetti pattern 
vcSmallGridPattern 2042	Cross-hatch pattern spaced 50% closer than vcCrossPattern 
vcSolidDiamondPattern 2046	Checkerboard pattern showing diagonal squares 
vcSpherePattern 2041	Checkerboard of spheres 
vcTrellisPattern 2040	Trellis pattern 

vcVerticalBottomLightedConvexPattern 43	Vertical color gradient from dark to bright 
vcVerticalConcavePattern 40	Vertical color gradient from dark to bright to dark 
vcVerticalConvexPattern 41	Vertical color gradient from bright to dark to bright 
vcVerticalGradientPattern 62	Vertical color gradient 
vcVerticalPattern 2	Vertical lines 
vcVerticalTopLightedConvexPattern 42	Vertical color gradient from bright to dark 
vcWavePattern 2031	Horizontal wave pattern 
vcWeavePattern 2034	Interwoven stripe pattern 
vcWideDownwardDiagonalPattern 2016	Diagonal lines slanting from top left to bottom right, showing the same spacing but three times the line width of vcF-DiagonalPattern 
vcWideUpwardDiagonalPattern 2017	Diagonal lines slanting from bottom left to top right right, showing the same spacing but three times the line width of vcBDiagonalPattern 
vcZigZagPattern 2030	Horizontal zig-zag lines 

PatternColorAsARGB

Property of 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. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

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 color in dependence of data.

	Data Type	Explanation
Property value	Color	ARGB color values ({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	Integer	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	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	Integer	Data field index

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	String	Name of the pattern map

Piles

Property of VcNodeAppearance

This property lets you assign/enquire the number of node piles in the chart. 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	Long	number of nodes piled or **-1**

Example Code

```
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	AppearanceShadowEnum	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

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Shadow = vcShOn
```

ShadowColorAsARGB

Property of VcNodeAppearance

This property lets you set or retrieve the color of the shadow 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. When casting an RGB value on an ARGB value, an alpha value of 255 has to be added.

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 VcNode-Appearance object).

	Data Type	Explanation
Property value	Color	ARGB color values ({0...255},0...255},{0...255},{0...255}) **Default value:** &hFFD8D8D8 (gray)

Example Code

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance

nodeAppearance.ShadowColor = MakeARGB(100, 100, 100, 100)
```

Specification

Read Only Property of VcNodeAppearance

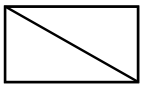
This property lets you retrieve the specification of a node appearance. A specification is a string that contains legible ASCII characters from 32 to 127 only, so it can be stored without problems to text files or data bases. This allows for persistency. A specification can be used to create a node appearance by the method **VcNodeAppearanceCollection.AddBy-Specification**.

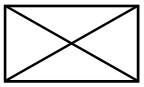
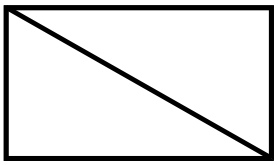
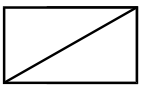
	Data Type	Explanation
Property value	String	Specification of the node appearance

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	AppearanceStrikeThroughEnum **Possible Values:** vcBackslashed 3	strike through pattern or **-1** Backslashed strike through 

vcCrossed 4	Crossed strike through 
vcDoubleBackslashed 8	Double backslashed strike through 
vcDoubleSlashed 7	Double slashed strike through
vcNoStrikeThrough 0	No strike through pattern
vcSlashed 2	Slashed strike through 
vcStrikeThroughNotSet -1	Strike through pattern not set

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Strikethrough = vcBackslashed
```

StrikeThroughColor

Property of VcNodeAppearance

This property lets you assign/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	Color	RGB color values or **-1** ({0...255},{0...255},{0...255})

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.StrikeThroughColor = RGB(255, 0, 0)
```

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	AppearanceThreeDEffectEnum **Possible Values:** vc3DNotSet -1 vc3DOff 0 vc3DOn 1	3DEffect setting Flag of 3D appearance not set Flag of 3D appearance set off Flag of 3D appearance set on

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.ThreeDEffect = vc3DOn
```

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	Boolean	node appearance visible in legend (True)/ invisible in legend (False) **Default value:** True

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.NodeAppearanceByName("Standard")

nodeAppearance.VisibleInLegend = False
```

Methods

PutInOrderAfter

Method of VcNodeAppearance

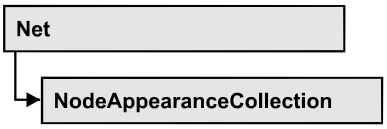
This method lets you set the node appearance behind a node appearance specified by name, within the NodeAppearanceCollection. If you set the name to "", the node appearence will be put in the first position. The order of the node appearances within the collection determines the order by which they apply to the nodes.

	Data Type	Explanation
Parameter: refNodeAppearanceName	String	Name of the node appearance behind which the current node appearance is to be put.
Return value	Void	

Example Code

```
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()
```

7.45 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.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstNodeAppearance
- NextNodeAppearance
- NodeAppearanceByIndex
- NodeAppearanceByName
- Remove

Properties

_NewEnum

Read Only Property of VcNodeAppearanceCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all node appearance objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim nodeApp As VcNodeAppearance

For Each nodeApp In VcNet1.NodeAppearanceCollection
   Debug.Print nodeApp.Name
Next
```

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	Long	Number of NodeAppearance objects

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim numberNodeAppColl As Integer

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection

numberNodeAppColl = nodeAppearanceCollectiont.Count
```

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	String	Name of the node appearance
Return value	VcNodeAppearance	New node appearance object

Example Code

```
Set 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	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" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ fromName	String	Name of the node appearance to be copied
⇨ newName	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

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance
```

NextNodeAppearance

Method of VcNodeAppearanceCollection

This method can be used in a forward iteration loop to retrieve subsequent node appearance objects 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	Subsequent node appearance object

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

While Not nodeAppearance Is Nothing
     Listbox.AddItem nodeAppearance.Name
     Set nodeAppearance = nodeAppearanceCollection.NextNodeAppearance
Wend
```

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	Integer	Index of the node appearance
Return value	VcNodeAppearance	Node appearance object returned

Example Code

```
Dim NodeAppearanceCltn As VcNodeAppearanceCollection

Set nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCltn.NodeAppearanceByIndex(2)
nodeApparance.LineThickness = 2
```

NodeAppearanceByName

Method of VcNodeAppearanceCollection

This method lets you retrieve a nodeAppearance object by its name. If a node appearance of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ nodeAppearanceName	String	Name of the node appearance object
Return value	VcNodeAppearance	Node appearance object returned

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.NodeAppearanceByName("Standard")
```

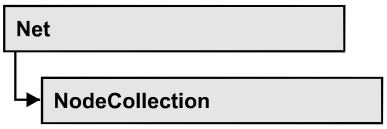
Remove

Method of VcNodeAppearanceCollection

This method lets you delete a node appearance. If the node appearance is used by a different object, it cannot be deleted. In the latter case **False** will be returned, otherwise **True**.

	Data Type	Explanation
Parameter:		
⇨ name	String	Name of the node appearance
Return value	Boolean	Node appearance deleted (True)/not deleted (False)

7.46 VcNodeCollection



An object of the type VcNodeCollection contains all nodes available in the diagram. You can select a part of them by using the method **SelectNodes**. You can access all objects in an iterative loop by **For Each node In Node-Collection** or by the methods **First...** and **Next...**. The number of nodes in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- FirstNode
- NextNode
- SelectNodes

Properties

_NewEnum

Read Only Property of VcNodeCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all node objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim node As VcNode

For Each node In VcNet1.NodeCollection
```

```
   Debug.Print node.Name
Next
```

Count

Read Only Property of VcNodeCollection

This property lets you retrieve the number of nodes in the NodeCollection object.

	Data Type	Explanation
Property value	Long	Number of Nodes in the node collection

Example Code

```
Dim nodeCltn As VcNodeCollection

Set nodeCltn = VcNet1.NodeCollection
MsgBox "Number of nodes: " & nodeCltn.Count
```

Methods

FirstNode

Method of VcNodeCollection

This method can be used to access the initial value, i.e. the first node of a NodeCollection, and then to continue in a forward iteration loop by the method **NextNode** for the nodes following. If there is no node in the Node-Collection, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNode	First node

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcNet1.NodeCollection
Set node = nodeCltn.FirstNode
```

NextNode

Method of VcNodeCollection

This method can be used in a forward iteration loop to retrieve subsequent nodes from a node collection after initializing the loop by the method **FirstNode**. If there is no node left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNode	Subsequent node

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

Set nodeCltn = VcNet1.NodeCollection
Set node = nodeCltn.FirstNode

While Not node Is Nothing
     node.MarkNode = False
     Set node = nodeCltn.NextNode
Wend
```

SelectNodes

Method of VcNodeCollection

This method lets you specify the nodes to be collected by the NodeCollection object.

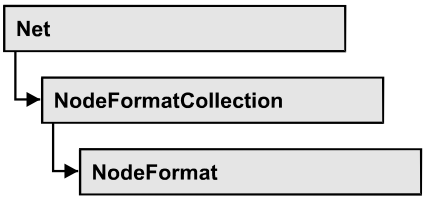
	Data Type	Explanation
Parameter:		
⇨ selType	SelectionTypeEnum	Nodes to be selected
	Possible Values:	
	vcAll 0	All objects in the diagram will be selected
	vcAllLinksCausingCycles 7	If this selection type is chosen, the link collection will contain all links that cause the existence of cycles. If these links are deleted, cycles will cede to exist in this chart.
	vcAllLinksInCycles 6	If this selection type is chosen, the link collection will contain all links that participate in forming cycles. Cycles are chains of nodes and links of which the beginning and end join.
	vcAllVisible 1	All visible objects will be selected
	vcMarked 2	All marked objects will be selected
Return value	Long	Number of nodes selected

Example Code

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode
```

```
Set nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes vcSelected
```

7.47 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 can get to reach by the **Nodes** property page.

Properties

- _NewEnum
- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification
- WidthOfExteriorSurrounding

Methods

- CopyFormatField
- RemoveFormatField

Properties

_NewEnum

Read Only Property of VcNodeFormat

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all node format field objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim formatField As VcNodeFormatField

For Each formatField In format
    Debug.Print formatField.Index
Next
```

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	Boolean	Fields inside the node separated by lines (True)/ not separated by lines (False)

Example Code

```
Dim format As VcNodeFormat

Set format = VcNet1.NodeFormatCollection.FormatByName("format1")
format.FieldsSeparatedByLines = True
```

FormatField

Read Only Property of VcNodeFormat

This property gives access to a VcNodeFormatField object by the index. The index has to be in the range from 0 to FormatFieldCount-1.

Note for users of a version earlier than 3.0: The index does **not** count from 1 to FormatFieldCount, as it does in more recent versions.

	Data Type	Explanation
Parameter: index	Integer	Index of the node format field 0 ... .FormatFieldCount-1
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	Integer	Number of fields of the node format

Example Code

```
Dim formatCollection As VcNodeFormatCollection
Dim format As VcNodeFormat
Dim nameofFormat As String

Set formatCollection = VcNet1.NodeFormatCollection
Set format = formatCollection.FormatByName("Standard")

numberofFormatField = format.FormatFieldCount
```

Name

Property of VcNodeFormat

This property lets you set or retrieve the name of the node format.

	Data Type	Explanation
Property value	String	Name of the node format

Example Code

```
Dim format As VcNodeFormat
Dim formatName As String

Set format = VcNet1.NodeFormatCollection.FirstFormat
formatName = format.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	String	Specification of the node format

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	Integer	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	FormatFieldPositionEnum	Position of the new node 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	Integer	Index of the reference node format field
Return value	VcNodeFormatField	Node format field object

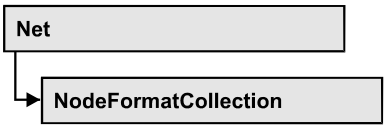
RemoveFormatField

Method of VcNodeFormat

This method lets you remove a layer format field by its index. After that, the program will update all layer format field indexes so that they are consecutively numbered again.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	Index of the node format field to be deleted

7.48 VcNodeFormatCollection



An object of the type VcNodeFormatCollection contains all available node formats. You can access all objects in an iterative loop by **For Each node In NodeCollection** or by the methods **First...** and **Next...**. You can access a single node formats by using the methods **FormatByName**. The number of node formats in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByIndex
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcNodeFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. This object allows to iterate over all node format objects. In Visual Basic this property is never indicated, but it can be used by the command **For Each** *element* **In** *collection*. In .NET languages the method **GetEnumerator** is offered instead. Some development environments replace this property by own language elements.

	Data Type	Explanation
Property value	Object	Reference object

Example Code

```
Dim format As VcNodeFormat

For Each format In VcNet1.NodeFormatCollection
   Debug.Print format.Name
Next
```

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	Long	Number of node formats

Example Code

```
Dim formatCltn As VcNodeFormatCollection
Dim numberOfFormats As Long

Set formatCltn = VcNet1.NodeFormatCollection
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 was not used before, the new VcNodeFormat object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

A node format by default has the below properties:

- It is a single field
- WidthOfExteriorSurrounding: 3 mm

A field has these 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	String	Name of the node format
Return value	VcNodeFormat	Node format object

Example Code

```
Set 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	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	String	Name of the node format to be copied
⇨ newName	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

```
Dim format As VcNodeFormat

Set format = VcNet1.NodeFormatCollection.FirstFormat
```

FormatByIndex

Method of VcNodeFormatCollection

This method lets you access a node format by its index. If a node format of the specified index does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	Integer	Index of the node format

Example Code

```
Dim nodeFormatCltn As VcNodeFormatCollection

Set nodeFormatCltn = VcNet1.NodeFormatCollection
Set nodeFormat = nodeFormatCltn.NodeFormatByIndex(2)
nodeFormat.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	String	Name of the node format
Return value	VcNodeFormat	Node format

Example Code

```
Dim formatCollection As VcNodeFormatCollection
Dim format As VcNodeFormat

Set formatCollection = VcNet1.NodeFormatCollection
Set format = formatCollection.FormatByName("Standard")
```

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

```
Dim formatCollection As VcNodeFormatCollection
Dim format As VcNodeFormat

Set formatCollection = VcNet1.NodeFormatCollection
```

```
Set format = formatCollection.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCollection.NextFormat
Wend
```

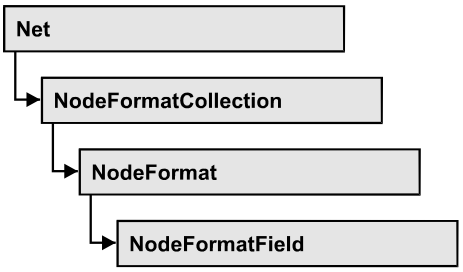
Remove

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	String	Node format name
Return value	Boolean	Node format deleted (True)/not deleted (False)

7.49 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
- BottomMargin
- CombiField
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName
- RightMargin
- 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	FormatFieldAlignmentEnum **Possible Values:** vcFFABottom 28 vcFFABottomLeft 27 vcFFABottomRight 29 vcFFACenter 25 vcFFALeft 24 vcFFARight 26 vcFFATop 22 vcFFATopLeft 21 vcFFATopRight 23	Alignment of the field content bottom bottom left bottom right center left right top top left top right

BottomMargin

Property of VcNodeFormatField

This property lets you set or retrieve the width of the bottom margin of the node format field.

	Data Type	Explanation
Property value	Integer	Width of the bottom margin of the node format field 0 ... 9

CombiField

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	Boolean	Combi field (True)/ no combi field (False)

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	String	Constant text

FormatName

Read Only Property of VcNodeFormatField

This property lets you retrieve the name of the node format to which this node format field belongs.

	Data Type	Explanation
Property value	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 be valid.

	Data Type	Explanation
Property value	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 read from the specified data field.

	Data Type	Explanation
Property value	Integer	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	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	Integer	Height of the graphics in mm 0 ... 99

Index

Read Only Property of VcNodeFormatField

This property lets you enquire the index of the node format field in the corresponding node format.

	Data Type	Explanation
Property value	Integer	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	Integer	Width of the left margin of the node format field 0 ... 9

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	Integer	Maximum number of lines 0 ... 9

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. When assigning a value by this property, please also remember to set the **MaximumTextLineCount** value anew, since otherwise the minimum value might overwrite the maximum value.

	Data Type	Explanation
Property value	Integer	Minimum number of lines 0 ... 9

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	Integer	Minimum width of the node format field in mm 0 ... 99

PatternBackgroundColorAsARGB

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 node format field shall have the background color of the node format, select the value **-1**.

If by the property **PatternBackgroundColorMapName** a map was specified, it will set the background color of the node format field in dependence on data.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	OLE_COLOR	Background color of the node format
Property value	Long	ARGB color values ({0...255},{0...255},{0...255},{0...255})

PatternBackgroundColorDataFieldIndex

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
Parameter: ⇨ Rückgabewert	Integer	Data field index
Property value	Long	Data field index

PatternBackgroundColorMapName

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 **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 **PatternBackgroundColor** will be used.

	Data Type	Explanation
Parameter: ⇨ Rückgabewert	String	Name of the color map
Property value	String	Name of the color map

PatternColorAsARGB

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	Integer	ARGB color values ({0...255},{0...255},{0...255},{0...255})

PatternColorDataFieldIndex

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	Integer	Data field index

PatternColorMapName

Property of VcNodeFormatField

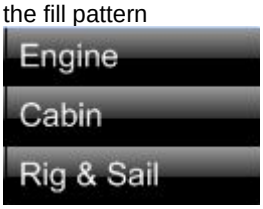
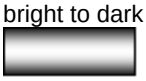
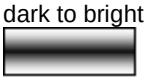
This property lets you set or retrieve the name of a color map (type vcColorMap). If set to "", no map will be used. Only if a map name and a data field index are specified in the property **PatternColorDataFieldIndex**, the pattern color is controlled by the map. If no data field entry applies, the pattern color of the calendar grid that is specified in the property **PatternColor** will be used.

	Data Type	Explanation
Property value	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	FieldFillPatternEnum	Pattern type **Default value:** As defined in the dialog
	Possible Values:	
	vcFieldNoPattern 1276	No fill pattern
	vcAeroGlassPattern 44	Vertical color gradient in the color of the 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

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	Long	Data field index

PatternExMapName

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	String	Name of the pattern map
Property value	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	Integer	width of the right margin of the node format field 0 ... 9

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 table format field. This property only works if the type of the data field is **vcFFTText**. If the value of the index equals **-1**, the content of the property **ConstantText** will be returned instead.

	Data Type	Explanation
Property value	Integer	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 in dependence of the data.

	Data Type	Explanation
Property value	StdFont	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 **TextFontMap-Name**, the map will control the text font color in dependence of the data.

	Data Type	Explanation
Property value	OLE_COLOR	font color of the node format **Default value:** -1

TextFontDataFieldIndex

Property of VcNodeFormatField

This property lets you set or retrieve the data field index required by the property **TextFontMapName** for a font map. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	data field index

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	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	Integer	width of the top margin of the node format field 0 ... 9

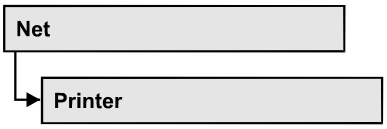
Type

Property of VcNodeFormatField

This property lets you enquire the type of the node format field.

	Data Type	Explanation
Property value	FormatFieldTypeEnum **Possible Values:** vcFFTGraphics 64 vcFFTText 36	type of the node format field graphics text

7.50 VcPrinter



The VcPrinter object offers a variety of properties to set up the printing process. You can enter the width of top, bottom, left and right margins, set a page frame, page numbers, a page description, cutting marks and the print date. Beside, you can specify the number of pages that the diagram is to be printed on. Zoom factor, alignment, orientation, paper size and color mode are more properties that you can vary for a perfect print.

Properties

- AbsoluteBottomMarginInCM
- AbsoluteBottomMarginInInches
- AbsoluteLeftMarginInCM
- AbsoluteLeftMarginInInches
- AbsoluteRightMarginInCM
- AbsoluteRightMarginInInches
- AbsoluteTopMarginInCM
- AbsoluteTopMarginInInches
- Alignment
- CurrentHorizontalPagesCount
- CurrentVerticalPagesCount
- CurrentZoomFactor
- CuttingMarks
- DefaultPrinterName
- DocumentName
- FitToPage
- FoldingMarksType
- MarginsShownInInches
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation
- PageDescription
- PageDescriptionString
- PageFrame
- PageNumberMode
- PageNumbers

- PagePaddingEnabled
- PaperSize
- PrintDate
- PrinterName
- RepeatTitleAndLegend
- StartUpSinglePage
- ZoomFactorAsDouble

Properties

AbsoluteBottomMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute height of the bottom margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Height of the bottom margin of the page in cm **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteBottomMarginInCM = 1.5
```

AbsoluteBottomMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute height of the bottom margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Height of the bottom margin of the page in inches **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteBottomMarginInches = 0.5
```

AbsoluteLeftMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute width of the left margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Width of the left margin of the page in cm **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteLeftMarginInCM = 1.5
```

AbsoluteLeftMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute width of the left margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Width of the left margin of the page in inches **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteLeftMarginInInches = 0.5
```

AbsoluteRightMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute width of the right margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Width of the right margin of the page in cm **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteRightMarginInCM = 1.5
```

AbsoluteRightMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute width of the right margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Width of the right margin of the page in inches **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteRightMarginInInches = 0.5
```

AbsoluteTopMarginInCM

Property of VcPrinter

This property lets you set or retrieve the absolute height of the top margin of the pages to be printed. The true width may be larger if the printer used has to print margins by obligation.

	Data Type	Explanation
Property value	Double	Height of the top margin of the page in cm **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5
```

AbsoluteTopMarginInInches

Property of VcPrinter

This property lets you set or retrieve the absolute height of the top margin of the pages to be printed in inches. The true width may be larger if the printer used has to print margins by obligation.

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Double	Height of the top margin of the page in inches **Default value:** 0

Example Code

```
VcNet1.Printer.AbsoluteTopMarginInInches = 0.5
```

Alignment

Property of VcPrinter

This property lets you set or retrieve the alignment of the diagram on a page. The property will be effective either if the diagram is put out onto a single page or if the **RepeatTitleAndLegend** property was set. In any other case the output will be centered.

	Data Type	Explanation
Property value	PrinterAlignmentEnum	Alignment of the output with its sheet **Default value:** vcPCenterCenter
	Possible Values:	
	vcPBottomCenter 28	Vertical alignment: bottom; horizontal alignment: center
	vcPBottomLeft 27	Vertical alignment: bottom; horizontal alignment: left
	vcPBottomRight 29	Vertical alignment: bottom; horizontal alignment: right
	vcPCenterCenter 25	Vertical alignment: center; horizontal alignment: center
	vcPCenterLeft 24	Vertical alignment: center; horizontal alignment: left
	vcPCenterRight 26	Vertical alignment: center; horizontal alignment: right
	vcPTopCenter 22	Vertical alignment: top; horizontal alignment: center
	vcPTopLeft 21	Vertical alignment: top; horizontal alignment: left
	vcPTopRight 23	Vertical alignment: top; horizontal alignment: right

Example Code

```
VcNet1.Printer.Alignment = 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 **CurrentVertical-PagesCount** and **MaxHorizontalPagesCount**.

	Data Type	Explanation
Property value	Long	Current number of pages counted in horizontal direction

CurrentVerticalPagesCount

Read Only Property of VcPrinter

This property lets you retrieve the actual number of pages in vertical direction onto which the chart is to be printed. Also see **CurrentHorizontal-PagesCount** and **MaxVerticalPagesCount**.

	Data Type	Explanation
Property value	Long	Current number of pages counted in vertical direction

CurrentZoomFactor

Read Only Property of VcPrinter

This property lets you retrieve the actual zoom factor for the setting **FitToPage = False**(zoom factor = 100: original size, zoom factor > 100: enlargement, zoom factor < 100: reduction).

	Data Type	Explanation
Property value	Double	Actual zoom factor

CuttingMarks

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) cutting marks are to printed onto a page.

	Data Type	Explanation
Property value	Boolean	Cutting marks are (True) / are not (False) printed **Default value:** False

Example Code

```
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	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	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	Boolean	Diagram is printed on a defined set of pages/is printed in a defined enlargement.

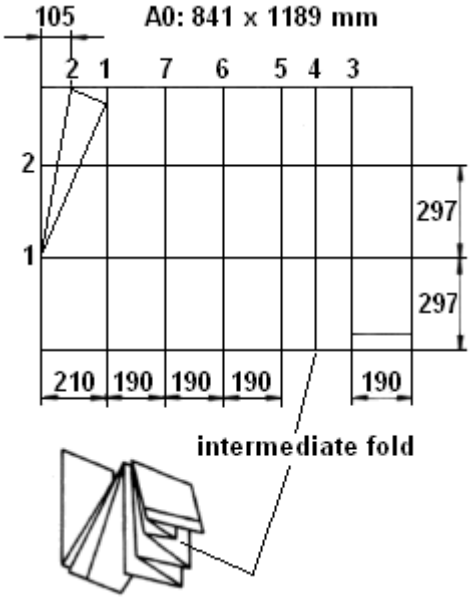
Example Code

```
VcNet1.Printer.FitToPage = True
```

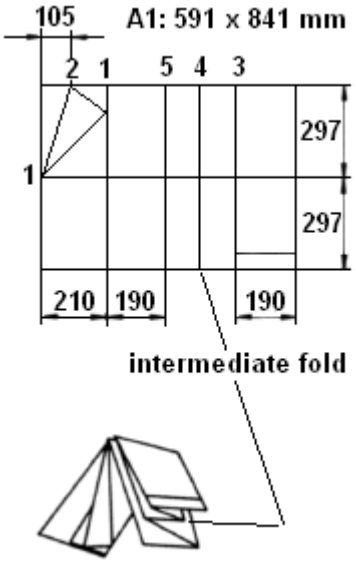
FoldingMarksType

Read Only Property of VcPrinter

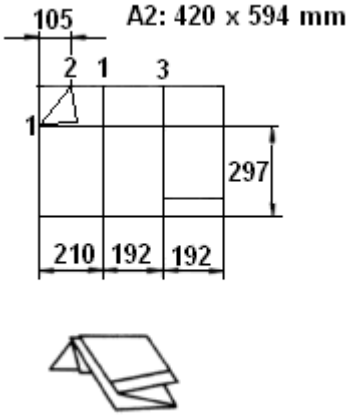
This property lets you set or retrieve the following folding marks according to DIN 824. The folding marks allow to fold paper sheets of the German DIN-A standard:



Folding of the DIN-A-0 format

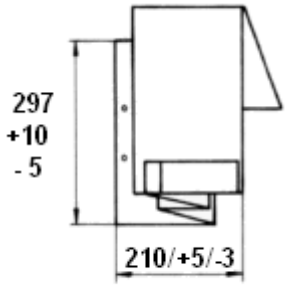
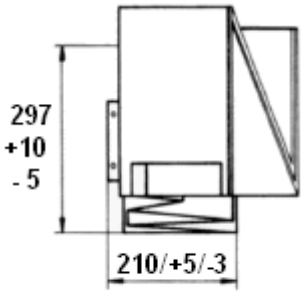
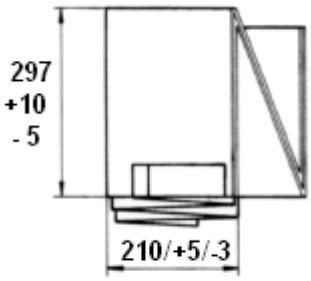


Folding of the DIN-A-1 format



Folding of the DIN-A-2 format

	Data Type	Explanation
Property value	FoldingMarksTypeEnum **Possible Values:**	Folding marks **Default value:** vcFMTNone

vcFMTDIN824FormA 65	Folding marks according to DIN824-A: the drawing can be punched and filed directly to a folder.  DIN 824-A way of folding
vcFMTDIN824FormB 66	Folding marks according to DIN824-B: the chart can be punched and filed to a folder by a flexi filing fastener.  DIN 824-B way of folding
vcFMTDIN824FormC 67	Folding marks according to DIN824-C: the folded chart is not to be punched but to be put in a sheet protector.  DIN 824-C way of folding
vcFMTNone 0	No folding marks

MarginsShownInInches

Property of VcPrinter

This property lets you set or retrieve whether the measuring unit of the margins in the <b"Page Layout dialog shall be switched to inches (at present only possible at runtime).

Tip: The internal conversion factor is 2.5 cm/inch instead of the actual correct 2.54 cm/inch so that the values shown in the **Page Setup** dialog will be smoother (1.5 cm so add up to 0.6 inches, 1 cm add up to 0.4 inches).

	Data Type	Explanation
Property value	Boolean	Measuring unit of the margins in the **Page Layout** dialog in inches (True)/ in cm (False) **Default value:** False

MaxHorizontalPagesCount

Property of VcPrinter

This property lets you set or retrieve the horizontal number of pages für printing and for the print preview. This property only works if the property **ScalingMode** was set to either **vcFitToPageCount** or to **vcZoomWithHorizontalFit**. Also see **MaxVerticalPagesCount** and **CurrentHorizontalPagesCount**.

	Data Type	Explanation
Property value	Long	Maximum number of pages counted in horizontal direction **Default value:** 1

Example Code

```
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** and **CurrentVerticalPagesCount**.

	Data Type	Explanation
Property value	Long	Maximum number of pages counted in vertical direction **Default value:** 1

Example Code

```
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	OrientationEnum	Orientation **Default value:** VcPortrait
	Possible Values: vcLandscape 42 vcPortrait 41	 Printing orientation **landscape** Printing orientation **portrait**

Example Code

```
VcNet1.Printer.Orientation = vcLandScape
```

PageDescription

Property of VcPrinter

This property lets you set or retrieve whether (True) or not (False) the page description string is to appear in the bottom left corner of a page. The contents of the page description string you can set by the **PageDescriptionString** property.

	Data Type	Explanation
Property value	Boolean	Page description is (True) / is not (False) printed **Default value:** False

Example Code

```
VcNet1.Printer.PageDescription = True
```

PageDescriptionString

Property of VcPrinter

This property lets you set or retrieve a page description string in the bottom left corner of each page. Whether or not the page description string is printed you can control by the **PageDescription** property. For numbering the pages you may enter the following place holders which will be replaced with the appropriate contents on the printout:

{PAGE} = consecutive numbering of pages

{NUMPAGES} = total number of pages

{ROW} = line position of the section in the complete chart

{COLUMN} = column position of the section in the complete chart

	Data Type	Explanation
Property value	String	Page description **Default value:** Empty string ""

Example Code

```
VcNet1.Printer.PageDescriptionString = "VARCHART chart"
```

PageFrame

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) a frame is to be drawn around the output. If the **RepeatTableTimeScale** property was set, the frame will be drawn around the part on each page, otherwise it will be drawn around the diagram as a whole.

	Data Type	Explanation
Property value	Boolean	Frame is (True) / is not (False) displayed **Default value:** True

Example Code

```
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	pageNumberModeEnum **Possible Values:** vcPageNOfM 1597 vcPRowColumn 1596	mode of page numbering **Default value:** vcPRowColumn "Page N of M pages" "x.y" (row no./column no.).

Example Code

```
Dim printer As VcPrinter

Set printer = VcNet1.printer
```

```
With printer
    .Orientation = vcLandscape
    .PageNumberMode = vcPageNOfM
    .PageNumbers = True
    .FitToPage = False
End With

VcNet1.PrintPreview
```

PageNumbers

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) a page number is printed. The mode of page numbering is set with the help of the property **PageNumberMode**.

	Data Type	Explanation
Property value	Boolean	Page numbers are (True) / are not (False) printed **Default value:** False

Example Code

```
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	Boolean	Space between diagram and boxes for legend/title is (True) / is not (False) left **Default value:** True

Example Code

```
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	PaperSizeEnum **Possible Values:** vcDIN_A2 66 vcDIN_A3 8 vcDIN_A4 9 vcISO_C 24 vcISO_D 25 vcISO_E 26 vcUS_LEGAL 5 vcUS_LETTER 1	Paper size DIN A2 DIN A3 DIN A4 ISO C ISO D ISO E US LEGAL US LETTER

Example Code

```
VcNet1.Printer.PaperSize = vcDIN_A3
```

PrintDate

Property of VcPrinter

This property lets you set or retrieve, whether (True) or not (False) the print date is to appear in the bottom left corner of a page.

	Data Type	Explanation
Property value	Boolean	Print date is/is not set

Example Code

```
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	String	Printer name

RepeatTitleAndLegend

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	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

```
VcNet1.Printer.RepeatTitleAndLegend = True
```

StartUpSinglePage

Property of VcPrinter

This property lets you set or retrieve the mode of starting the page preview: either all pages of the diagram will be displayed (False) or only the first page will be displayed (True).

	Data Type	Explanation
Property value	Boolean	at the start of the page preview: only first page of the diagram (True)/ all pages of the diagram (False)

Example Code

```
Dim printer As VcPrinter

Set printer = VcNet1.printer

With printer
    .Orientation = vcLandscape
    .StartUpSinglePage = True
    .FitToPage = False
End With

VcNet1.PrintPreview
```

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	Double	Zoom factor of the diagram **Default value:** 100

Example Code

```
VcNet1.Printer.ZoomFactorAsDouble = 150
```

7.51 VcRect

Rect

An object of the type **VcRect** designates a rectangle object and is only passed by the event VcNet.OnShowInPlaceEditor.

Properties

- Bottom
- Height
- Left
- Right
- Top
- Width

Properties

Bottom

Property of VcRect

This property returns/sets the bottom coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	Position of the bottom border of the rectangle

Height

Read Only Property of VcRect

This property returns the height of the VcRect object.

	Data Type	Explanation
Property value	Long	Height of the rectangle

Left

Property of VcRect

This property returns/sets the left coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	Position of the left border of the rectangle

Example Code

```
Private Sub VcNet1_OnShowInPlaceEditor(ByVal editObject As Object, _
                           ByVal editObjectType As _
                           VcNetLib.VcObjectTypeEnum, _
                           ByVal fieldIndex As Long, ByVal objRectComplete As _
                           VcNetLib.VcRect, ByVal objRectVisible As _
                           VcNetLib.VcRect, ByVal fldRectComplete As _
                           VcNetLib.VcRect, ByVal fldRectVisible As _
                           VcNetLib.VcRect, returnStatus As Variant)

   Dim oldScaleMode As Long

   If editObjectType = vcObjTypeNodeInTable Then
      returnStatus = vcRetStatFalse

      Set myEditObject = editObject
      myEditObjectType = editObjectType
      myEditObjectFieldIndex = fieldIndex

      oldScaleMode = Me.ScaleMode
      Me.ScaleMode = vbPixels

      Select Case fieldIndex
         Case 1   'Name
            Text1.Left = fldRectVisible.Left + VcNet1.Left
            Text1.Top = fldRectVisible.Top + VcNet1.Top
            Text1.Width = fldRectVisible.Width
            Text1.Height = fldRectVisible.Height

            Text1.Text = editObject.DataField(fieldIndex)
            Text1.Visible = True
            Text1.SetFocus

         Case 2, 3     'Start or End
            MonthView1.Left = fldRectVisible.Left + VcNet1.Left
            MonthView1.Top = fldRectVisible.Top + VcNet1.Top

            MonthView1.Value = editObject.DataField(fieldIndex)
            MonthView1.Visible = True
            MonthView1.SetFocus

         Case 13   'Employee
            Combo1.Left = fldRectVisible.Left + VcNet1.Left
            Combo1.Top = fldRectVisible.Top + VcNet1.Top
            Combo1.Width = fldRectVisible.Width

            Combo1.Text = editObject.DataField(fieldIndex)
            Combo1.Visible = True
            Combo1.SetFocus

      End Select

      Me.ScaleMode = oldScaleMode
```

```
   End If
End Sub
```

Right

Property of VcRect

This property returns/sets the right coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	position of the right border of the rectangle

Top

Property of VcRect

This property returns/sets the top coordinate of the VcRect object.

	Data Type	Explanation
Property value	Long	position of the top border of the rectangle

Example Code

```
MonthView1.Top = fldRectVisible.Top + VcNet1.Top
```

Width

Read Only Property of VcRect

This property returns the width of the VcRect object.

	Data Type	Explanation
Property value	Long	width of the rectangle

Example Code

```
Text1.Width = fldRectVisible.Width
```

7.52 VcScheduler

An object of the type **VcScheduler** represents a module for calculating simple project data, such as the early end of a project or its early start (if calculations are performed backward), or its free float and total float.

Properties

- ActualEndDateDataFieldIndex
- ActualStartDateDataFieldIndex
- AutomaticSchedulingEnabled
- DurationDataFieldIndex
- EarlyEndDateDataFieldIndex
- EarlyStartDateDataFieldIndex
- EndDateForAutomaticScheduling
- EndDateNotLaterThanDataFieldIndex
- FreeFloatDataFieldIndex
- LateEndDateDataFieldIndex
- LateStartDateDataFieldIndex
- LinkDurationDataFieldIndex
- ScheduledProjectEndDate
- ScheduledProjectStartDate
- ScheduleSuccessorsOnlyEnabled
- StartDateForAutomaticScheduling
- StartDateNotEarlierThanDataFieldIndex
- TotalFloatDataFieldIndex

Methods

- ScheduleProject

Properties

ActualEndDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the present end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the valid end date

ActualStartDateDataFieldIndex

Property of VcScheduler

This property lets you set/retrieve the index of the data field which contains the start date set to the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the valid start date

AutomaticSchedulingEnabled

Property of VcScheduler

This property lets you set or retrieve whether automatic time scheduling is switched on or off.

	Data Type	Explanation
Property value	Boolean	Automatic time scheduling is switched on (True) or off (False) **Default value:** False

DurationDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the duration of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the duration of the activity

EarlyEndDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated earliest possible end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the earliest possible end date of an acitivity

EarlyStartDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated earliest possible start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the earliest possible start date of an activity

EndDateForAutomaticScheduling

Property of VcScheduler

In case **Automatic scheduling** is activated, this property lets you set or retrieve the end date of the project.

	Data Type	Explanation
Property value	Date	Desired end date for automatic scheduling

EndDateNotLaterThanDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the desired latest end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the desired latest end date

FreeFloatDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated free float of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the free float

LateEndDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated latest possible end date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the latest possible end date

LateStartDateDataFieldIndex

Property of VcScheduler

With this property you can set/retrieve the index of the data field which contains the calculated latest possible start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the latest possible start date

LinkDurationDataFieldIndex

Property of VcScheduler

This property lets you set or retrieve the index of a data field in the project in which a minimum temporal distance between predecessor and successor can be stored. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the minimum time space between a predecessor and a successor node

ScheduledProjectEndDate

Read Only Property of VcScheduler

This property returns the **early end** of a project after having calculated the project dates by **VcScheduler.ScheduleProject** if the start date was set before.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Date	Index of the data field which holds the scheduled end date of the project

ScheduledProjectStartDate

Read Only Property of VcScheduler

This property returns the **late start** of a project after the project dates were calculated by **VcScheduler.ScheduleProject** if an end date was set before.

This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Date	Index of the data field which holds the scheduled start date of the project

ScheduleSuccessorsOnlyEnabled

Property of VcScheduler

With this property you can set/retrieve whether the scheduling of only those nodes that have a predecessor node is switched on or off; otherwise all nodes will be scheduled. A "project start" will thus be ignored.

	Data Type	Explanation
Property value	Boolean	Scheduling of nodes only with predecessors is switched on/off

StartDateForAutomaticScheduling

Property of VcScheduler

In case **Automatic scheduling** is activated, this property lets you set or retrieve the start date of the project.

	Data Type	Explanation
Property value	Date	Desired start date for automatic scheduling

StartDateNotEarlierThanDataFieldIndex

Property of VcScheduler

This property lets you set or retrieve the index of the data field which contains the desired earliest start date of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the desired early start date

TotalFloatDataFieldIndex

Property of VcScheduler

This property lets you set or retrieve the index of the data field which contains the calculated total float of the activity. This is only possible as long as no data has been loaded.

	Data Type	Explanation
Property value	Long	Index of the data field which holds the total float

Methods

ScheduleProject

Method of VcScheduler

This method lets you calculate the dates of a project (early / late start, early / late end, free float, total float) of a project. The desired start and end date can be set by this method. By passing only the end date, the project start will be calculated, by passing only the start date, the project end will be calculated. You can pass both dates, which will add the corresponding float to the activities. (This only works with matching dates, which means that the end date for example should not be within the project time period.) At least one date must be passed, otherwise an error message will occur. If a cycle amongst the nodes and links is identified, the ones affected will be marked.

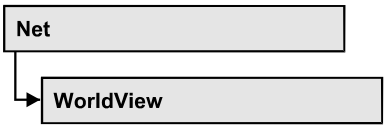
The results will be stored to fields that you can set by the properties**EarlyStartDateDataFieldIndex**, **LateStartDateDataFieldIndex**, **EarlyEndDateDataFieldIndex**, **LateEndDateDataFieldIndex**, **FreeFloatDataFieldIndex** and **TotalFloatDataFieldIndex**.

	Data Type	Explanation
Parameter:		
⇨ startDate	Date	Desired start date
⇨ endDate	Date	Desired end date
Return value	Boolean	The project data were successfully calculated (true) / were not calculated (False)

Example Code

```
VcScheduler.ScheduleProject (3.5.2012,1.10.2012)
```

7.53 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 should have a frame (not valid for **vcPopupWindow** mode). The color of the frame is **Color.Black**. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean	World view with a border line (True)/without border line (False) **Default value:** True

Example Code

```
VcNet1.WorldView.Mode = vcNotFixed
VcNet1.WorldView.Border = True
```

Height

Property of VcWorldView

This property lets you retrieve the vertical extent of the world view. In the modes **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Height of the world view {0, ...} **Default value:** 100

Example Code

```
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.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/in Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual height of the world view {0, ...} **Default value:** 100

Example Code

```
VcNet1.LegendView.Height = 300
```

Left

Property of VcWorldView

This property lets you retrieve the left position of the Additional Views. In the modes **vcNotFixed** and **vcPopupWindow** of the property **Mode** it can also be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Left position of the world view **Default value:** 0

Example Code

```
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.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual left position of the world view **Default value:** 0

Example Code

```
VcNet1.LegendView.LeftActualValue = 150
```

MarkingColor

Property of VcWorldView

This property lets you enquire/set the line color of the rectangle that indicates in the Additional Views the currently selected section. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Color	RGB color values **Default value:** RGB(0, 0, 255)

Example Code

```
VcNet1.WorldView.MarkingColor = RGB(255, 0, 0)
```

Mode

Property of VcWorldView

This property lets you enquire/set the Additional Views mode. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	WorldViewModeEnum	Mode of the world view **Default value:** vcPopupWindow
	Possible Values:	
	vcFixedAtBottom 4	The world view is displayed on the bottom of the control window. The reference system of the coordinates is the control. With this value set, the height can be specified, whereas the position and the width are fixed.
	vcFixedAtLeft 1	The world view is displayed on the left side of the control window. The reference system of the coordinates is the control. With this value set, the width can be specified, whereas the position and the height are fixed.
	vcFixedAtRight 2	The world view is displayed on the right side of the control window. The reference system of the coordinates is the control. With this value set, the width can be specified, whereas the position and the height are fixed.

vcFixedAtTop 3	The world view is displayed on the top of the control window. The reference system of the coordinates is the control. With this value set, the height can be specified, whereas the position and the width are fixed.
vcNotFixed 5	The world view is a child window of the current parent window of the control. It can be positioned at any position with any extension. The reference system of the coordinates is the parent window. The child window does not have a frame of its own and cannot be moved interactively by the user. The parent window can be modified by the property **VcWorldView.ParentHWnd**.
vcPopupWindow 6	The world view is a popup window with its own frame. The reference system of the coordinates is the screen. The user can modify its position and extension, open it by the default context menu and close it by the **Close** button in the frame.

Example Code

```
VcNet1.WorldView.Mode = vcFixedAtBottom
```

ParentHWnd

Property of VcWorldView

In the **vcNotFixed** mode, this property lets you set the HWnd handle of the parent window, for example, if the world view is to appear in a frame window implemented by your own. By default, the frame window is positioned on the HWnd handle of the parent window of the VARCHART ActiveX main window. This property can be used only at run time.

	Data Type	Explanation
Property value	OLE_HANDLE	Handle

Example Code

```
MsgBox (VcNet1.worldview.ParentHWnd)
```

ScrollBarMode

Property of VcWorldView

This property lets you set or retrieve the scroll bar mode of the world view. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	WorldViewScrollBarModeEnum **Possible Values:**	Scrollbarmode **Default value:** NoScrollBar

vcAutomaticScrollBar 3	Display of a horizontal or vertical scrollbar if required.
vcHorizontalScrollBar 1	Display of a horizontal scrollbar if required.
vcNoScrollBar 0	The complete chart is displayed without scrollbars.
vcVerticalScrollBar 2	Display of a vertical scrollbar if required.

Example Code

```
VcNet1.WorldView.ScrollBarMode = vcAutomaticScrollBar
```

Top

Property of VcWorldView

This property lets you retrieve the top position of the world view. n the modes **vcNotFixed** und **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Top position of the world view

Example Code

```
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.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual top position of the world view **Default value:** 0

Example Code

```
VcNet1.LegendView.TopActualValue = 40
```

UpdateBehaviorName

Property of VcWorldView

This property lets you set or retrieve the name of the UpdateBehavior.

	Data Type	Explanation
Property value	String	Name of the UpdateBehavior

Visible

Property of VcWorldView

This property lets you enquire/set whether the worldview is visible or not. This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Boolean	World view visible (True)/not visible (False) **Default value:** False

Example Code

```
VcNet1.WorldView.Visible = True
```

Width

Property of VcWorldView

This property lets you retrieve the horizontal extent of the world view. In the modes **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow** of the property **Mode** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **Additional Views** property page.

	Data Type	Explanation
Property value	Long	Horizontal extension of the world view {0, ...} **Default value:** 100

Example Code

```
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.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a conversion from/to Twips by the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

	Data Type	Explanation
Property value	Long	Actual horizontal extension of the world view {0, ...} **Default value:** 100

Example Code

```
VcNet1.LegendView.WidthActualValue = 600
```

8 Index

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z