VARCHART XTree ActiveX Edition

Version 3.1

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: 7. July 2005

Table of Contents

1 Introduction

1.1 General Information on VARCHART XTree

VARCHART XTree 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 XGanttLC, VARCHART XNet, VARCHART XTree).

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

VARCHART XTree lets you visualize, edit, export and print your data in the form of tree diagrams. VARCHART XTree is the perfect tool to display any kind of hierarchical structure, such as work breakdown structures or file systems or classifications in general. Not only general tree structures, but also specific styles such as the look of the Microsoft Explorer can be displayed (TreeView Style).

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 formats of the different VARCHART ActiveX controls can be made compatible (depends on the settings on the **DataDefinition** property page), 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 graphical settings of nodes and their data.
- Users can interactively create, edit, delete or move nodes.
- Nodes can be displayed in TreeView style. It makes vertical levels show a plus or minus symbol. Clicking on a symbol will expand/collapse the subtree and transform the symbol into the opposite symbol.
- The total height of a tree diagram can be limited by the number of levels.
- The layout of a tree structure can be optimized by a combination of horizontal and vertical arrangements of subtrees. You can select the level, from that on the nodes are arranged vertically.
- You can collapse and expand subtrees.
- You can choose, whether the structure of the tree is to be defined by a structure code or by the ID of the parent node.
- OLE Drag & Drop operations in VARCHART XTree are compatible to the ones in Visual Basic. Methods, properties and events show identical names and results as the default objects of Visual Basic. Nodes or subtrees can be moved or copied via the OLE drag&drop mode.
- You can zoom your diagrams smoothly. Sections of your diagram can be zoomed interactively to full screen size. You can move within the diagram via the scroll bars and view other sections in the same enlargement.
- If you move a node behind a margin of the control form, the diagram will be scrolled automatically (Autoscrolling).
- 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.

1.2 Technical Requirements

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

- operating system Microsoft Windows NT 4.0, 2000, XP or Server 2003
- a development environment that supports the integration of ActiveX such as Visual C++, Visual Basic, Visual Fox Pro, Delphi, Centura, Oracle Forms, Progress, HTML (Visual Basic Script)
- about 5 MB hard disk space.

1.3 Installation

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

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

- c:\windows\system\regsvr32 c:\varchart\xtree\vctree.ocx

Naturally, the specified paths depend on the settings of your computer.

The installation procedure is logged in the file *install.log* allowing you to trace where files were copied.

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

You can remove the registration entry yourself using the Windows system program **regsvr32.exe**:

- c:\windows\system\regsvr32 /u c:\varchart\xtree\vctree.ocx

1.4 Shipping the Application

When you deliver your application, you must include the following files or check whether they are already installed in your customer´s Windows directory (e.g. in the directory C:\Winnt\System32\):

VARCHART XTree files:

- *vctree.ocx*
- *vcpane32.dll* (Version 3.150)
- *vcprct32.dll* (Version 3.150)
- *vcwin32.dll* (Version 3.150)
- *vxcsv32.dll* (Version 1.280)

Windows files:

- *msvcp60.dll* (Version 6.0)
- *mfc42.dll* (Version 6.0)
- *msvcrt.dll* (Version 6.0)
- *olepro32.dll* (Version 2.20)
- *oleaut32.dll* (Version 2.20)
- *comctl32.dll* (Version 4.72)
- *wininet.dll* (Version 4.72)
- *shlwapi.dll* (Version 4.72)

The files *mfc42.dll*, *oleaut32.dll*, *olepro32.dll* and *vctree.ocx* have to be registrated (Start-Menü, Programs, DOS Prompt, via the command *regsvr32*).

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

- *vctree.lic* (necessary for activating the property pages in the design mode)
- *vctree.hhc* (online help contents file)
- *vctree.chm* (online help file)

1.5 Data Exchange by VARCHART XTree

At present, data are exchanged by the VARCHART ActiveX controls via variants. For this, you can address a file or communicate via the API. Via the API, you can either enter or read a complete data record, or, for vcNode objects, address the data as properties of the VcNode object by fields.

1.5.1 Definition of the Interface

By default 20 data fields are available in the Maindata table. On the **DataDefinition** property page you can generate new data fields by editing the **New** entry at the bottom of the list. As soon as you leave the field, a new one is created.

You can name the fields and specify their data type (alphanumeric, integer, date/time) on the property page **DataDefinition**. 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 and separate the data fields by semicolons.

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 **SaveAs** 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 (corresponds to the **Save** method).

Note: CSV-Files may be retrieved in ANSI as well as in Unicode (automatic recognition), but only written in ANSI.

Example Code

```
VcTree1.Open "c:\data\example1.tre"
...
VcTree1.SaveAs ""
' or
VcTree1.SaveAs "c:\data\example2.tre"
```

1.5.4 Transferring Node Data to VARCHART XTree via API

When you use the call interface, each node has to be passed by the **Insert-NodeRecord** 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;0;100;03.11.00;10.11.00;;0;;"
VcTree1.InsertNodeRecord data
VcTree1.EndLoading
```

1.5.5 Retrieving Node Data from VARCHART XTree

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 = VcTree1.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
```

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

To insert a VARCHART ActiveX control in your MFC project, 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: *xtree.h* and *xtree.cpp*. These files are 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 *xtree.cpp* from the subdirectory MFC of the installation directory of the VARCHART ActiveX control. The corresponding header file (*xtree.h*) you will also find there.

If you use only one control in one class, the following two code lines are sufficient:

Example Code

```
#include "xtree.h"
using namespace XTree;
```

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

```
XTree::CVcTitle title = VcTree1.GetTitle();
```

If you want to create an ActiveX control dynamically (i. e. not via statical insertion into a dialog resource), than in the **Create** method of the wrapper of the control you have to insert the content of the first line of the NETRONIC LIC file for the parameter **bstrLicKey**.

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 ActiveX Controls in Browser Environment

In this chapter it is shown how to get VARCHART ActiveX controls working in an HTML page and how to control them via script. There are some restrictions in comparison with the programming of normal applications:

- Javascript/JScript (ECMAScript) is not suitable as script language because it does not offer parameters by-reference, so that it is impossible to return other values in functions besides the return value itself. Considerable are the methods **IdentifyObjectAt** and most of the events, e.g. **OnNodeCreate**.
- VBScript is suitable however.
- Netscape browsers are not suitable at all because of the following reasons:

1. The script language supported is Javascript which is not suitable. Since Netscape browsers do not support VBScript, any possibility of control via the methods and events mentionned above is not available.
2. ActiveX controls are not supported. However, a plug-in free of charge is available: the Esker ActiveX Plug-In (www.esker.com). The Esker plug-in cannot handle the Windows License Manager which is required to run licensed ActiveX controls on any computer.
3. Netscape does not support the OBJECT tag. Instead of this, the EMBED tag is offered.
4. The CAB format used by the Internet Explorer cannot be used by the Esker Plug-In. Therefore you have to write your own Setup if an ActiveX control is to be displayed without installing it explicitely before.

Earlier, an alternative to the Esker Plug-In for Netscape browsers existed: the ScriptActive-Plug-In of NCompass. But this Plug-In is not on the market any more. It allowed to model any property missing in the Netscape browser like VBScript, OBJECT tag and the support of ActiveX controls.

Please consider that the management of VARCHART ActiveX controls via script is no substitute for a real application. Scripts are only suitable for relatively small applications. If you plan a larger application, you should develop your own ActiveX control, e.g. via Visual Basic, containing one or several VARCHART ActiveX controls. For example you normally cannot access the mass storage of the target computer using a script , whereas an ActiveX control is able to do this (even if it is not supposed to).

In the following section will be described how to implement VARCHART ActiveX controls into HTML pages in the Microsoft Internet Explorer using the script language VBScript.

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

Example Code

```
<OBJECT ID="VcTree1" WIDTH=700 HEIGHT=350
 CLASSID="CLSID:CA393F28-3DE9-11D3-B22E-0080AD0058C7"
 CODEBASE="vctree.cab#version=3,000,0,0">
</OBJECT>
```

The command specifies the size and the Class-ID of the VARCHART ActiveX control. Each VARCHART ActiveX control has got an unique Class-ID by which it is identified when it is registered. If an ActiveX control is to be displayed without explicite installation, the Codebase parameter will be used. Here is specified where the according installation file is stored on the server. The CAB file that has 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 only an older version available on the target computer.

The CAB file was signed by NETRONIC Software GmbH, so that the user will receive a message about the certificate in the Internet Explorer, when the browser starts to install the control. In addition, the VARCHART ActiveX control is signed as safe for installation and for the usage in script languages ("Safe for Scripting and Initializing").

Inserting the OBJECT tag is not enough however. You also have to provide a LPK file. This LPK file contains the combined licensing information of the embedded controls. The LPK file is generated by the LPKTool from Microsoft and is stored together with the HTML page and if necessary with the CAB file on the server. You can receive the LPKTool free of charge from Microsoft together with the Internet Client SDK. The LPK file is read by the Microsoft License Manager via the following OBJECT tag, which has to be inserted on the HTML page before any other OBJECT tag:

Example Code

```
<OBJECT CLASSID="clsid:5220cb21-c88d-11cf-b347-00aa00a28331">
<PARAM NAME="LPKPath" VALUE="xtree_sample.lpk">
</OBJECT>
```

After embedding the VARCHART ActiveX control in the HTML page, you have to provide your own configuration file so that the VARCHART ActiveX control will show the desired appearance. This is done via a script, in which the **ConfigurationName** property of the VARCHART ActiveX control is set to an URL that writes a file that is stored together with the others on the server:

Example Code

```
VcTree1.ConfigurationName = "www.netronic_test.com/xtree_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. These files can be generated in the following way: Drag the VARCHART ActiveX control into a development environment and configure its property pages. Then save the configuration files via the property page **General**.

A complete example is delivered by NETRONIC Software GmbH on the installation CD.

If the URL of the INI file is known when the HTML page is written (i. e. if it has not to be determined via script), then you also can assign the configuration file via <PARAM> tag within the <OBJECT> tag. The advantage of this way is that the ActiveX control will be displayed even at the first display with the valid settings (colors, size etc.). Otherwise the default settings will appear briefly.

Example Code

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

➤ References for solving problems and for further technical information:

Microsoft Knowledge Base (http://msdn.microsoft.com):

- Q159923 – “FILE: Using Licensed ActiveX Controls in Internet Explorer”
- Q167597 – “HOW TO: Specifying FileVersion and #Version for Component Download”
- Q169438 – “PRB: ActiveX Control Does Not Display Correctly on Web Page”
- Q182598 – “HOW TO: Implementing IObjectSafety in Visual Basic Controls”

YahooGroups forum for Esker ActiveX Plug-In

(http://groups.yahoo.com/group/esker-activex-plugin/info.html):

- in Section "Files”: Esker, “DPL file reference”
- Message 211, “Re: Microsoft License Manager”
- Message 560, “Re: Registering DLLs”

- Message 656, “Re: Problem with .dpl file”

1.8 Support and Advice

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

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

Have you just started testing VARCHART XTree and are you wondering how to get to a special feature of your tree 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 support@netronic.com

www.netronic.com

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

- Support hotline
- Detailed expert advice to questions of application
- Quick fixing of possible bugs in the software
- Upgrade to a new VARCHART XTree release for development and runtime versions.

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

2 Tutorial

2.1 Overview

This tutorial teaches you how to utilise VARCHART XTree in your programming environment. We have chosen Visual Basic as our sample environment because it is widespread and the syntax is simple and easy to understand.

Naturally, some items are specific to Visual Basic alone. However, this is only in the case of simple procedures that can be realised in a similar way in other development environments.

In the tutorial you will learn how to adjust the data interface to the structure of your data, how to adapt the various graphical elements and which types of interaction are possible.

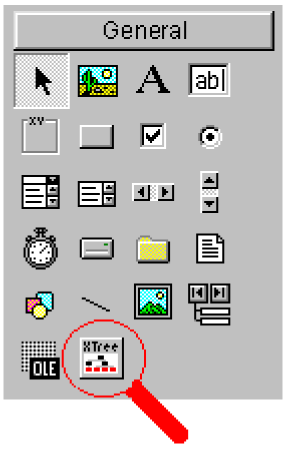
Use the sample data supplied in the file *tutorial.tre* which you can find in the "Samples" subdirectory below the VARCHART XTree directory.

2.2 Adding VARCHART XTree to the Toolbox

For adding VARCHART XTree to the toolbox proceed as following:

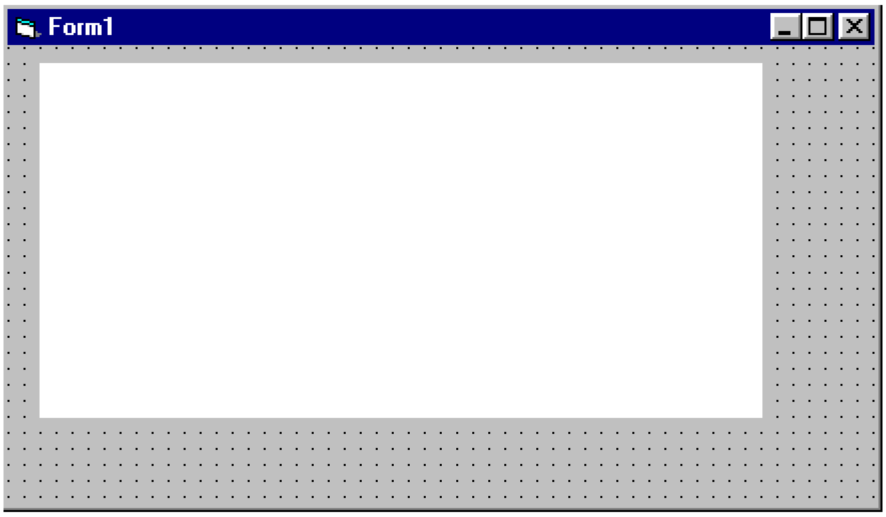
1. In the Project menu of Visual Basic, choose the **Components** option.
2. On the record card **Controls**, choose **NETRONIC VARCHART XTree** from the list and confirm your choice by **OK**.

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

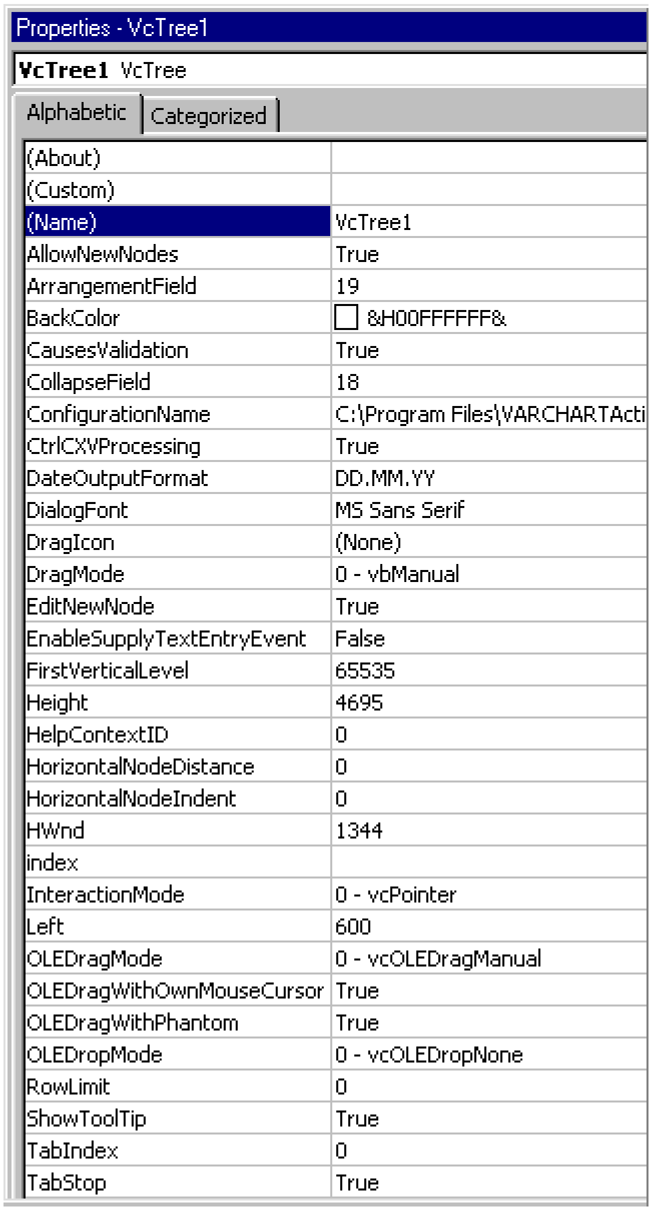


2.3 Placing the VARCHART XTree control in a Form

To place the VARCHART XTree control in a Visual Basic form you simply have to click on it in the toolbox after inserting VARCHART XTree in the toolbox and then, with the mouse, draw a frame for the VARCHART XTree control at the position in the form where you want it to appear. Then the VARCHART XTree control will be displayed in the size you specified. Naturally, you can readjust the size with the help of the mouse.



In the **Properties** dialog of the control, you can activate the VARCHART XTree property pages via the **Custom** entry.

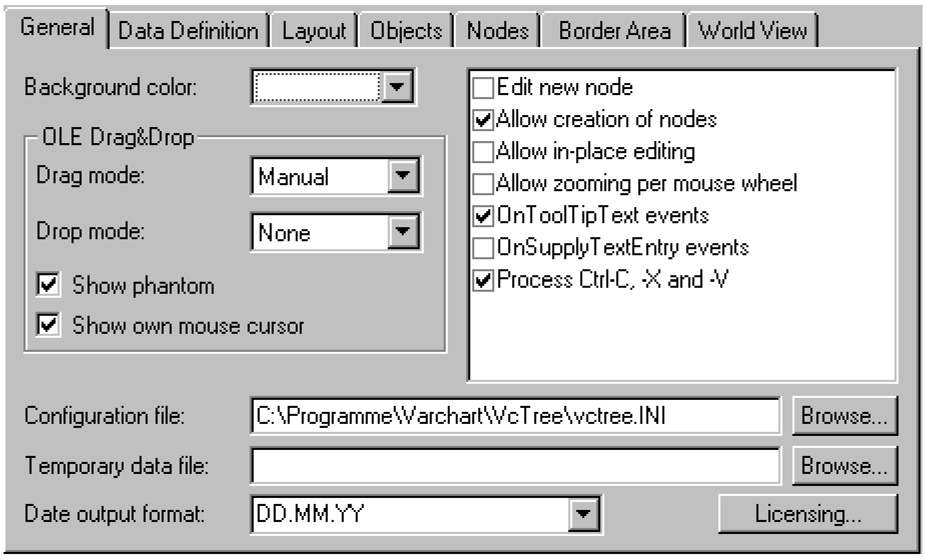

Properties - VcTree1

VcTree1 VcTree

Alphabetic | Categorized

(About)	
(Custom)	
(Name)	VcTree1
AllowNewNodes	True
ArrangementField	19
BackColor	☐ &H00FFFFFF&
CausesValidation	True
CollapseField	18
ConfigurationName	C:\Program Files\VARCHARTActi
CtrlCXVProcessing	True
DateOutputFormat	DD.MM.YY
DialogFont	MS Sans Serif
DragIcon	(None)
DragMode	0 - vbManual
EditNewNode	True
EnableSupplyTextEntryEvent	False
FirstVerticalLevel	65535
Height	4695
HelpContextID	0
HorizontalNodeDistance	0
HorizontalNodeIndent	0
HWnd	1344
index	
InteractionMode	0 - vcPointer
Left	600
OLEDragMode	0 - vcOLEDragManual
OLEDragWithOwnMouseCursor	True
OLEDragWithPhantom	True
OLEDropMode	0 - vcOLEDropNone
RowLimit	0
ShowToolTip	True
TabIndex	0
TabStop	True

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



Background color:

OLE Drag&Drop
Drag mode: Manual
Drop mode: None
[✓] Show phantom
[✓] Show own mouse cursor

[ ] Edit new node
[✓] Allow creation of nodes
[ ] Allow in-place editing
[ ] Allow zooming per mouse wheel
[✓] OnToolTipText events
[ ] OnSupplyTextEntry events
[✓] Process Ctrl-C, -X and -V

Configuration file: C:\Programme\Varchart\VcTree\vctree.INI
Temporary data file:
Date output format: DD.MM.YY

Note: Here and in the example code, the inserted VARCHART XTree control is called **VcTree1**.

2.4 Automatic Scaling of VARCHART XTree

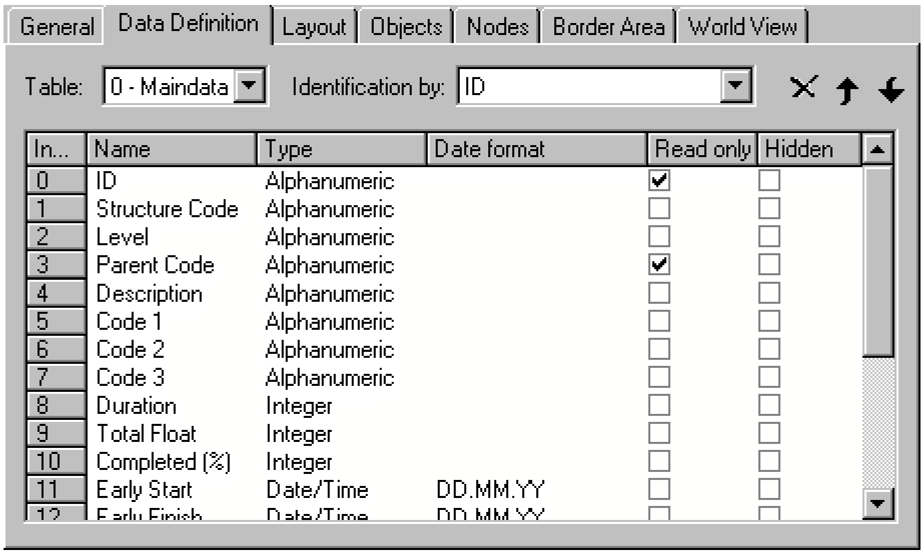
If you wish the bottom and right-hand side of the VARCHART XTree control to be adjusted to the full size of the window during runtime, add the below code:

Example Code

```
Private Sub Form_Resize()
   If ScaleWidth - VcTree1.Left > 0 And _
              ScaleHeight - VcTree1.Top > 0 Then
      VcTree1.Width = ScaleWidth - VcTree1.Left
      VcTree1.Height = ScaleHeight - VcTree1.Top
   End If
End Sub
```

2.5 Preparing the Interface

Prepare the interface now by defining the data fields of the **Maindata** table (node data). For this, please open the **DataDefinition** VARCHART XTree property page.



General | Data Definition | Layout | Objects | Nodes | Border Area | World View

Table: 0 - Maindata   Identification by: ID

In...	Name	Type	Date format	Read only	Hidden
0	ID	Alphanumeric		✓	☐
1	Structure Code	Alphanumeric		☐	☐
2	Level	Alphanumeric		☐	☐
3	Parent Code	Alphanumeric		✓	☐
4	Description	Alphanumeric		☐	☐
5	Code 1	Alphanumeric		☐	☐
6	Code 2	Alphanumeric		☐	☐
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	☐	☐

The field of the index "0" by default is named "ID" and is of the type "alphanumeric". To adapt your interface for the tutorial, please replace "ID" by "Number" and select the data type "Integer". The name can be edited via a double-click or when it is marked via a left-click. The type you can select from a combo box that appears after clicking on the **Type** field.

You can create a new data field by editing the "New..." field in the last row.

Please modify the fields of the Maindata table as shown below:

Index	Name	Type
0	Number	Integer
1	Structure code	alphanumeric
2	Level	Integer
3	Parent node	alphanumeric
4	Name	alphanumeric
5	Group code	alphanumeric
6	Code	Integer
7	Group name	alphanumeric
8	Duration	Integer
9	Float	Integer
10	completed (%)	Integer
11	Early start	Date/Time

Index	Name	Type
12	Early finish	Date/Time
13	Late start	Date/Time
14	Late finish	Date/Time
15	Free float	Integer
16	Calculated Start	Date/Time
17	Calculated Finish	Date/Time
18	Collapsed	Integer
19	Arrangement	Integer

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

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

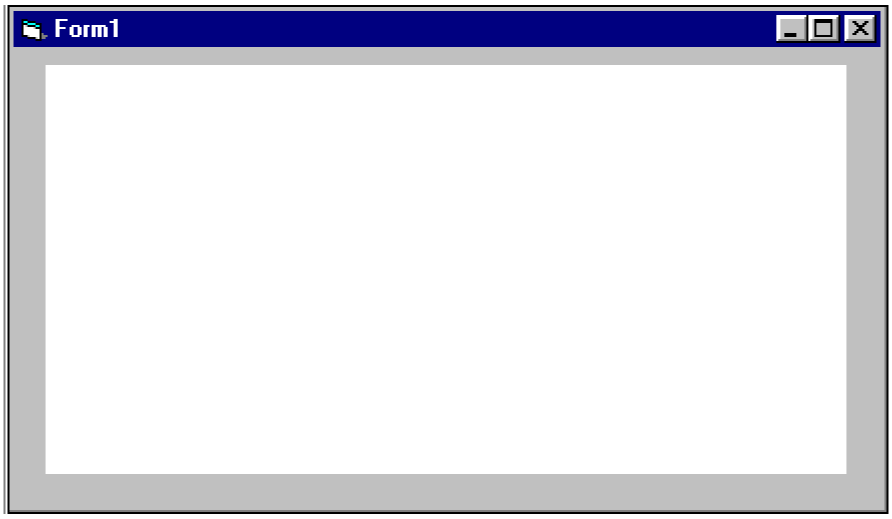
Now select a field that the node is to be identified by. From the field **Identification by** please select the field "Number".

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

By clicking on the **Apply** button the modificatiosn of this configuration will be stored. They will also be stored by clicking on the **OK** button and by changing to a different property page, thus being available to other property pages immediately.

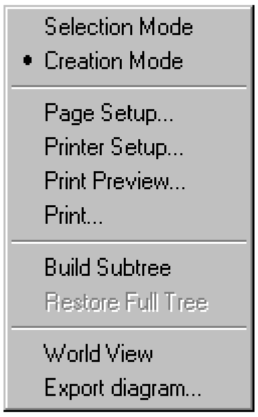
2.6 Your First Run

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

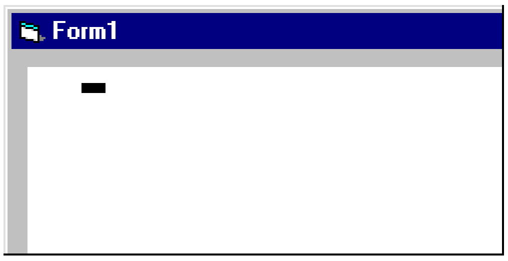


▸ Creating Nodes

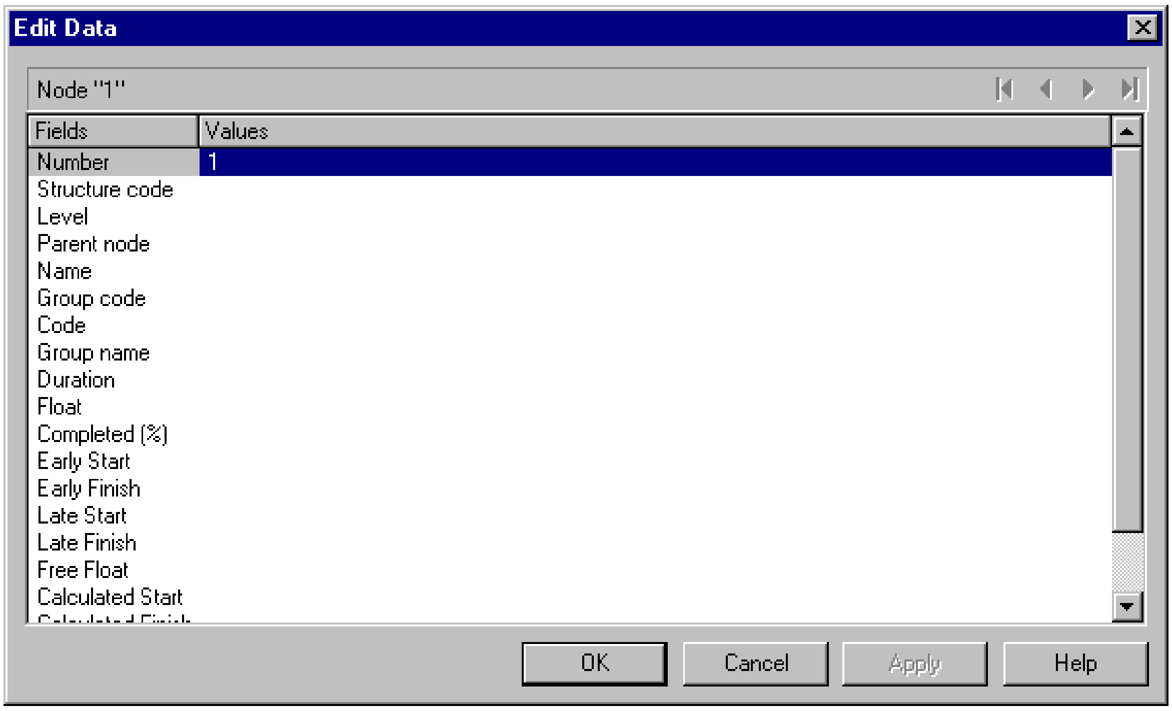
There are two modes that you can toggle between in VARCHART XTree: The **Selection Mode** and the **Creation Mode**. Nodes can be generated in **Creation Mode** only. To change modes, press the right mouse button on an empty area in the diagram and select the appropriate menu item from the context menu popping up.



In creation mode the cursor will transform into a small black rectangle.

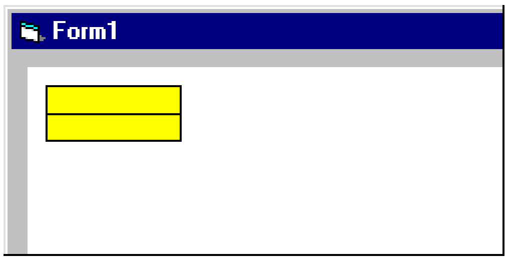


If you click on the left mouse button, two different things may happen, depending on the settings on the **General** property page. If the check box **Edit new node** was ticked, the **Edit Data** dialog will appear that displays the node data.



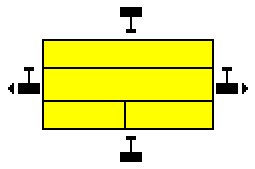
Fields	Values
Number	1
Structure code	
Level	
Parent node	
Name	
Group code	
Code	
Group name	
Duration	
Float	
Completed (%)	
Early Start	
Early Finish	
Late Start	
Late Finish	
Free Float	
Calculated Start	

The left column lists the field names of the node record, whereas the right column displays the corresponding values. Most of the values do not exist, only the "Number" field has a value at this point, which is "1". You can add values, such as dates or a description. As soon as you confirm the data by the **OK** button, the node will be generated. The dialog will disappear and the node will be displayed.



If on the **General** property page the check box **Edit new node** was not ticked, a node will be displayed as soon as you click the left mouse button (provided you are in **Creation Mode**) in an empty place of the diagram. The **Edit Data** dialog will not appear.

The next nodes you can generate by placing the cursor near the existing node. The cursor will change its shape according to whether the new node is going to be a parent node, a child node or a left or right brother node.



➤ Editing Nodes

You can edit a node by opening the the **Edit Data** dialog via a double-click. In this dialog you will find the data fields defined on the **DataDefinition** property page. Data fields defined as **Hidden** will not appear in this dialog. Data fields defined as **read only** cannot be edited in this dialog.

➤ Back to Design Mode

Finish your first run by closing the form.

2.7 Loading Data from a File

To feed data into VARCHART XTree, load the file *tutorial.tre*. You can do this automatically on the start. *Tutorial.tre* 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:

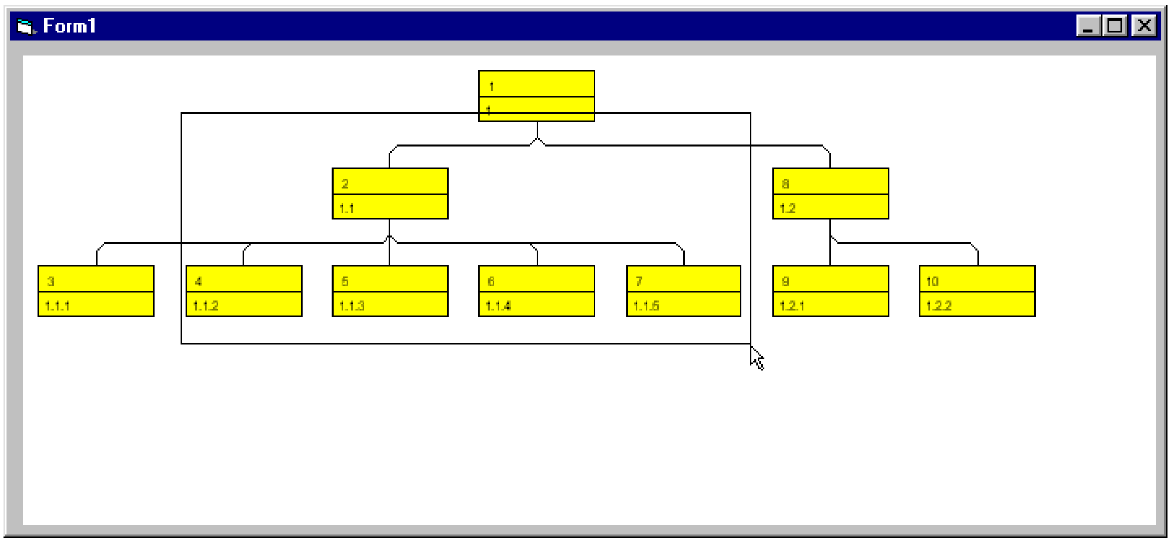
Example Code

```
Private Sub Form_Load()
    VcTree1.Open "C:\Programs\Varchart\xtree\tutorial.tre"
End Sub
```

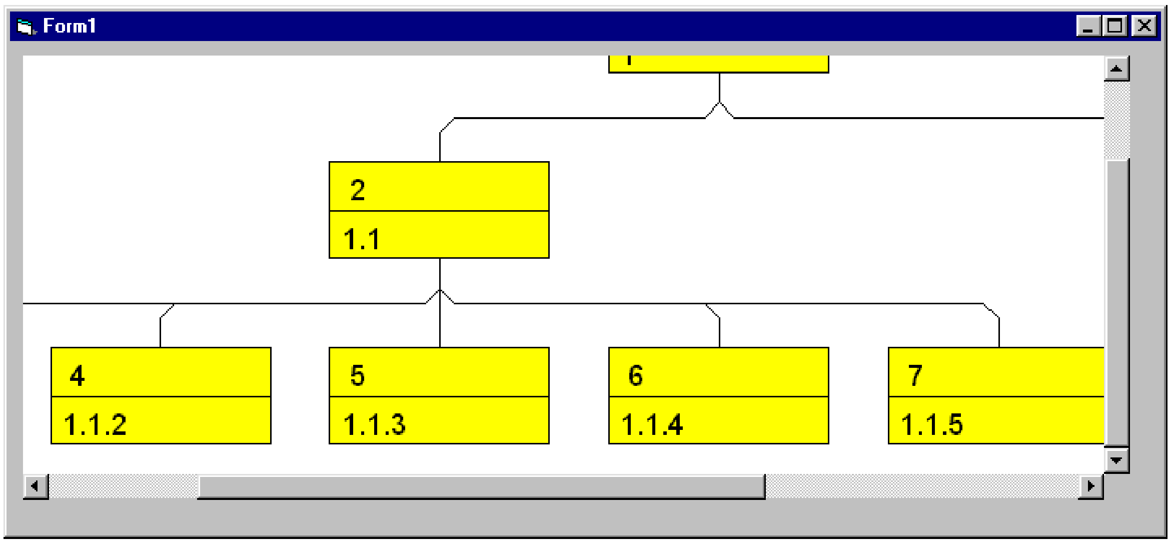
The path depends on the installation of your program. Please save the project now. If you start the program, the nodes and links of the project will be displayed.

VARCHART XTree will display a tree 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()
   VcTree1.Zoomfactor = 100
End Sub
```

If you want VARCHART XTree to cover the form completely, verify the following:

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

2.8 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 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** (*.tre).

Please select the delivered sample file *tutorial.tre* 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 **Objects** property page and click on the **Node Appearances** button. The **Administrate Node appearances** dialog will open, with the "Standard" 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 into a tree diagram by the method **InsertNodeRecord** or **InsertNodeRecordEx**.

2.9 Specifying the Marking Type of Nodes

On the **Nodes** property page you can specify the appearance of marked nodes. Just select an entry of the **Marking type** combo box.

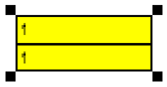
Start the program, switch to the creation mode and generate some nodes for marking.

You can mark nodes by clicking on them with the left mouse button. By simultaneously pressing the Ctrl key you can mark several nodes. Each time you click on a node you toggle the marking on or off.

To mark a subtree, press the Shift button and click the left mouse button on the subtree´s parent node.

Click the left mouse button in an empty space of the diagram to demark the marked nodes.

Try different options of node marking. The picture below shows marking by pickmarks:

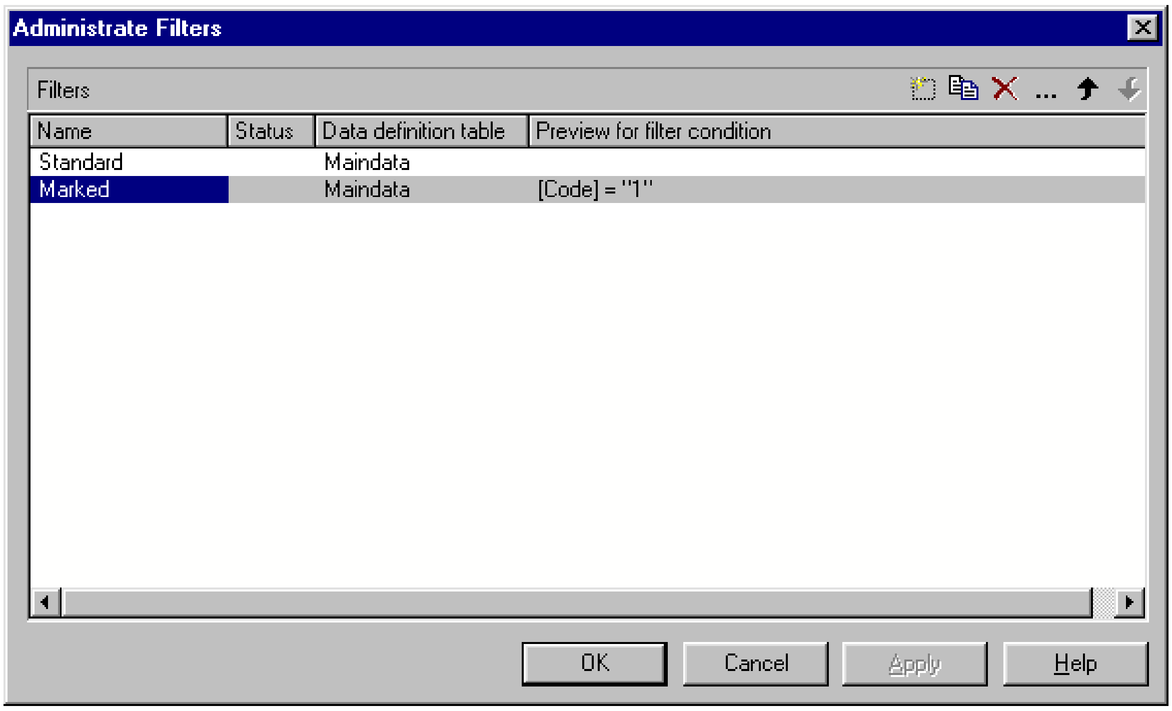


2.10 Setting Filters for Nodes

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

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.



Name	Status	Data definition table	Preview for filter condition
Standard		Maindata	
Marked		Maindata	[Code] = "1"

▸ Buttons in the "Administrate Filters" dialog box

Add filter

Copy filter

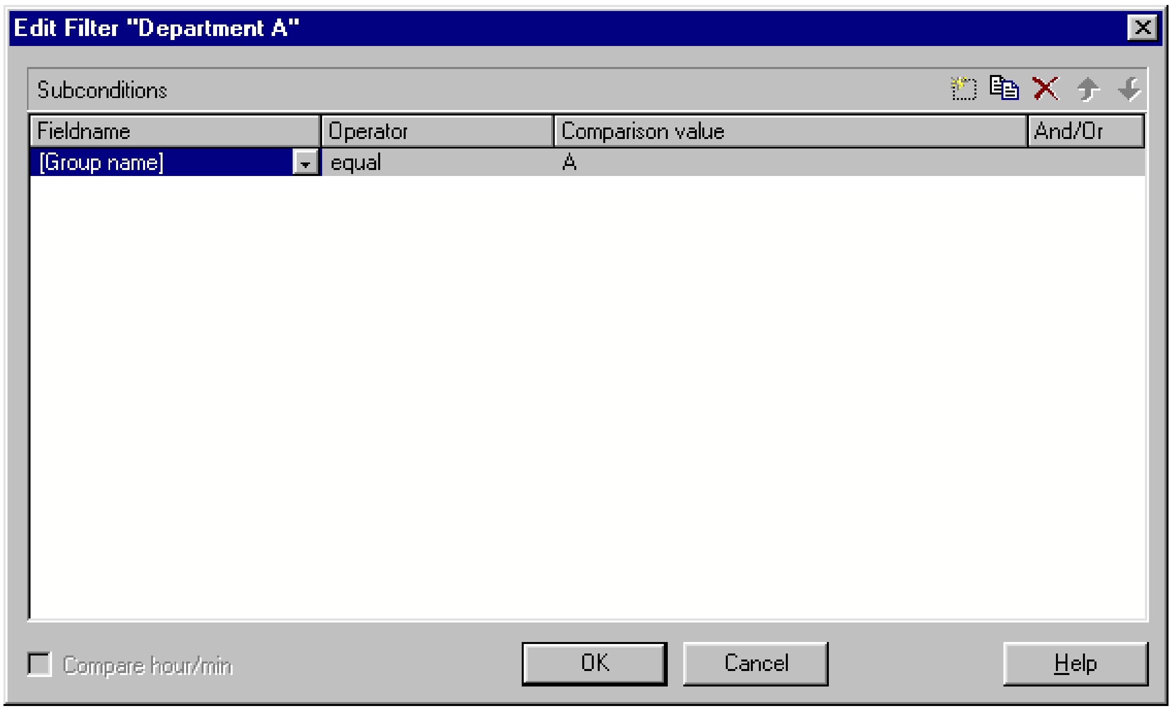
Delete filter

... Edit filter

▸ Creating and editing filters

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

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


Edit Filter "Department A"

Subconditions

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

The head line indicates the name of the current filter.

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

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

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

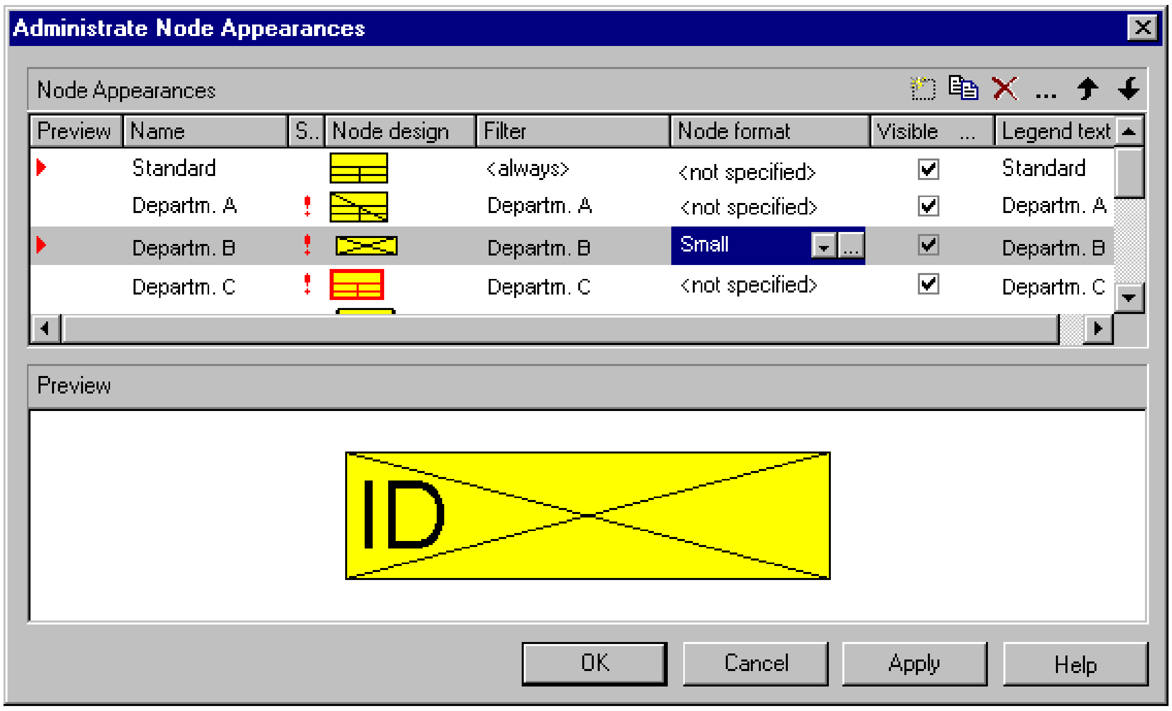
In the **And/Or** column you can choose the operators to combine the condition of the current row with the one in the row following, if necessary.

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

2.11 Setting Node Appearances

VARCHART XTree offers a variety of options to modify node appearances. You can define the appearance of a node depending on its data. For example, you can define a different node appearance for each department. A defined set of graphical attributes is called an appearance. A node may have several appearances of different priorities.

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



Preview	Name	S..	Node design	Filter	Node format	Visible ...	Legend text
▶	Standard			<always>	<not specified>	✓	Standard
	Departm. A	!		Departm. A	<not specified>	✓	Departm. A
▶	Departm. B	!		Departm. B	Small	✓	Departm. B
	Departm. C	!		Departm. C	<not specified>	✓	Departm. C

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.

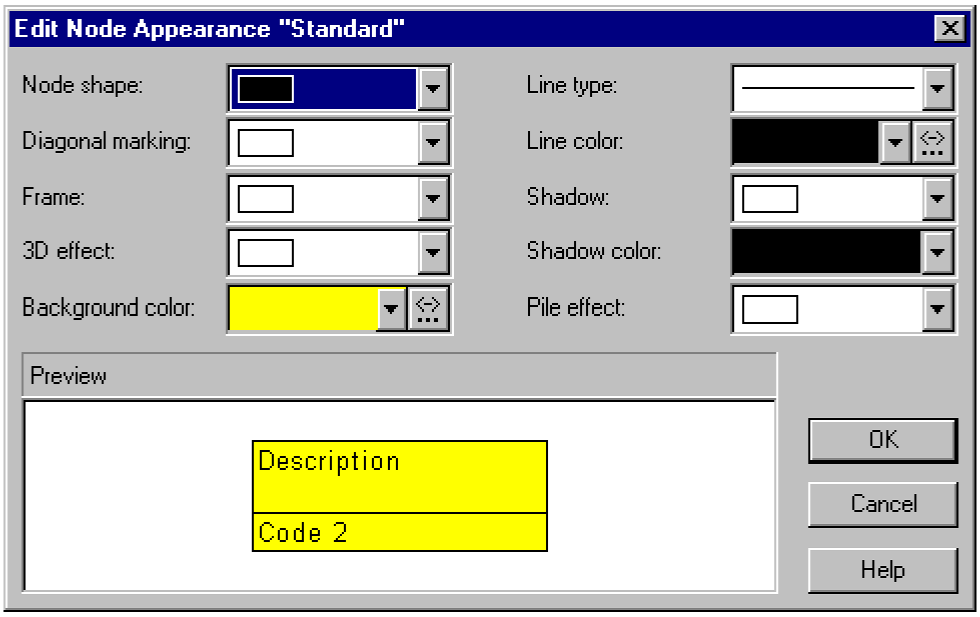
▸ Using node appearances and filters

This paragraph is about handling node appearances and their associated filters.

Please create the new node appearances "Department A" and "Department B" as copies of the node appearance "Standard".

Assign to the node appearance "Department A" the top priority by placing it at the bottom. Place the "Department B" node appearance right above it to receive second place priority.

Please edit the new 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.



Edit Node Appearance "Standard"

Node shape:

Diagonal marking:

Frame:

3D effect:

Background color:

Line type:

Line color:

Shadow:

Shadow color:

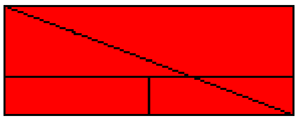
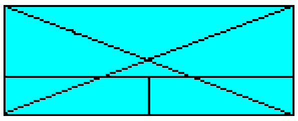
Pile effect:

Preview

Description

Code 2

Please enter the below settings:

Node appearance	Department A	Department B
Filter	Department A	Department B
Filter criterion	Group name equal A	Group name equal B
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 by steps in the **Edit Data** dialog.

- Please enter "A" into "Group name": The node will show the "Department A" node appearance with a red background and a downward strike-through pattern.
- Next, please enter "B" into "Group name": The node will show the "Department B" appearance, that has a crossed-lines strike-through pattern and a blue blackground.

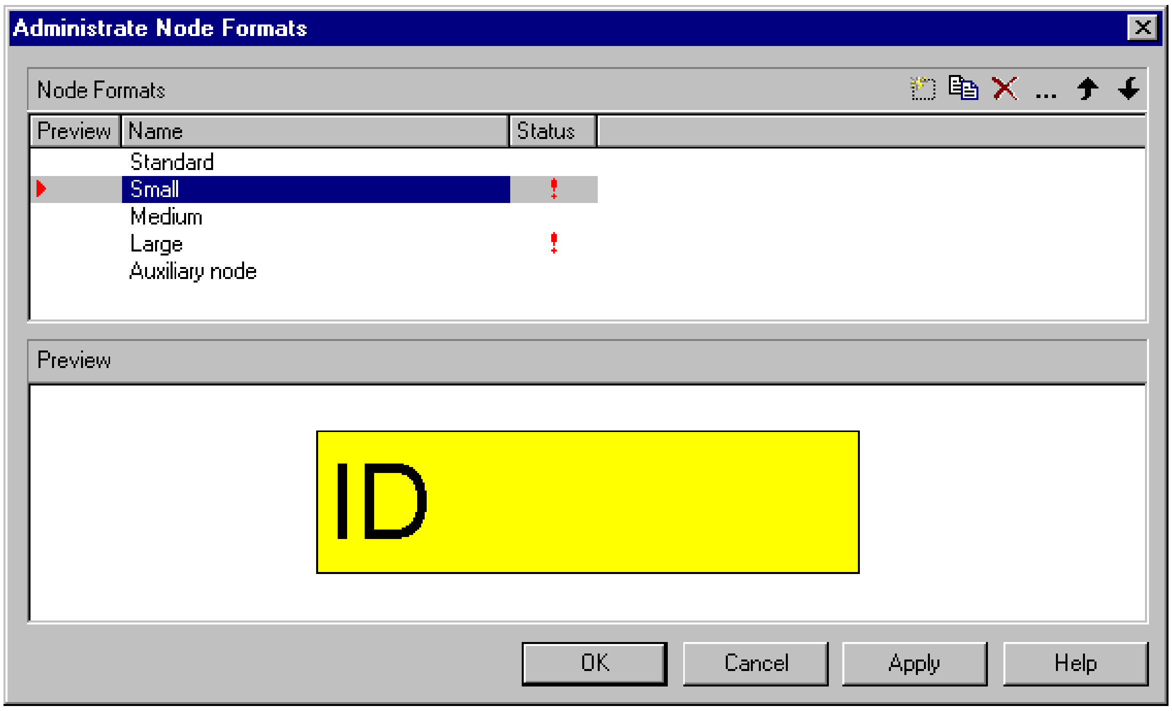
► Specifying the node appearance in dependence on its data

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".

2.12 Setting Node Formats

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

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



Preview	Name	Status
	Standard	
▶	Small	!
	Medium	
	Large	!
	Auxiliary node	

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

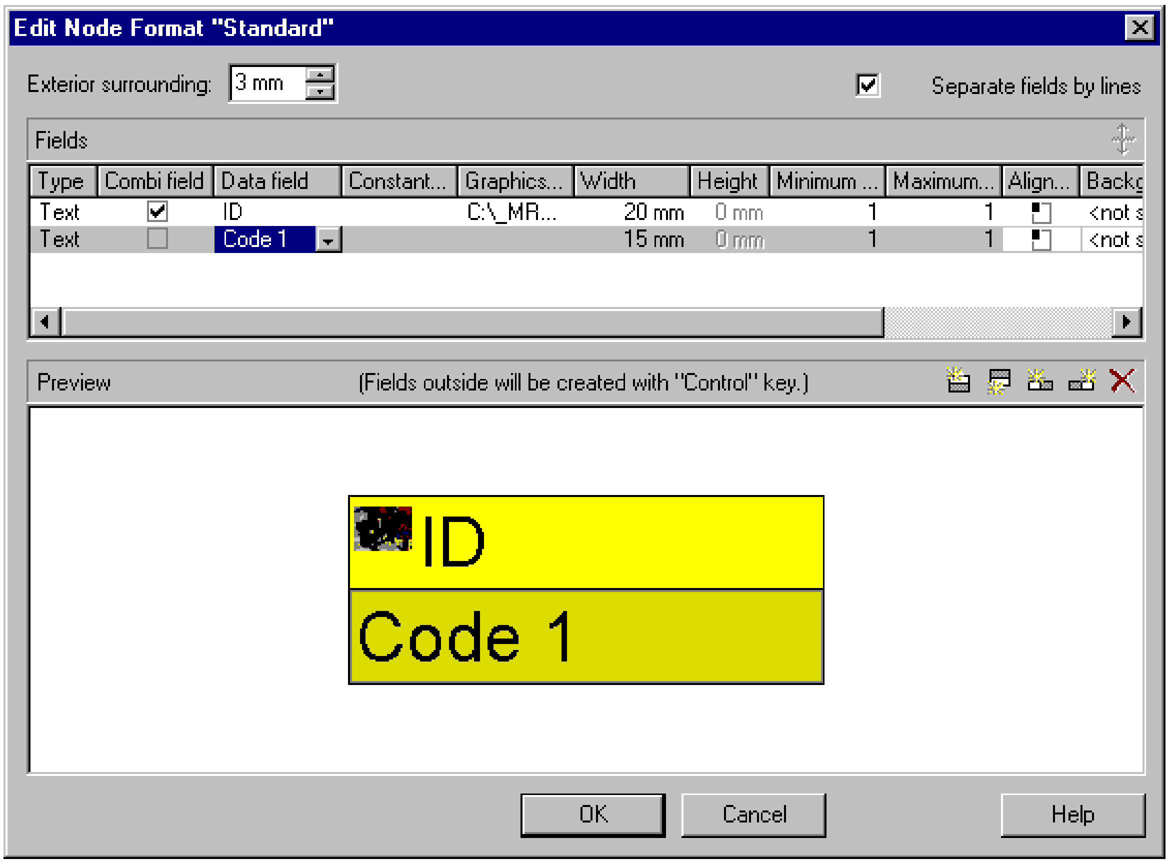
In the **Administrate Node Formats** dialog box you can create, copy, delete and edit node formats via the following buttons:

- **Add node format**
- **Copy node format**
- **Delete node format**
- ... **Edit node format**

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

► Editing Node Formats

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



Edit Node Format "Standard"

Exterior surrounding: 3 mm

☑ Separate fields by lines

Fields

Type	Combi field	Data field	Constant...	Graphics...	Width	Height	Minimum ...	Maximum...	Align...	Backg
Text	☑	ID		C:_MR...	20 mm	0 mm	1	1		<not s
Text	☐	Code 1			15 mm	0 mm	1	1		<not s

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

In this dialog box you can specify the following:

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

➤ Displaying graphics in node fields

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

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

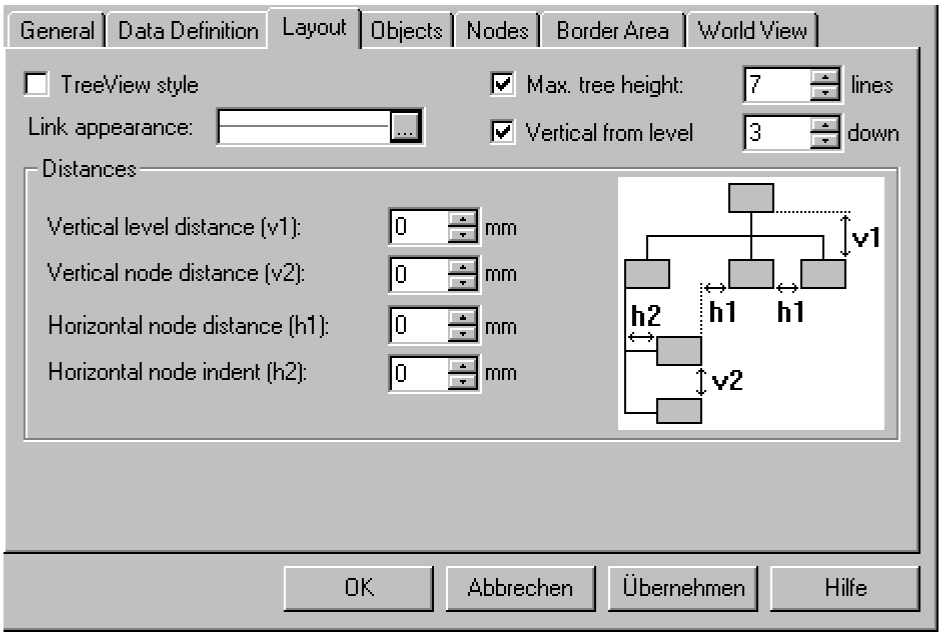
To configure a mapping from data field entries to graphics files, click the second button. Then the **Configure Mapping** dialog box will open.

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

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

2.13 Setting the Link Appearances

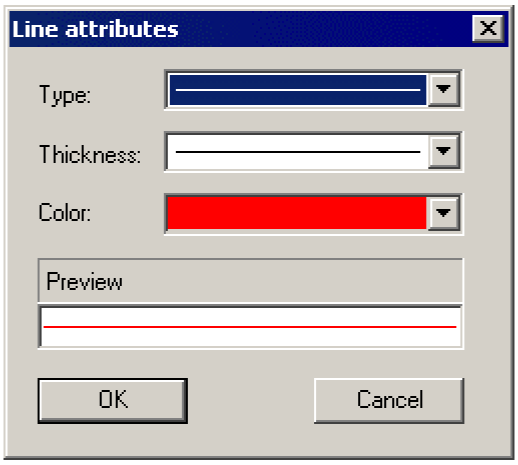
The field **Link Appearance** on the **Layout** property page displays the current link appearance.



☐ TreeView style
Link appearance:
☑ Max. tree height: 7 lines
☑ Vertical from level 3 down

Distances
Vertical level distance (v1): 0 mm
Vertical node distance (v2): 0 mm
Horizontal node distance (h1): 0 mm
Horizontal node indent (h2): 0 mm

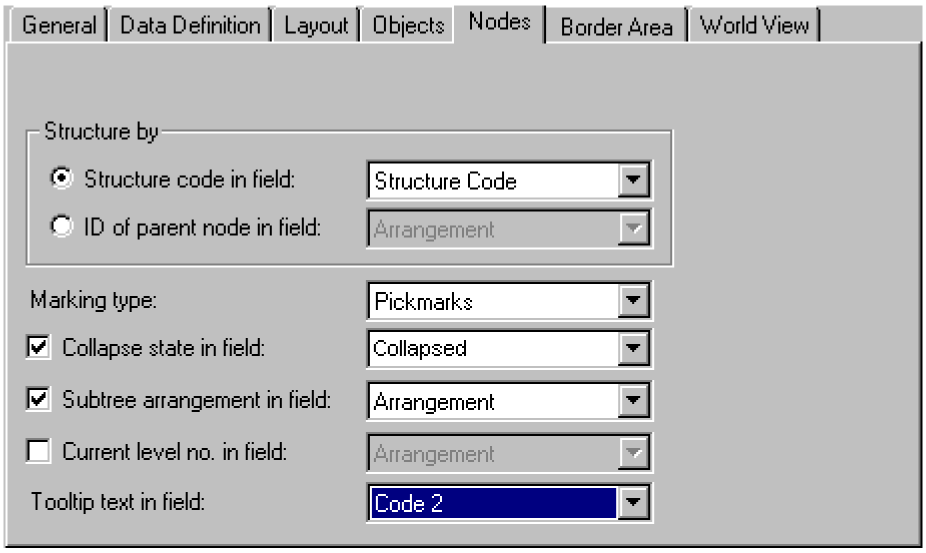
To modify it, click on the **Edit** button. You will get to the **Line Attributes** dialog, where you can set **Type**, **Thickness** and **Color** of the lines.



Type:
Thickness:
Color:
Preview

2.14 Storing the Tree Structure

On the **Nodes** property page you can enter the settings for storing the tree structure.



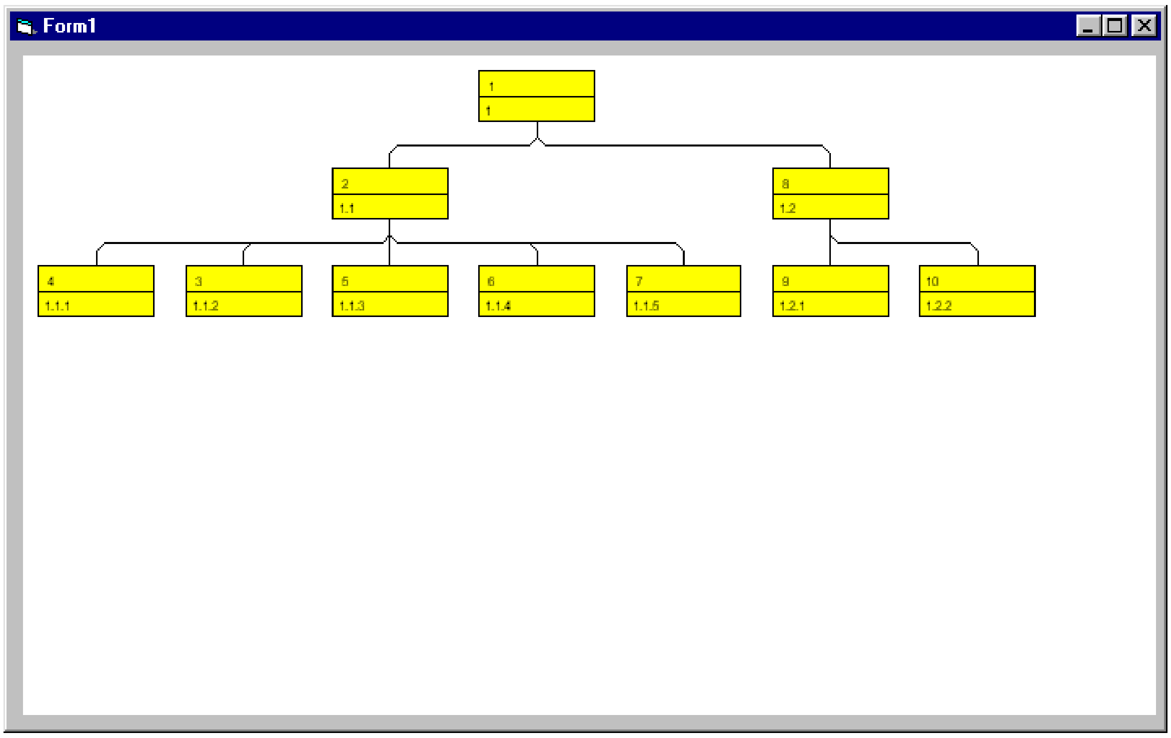
Structure by
- (●) Structure code in field: Structure Code
- ( ) ID of parent node in field: Arrangement

Marking type:	Pickmarks
[✓] Collapse state in field:	Collapsed
[✓] Subtree arrangement in field:	Arrangement
[ ] Current level no. in field:	Arrangement
Tooltip text in field:	Code 2

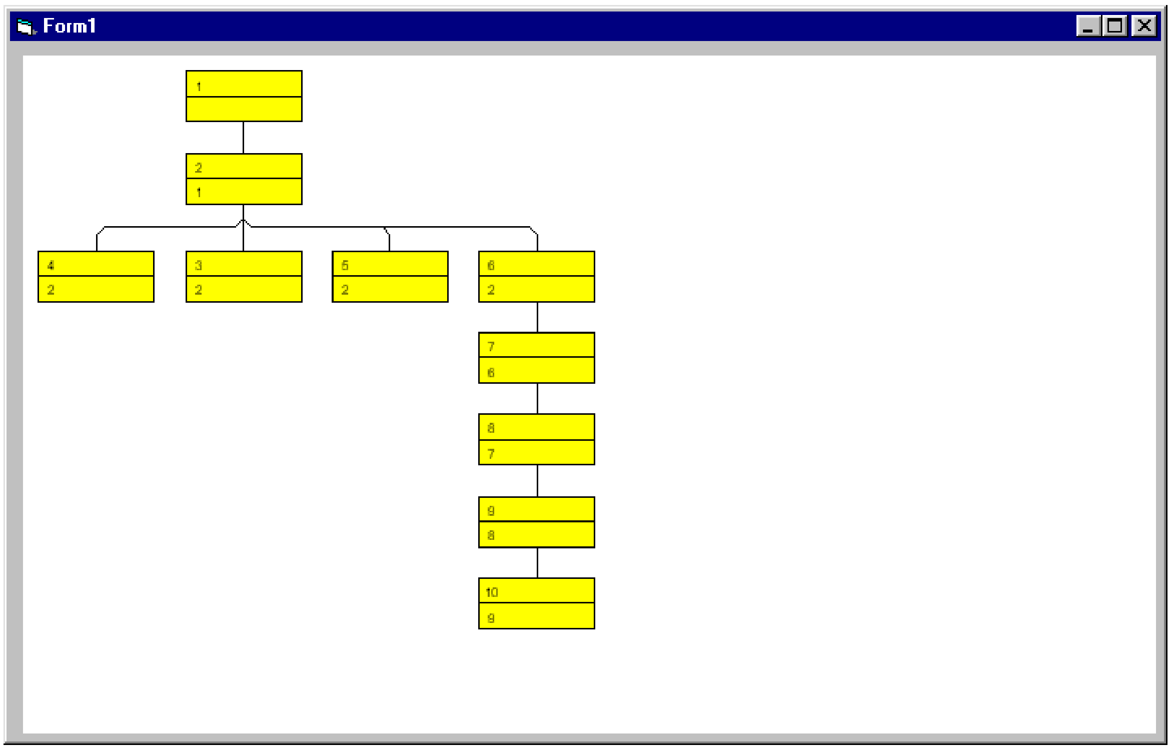
Basically, you need to decide whether the tree structure is to be defined via structure codes or via the IDs of parent nodes.

1. **Structure code in field:** The tree structure is built according to a structure code. You can select a field to hold the ID of the structure node. The levels are separated by a separator (point).
2. **ID of parent node in field** The tree structure is defined for each node by the ID of the parent node. You can select a field to hold the ID of the parent node.

Please set the radio button to **Structure code in field** and select the field **Structure code**. Its entry will determine the tree structure and will show the following result:



Please return to the design mode and select **ID of parent node in field**. Please select **parent node** as the field to hold the ID of the parent node. Please run the program to obtain the result below:



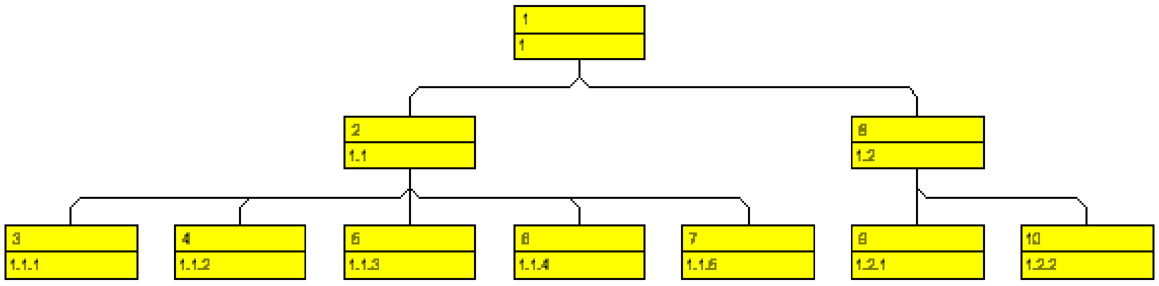
2.15 Vertical and Horizontal Arrangements in Tree Structures

You will learn in this paragraph how to optimize a tree diagram by combining horizontal and vertical arrangements of subtrees.

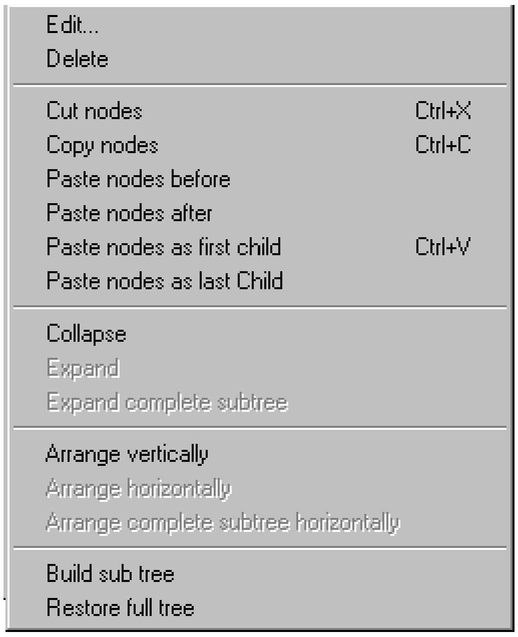
- *Horizontal arrangement:* Horizontally arranged subtrees will reduce the height of a tree diagram. All nodes of a level will be placed next to each other. The ports to connect a link will be placed in the center of the bottom line of a parent node, and in the center of the top line of a child node.
- *Vertical arrangement:*Vertically arranged subtrees will reduce the width of a tree diagram. All nodes of a level and its sublevels will be placed beneath each other. The ports to connect a link to a node will be placed in the bottom left corner of the parent node, and in the center of the left line of the child node.

On the **Nodes** property page, select **Structure code in field:** "Structure code".

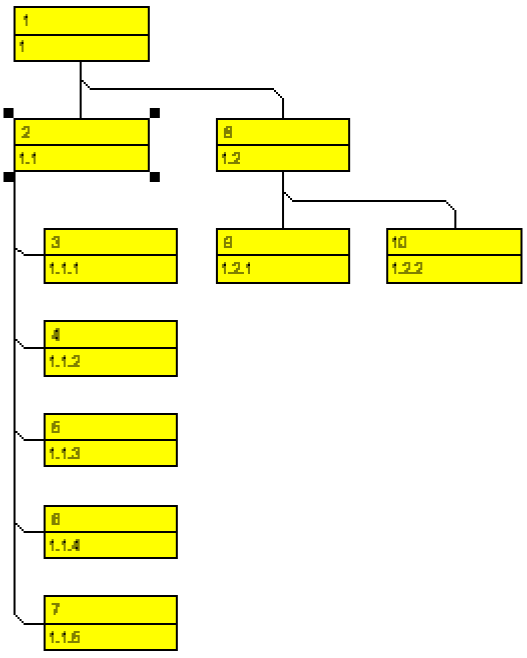
Please run the program now using the data file *tutorial.tre* (please see "Tutorial: Loading Data from a File").



Please mark the first node and press the right mouse button. In the context menu popping up available commands will be activated.



Please select the menu item **Arrange vertically**. The subtree beneath the marked node will be arranged vertically.



If not all levels of a subtree have been arranged vertically, please check the maximum height of the tree set on the **Layout** property page. The number of levels is limited by the **Max. tree height** check box and field.

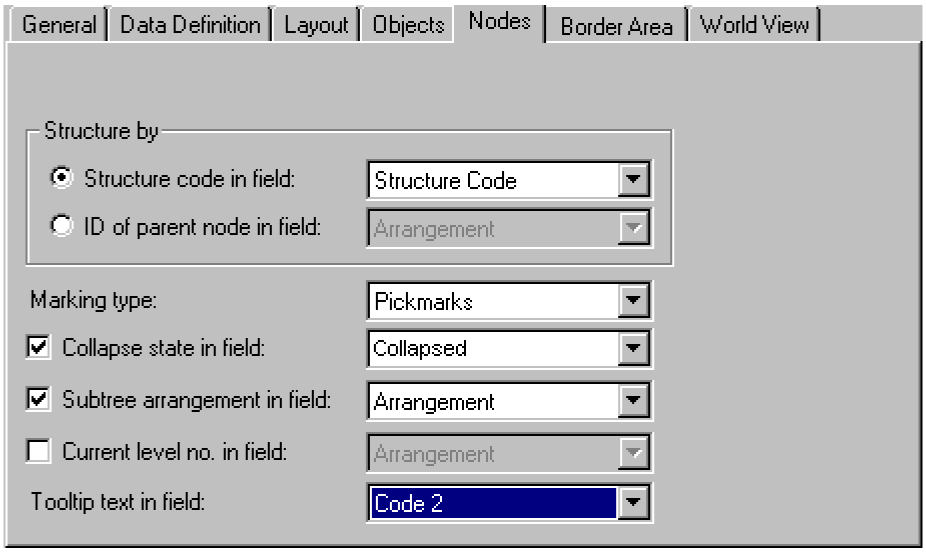
These settings will influence vertical arrangements only. If in a vertical branch more levels exist than set, another branch will be generated by the parent node to adopt the remaining child nodes.

Please tick the check box **Max. tree height** and select the value "10".

To return the vertical subtree into a horizontal subtree, please mark the top node of the subtree and press the right mouse button. In the context menu popping please select the menu item **Arrange horizontally**. The first level of the subtree will be arranged horizontally. If you wish all levels of the subtree to be arranged horizontally, please select **Arrange complete subtree horizontally**.

▸ Storing the Subtree Arangement to a data field

You can store the information on whether a subtree is arranged vertically or horizontally to a data field. Please return to design mode and open the **Nodes** property page.



Structure by
- (●) Structure code in field: Structure Code
- ( ) ID of parent node in field: Arrangement

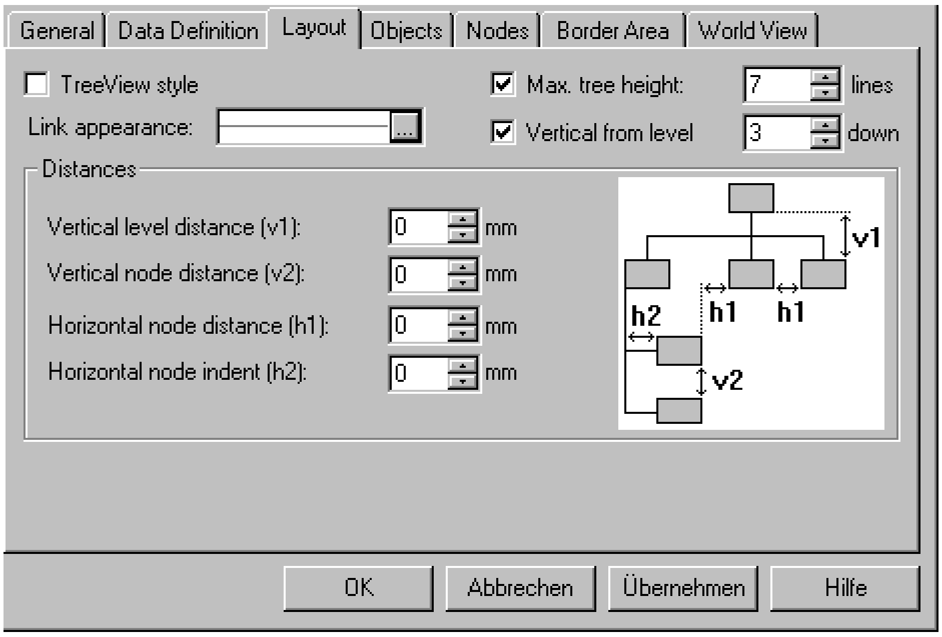
Marking type: Pickmarks
[✓] Collapse state in field: Collapsed
[✓] Subtree arrangement in field: Arrangement
[ ] Current level no. in field: Arrangement
Tooltip text in field: Code 2

Activate the **Subtree arrangement in field** check box and select the data field **Arrangement** from the combo box to keep the orientation of a subtree stored to that field. It may contain "0" for a subtree arranged horizontally, or "1" for a subtree arranged vertically. Horizontal arrangements in subtrees are visible only if the parent node is a part of a vertical arrangement.

Please start the program now and generate some nodes. Mark a node and double-click the right mouse button on it to open the **Edit Data** dialog. If its subtree is arranged horizontally, the **Arrangement** data field will display "0". If the subtree is arranged vertically, the data field will display "1".

▸ Distances

On the **Layout** property page you can specify the following distances (unit: mm):



☐ TreeView style

Link appearance:

☑ Max. tree height: 7 lines

☑ Vertical from level 3 down

Distances

Vertical level distance (v1): 0 mm

Vertical node distance (v2): 0 mm

Horizontal node distance (h1): 0 mm

Horizontal node indent (h2): 0 mm

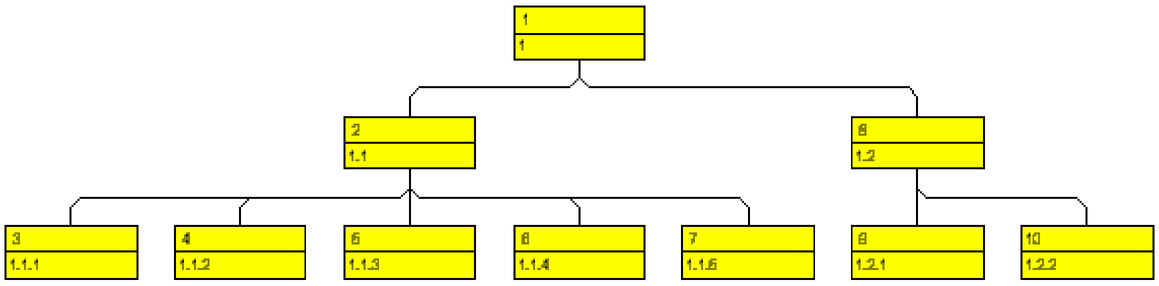
- **Vertical level distance (v1):** This field lets you enter the vertical distance between horizontally arranged levels.
- **Vertical node distance (v2):** This field lets you enter the vertical distance between vertically arranged nodes.
- **Horizontal node distance (h1):** This field lets you set the horizontal node distance between two horizontally arranged nodes.
- **Horizontal node indent (h2):** This field lets you enter the horizontal indent of vertically arranged nodes.

2.16 Collapsing and Expanding Tree Structures

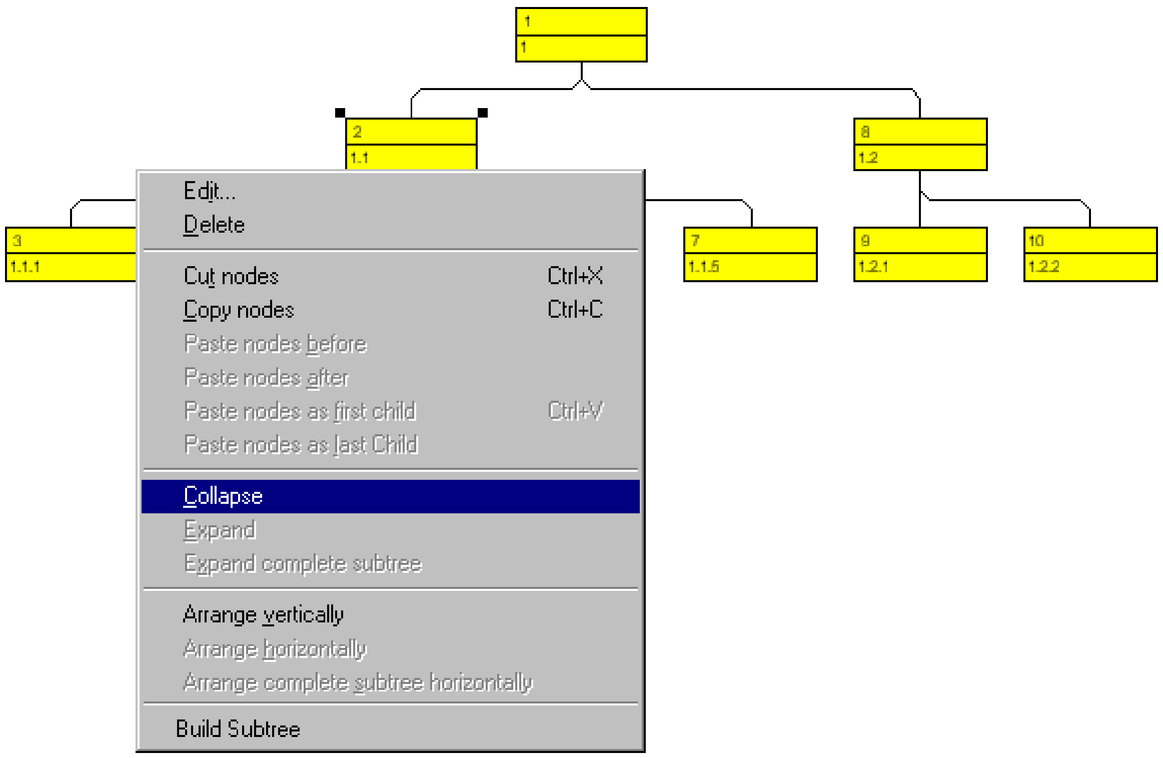
This chapter is about collapsing and expanding subtrees. A subtree can be collapsed, minimizing its extent to the top node of the subtree, to be expanded and displayed in its full size again. The top node of a subtree is called "structure node". It will remain visible while any other node of the subtree disappears when the subtree is collapsed.

Collapsing subtrees helps to keep complex trees well structured. It enables you to focus on certain parts of a structure while others can visually disappear. Because the information on the structure of collapsed trees is saved, no part of the total structure will be lost.

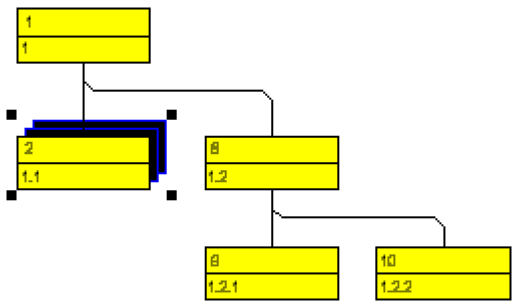
Please run the program now using the data file *tutorial.tre*. (Please see "Tutorial: Loading Data from a File".)



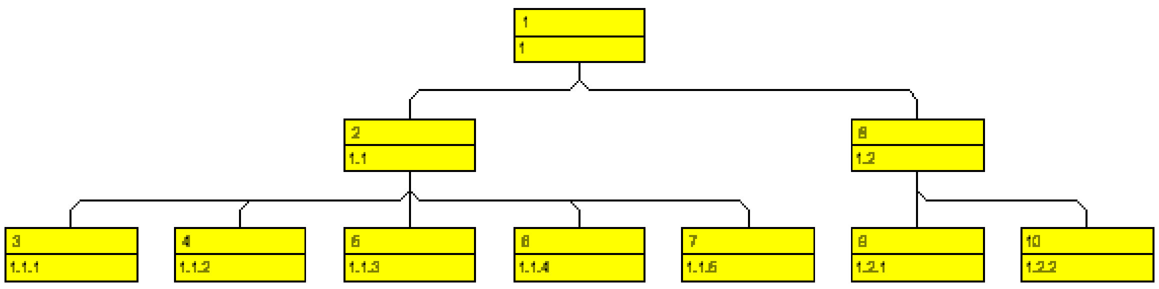
Please mark the first node on the second level and pop up the context menu by clicking on the right mouse button.



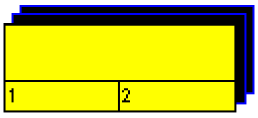
Select the **Collapse** menu item. This will collapse all subtrees that belong to the marked node. It will turn into the structure node, representing the hidden subtree.



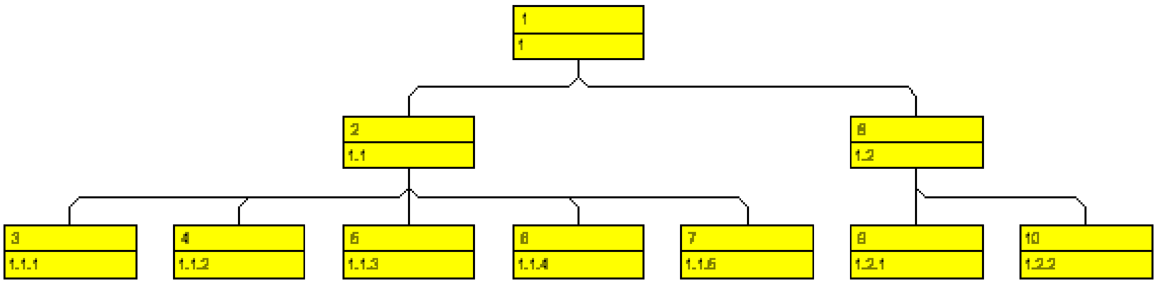
Please select the menu item **Expand** from the context menu now, which lets you expand the subtree collapsed. Only the marked structure node will be expanded. Collapsed structure nodes further down the subtree will remain collapsed.



You can expand the subtree including all collapsed structure nodes further down by the menu item **Expand complete subtree**. To test this command, please collapse the first node on the second level and then the node on the first level.

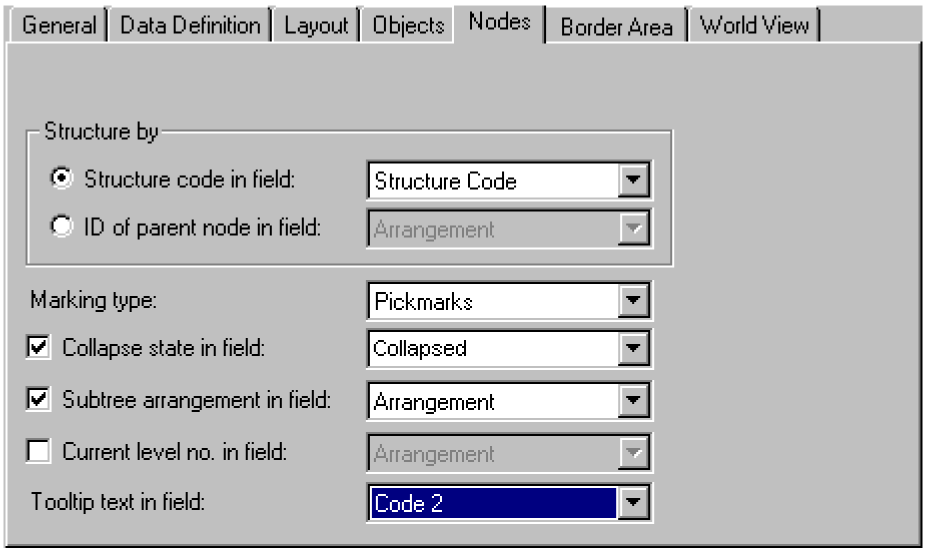


Then select the menu item **Expand complete subtree** to expand the subtree completely.



▸ Store "Collapse State" to data field

You can store to a data field, whether the subtree of a node is collapsed or expanded ("collapse state" of the node). Please return to design mode and open the property page **Nodes**.



Structure by
(●) Structure code in field: Structure Code
( ) ID of parent node in field: Arrangement
Marking type: Pickmarks
[✓] Collapse state in field: Collapsed
[✓] Subtree arrangement in field: Arrangement
[ ] Current level no. in field: Arrangement
Tooltip text in field: Code 2

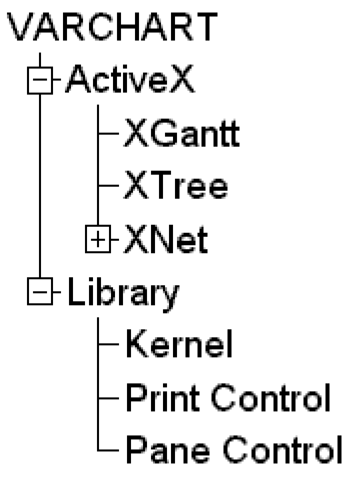
Activate the check box **Collapse state in field** and select the data field "Collapsed" from the combo box associated. The collapse state will now be continously stored (synchronized) to the "Collapsed" data field. The field may contain the values "0" (node expanded) or "1" (node collapsed).

Run the program. Mark a node and double-click the left mouse button on the node to pop up the **Edit Data** dialog. If the subtree of the node is collapsed, the "Collapsed" data field will contain the value "1". If the subtree of the node is expanded, the field will contain the value "0".

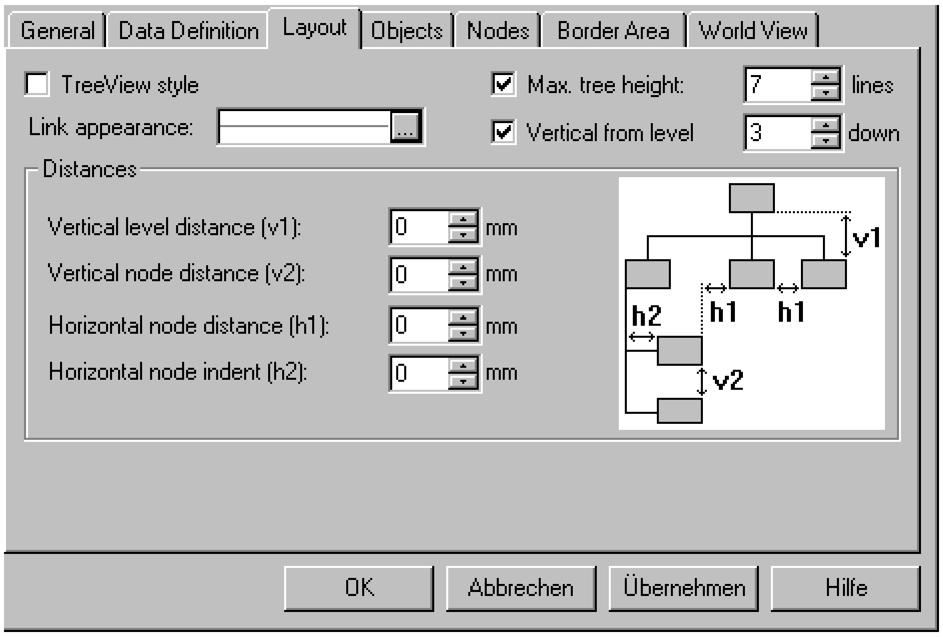
2.17 TreeView Style

This paragraph is about how to arrange nodes in TreeView style. Similar to the appearance of the Microsoft Explorer directory tree, the tree view style lets you add a plus or minus symbol to vertically arranged node levels. The plus symbol indicates that the subtree of this node is collapsed, the minus symbol indicates that it is expanded. The symbols are set to those nodes only that are no leave nodes, i.e. that do have child nodes. Clicking on a plus symbol will expand a tree and transform the symbol into a minus. Clicking on a minus symbol will collapse the tree and transform the symbol into a plus.

Example of a tree displayed in TreeView style:



To activate the TreeView style, tick the **TreeView style** check box on the **Layout** property page.



☐ TreeView style

Link appearance:

☑ Max. tree height: 7 lines

☑ Vertical from level 3 down

Distances

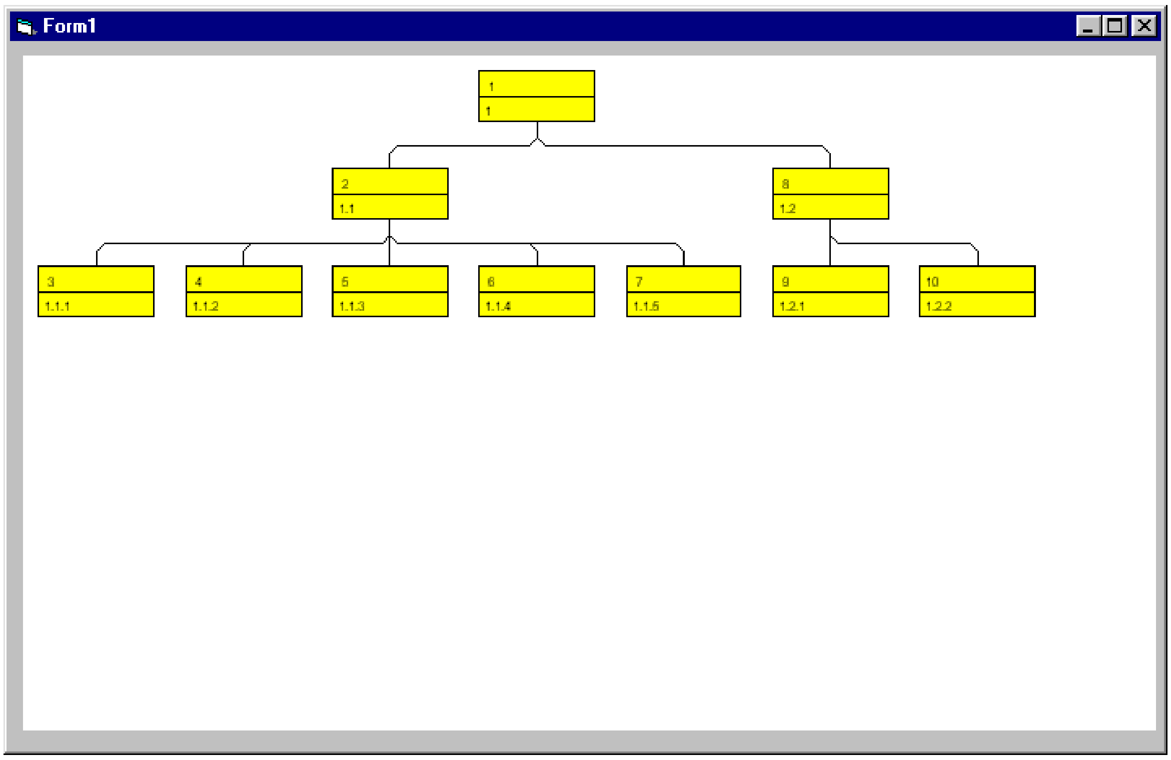
Vertical level distance (v1): 0 mm

Vertical node distance (v2): 0 mm

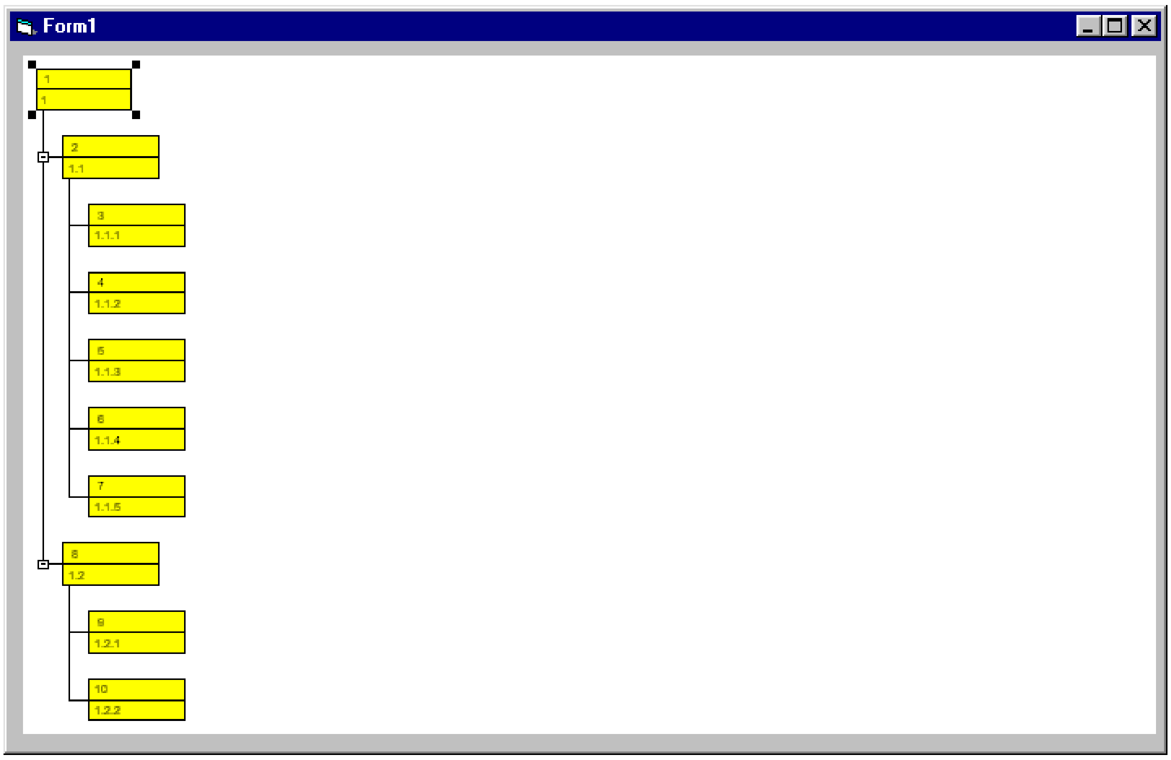
Horizontal node distance (h1): 0 mm

Horizontal node indent (h2): 0 mm

Please deactivate the check box and run the program. The tree will not be displayed in TreeView style:

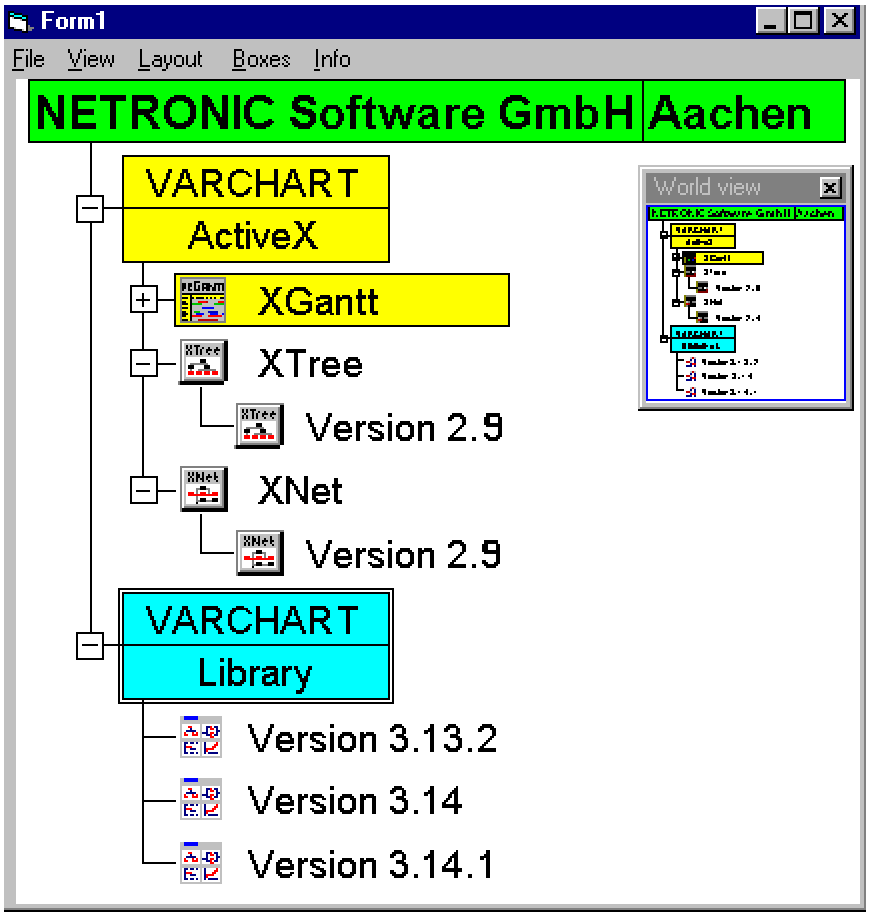


Now change to design mode and activate the **TreeView style** check box on the **Layout** property page to have the nodes arranged in TreeView style:

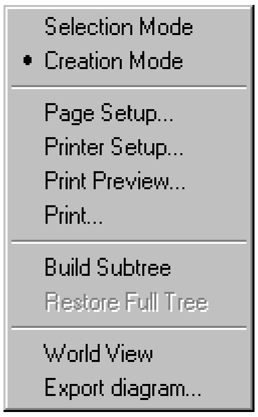


2.18 Using the Worldview

The world view is an additional window that shows the complete diagram. A frame shows the diagram section currently displayed in the main window. When you move the frame or change its size, the corresponding diagram section will be adapted as soon as you release the mouse button. Vice versa, the position or the size of the frame will be changed when you scroll or zoom the section in the main window.

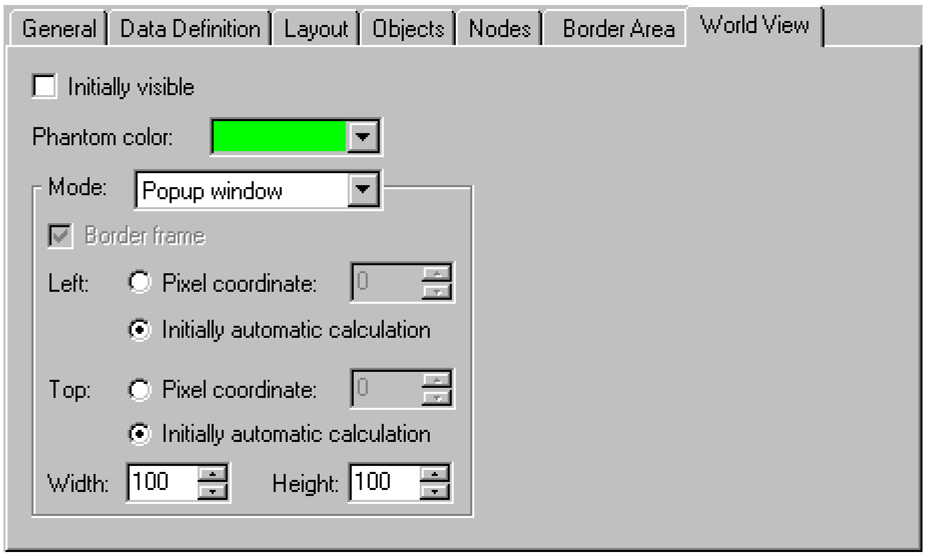


At runtime, you can switch on/off the world view via the item **World View** of the default context menu.



On the **World View** property page you can specify the following properties of the World View:

- World view initially visible
- Phantom color (color of the rectangle line)
- Mode of the World View (fixed at left/right/top/bottom side, Position not fixed or Popup window)
- Border frame
- possibly the position and extension of the World View



☐ Initially visible

Phantom color: 🟩

Mode: Popup window

☑ Border frame

Left: ○ Pixel coordinate: 0
[● Initially automatic calculation]

Top: ○ Pixel coordinate: 0
[● Initially automatic calculation]

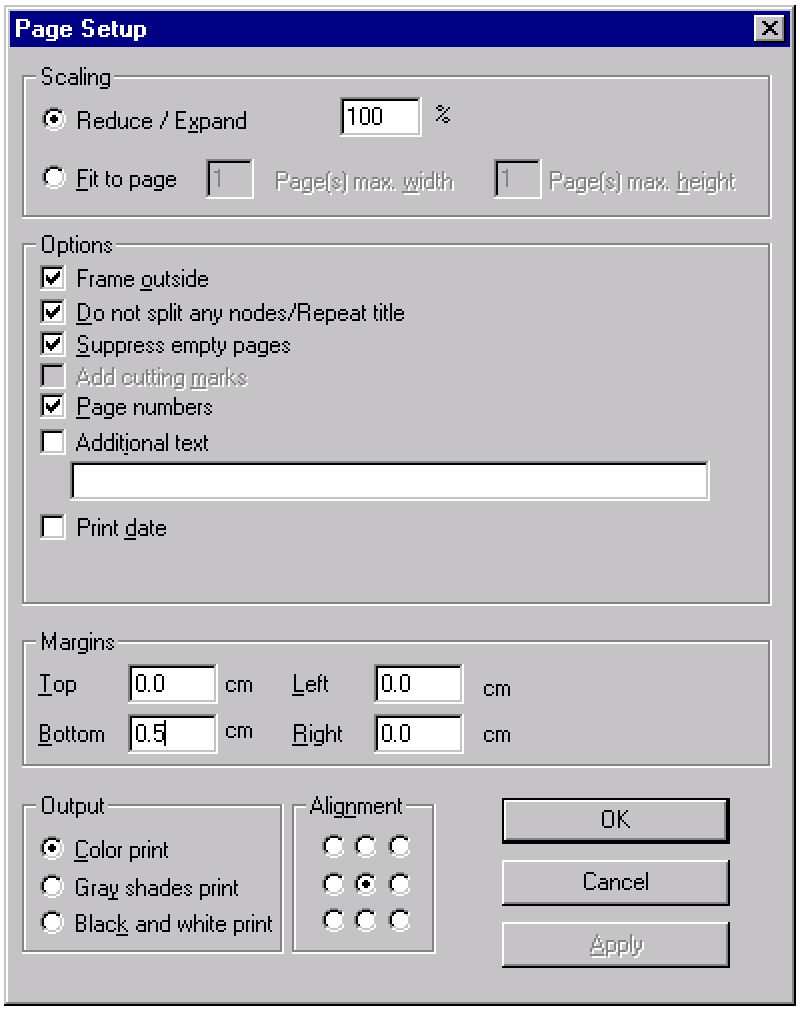
Width: 100 Height: 100

2.19 Setting up Pages for Printing

There are three ways to call the **Page Setup** dialog:

- By clicking with the right mouse key in the diagram area during runtime, you will open a context menu. Select **Page setup...** here.
- Select **Print Preview** in the context menu and there click on the **Page Setup...** button
- Use the VARCHART Windows forms control method **ShowPageSetupDialog**.

The following dialog will pop up:



Page Setup

Scaling
- [✓ Reduce / Expand] 100 %
- Fit to page 1 Page(s) max. width 1 Page(s) max. height

Options
- [✓] Frame outside
- [✓] Do not split any nodes/Repeat title
- [✓] Suppress empty pages
- [ ] Add cutting marks
- [✓] Page numbers
- [ ] Additional text
- [ ] Print date

Margins

Top	0.0	cm	Left	0.0	cm
Bottom	0.5	cm	Right	0.0	cm

Output
- [✓ Color print]
- Gray shades print
- Black and white print

Alignment: ○ ○ ○ / ○ ● ○ / ○ ○ ○

After selecting the **Reduce/Expand** radio button you can enter the scale factor into the associated field. 100% is equivalent to the original size; a smaller value correspondingly reduces the size of the diagram for the output, a greater value increases the size of the diagram for the output.

Activating the **Fit to page** radio button lets you specify the maximum number of pages, both width-wise and height-wise, that the diagram may be split into for the output. If necessary, one of the two values may be ignored in order to print the diagram as large as possible while preventing it from being distorted.

Beside, you can specify

- whether a frame is to be drawn around the diagram (**Frame outside**)
- whether nodes of a partitioned diagram are to be split and whether a title and a legend are to be added to each page (**Do not split any nodes/Repeat title**)
- whether empty pages are to be printed (**Suppress empty pages**) (only activated if **Do not split any nodes/Repeat title** has been ticked)
- whether cutting marks are to be added to each page for reassembling them after printing (**Add cutting marks**) (only activated if you have opted to output the diagram on more than a single page and if the check box **Do not split any nodes/Repeat title** has not been ticked)
- whether **Page numbers** and probably **Additional text** are to be added
- 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:
- **Additional text**:

{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

- whether the date of printing will be printed in the bottom left corner (**Print date**)
- the width of **Margins**
- the type of output (**Color print, Gray shades print** or **Black and white print**)
- the **Alignment** of the diagram.

2.20 Printing the Diagram

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

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

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

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

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

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

In the **Page Setup** dialog you can specify e.g. the scaling, whether the pages shall be numbered, the margins, the alignment etc. For further information see chapter 5.26 The "Page Setup" Dialog.

2.21 Exporting a Diagram

Your diagram can be exported as a graphics file as follows:

- 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 **GraphicExport** or **GraphicExportToFile**.

Available formats:

- *.BMP (Microsoft Windows Bitmap)
- *.EPS (Encapsulated Postscript)
- *.GIF (Graphics Interchange Format) (because of the used export library file names with any unicode characters whatever are not possible)
- *.JPG (Joint Photographic Experts Group) (because of the used export library file names with any unicode characters whatever are not possible)
- *.PCX (Microsoft Paint) (because of the used export library file names with any unicode characters whatever are not possible)
- *.PNG (Portable Network Graphics) (because of the used export library file names with any unicode characters whatever are not possible)
- *.TIF (Tagged Image File Format) (because of the used export library file names with any unicode characters whatever are not possible)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile)

EPS, VMF and WMF are vector formats that allow to store the file independent of pixel resolution. All other formats are pixel-orientated and confined to a limited resolution. Files of the VMF format can be displayed via the NETRONIC WebViewer on any platform by Java-compatible internet browsers. For more information please see www.netronic.de.

2.22 Saving the Configuration

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

A configuration consists of two files with the same name but two different endings, an ini- and an ifd-file, which are both indispensable.

How to save your current configuration:

In the 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 you click on **Apply**, the ini-file will be created and linked to the VARCHART ActiveX instance.

How to load a saved configuration:

In the 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 now on be linked to the VARCHART ActiveX instance. All current settings will expire irrevocably.

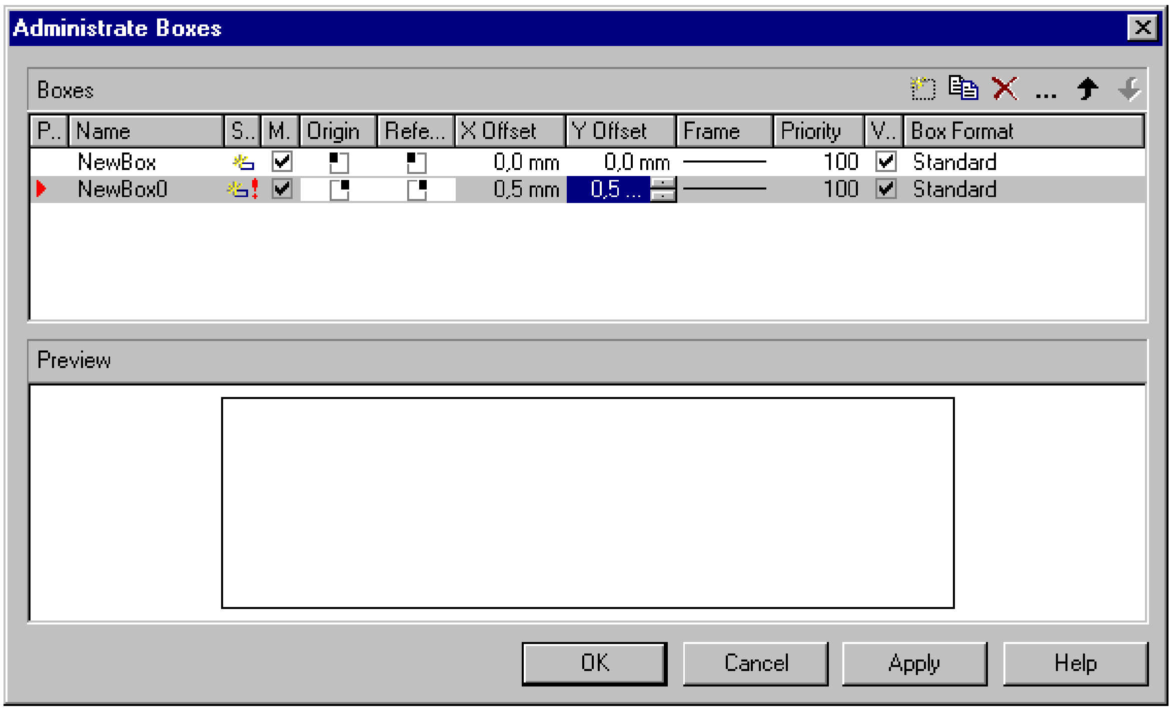
Note: The settings of the configuration file are loaded once only .The VARCHART ActiveX will not read them for a second time from the same file. Instead, the settings will be loaded from the internal storing, which are the same as those in the configuration file.

Thus, modifying the data of the configuration file with an editor will not be effective. If you do want VARCHART ActiveX to accept a modified configuration file, you have to rename the modified *ini* file and the corresponding *ifd* file and specifiy the name of the modified *ini* file on the **General** property page under **Configuration file**.

3 Important Terms

3.1 Boxes

In the diagram area, any number of boxes containing text or graphics can be displayed. Via the property page **Objects**, **Edit box** button you open the dialog **Administrate Boxes** where you can add, copy, delete or edit boxes..



P..	Name	S..	M.	Origin	Refe...	X Offset	Y Offset	Frame	Priority	V..	Box Format
	NewBox		✓	▪	▪	0,0 mm	0,0 mm	—	100	✓	Standard
▶	NewBox0	!	✓	▪	▪	0,5 mm	0,5 ...	—	100	✓	Standard

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

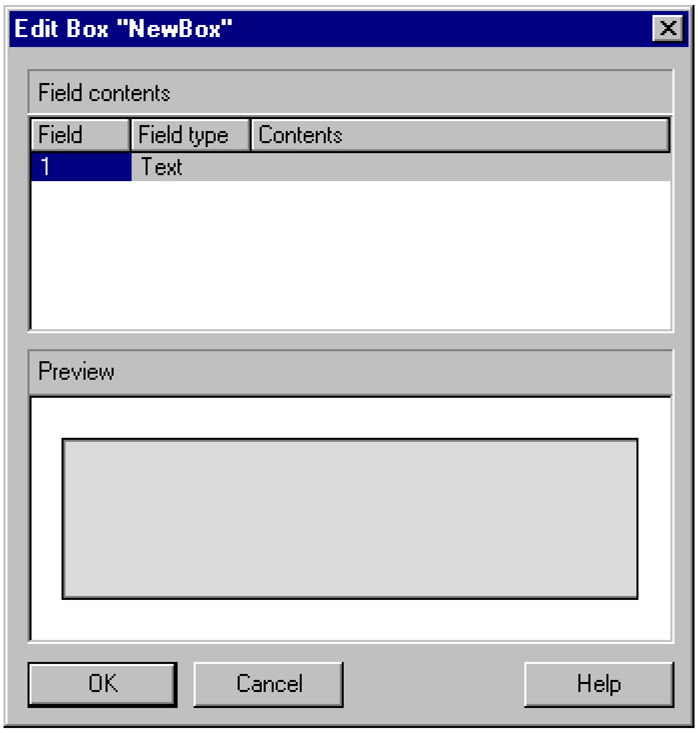
For each box you can specify the following:

- its name
- whether it is movable in the diagram at run time
- its origin (the point of the diagram from which the offset to the reference point of the box will be measured)
- its reference point, i. e. the point of the box from which the offset to the origin will be measured
- 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 relative priority in comparison with the other diagram objects (nodes, grids, etc.)
- whether it is visible
- its format

▸ Editing boxes

The **Edit Box** dialog lets you edit a box. This dialog box will appear at design time when you click the **Edit box** button in the **Administrate Boxes** dialog box. At run time it will appear when you double-click the box to be edited. You also can edit the texts of boxes directly at run time.



Field	Field type	Contents
1	Text	

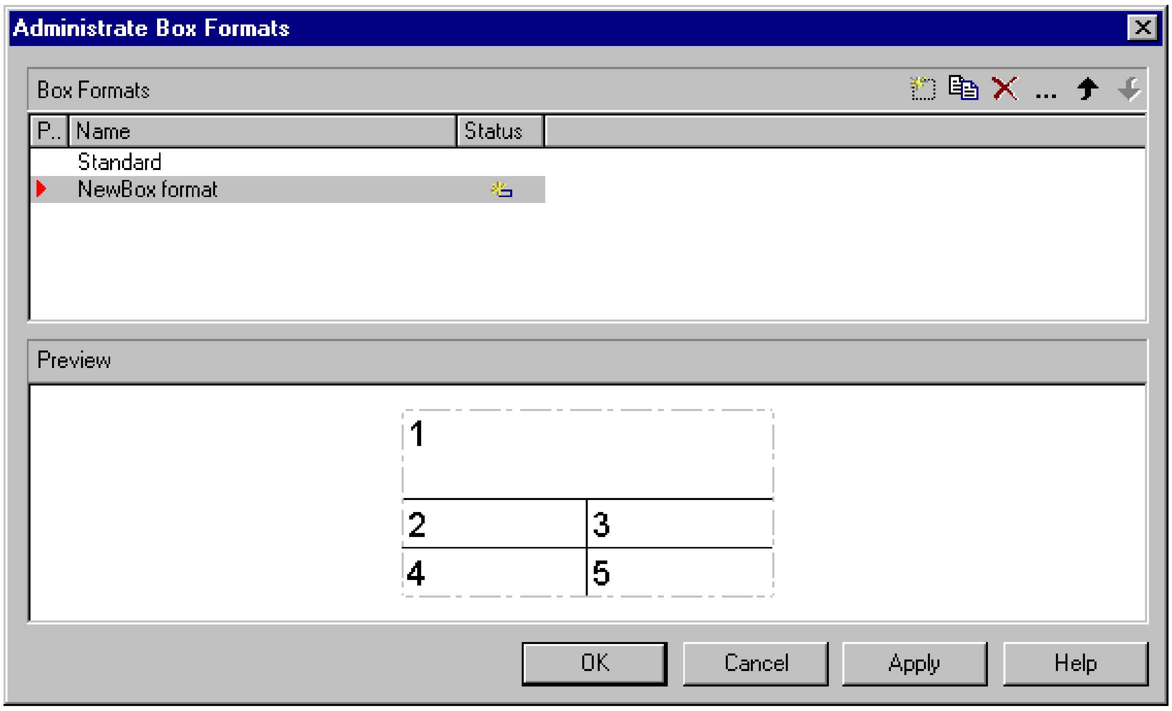
The **Field** column contains the numbers of the box fields. (The number of fields depends on the selected box format.)

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

You can type the contents of the field or a graphics file name into the **Content** column. If a text field contains more than one line, you can use "\n" in the text string to separate two lines of the text field (Example: "Line1\nLine2"). Otherwise the lines will be separated at blanks.

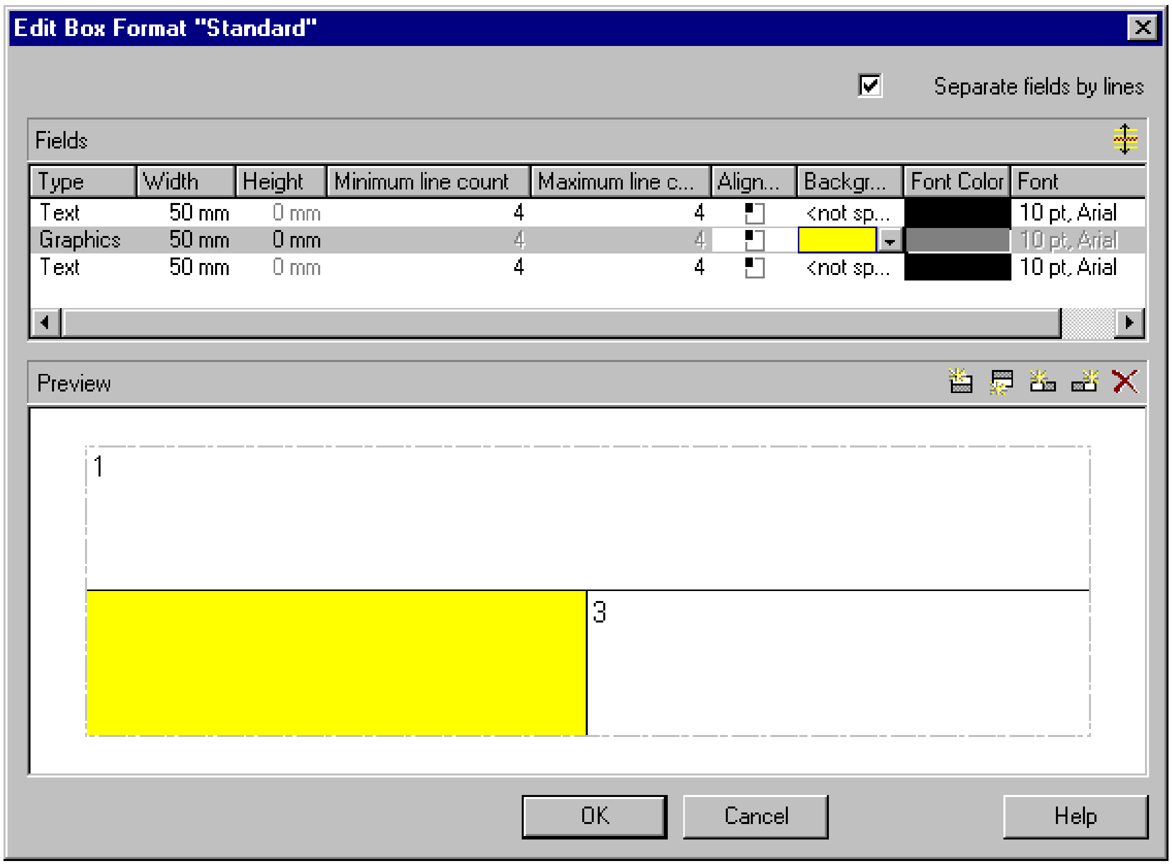
For each box you can select a box format, and you can specify the box formats.

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



P..	Name	Status
	Standard	
▶	NewBox format	

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



☑ Separate fields by lines

Type	Width	Height	Minimum line count	Maximum line c...	Align...	Backgr...	Font Color	Font
Text	50 mm	0 mm	4	4	◰	<not sp...	■	10 pt, Arial
Graphics	50 mm	0 mm	4	4	◰	🟨	■	10 pt, Arial
Text	50 mm	0 mm	4	4	◰	<not sp...	■	10 pt, Arial

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

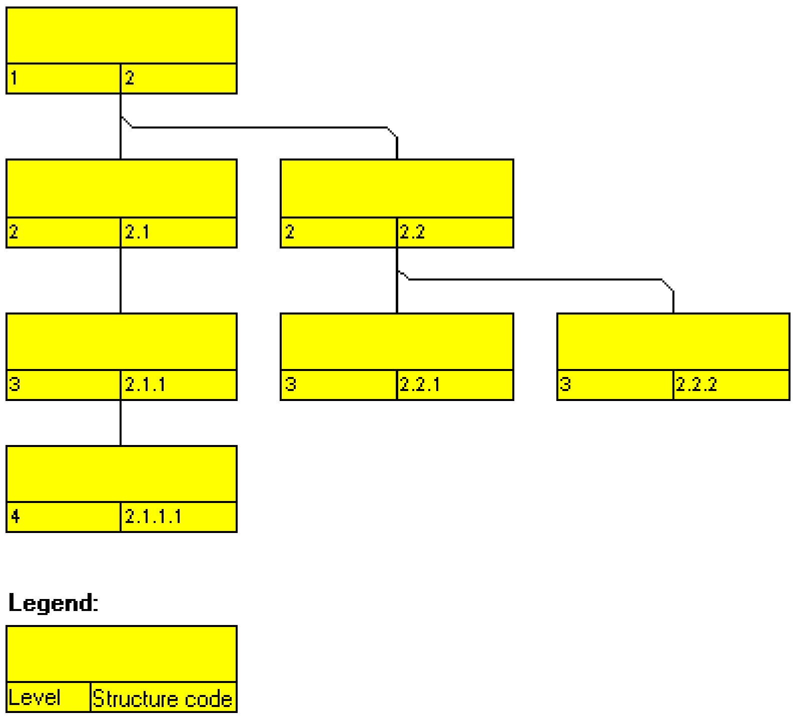
Furthermore you can specify for each box the following:

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

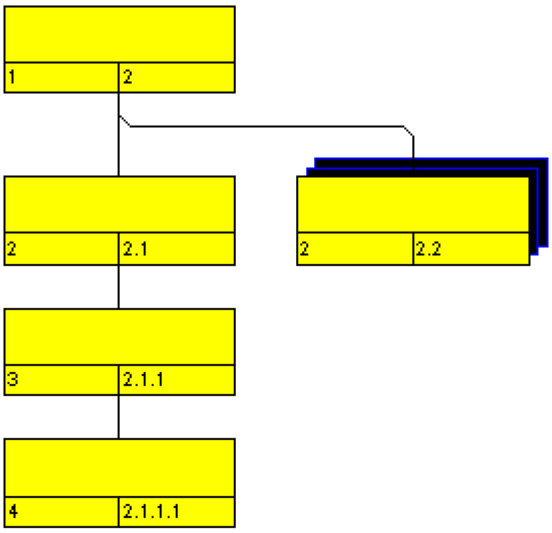
3.2 Collapsing and Expanding

A subtree can be collapsed, minimizing its extent to the top node of the subtree, and expanded and displayed in its full size again. The top node of a subtree is called "structure node". It will remain visible while any other node of the subtree disappears when the subtree is collapsed.

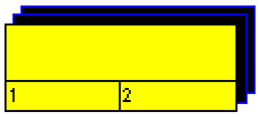
Collapsing subtrees helps to keep complex trees well structured. It enables you to focus on certain parts of a structure while others can visually disappear. Because the information on the structure of collapsed trees is saved, no part of the total structure will be lost.



Expanded tree



Subtree collapsed to the structure node



Completely collapsed tree

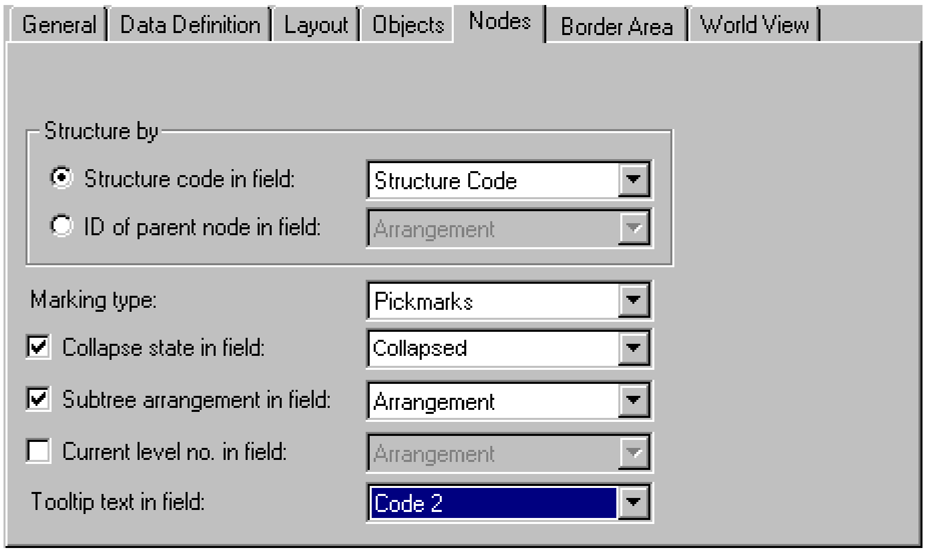
The menu item **Collapse** of the context menu of a node lets you collapse the subtrees that depend on the marked structure nodes. The structure nodes then represent the hidden subtrees.

The menu item **Expand** lets you expand the subtrees that are represented by the marked structure nodes. Only the collapsed nodes will be expanded. Collapsed structure nodes further down the subtree will remain collapsed.

You can expand the subtree including all collapsed structures further down by the menu item **Expand complete subtree**.

► Storing the Collapse State of a Node to a Data Field

You can store the collapse state of a node, that is whether the node is collapsed or expanded, to a data field. Activate the **Collapse state in field** check box and select a data field (e.g. the field **Collapsed**) from the combo box.



Structure by

- [✓ Structure code in field:] Structure Code
- ID of parent node in field: Arrangement

Marking type: Pickmarks

[✓] Collapse state in field: Collapsed

[✓] Subtree arrangement in field: Arrangement

[ ] Current level no. in field: Arrangement

Tooltip text in field: Code 2

The data field will now continuously store the state of the node and may contain "0" for a node to be expanded, or "1" for a node to be collapsed.

3.3 Data

Nodes can be generated interactively or via the API. If you generate them via the API, you can either load their data by a file via the method **Open** or you can create new ones by using the method **InsertNodeRecord**. A node record is passed as a string or in a Variant array, with its data fields defined according to the settings of the data definition. 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

```
VcTree1.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 node by **NodeCollection**.

3.4 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 in the program code of your application.

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

Note: By means of the events, via the **returnStatus** parameter 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 following table contains the return status values of VARCHART ActiveX events:

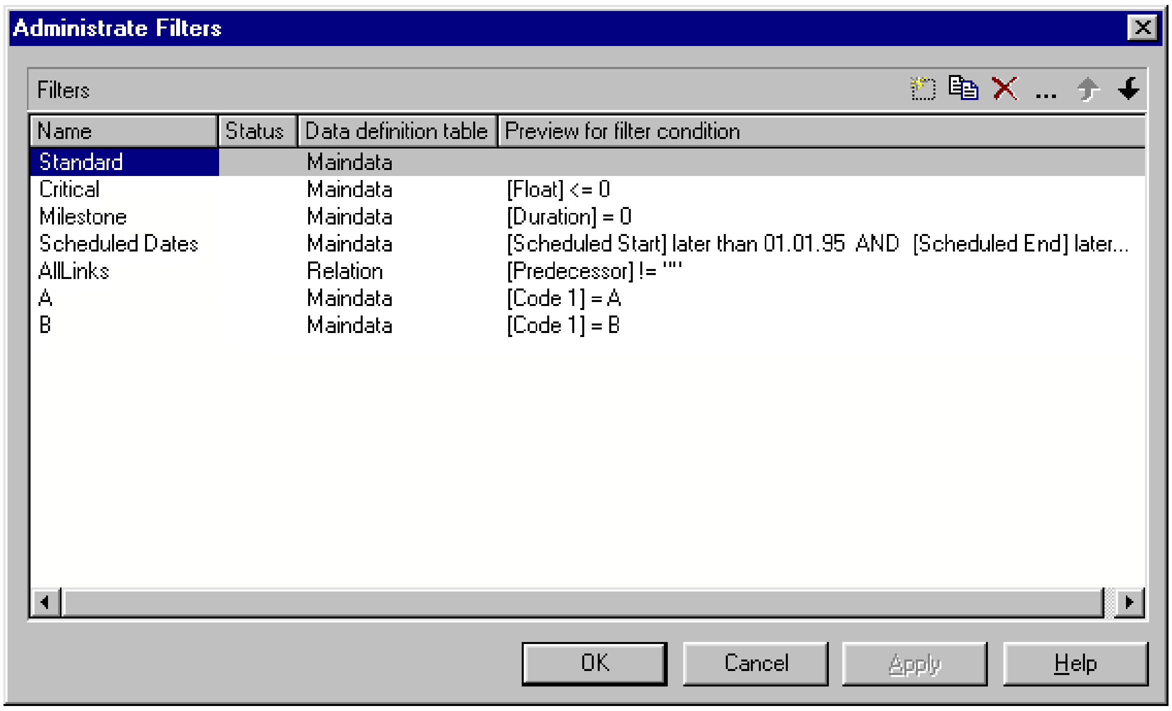
Constant	value	description
vcRetStatDefault	2	default value
vcRetStatFalse	0	revoking the action
vcRetStatNoPopup	4	revoking the popup menu

3.5 Filters

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

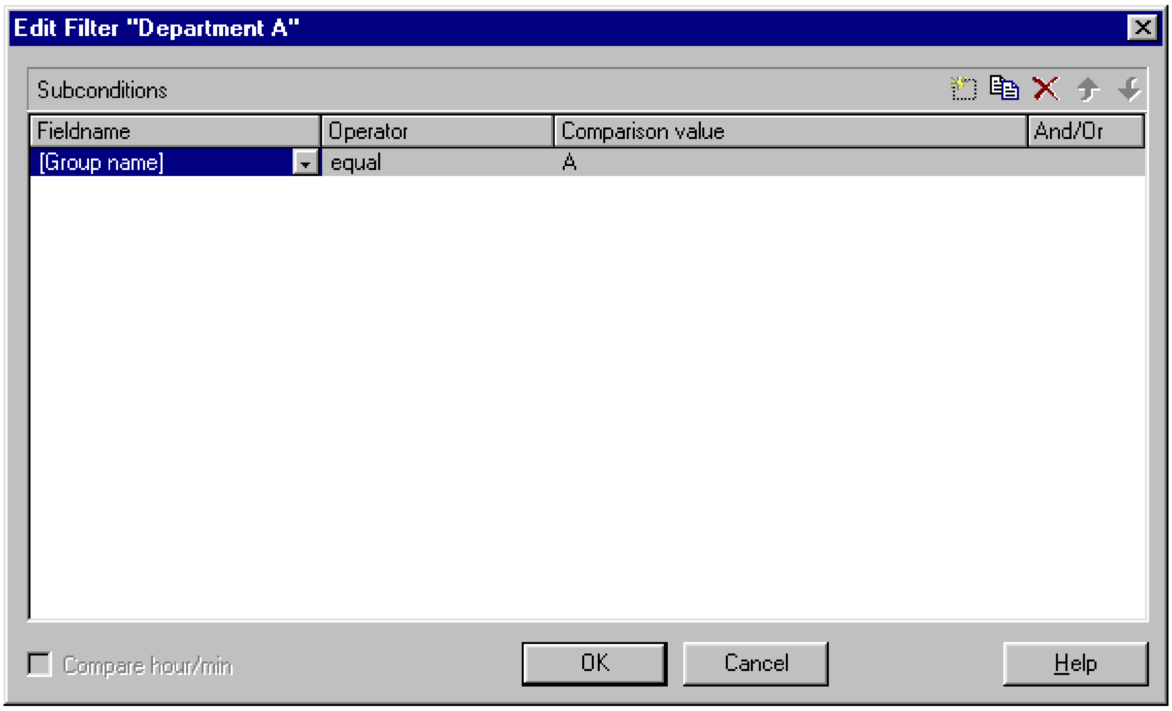
When you apply a filter, the data of the record is compared with the criteria of the filter. Those activities that fulfill the filter criteria will be selected. For example, you can define a filter "Nodes of Department A".

Filters can only be handled in design mode. You can get to the **Administrate Filters** dialog box via the **Objects** property page. Use the **Administrate Filters** dialog box to rename, create, copy, delete or edit filters.



Name	Status	Data definition table	Preview for filter condition
Standard		Maindata	
Critical		Maindata	[Float] <= 0
Milestone		Maindata	[Duration] = 0
Scheduled Dates		Maindata	[Scheduled Start] later than 01.01.95 AND [Scheduled End] later...
AllLinks		Relation	[Predecessor] != ""
A		Maindata	[Code 1] = A
B		Maindata	[Code 1] = B

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



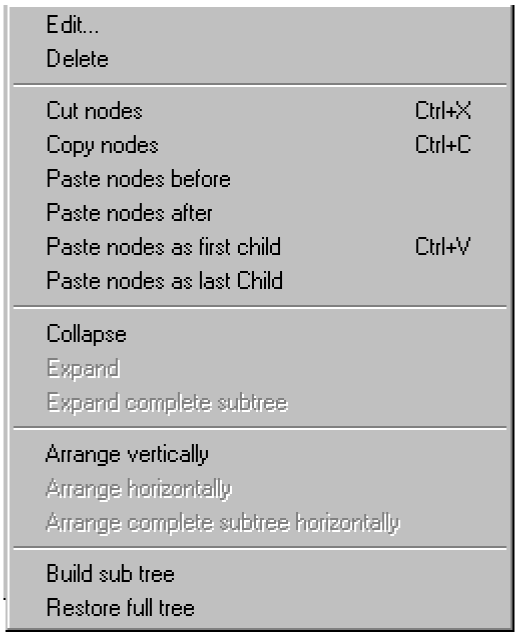
Fieldname	Operator	Comparison value	And/Or
[Group name]	equal	A	

3.6 Horizontal/Vertical Arrangement

If you display your tree diagram for the first time, it may not show its optimum layout. You will obtain it by an appropriate combination of horizontal and vertical subtrees, that will make the tree look more compact.

- *Horizontal arrangement:* Horizontally arranged subtrees will reduce the height of a tree diagram. All nodes of a level will be placed next to each other. The ports, i.e. the places where links join the nodes, will be placed in the center of the bottom line of a parent node, and in the center of the top line of a child node.
- *Vertical arrangement:* Vertically arranged subtrees will reduce the width of a tree diagram. All nodes of a level and its sublevels will be placed beneath each other. The ports will be placed in the bottom left corner of the parent node, and in the center of the left left of the child node.

Menu items to set arrangements will be at your disposition after marking a node and pressing the right mouse button. In the context menu popping up only activated commands are available at this time.

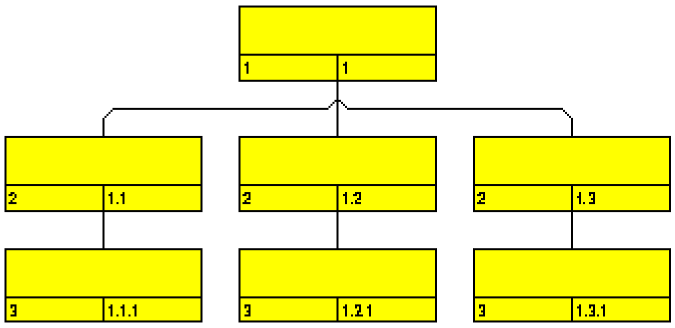


To arrange subtrees horizontally, please mark the top node(s) of these subtrees and select the context menu item **Arrange horizontally**. The first level of the subtree will be arranged horizontally whereas the next levels will not be influenced.

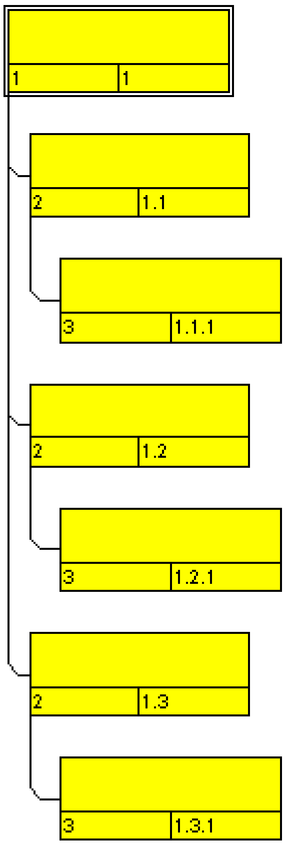
If you wish all levels of the subtree to be arranged horizontally, please select **Arrange complete subtree horizontally**.

By selecting the menu item **Arrange vertically**, all subtrees will be arranged vertically, starting by the first parent node marked.

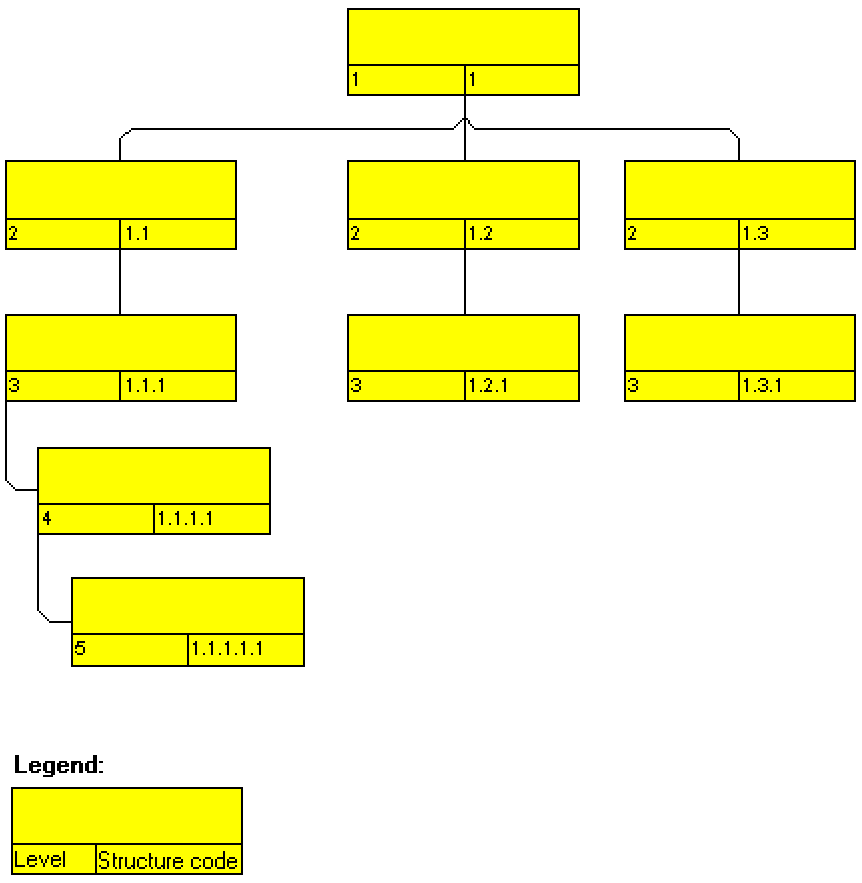
Note: If not all levels of a subtree have been arranged vertically, please check the maximum height of the tree set on the **Layout** property page. The number of levels is limited by the **Max. tree height** check box and field.



All levels arranged horizontally



All levels arranged vertically



Example of a tree diagram with vertically and horizontally arranged subtrees.

Note: Modifications of the arrangement settings will also apply to collapsed subtrees.

► Subtree arrangement in field

On the **Nodes** property page, activate the check box **Subtree arrangement in field** to keep the orientation of a subtree stored to a field. The data field may contain "0" for a subtree arranged horizontally, or "1" for a subtree arranged vertically. A horizontal arrangement of the subtree can be visible only if the parent node is a part of a vertical arrangement.

Alternatively, you can use the VcTree property **ArrangementField** to trace the arrangement of a subtree in a data field.

► Vertical from level

If you tick the check box **Vertical from level** on the **Layout** property page, all nodes from the level selected will be arranged vertically. To trigger the setting, the API method **Arrange** needs to be invoked.

The VcTree property **FirstVerticalLevel** lets you set/enquire the level, from that on the nodes are arranged vertically. If set to "-1", the property is disabled.

3.7 Identification

On the **DataDefinition** property page, please select fom the **Maindata** table a data field to be used for the identification of nodes (**Identification by**).

The identification of nodes serves to refer nodes correctly to a data base that works in the background.

Identifications are used to access nodes by the method **GetNodeByID**.

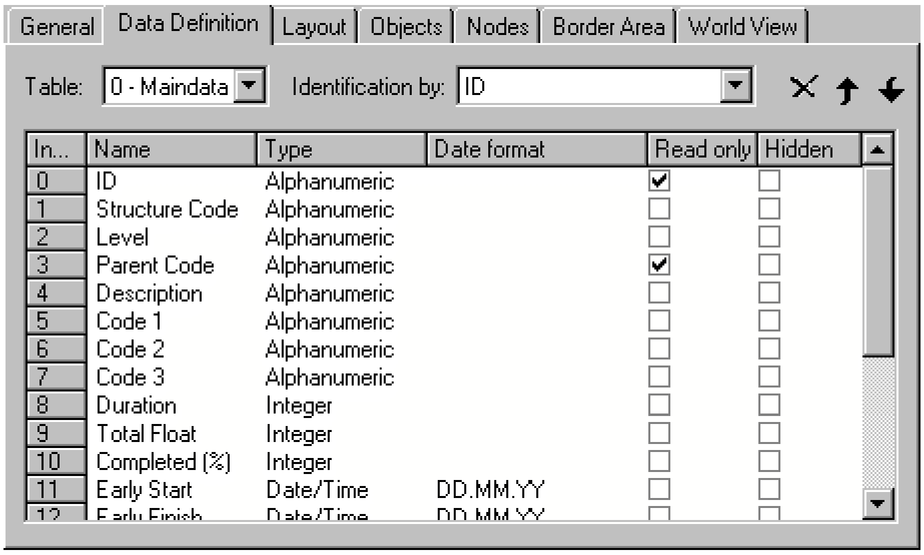


Table: 0 - Maindata

Identification by: ID

In...	Name	Type	Date format	Read only	Hidden
0	ID	Alphanumeric		✓	☐
1	Structure Code	Alphanumeric		☐	☐
2	Level	Alphanumeric		☐	☐
3	Parent Code	Alphanumeric		✓	☐
4	Description	Alphanumeric		☐	☐
5	Code 1	Alphanumeric		☐	☐
6	Code 2	Alphanumeric		☐	☐
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	☐	☐

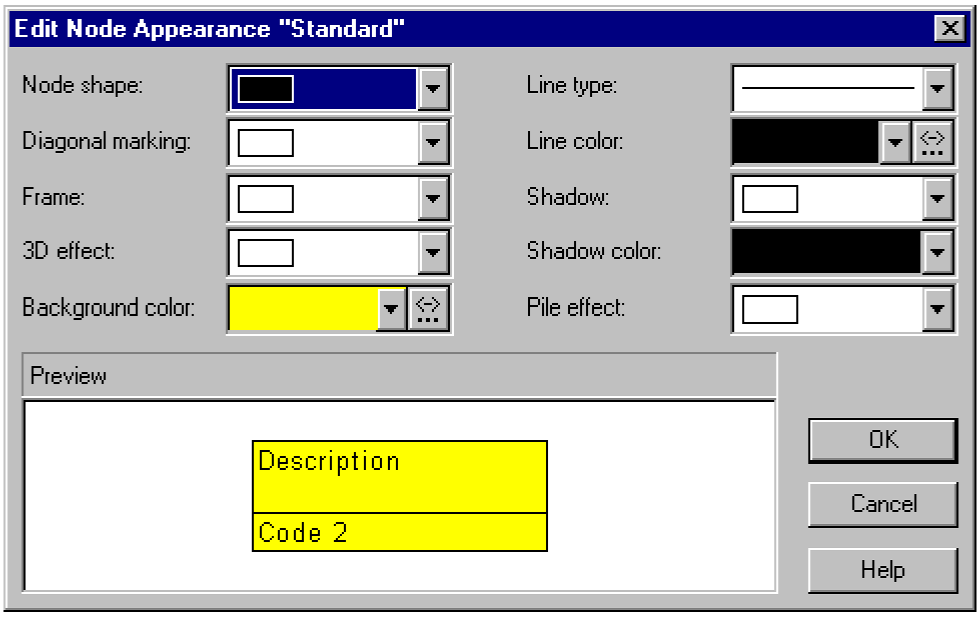
3.8 Maps

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

► Node Appearance in Dependence on Node Data

For each node appearance the background color and the line color can be assigned in dependence on the node data via a map.

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

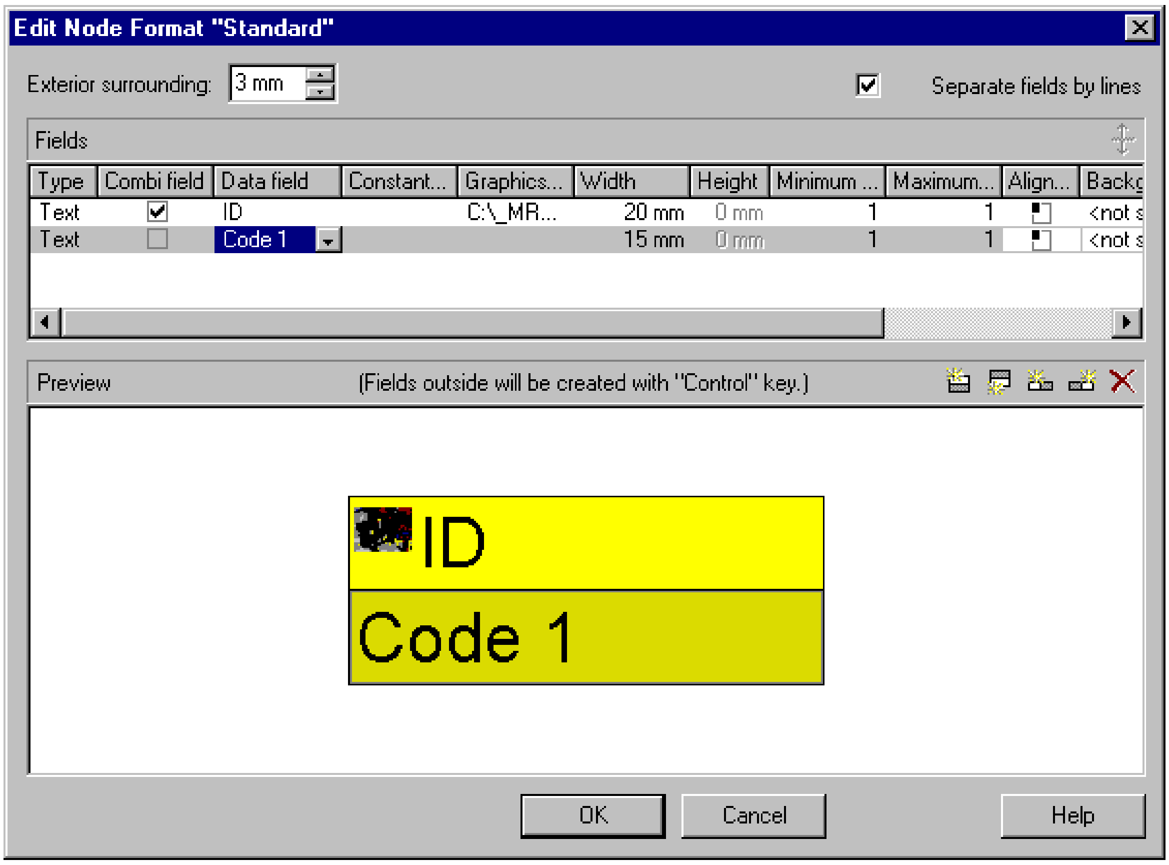

Edit Node Appearance "Standard"

Node shape:
Diagonal marking:
Frame:
3D effect:
Background color:
Line type:
Line color:
Shadow:
Shadow color:
Pile effect:
Preview
Description
Code 2

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

► Graphics file for node formats in dependence on node data

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


Edit Node Format "Standard"
Exterior surrounding: 3 mm
Separate fields by lines
Fields
Preview
(Fields outside will be created with "Control" key.)
ID
Code 1
OK
Cancel
Help

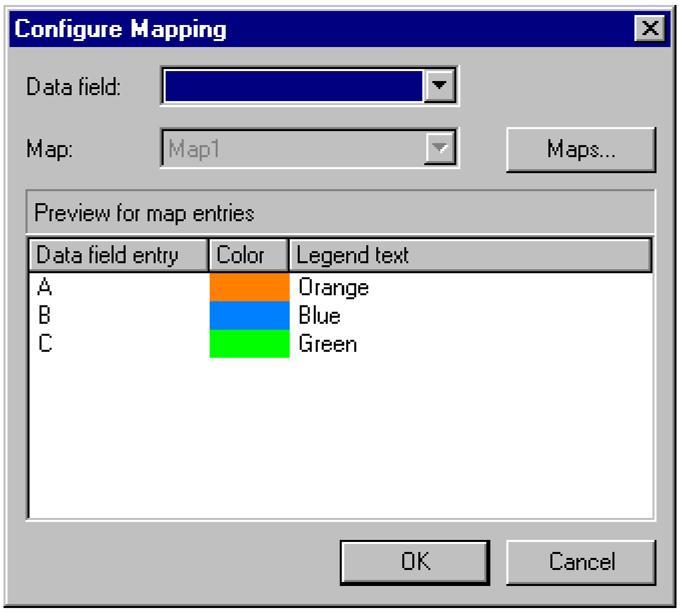
Type	Combi field	Data field	Constant...	Graphics...	Width	Height	Minimum ...	Maximum...	Align...	Backg
Text	✓	ID		C:_MR...	20 mm	0 mm	1	1		<not s
Text		Code 1			15 mm	0 mm	1	1		<not s

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

If a mapping has been configured, a symbol (↔) is 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.



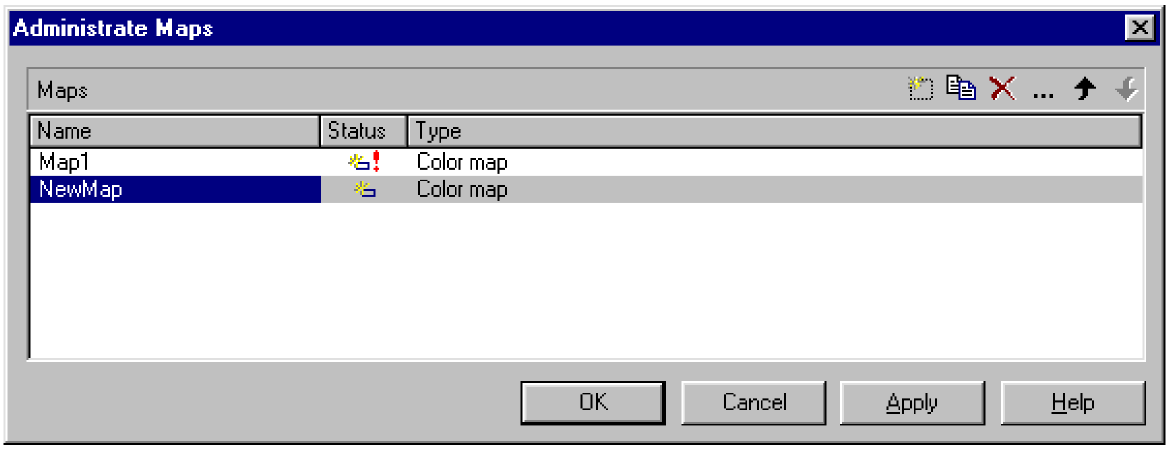
Data field entry	Color	Legend text
A	🟧	Orange
B	🟦	Blue
C	🟩	Green

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

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

▸ Administration of Maps

To create a new map or to edit a map, click on the **Maps** button. Then the **Administrate Maps** dialog box will open. This dialog box lets you create, copy, edit or delete maps.

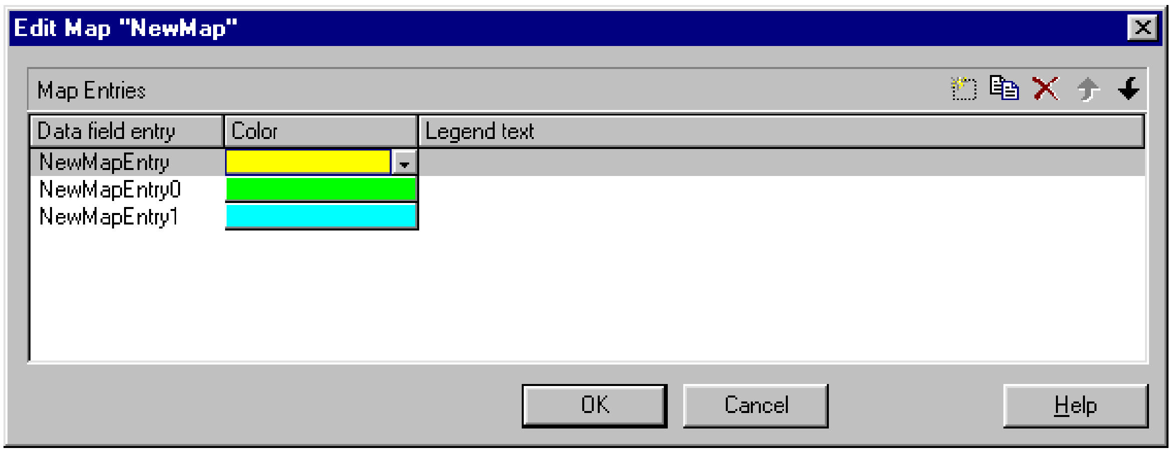


Name	Status	Type
Map1		Color map
NewMap		Color map

Alternatively, you can open this dialog box via the **Maps** button of the **Objects** property page.

▸ Editing Maps

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



Data field entry	Color	Legend text
NewMapEntry	🟨	
NewMapEntry0	🟩	
NewMapEntry1	🟦	

The **Map entries** table shows for each data field entry e.g. the color and the legend text assigned.

Via the buttons above the **Map entries** table you can create, copy or delete map entries or modify their position in the table.

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.

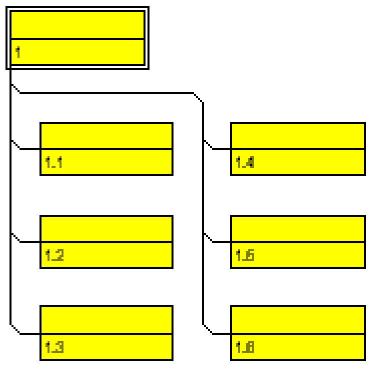
For further details please read the chapters "Property Pages and Dialog Boxes".

▸ Adjusting the Map during Runtime

You can adjust the map during runtime using VcMap methods, which lets the user modify your default settings via a dialog designed by you.

3.9 Maximum Height of the Tree Diagram

The total height of a tree diagram can be limited by the number of levels. For this, please activate the check box **Max. tree height** and enter the maximum tree height as number of levels. These settings will influence vertical arrangements only. If in a vertical branch more levels exist than set, another branch will be generated by the parent node to adopt the remaining child nodes.



Vertically arranged tree structure. Maximum height limited to 4 lines

The total height of a tree diagram can also be set/retrieved via the VcTree property **RowLimit**.

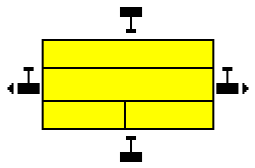
3.10 Node

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.

► Creating Nodes

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

Further nodes you can generate from the existing node by placing the cursor near it. The cursor will change its shape according to whether the new node is going to be a parent node, a child node or a brother node.



If on the **General** property page the check box **Edit new node** was ticked, the dialog box **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 Data** 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**.

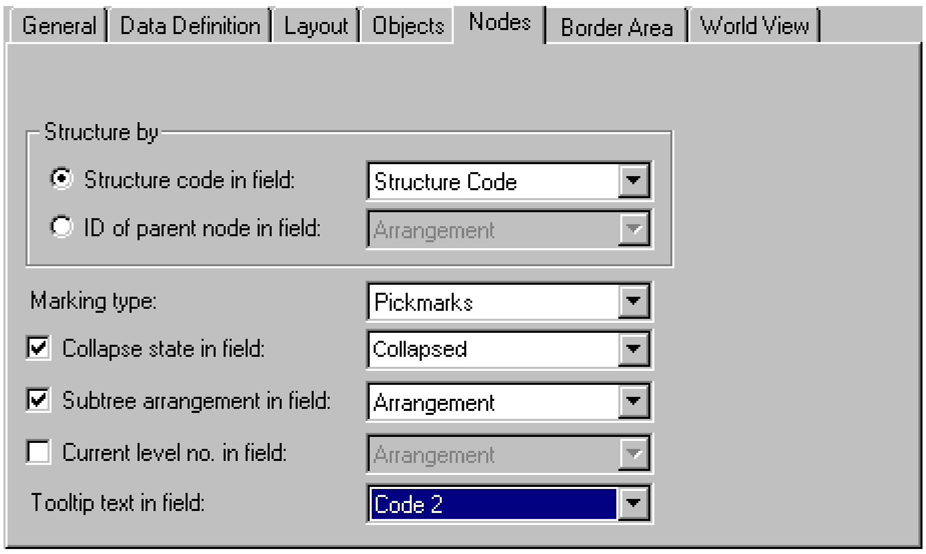
➤ Events

You can react to the events:

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

➤ Defining Data Fields for Tree Structures

You can set the data of the tree structure on the property page **Nodes**.



Structure by

[✓ Structure code in field:] Structure Code

ID of parent node in field: Arrangement

Marking type: Pickmarks

[✓] Collapse state in field: Collapsed

[✓] Subtree arrangement in field: Arrangement

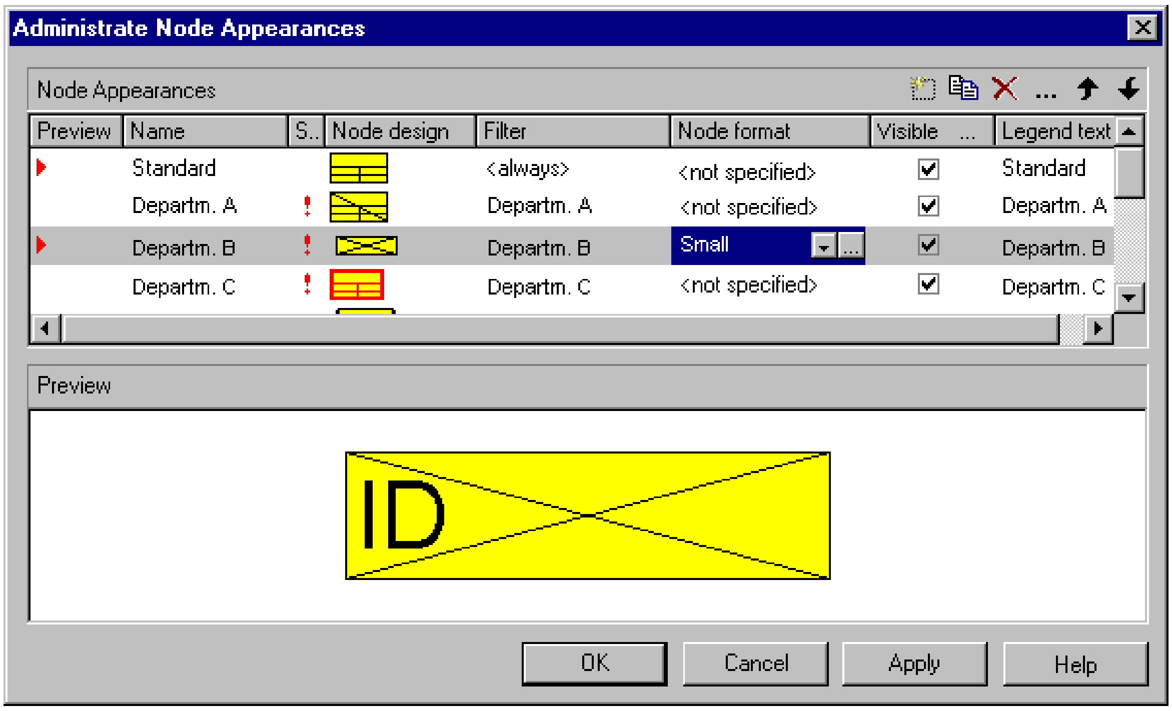
[ ] Current level no. in field: Arrangement

Tooltip text in field: Code 2

- If the tree structure is to be defined by a structure code, set the radio button to **Structure code in field** and select an appropriate data field for the structure code.
- If the tree structure is to be defined by the ID of the parent node, set the radio button to **ID of parent node in field** and select an appropriate data field for the ID of the parent node.
- If you wish to keep the state of collapsing/expanding of a node stored to a field, activate the **Collapse state in field** check box. The data field may contain "0" for an expanded node, or "1" for a collapsed node.
- Activate the **Subtree arrangement in field** check box to keep the orientation of a subtree stored to a field. The data field may contain "0" for a subtree arranged horizontally, or "1" for a subtree arranged vertically. A horizontal arrangement of the subtree will be visible only if the direct or indirect parent node is a part of a vertical arrangement.
- **Current level no. in field:** Activate this check box to specify the data field that contains the level number of nodes. The level numbers are counted from 0 upwards. At run time it is not possible to modify the level of a node by modifying the value of the level number data field.

3.11 Node Appearance

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


Administrate Node Appearances

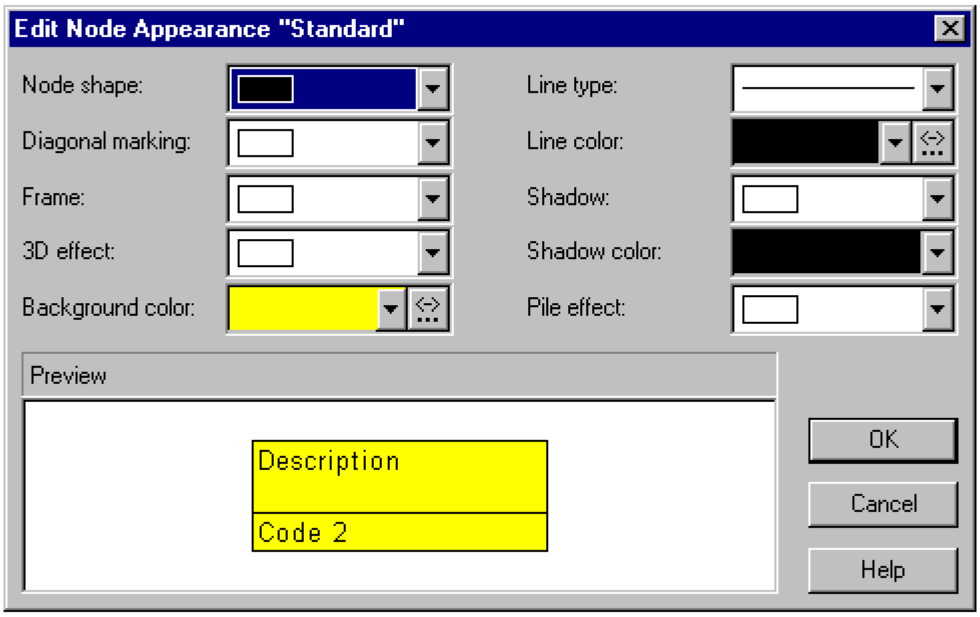
Node Appearances

Preview	Name	S..	Node design	Filter	Node format	Visible ...	Legend text
▶	Standard			<always>	<not specified>	✓	Standard
	Departm. A	!		Departm. A	<not specified>	✓	Departm. A
▶	Departm. B	!		Departm. B	Small	✓	Departm. B
	Departm. C	!		Departm. C	<not specified>	✓	Departm. C

Preview

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. Then the following 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.

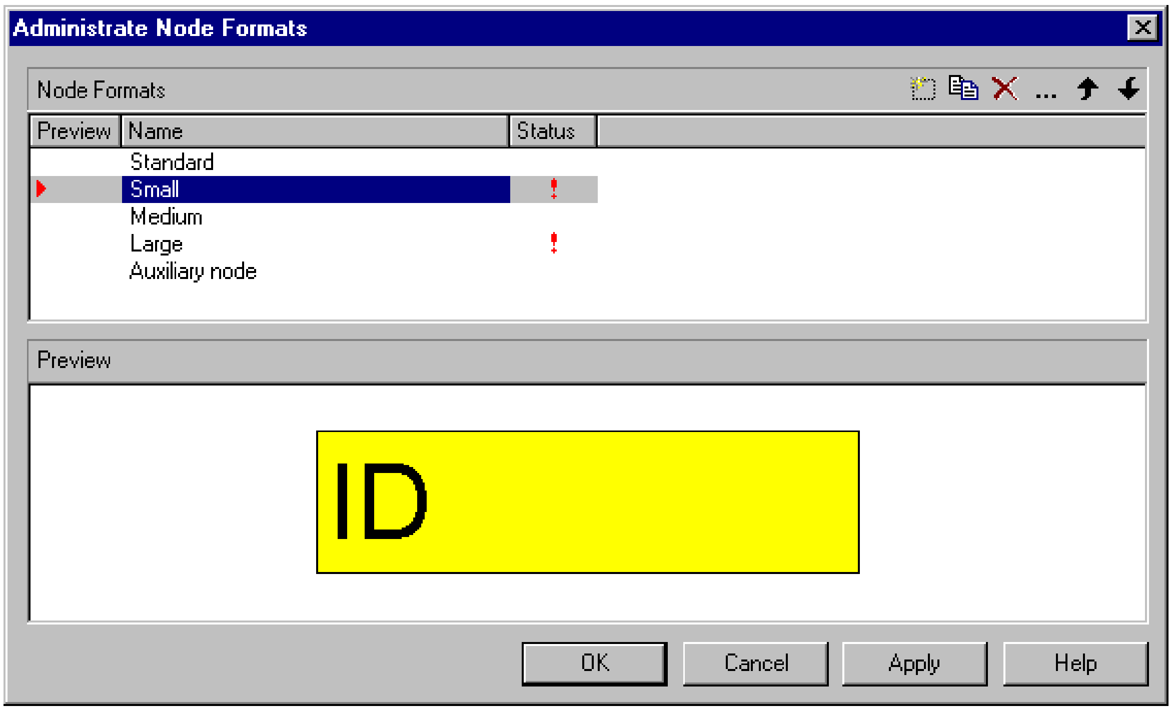
You can modify the order of working off the node appearances with the help of the arrow buttons.

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".

3.12 Node Format

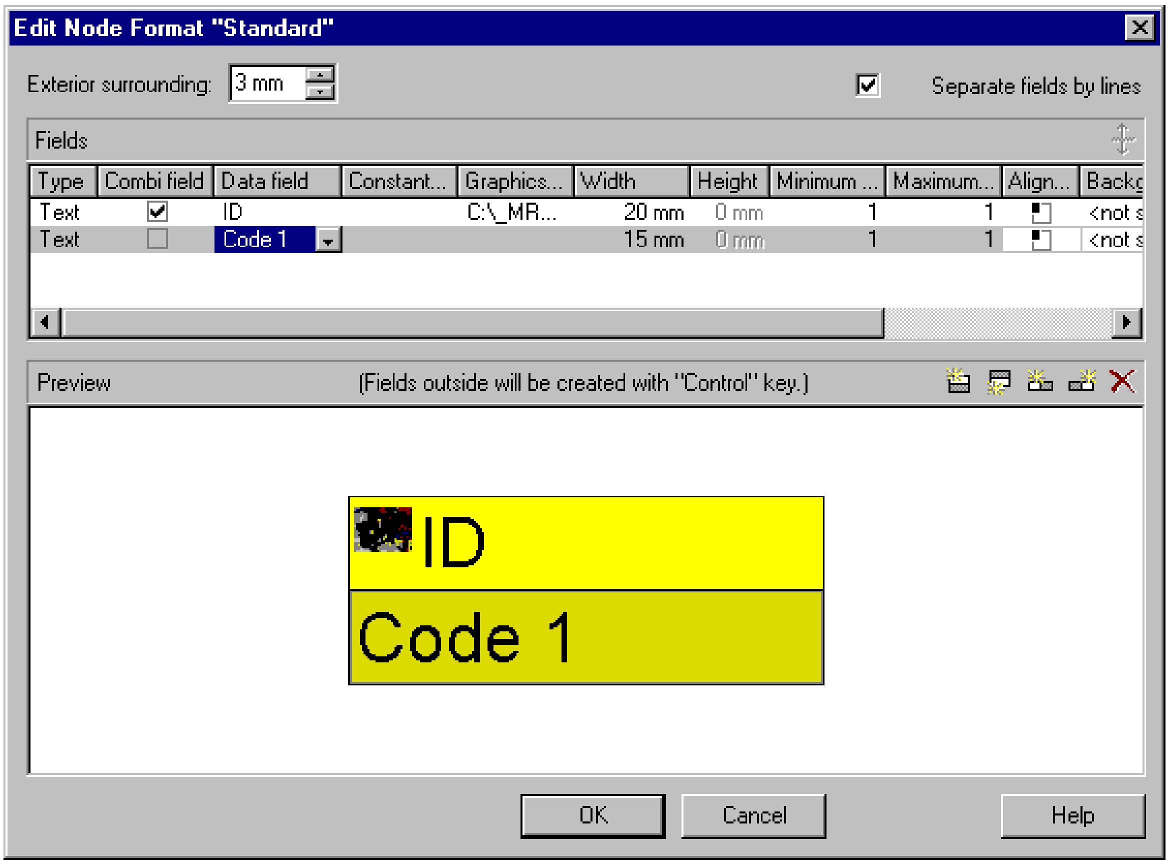
A node appearance always is combined with a node format. The **Node format** combo box in the **Administrate Node Appearances** dialog box lets you select the node format to be assigned to the node appearance.

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



Preview	Name	Status
	Standard	
▶	Small	!
	Medium	
	Large	!
	Auxiliary node	

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


Edit Node Format "Standard"

Exterior surrounding: 3 mm

[✓] Separate fields by lines

Fields

Type	Combi field	Data field	Constant...	Graphics...	Width	Height	Minimum ...	Maximum...	Align...	Backg
Text	✓	ID		C:_MR...	20 mm	0 mm	1	1	▞	<not s
Text	☐	Code 1			15 mm	0 mm	1	1	▞	<not s

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

ID

Code 1

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 background color 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".

3.13 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 leaf nodes and subtrees can be moved. The drag & drop mode is either started automatically or can be started manually by the VcTree 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.

- **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**, it is your job to produce a result that corresponds to the drop effect. After the event in the source control the **OLECompletDrag** 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.

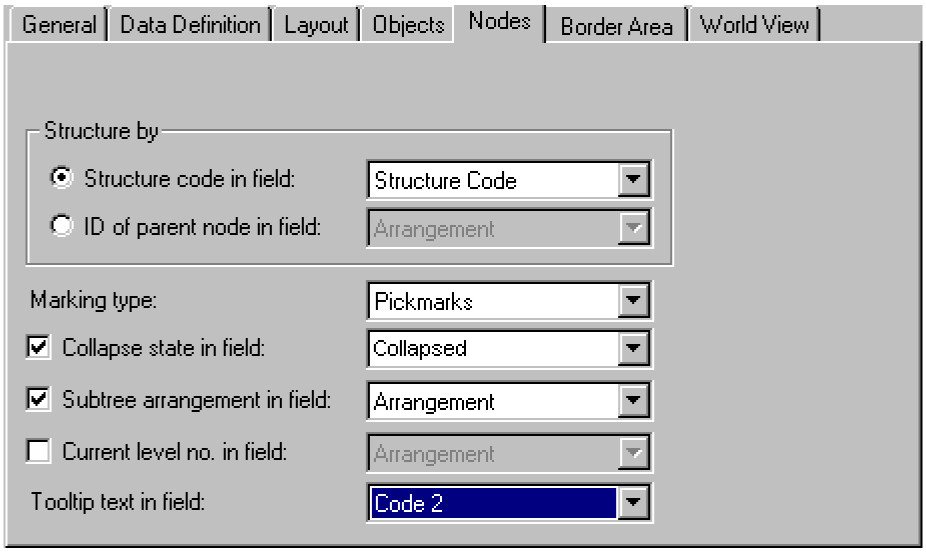
When you want to drag & drop file names, the **DataObjectFiles** object becomes interesting. To drag a file name, you first have to define the file format **vbCFFiles** (resp. **CF_HDROP**) in the **OLEStartDrag** event using **dataObject.SetData Empty, vbCFFiles**. Now you can add files using the **DataObject.Files.Add** 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)**.

3.14 Status Line Text

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

3.15 Structure

On the **Nodes** property page you can set the structure of tree diagrams.



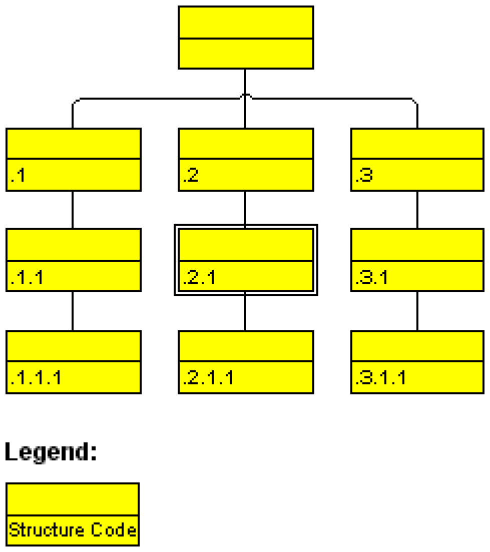
Structure by
- [✓ Structure code in field:] Structure Code
- ID of parent node in field: Arrangement

Marking type: Pickmarks
- [✓] Collapse state in field: Collapsed
- [✓] Subtree arrangement in field: Arrangement
- [ ] Current level no. in field: Arrangement

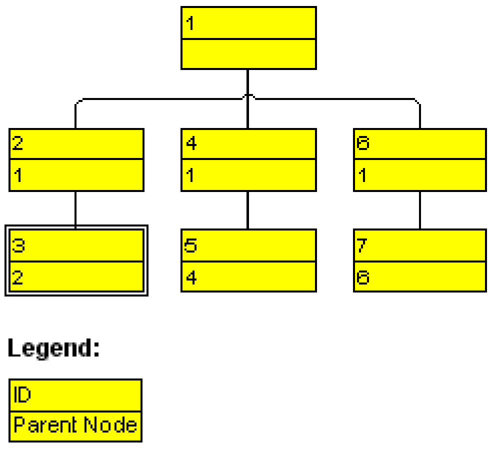
Tooltip text in field: Code 2

There are two options:

1. **Strukture code in field:** The tree is built according to a structure code. You can select a field that the structure code is stored to (Separator: ".").
2. **ID of parent node in field:** The tree is defined by the ID of the parent node. You can select a field that the ID of the parent is stored to.



Tree that was defined by a structure code



Tree that was defined by the IDs of parent nodes

3.16 Text Output During Runtime

The **OnSupplyTextEntry** event allows to replace all items in context menus, dialogs, information boxes and error messages, in order to, for example, translate them into a different language. For this, set the VcTree property **EnableSupplyTextEntryEvent** to **True** to activate the event.

Example Code

```
VcVcTree1.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 VcTree1_OnSupplyTextEntry(ByVal controlIndex As _
                                VcTreeLib.TextEntryIndexEnum, _
                                TextEntry As String, _
                                returnStatus As Variant)
    Select Case controlIndex
        Case vcTXECtxmenCollapse
            TextEntry = "Collapse nodes"
        Case vcTXECtxmenExpand
            TextEntry = "Expand nodes"
        End Select
End Sub
```

3.17 Tooltips During Runtime

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

(The event **OnToolTipTextAsVariant** has to be used if you use a Script language that does not allow that strings are returned, e.g. VBScript.)

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

Example Code

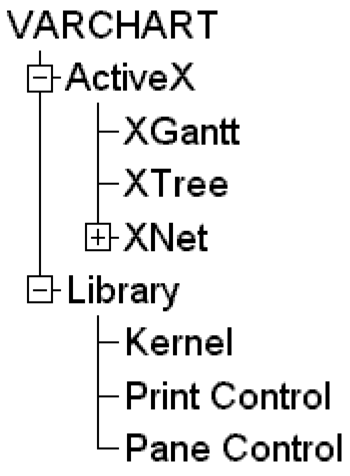
```
VcTree1.ShowToolTip = True
```

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.

3.18 TreeView Style

A tree can be displayed in TreeView style. Similar to the appearance of the Microsoft Explorer directory tree, the tree view style lets you add a plus or minus symbol to vertically arranged node levels. The plus symbol indicates that the subtree of this node is collapsed, the minus symbol indicates that it is expanded. The symbols are set to those nodes only that are no leave nodes, i.e. that do have child nodes. Clicking on a plus symbol will expand a tree and transform the symbol into a minus. Clicking on a minus symbol will collapse the tree and transform the symbol into a plus.

Example of a tree displayed in TreeView style:



To activate the TreeView style, tick the **TreeView style** check box on the **Layout** property page.

3.19 Unicode

In order to display Unicode characters you need a font which supports Unicode as e.g. "Arial Unicode MS".

For displaying Unicode characters on the property pages at design time, an appropriate font has to be set under

Start/Control Panel/Settings/Display/Appearance

for the element **Message Box**.

Every object in VARCHART XTree which contains texts can display Unicode characters if an appropriate font was set in the corresponding property **Font**.

Via the property **DialogFont** of the object **VcTree**, a Unicode font can be assigned to the context menus, tooltips and run time dialogs.

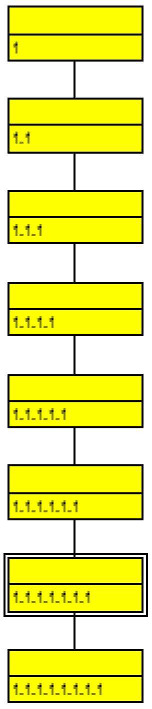
While retrieving CSV files, the method **VcTree.Open** recognizes automatically whether it is a Unicode or an ANSI file.

Note: Visual Studio 6 isn't able to use Unicode characters in the source code files. Internally however, the strings of VB6 are displayed in Unicode. If you use Visual C++ in combination with MFC you have to set the Defines_UNICODE and UNICODE to use strings in Unicode. From the version Visual Studio .NET 2002 onward, the editing of Unicode characters is possible. For the saving of files you have to select the coding "Unicode".

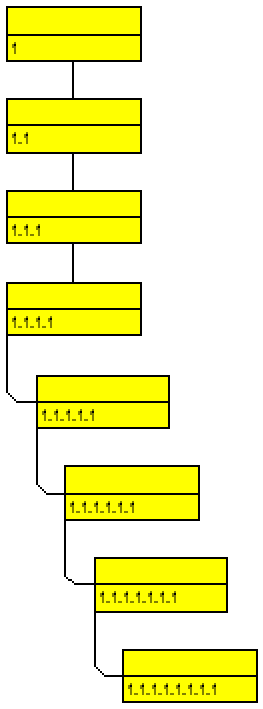
3.20 Vertical Levels

In vertical arrangements nodes of the same level are placed below one another. To most applications it is useful to arrange nodes vertically from a certain level onwards, in order to limit the width of a tree diagram.

On the **Layout** property page, you can tick the check box **Vertical from level** and then enter the number of the level from that on the tree is to be arranged vertically. To trigger the settings, the API method **Arrange** needs to be invoked.



Tree arranged horizontally in all parts



4th level arranged vertically after invoking ***Arrange***

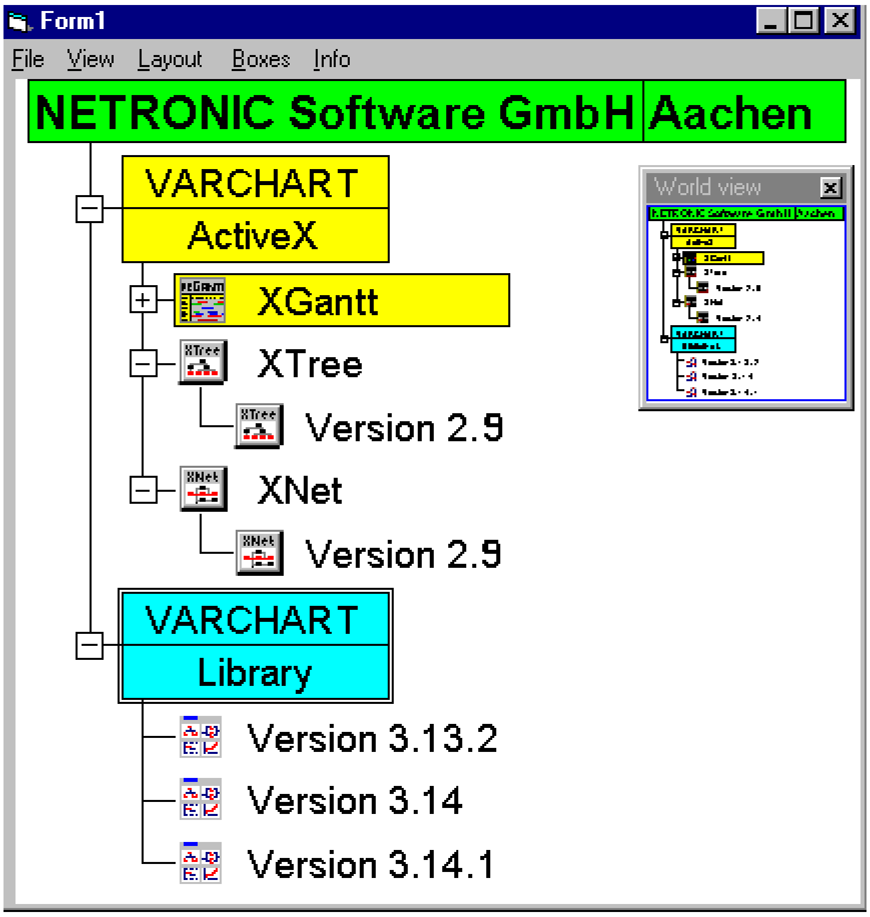
3.21 Viewer Metafile (*.vmf)

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

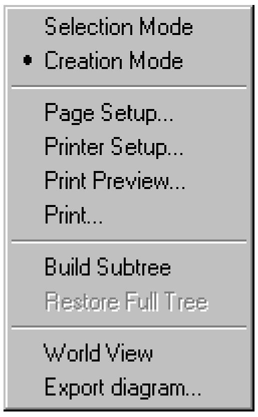
The method **ExportGraphicsToFile** of object VcGantt or the default context menu for the diagram lets you store the diagram to a file.

3.22 World View

The world view is an additional window that shows the complete diagram. A frame shows the diagram section currently displayed in the main window. When you move the frame or change its size, the corresponding diagram section will be adapted as soon as you release the mouse button. Vice versa, the position or the size of the frame will be changed when you scroll or zoom the section in the main window.



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



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

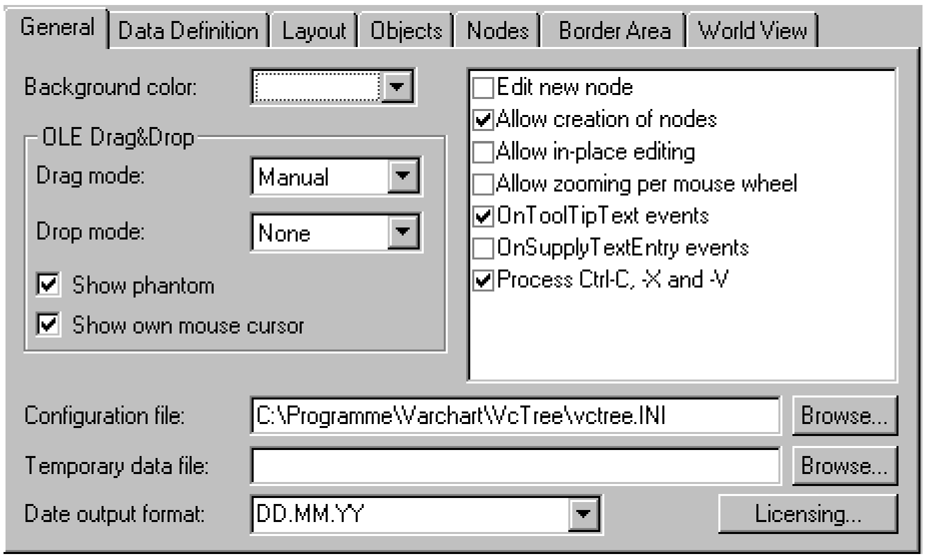
You also can specify the properties of the World View via the API (**VcWorldView**).

4 Property Pages and Dialog Boxes

4.1 General Information

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



Background color:

OLE Drag&Drop
Drag mode: Manual
Drop mode: None
☑ Show phantom
☑ Show own mouse cursor

☐ Edit new node
☑ Allow creation of nodes
☐ Allow in-place editing
☐ Allow zooming per mouse wheel
☑ OnToolTipText events
☐ OnSupplyTextEntry events
☑ Process Ctrl-C, -X and -V

Configuration file: C:\Programme\Varchart\VcTree\vctree.INI
Temporary data file:
Date output format: DD.MM.YY

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

Background Color

Please select a background color for the tree diagram. The default color is white.

Drag mode

By this property you can set/retrieve, whether dragging a node beyond the limits of the current VARCHART XTree 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 XTree 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 XTree 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 XTree control can be dropped in 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 you select **Automatic**, the dropping will automatically be processed by the control, displaying 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 when generating a new object is omitted but merely the attributes of the object in the target control are modified.

Show own mouse cursor

This property lets you enable/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 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 via this checkbox.

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.

Edit new node

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

Allow creation of nodes

Please activate this option if you want the user to be able to create new nodes in an open project interactively.

Allow in-place editing

Tick this option if the in-place editing of node fields is to be allowed. If for certain data fields in-place editing is not to be allowed, please specify these fields as "read only" in the data definition.

Allow zooming per mouse wheel

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

OnToolTipText events

This feature lets you activate/deactivate the event **OnToolTipText**. It also can be set by the **ShowToolTip** property of the vcTree class. The event **OnToolTipText** lets you edit the tooltip texts.

OnSupplyTextEntry events

By ticking this box you can trigger the **OnSupplyTextEntry** event. This event lets you modify the texts of context menus, dialog boxes, error messages that occur during run time, for example for translation into different languages. This feature can also be set by the vcTree property **EnableSupplyTextEntryEvent**.

Process Ctrl-X, -C and -V

If you activate this check box, the key combinations Ctrl+C, Ctrl+X and Ctrl+V will be translated automatically into the clipboard commands **Copy-NodesToClipboard**, **CutNodesToClipboard** and **PasteNodesFrom-Clipboard**, respectively. You can revoke this feature by leaving the check box blank, in order to avoid interfering with menu commands in Visual Basic. This feature can also be set by the **CtrlCXVProcessing.Enabled** property of the object **VcGantt**.

Configuration file

In this field the configuration file is displayed. It serves to initialize and save your settings on the property pages. Beside, your settings are saved internally to the VARCHART ActiveX.

To set a configuration file, you can select an *.ini* file via the **Browse** button. Alternatively, you can create an *.ini* file. Please enter the name *tutorial.ini* into the **Configuration file** field and click on the **Apply** button. If the file does not exist yet, you will receive the message that it is being created. The file will be generated as soon as you confirm the message.

Browsing for a different file by pressing the **Browse** button will open the **Load/Save** dialog. After leaving the dialog via the **Save** button, you will return to the property page, where you need to click on the **Apply** button. The file will be loaded, and any modification on the property pages from now on will be stored to it.

The settings of the configuration file are loaded only for once. The VARCHART ActiveX will not read them for a second time from the same file. Instead, the settings will be loaded from internal storings, that are the same as the ones in the configuration file.

So modifying the data of the configuration file from outside will not be effective. If you do want VARCHART ActiveX to accept a modified configuration file, in addition you need to rename it and import this configuration file.

Temporary data file

During desing mode, you can enter or select the name of a file that contains node data to control your settings. By the **Browse** button you can open the Windows dialog **Load/Save** that will display the preset file type "Data files (*.tre)". The Data file selected here will be valid for design mode only. During runtime, a file has to be opened by the **Open** method.

Date output format

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

The format will also apply to the dialogs at runtime. You can compose the format of the below 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)

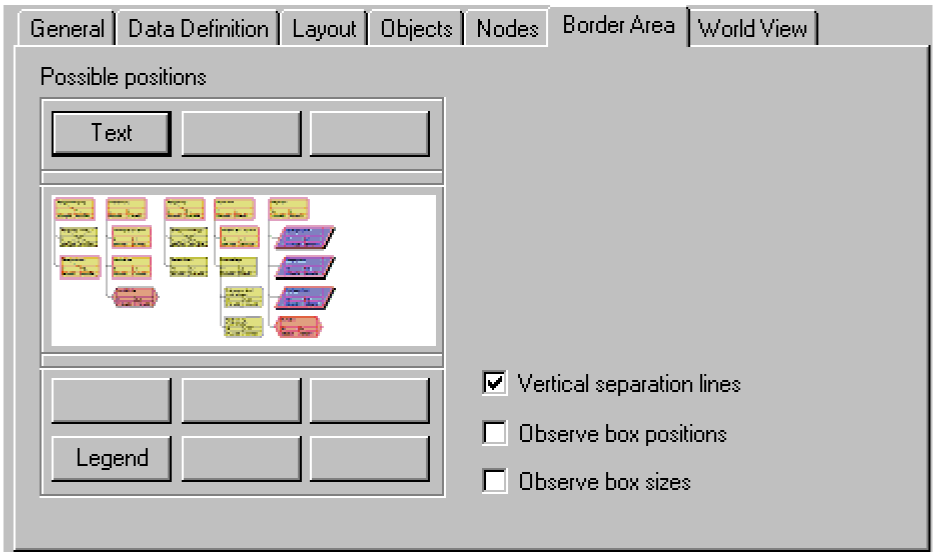
- **DD** (two-digit figure for the day)
- **hh** (two-digit figure for the hour)
- **mm** (two-digit figure for the minute)
- **ss** (two-digit figure for the second)

Any other character can be put between the strings.

Licensing

Press this button to reach the **Licensing** dialog box.

4.3 The "Border Area" Property Page



[✓] Vertical separation lines
[ ] Observe box positions
[ ] Observe box sizes

Possible positions

There are three areas above and six areas below the diagram which you can 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 "Data Definition" Property Page

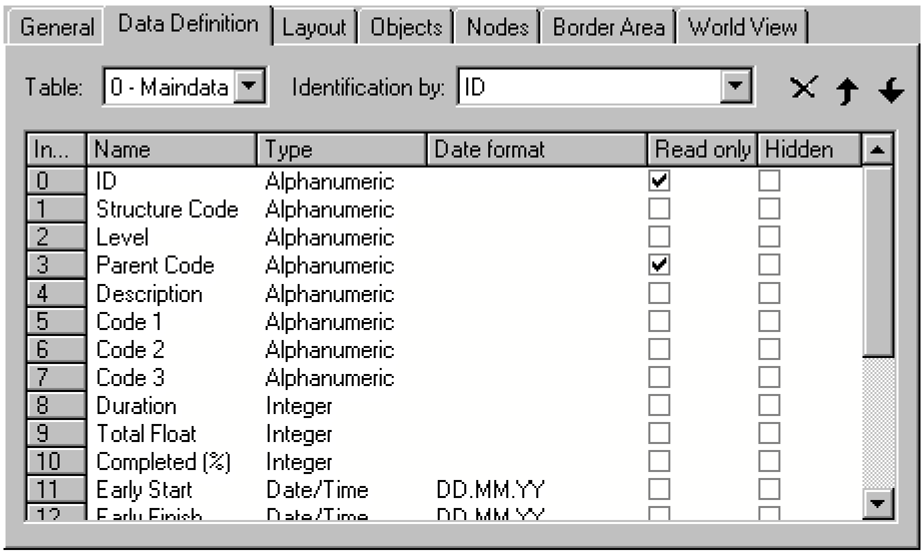


Table: 0 - Maindata Identification by: ID

In...	Name	Type	Date format	Read only	Hidden
0	ID	Alphanumeric		☑	☐
1	Structure Code	Alphanumeric		☐	☐
2	Level	Alphanumeric		☐	☐
3	Parent Code	Alphanumeric		☑	☐
4	Description	Alphanumeric		☐	☐
5	Code 1	Alphanumeric		☐	☐
6	Code 2	Alphanumeric		☐	☐
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	☐	☐

This property page lets you define the data of your interface. Logically, you should make your entries on the **Data Definition** property page before setting the options on the other property pages.

You can delete data fields via the ✕ button or via the **Del** key. To create a new data field, overwrite the "New..." entry at the end of the list. As soon as you leave this line, the new data field will be created and another "New..." entry will be created. With the help of the arrow buttons (↑ ↓) you can move data fields in the list.

Table

This field displays the name of the table currently used:

0 - Maindata (node data)

Identification by

This field lets you select the data field that a node is to be identified by.

Index

The index of the data fields (on gray background) cannot be modified, since internally, it serves as a reference. In the API, data fields are referred to by the index. The **Maindata** table contains 20 data fields by default.

Name

This column displays the names of the data fields of the **Maindata** table. You can change the field names or create a new data field by editing the name "New..." at the end of the list. As soon as you leave the field edited, your modifications will be accepted and another "New..." field will occur.

Type

Specify the data type for the selected data field. Choose between:

- **Alphanumeric**
- **Integer**
- **Date/Time**

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.

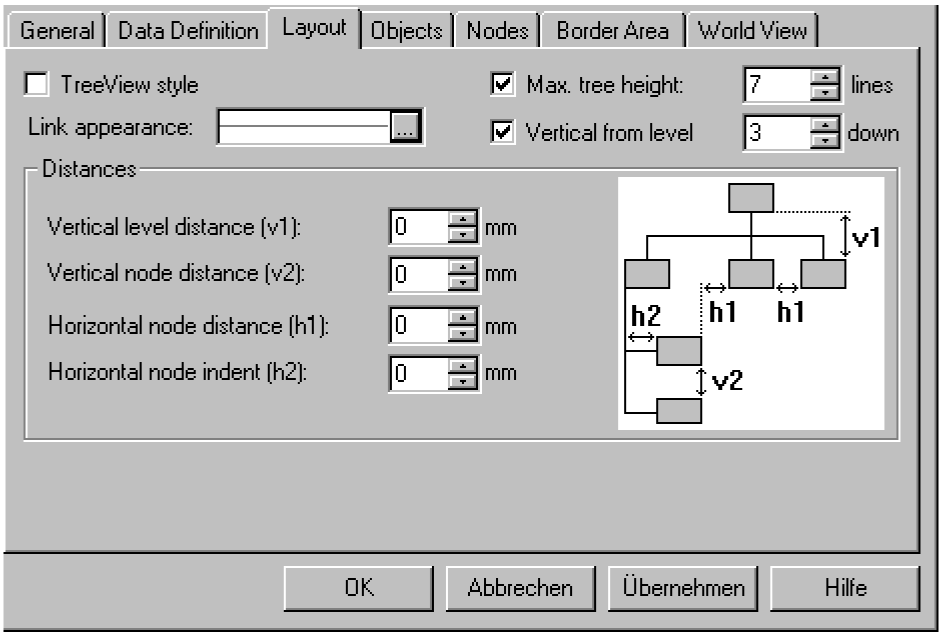
Read only

Please activate this checkbox for all data fields that you do not wish the user to edit in the dialog **Edit Data**.

Hidden

Please activate this checkbox that you wish to remain hidden to the user in the dialog **Edit Data**.

4.5 The "Layout" Property Page



☐ TreeView style
Link appearance:
☑ Max. tree height: 7 lines
☑ Vertical from level 3 down

Distances
Vertical level distance (v1): 0 mm
Vertical node distance (v2): 0 mm
Horizontal node distance (h1): 0 mm
Horizontal node indent (h2): 0 mm

This property page lets you specify the settings of your diagram.

TreeView style

This check box lets you display the nodes in TreeView style, that looks like e. g. the Microsoft Explorer directory tree. Typical for the TreeView style are the minus and plus symbols on parent nodes, that indicate whether a subtree is expanded or collapsed, respectively. Clicking on the minus or plus symbol turns an expanded subtree into a collapsed one or vice versa.

The symbols are only displayed for nodes which have children. A mouse click will change the status from collapsed to expanded or vice versa.

Link appearance

This field displays the current link appearance. To modify it, click on the **Edit** button. You will get to the **Line attributes** dialog, where you can set **Type**, **Thickness** and **Color** of the lines.

Max. tree height

If you tick this check box, you can enter the maximum tree height by number of levels. These settings will influence vertical arrangements only. If in a vertical branch more levels exist than set, another branch will be generated by the parent node to adopt the remaining child nodes.

Vertical from level

If you tick this check box, all nodes from the level selected will be arranged vertically. To trigger the setting, the API method **Arrange** needs to be invoked.

Vertical level distance (v1)

This field lets you enter the vertical distance between horizontally arranged levels. Unit: mm.

Vertical node distance (v2)

This field lets you enter the vertical distance between vertically arranged nodes. Unit: mm.

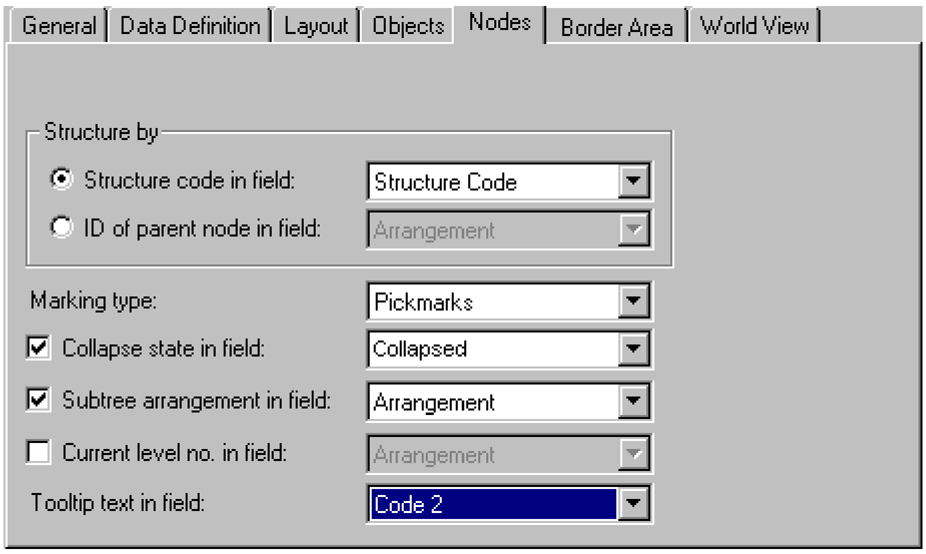
Horizontal node distance (h1)

This field lets you set the horizontal node distance between two horizontally arranged nodes. Unit: mm

Horizontal node indent (h2)

This field lets you enter the horizontal indent of vertically arranged nodes. Unit: mm.

4.6 The "Nodes" Property Page



Structure by

(●) Structure code in field: Structure Code

( ) ID of parent node in field: Arrangement

Marking type: Pickmarks

[✓] Collapse state in field: Collapsed

[✓] Subtree arrangement in field: Arrangement

[ ] Current level no. in field: Arrangement

Tooltip text in field: Code 2

Structure code in field

By ticking this check box, you can select the data field to set the structure code.

ID of parent node in field

If you tick this check box, you can select the data field to store the ID of the parent node.

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.

Collapse state in field

Activate this check box to continuously store the state of collapsing/ expanding a node to a field. The data field may contain "0" for an expanded node or "1" for a collapsed node.

Subtree arrangement in field

Activate this check box to keep the orientation of a subtree stored to a field. The data field may contain "0" for a subtree arranged horizontally, or "1" for a subtree arranged vertically. A horizontal arrangement is visible only if the immediate or mediate parent nodes are arranged horizontally.

Current level no. in field

Activate this check box to specify the data field that contains the level number of nodes. The level numbers are counted from 0 upwards.

This property also can be set via the VcTree property **LevelField**.

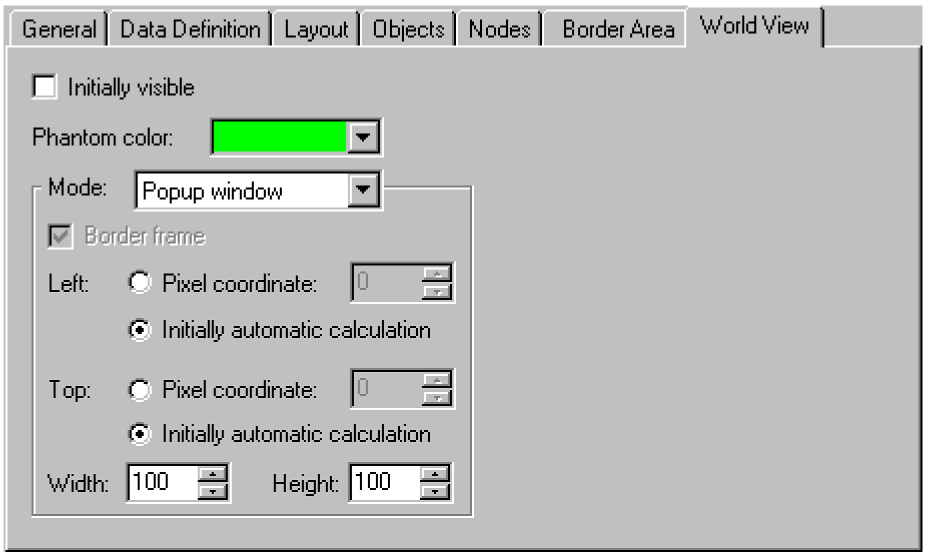
Note: At run time it is not possible to modify the level of a node by modifying the value of the level number data field.

Tooltip text in field

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

To show tooltips in your application in the VARCHART XTree, activate the checkbox **OnToolTipText events** on the **General** property page or set the property **ShowToolTip** = True and specify in the **OnToolTipText** event which data fields are to be displayed.

4.7 The "World View" Property Page



☐ Initially visible

Phantom color: 🟩

Mode: Popup window

☑ Border frame

Left: ○ Pixel coordinate: 0
● Initially automatic calculation

Top: ○ Pixel coordinate: 0
● Initially automatic calculation

Width: 100 Height: 100

On this property page you can specify the properties of the World View. The world view is an additional window that shows the complete diagram. A frame indicates the section currently displayed in the main window.

If you have selected the **Popup window** mode, you can switch on/off the world view at run time via the item **Show world view** of the default context menu.

Initially visible

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

Phantom color

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

Mode

Select the world view mode. There are the following options:

- **Fixed at left side:** The world 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.
- **Fixed at right side:** The world 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.

- **Fixed at top side:** The world 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.
- **Fixed at bottom side:** The world 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.
- **Position not fixed:** The world 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**.
- **Popup window:** The world 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.

Border frame

Not active if the mode ***Popup window*** *has been selected.* Activate this check box if the world view is to have a frame.

Left

Only active if the mode ***Position not fixed*** *or* ***Popup window*** *has been selected.* Select the left position of the world 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.

Top

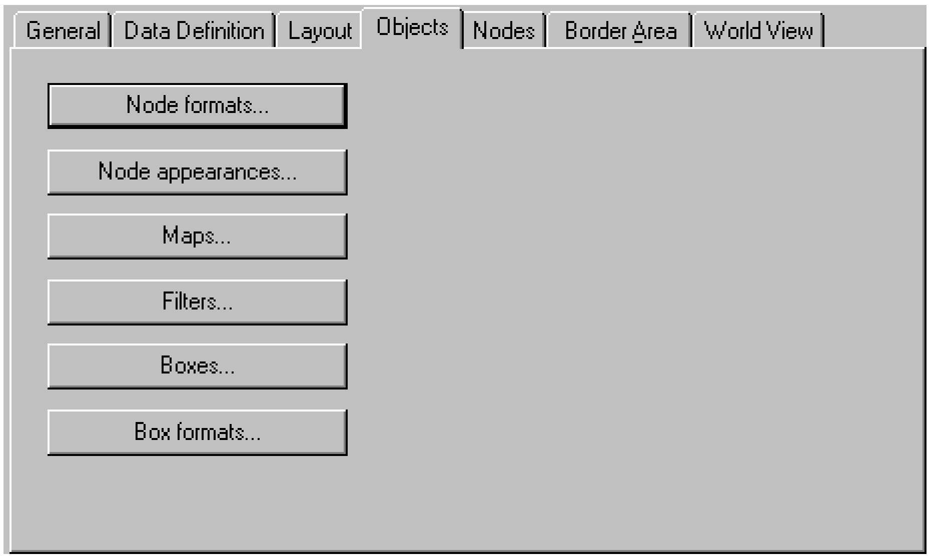
Only active if the mode ***Position not fixed*** *or* ***Popup window*** *has been selected.* Select the top position of the world view. There are two possibilities:

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

Width

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

4.8 The "Objects" Property Page



Node formats

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

Node appearances

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

Maps

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

Filters

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

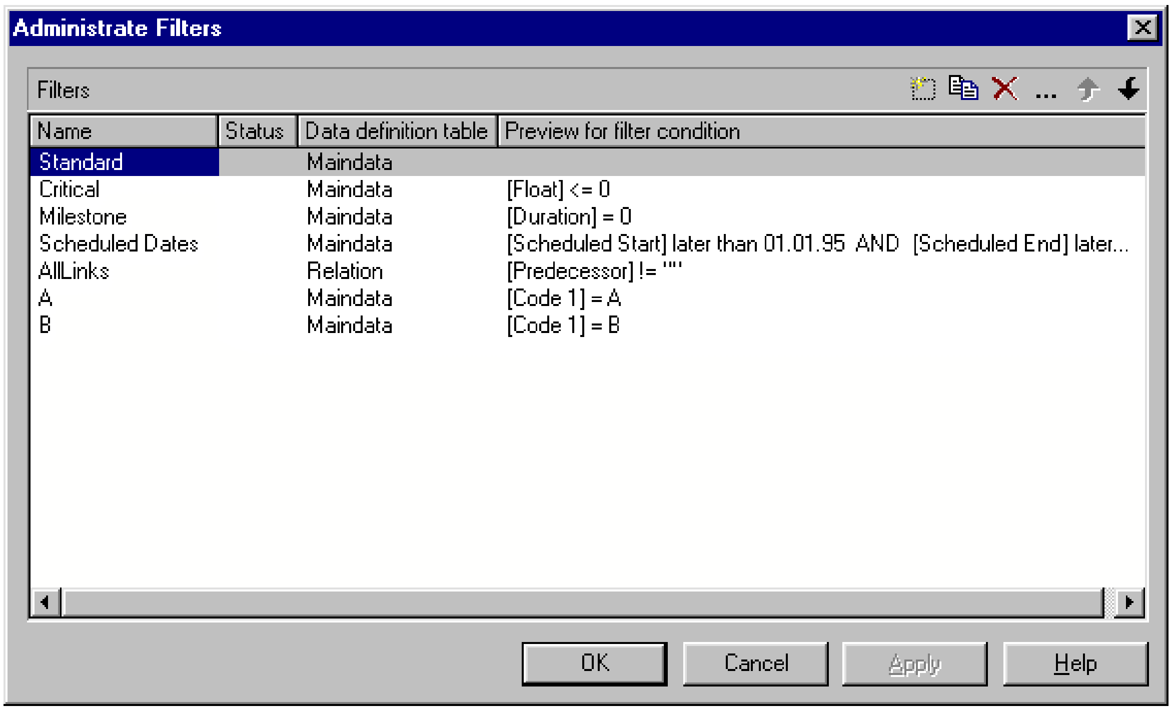
Boxes

Opens the dialog **Administrate Boxes**.

Box formats

Opens the dialog **Administrate Box Formats**.

4.9 The "Administrate Filters" Dialog



Name	Status	Data definition table	Preview for filter condition
Standard		Maindata	
Critical		Maindata	[Float] <= 0
Milestone		Maindata	[Duration] = 0
Scheduled Dates		Maindata	[Scheduled Start] later than 01.01.95 AND [Scheduled End] later...
AllLinks		Relation	[Predecessor] != ""
A		Maindata	[Code 1] = A
B		Maindata	[Code 1] = B

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

Name

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

Status

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

Data definition table

This column shows the data definition table (**Maindata**) for each filter (see property page **DataDefinition**).

Preview for the filter condition

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

Add filter

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

Copy filter

Copies the selected filter.

Delete filter

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

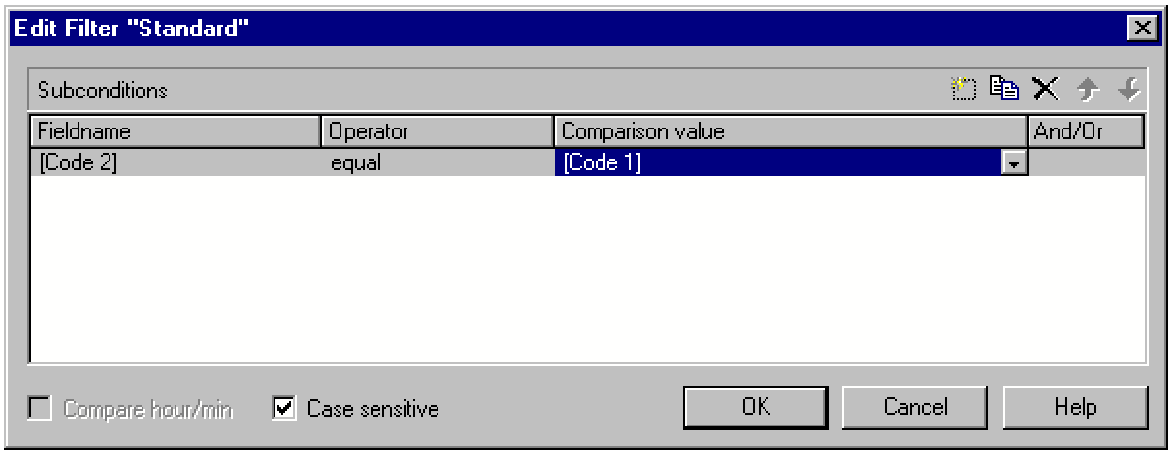
Edit filter

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

Filter one line up/down

The selected filter will be moved up/down one position in the list.

4.10 The "Edit Filter" Dialog



Fieldname	Operator	Comparison value	And/Or
[Code 2]	equal	[Code 1]	

☐ Compare hour/min ☑ Case sensitive

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

Add subcondition

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

Copy subcondition

Copies the selected subcondition.

Delete subcondition

Deletes the selected subcondition.

Evaluate subcondition earlier/later

If a filter consists of several subconditions, 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 one position up/down in the table.

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 via 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 "inequal" 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).

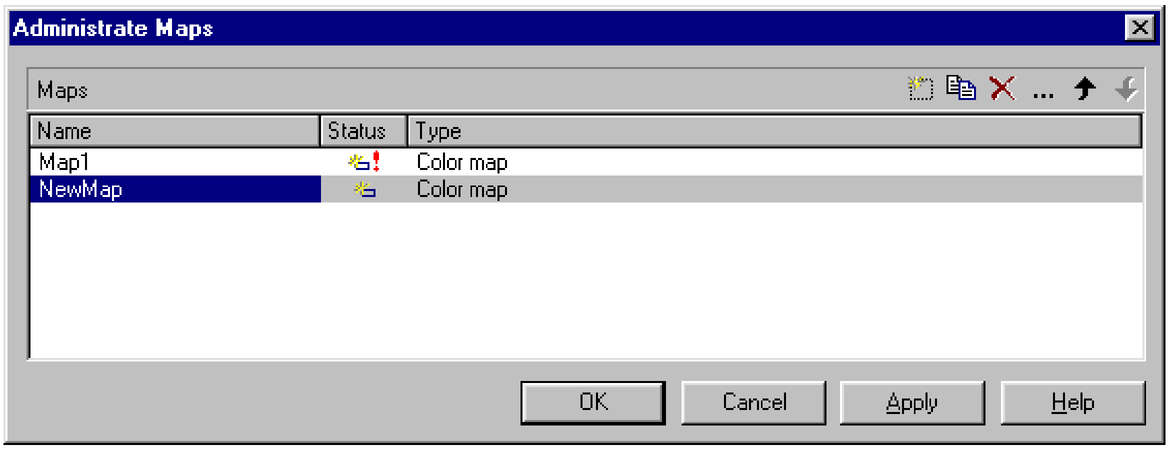
Compare hour/min

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.

4.11 The "Administrate Maps" Dialog


Administrate Maps

Maps

Name	Status	Type
Map1	!	Color map
NewMap		Color map

You reach this dialog via the **Objects** property page or when you press the **Maps** button of 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.

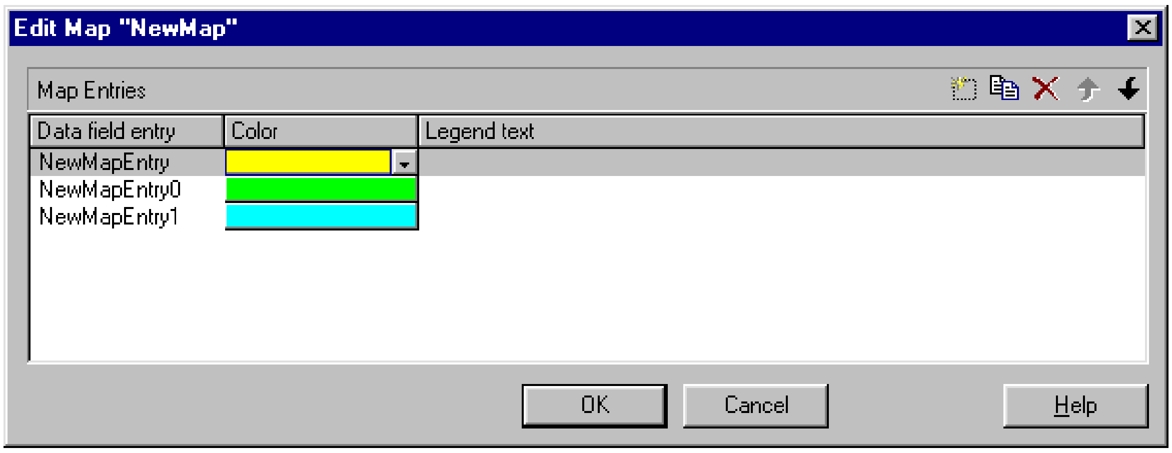
Edit map

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

Map one line up/down

↑ ↓ The selected map will be moved up/down one position in the list.

4.12 The "Edit Map" Dialog



Data field entry	Color	Legend text
NewMapEntry	🟨	
NewMapEntry0	🟩	
NewMapEntry1	🟦	

This dialog box will open when you press the **Edit map** button (...) of the **Administrate Maps** dialog box.

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.

Data field entry

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

Color/Graphics File Name

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

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

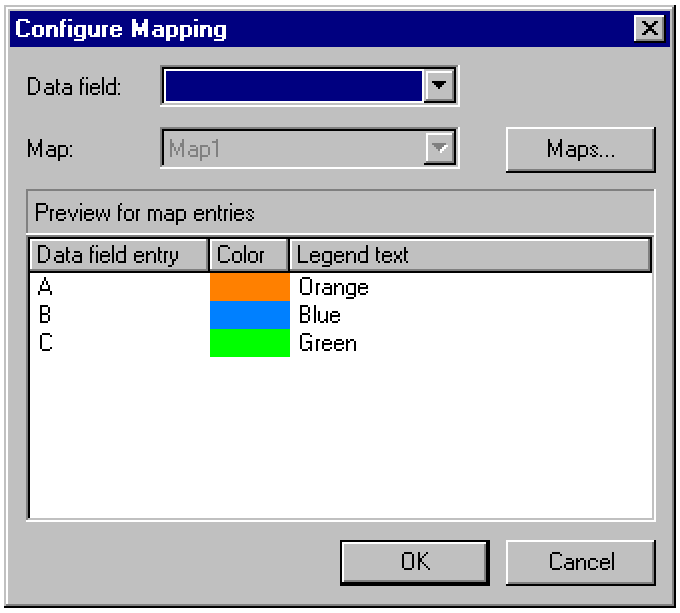
Delete map entry

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

Map entry one line up/down

The selected map entry will be moved up/down one position in the list.

4.13 The "Configure Mapping" Dialog



Data field entry	Color	Legend text
A	🟧	Orange
B	🟦	Blue
C	🟩	Green

In this dialog box you can assign a map to a data field.

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

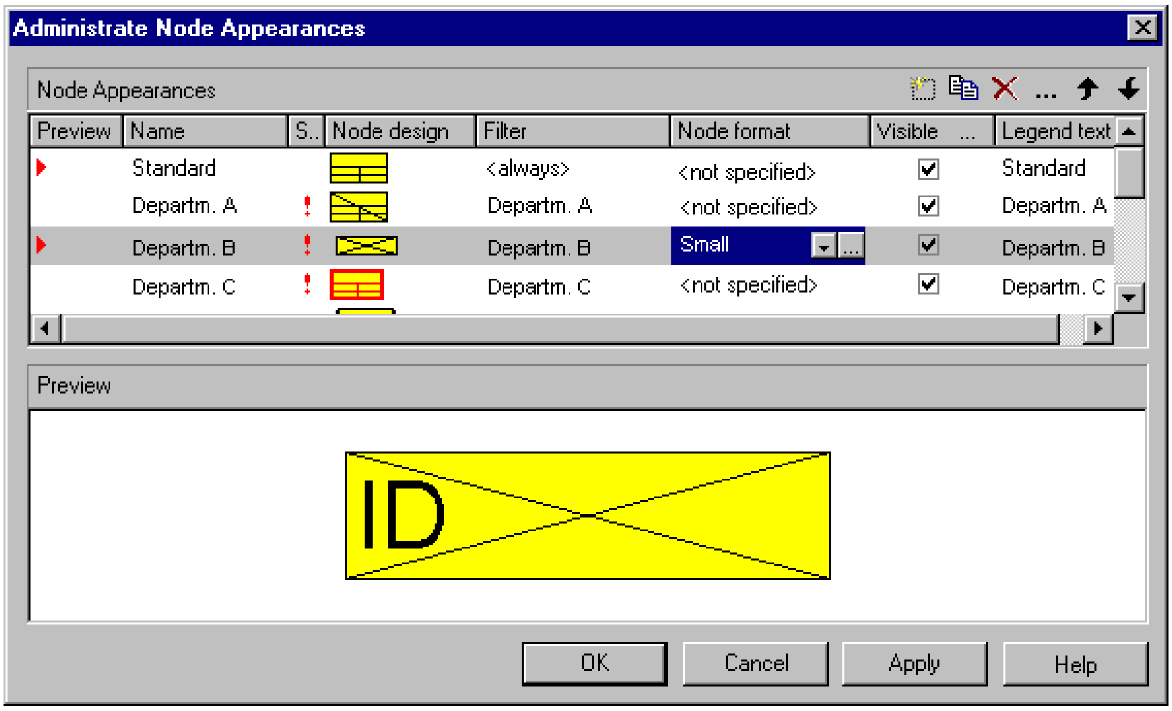
Maps

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

Preview for map entries

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

4.14 The "Administrate Node Appearances" Dialog



Preview	Name	S..	Node design	Filter	Node format	Visible	...	Legend text
▶	Standard			<always>	<not specified>	✓		Standard
	Departm. A	!		Departm. A	<not specified>	✓		Departm. A
▶	Departm. B	!		Departm. B	Small	✓		Departm. B
	Departm. C	!		Departm. C	<not specified>	✓		Departm. C

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

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

Preview

All node appearances marked by a small arrowhead in the **Preview** column are displayed and 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 appearances you can select the filter of your choice. Only for the node appearances "Standard" and "Collapsed" the filters are fixed ("<always>" or "<CollapsedNode>").

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 checkbox 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.

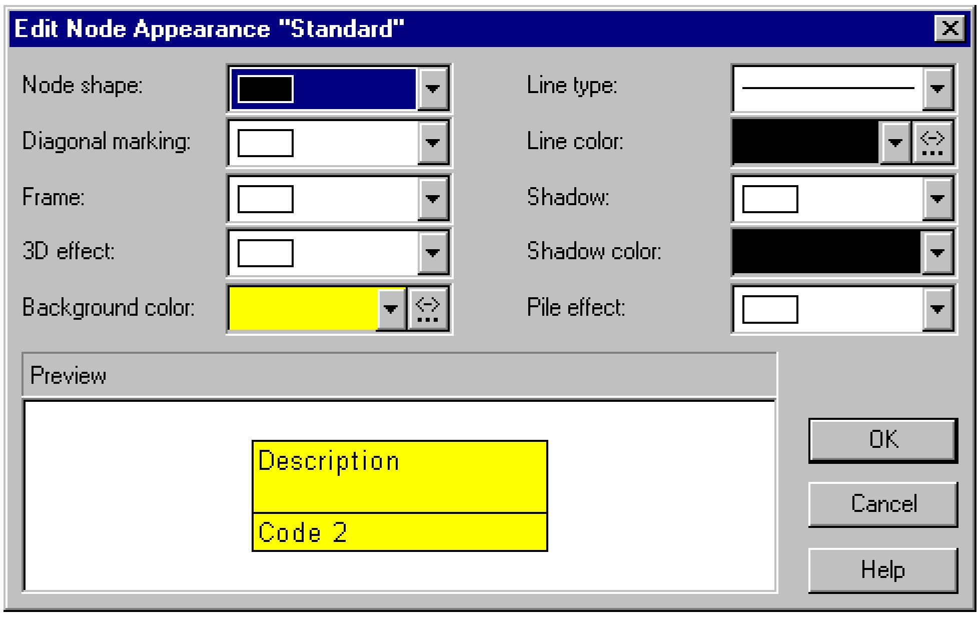
If several node appearances apply to a node, the attributes of each node appearance are replaced by the attributes of the node appearances that are processed later. Only the attributes whose value is "not specified" do not replace the attributes of their predecessors.

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.

4.15 The "Edit Node Appearance" Dialog



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>.

Diagonal marking

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

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.

Background color

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

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

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

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

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.

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

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

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

Shadow

This field lets you add a shadow to the nodes.

Shadow color

Select the color for the shadow or the pile effect.

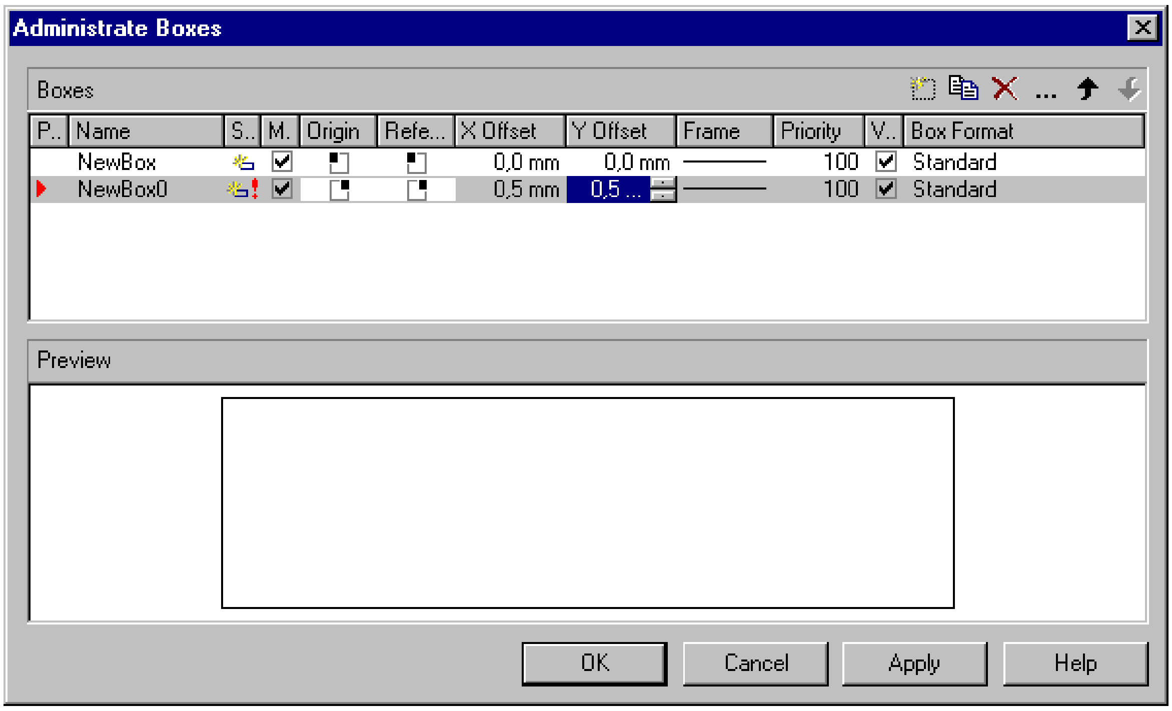
Pile effect

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.

4.16 The "Administrate Boxes" Dialog



P..	Name	S..	M.	Origin	Refe...	X Offset	Y Offset	Frame	Priority	V..	Box Format
	NewBox		✓			0,0 mm	0,0 mm		100	✓	Standard
▶	NewBox0	!	✓			0,5 mm	0,5 ...		100	✓	Standard

In the diagram area, any number of boxes can be displayed. With the help of this dialog box you can administrate these boxes. You can get to this dialog box via the **Objects** property page.

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.

Moveable free

By moving a box, its offset will be modified. Activate this checkbox if the box is to be moveable in the diagram at run time. Deactivate the checkbox if you have positioned a box correctly and do not want it to be moved at run time.

Origin

With the help of the properties **Origin**, **Reference point**, **X Offset** and **Y Offset** you can position each 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

Specify 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

Specify the distance between origin and reference point in x direction.

Y Offset

Specify the distance 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 checkbox 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.

... Via 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.

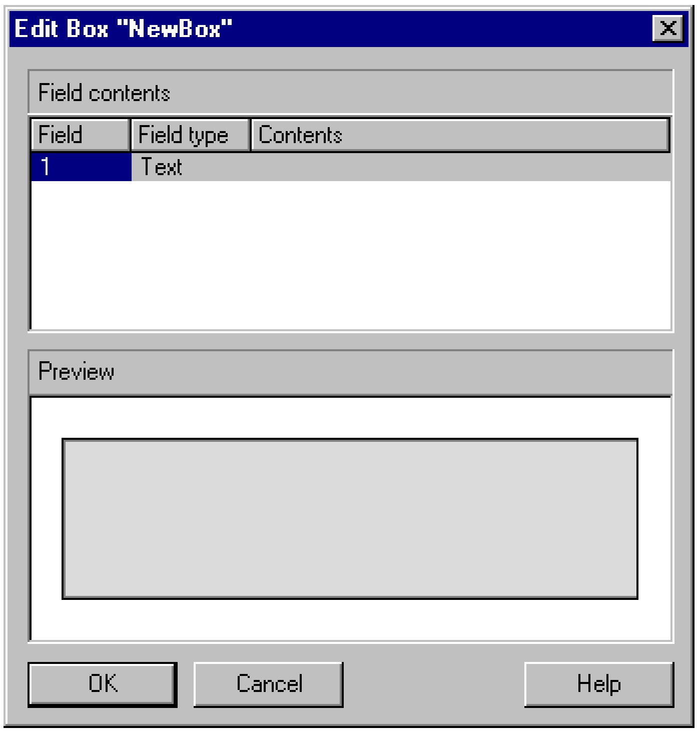
Edit box

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

Box one line up/down

↑ ↓ The selected box will be moved up/down one position in the list.

4.17 The "Edit Box" Dialog



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

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

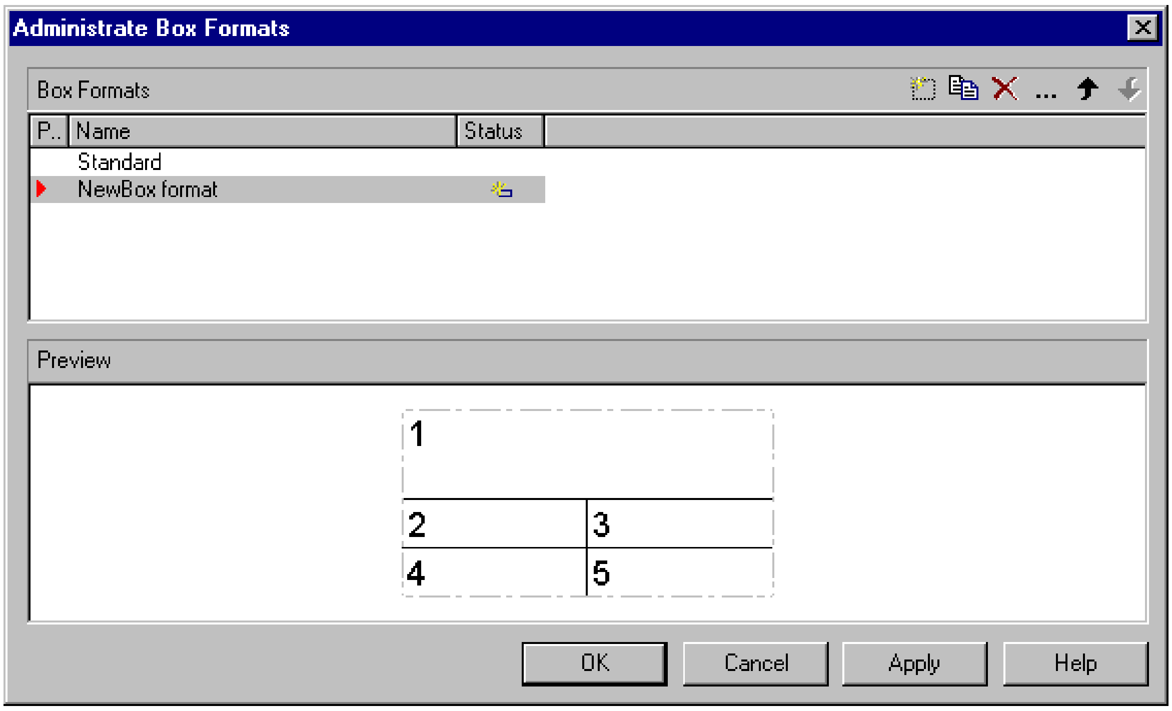
Contents

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 availabe: VMF, WMF, JPG, BMP, EPS, GIF, PCX, PNG, TIF.

4.18 The "Administrate Box/Node Formats" Dialog



This dialog you can get to via 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.

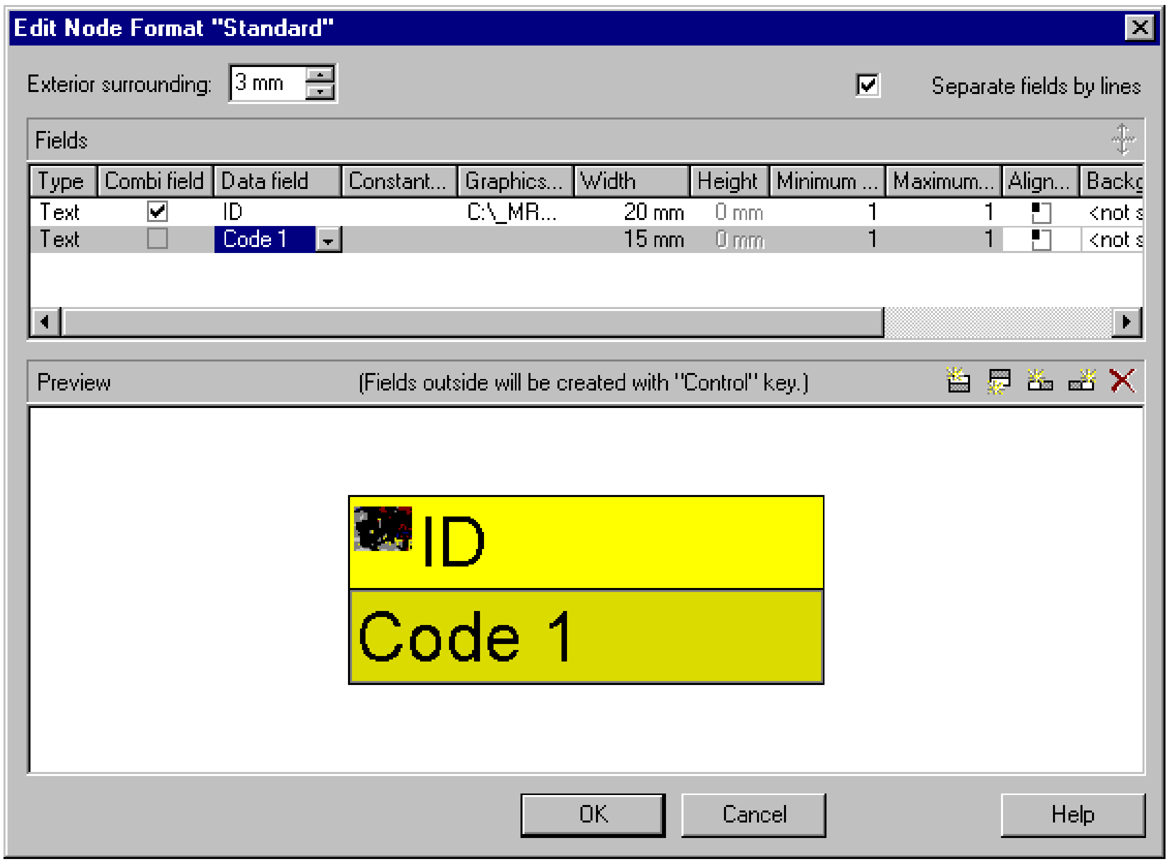
Edit box/node format

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

Box/node format one line up/down

The selected format will be moved up/down one position in the list.

4.19 The "Edit Node Format" Dialog


Edit Node Format "Standard"

Exterior surrounding: 3 mm

Separate fields by lines ✓

Fields

Type	Combi field	Data field	Constant...	Graphics...	Width	Height	Minimum ...	Maximum...	Align...	Backg
Text	✓	ID		C:_MR...	20 mm	0 mm	1	1		<not s
Text		Code 1			15 mm	0 mm	1	1		<not s

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

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

Exterior surronding

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 combobox 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.

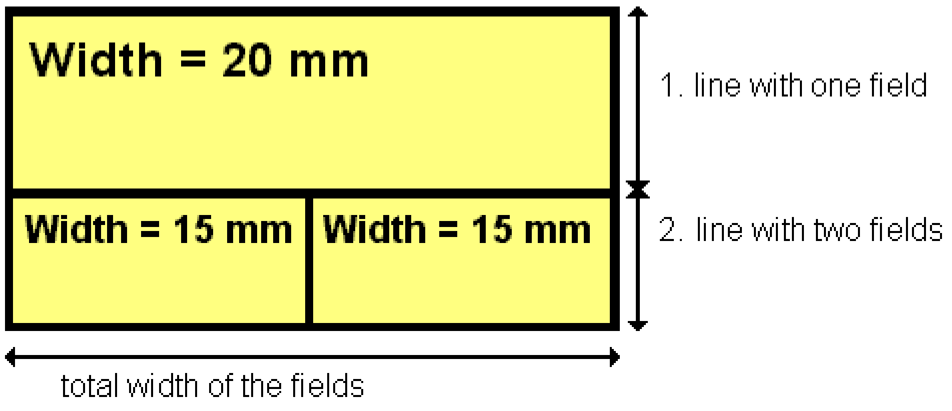
If a mapping has been configured, the arrow on the second button will be displayed in bold (▤).

↔ 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.

Background color

Select the background color for the current format field. You can define your own colors in addition to the ones suggested. If you click on the field, two buttons will apperar:

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

Via the second button you reach the **Configure Mapping** dialog box. Here you can configure data-dependent background colors.

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

If you do not specify a background color, the background color specified for the node appearance will be applied.

Font Color

(only for the type text) Specify the font color for the field. If you click on the field, two buttons will apperar:

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

Via 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

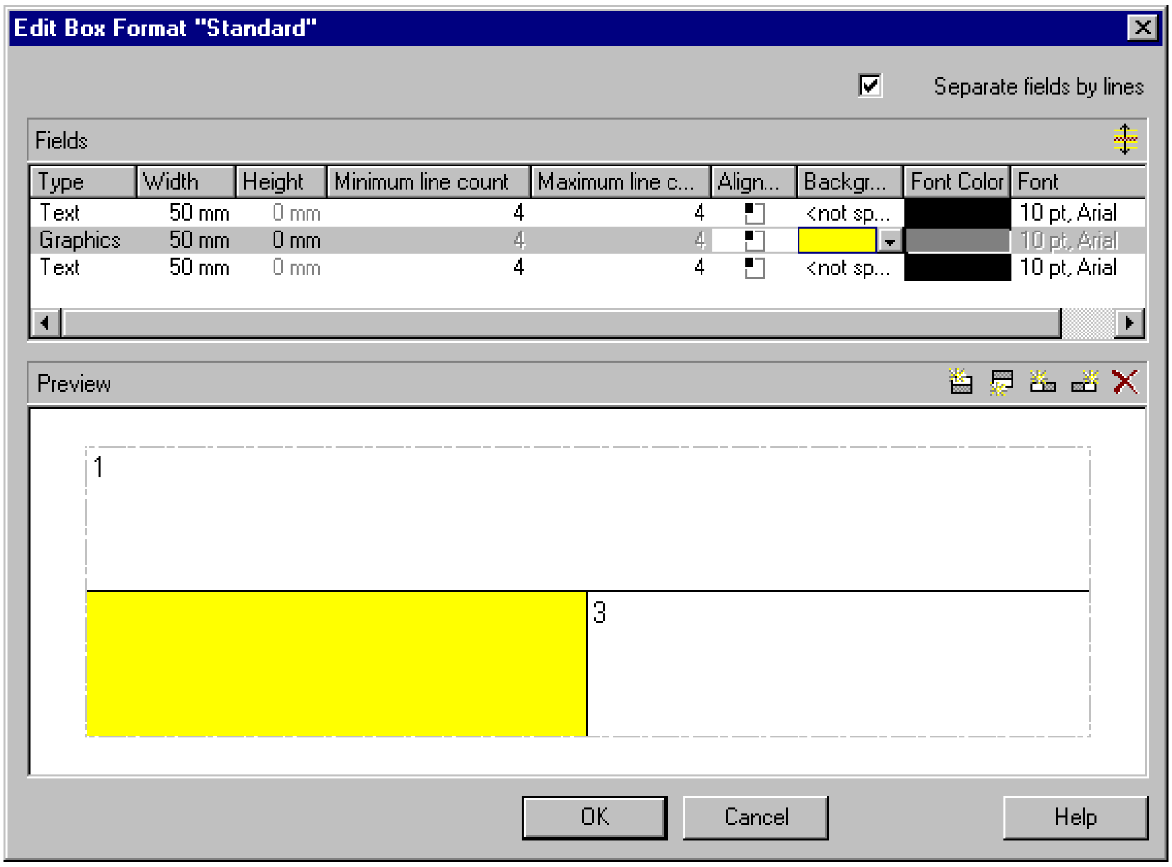
The current node format is displayed in the preview window. If you click on a field in the preview window you can modify its attributes in the **Fields** table.

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.

4.20 The "Edit Box Format" Dialog



[✓] Separate fields by lines

Type	Width	Height	Minimum line count	Maximum line c...	Align...	Backgr...	Font Color	Font
Text	50 mm	0 mm	4	4		<not sp...	■	10 pt, Arial
Graphics	50 mm	0 mm	4	4		🟨	■	10 pt, Arial
Text	50 mm	0 mm	4	4		<not sp...	■	10 pt, Arial

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

Separate fields by lines

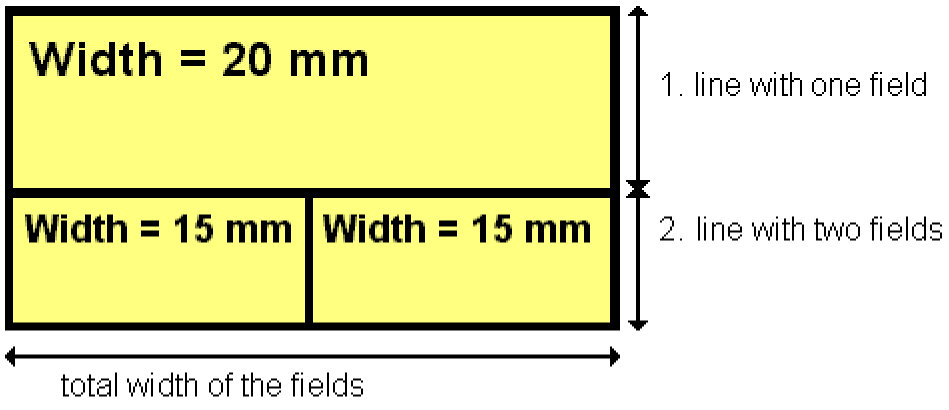
Activate this check box if the box fields are to be separated by lines.

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).

Background color

Select the background color for the current field. You can define your own colors in addition to the ones suggested.

Font Color

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

▾ Via 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

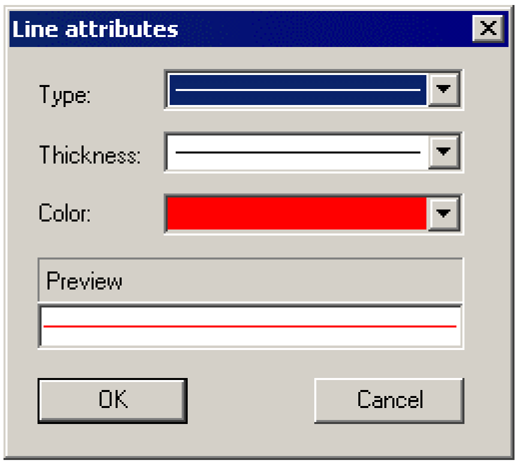
Applies the marked property to all fields.

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.

4.21 The "Line Attributes" Dialog



Preview

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

Type

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

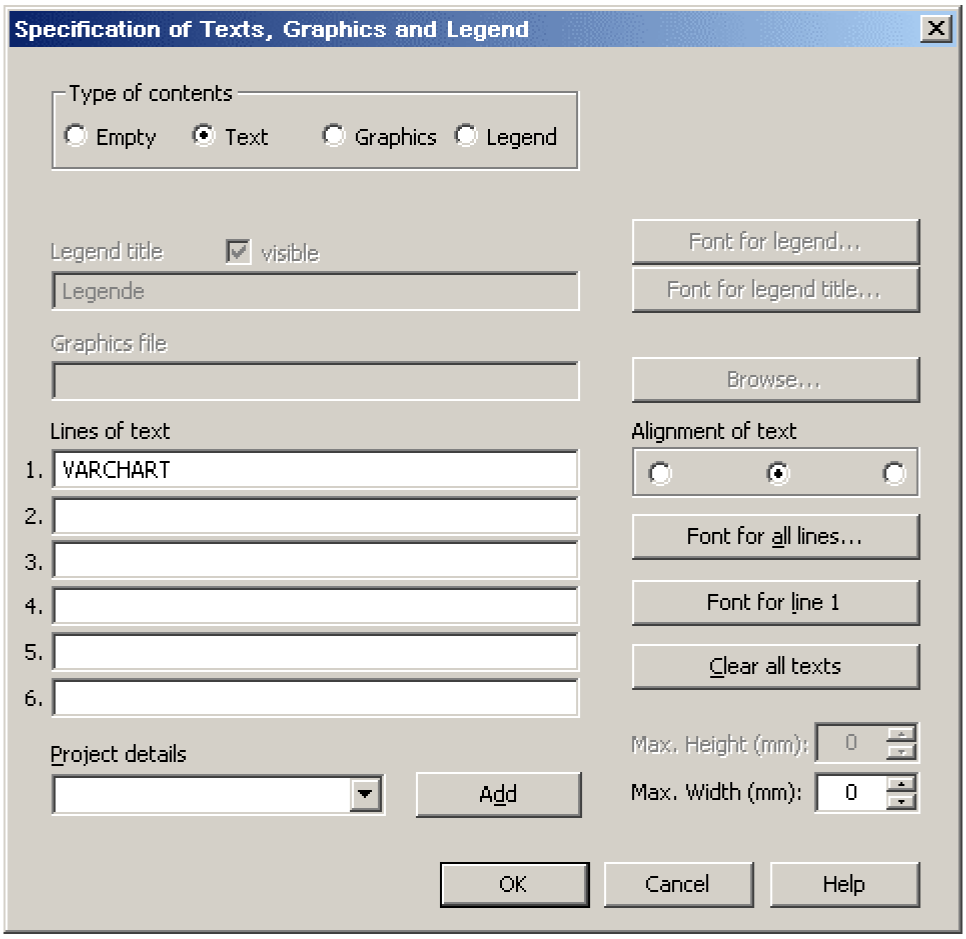
Thickness

Define the line thickness.

Color

Select the line color.

4.22 The "Specification of Texts, Graphics and Legend" Dialog



Type of contents: Empty [✓ Text] Graphics Legend

Legend title: Legende ✓ visible

Graphics file:

Lines of text:
1. VARCHART
2.
3.
4.
5.
6.

Project details:

Max. Height (mm): 0

Max. Width (mm): 0

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

Type of contents

Specify the type of information you want to display at the chosen 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 (via 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 title

*Only activated when the check box **Legend** has been ticked.* Type a legend title.

visible

*Only activated when the check box **Legend** has been ticked.* Activate this check box if the legend title is to be visible.

Font for legend

*Only activated when the check box **Legend** has been ticked.* You will reach the Windows **Font** dialog box where you can specify the font attributes for the legend texts.

Font for legend title

*Only activated when the check box **Legend** has been ticked.* You will reach the Windows **Font** dialog box where you can specify the font attributes for the legend title.

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 form the list and clicking on the **Add** button.

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

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.

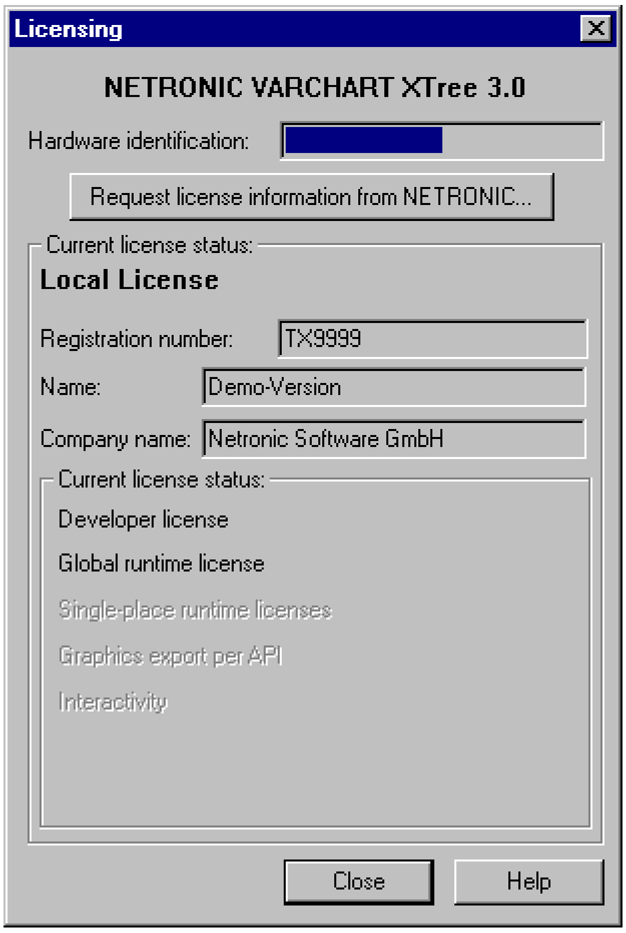
Max. Height (mm)

*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.

4.23 The "Licensing" Dialog


Licensing
NETRONIC VARCHART XTree 3.0
Request license information from NETRONIC...
Close
Help

Hardware identification:

Current license status:

Local License

Registration number: TX9999

Name: Demo-Version

Company name: Netronic Software GmbH

Current license status:

Developer license

Global runtime license

Single-place runtime licenses

Graphics export per API

Interactivity

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

Before licensing, the program is automatically licensed as a demo version. The demo version has the following restriction compared with the full version: The trial period for testing the product is limited to 30 days. After this period, all diagrams will have 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

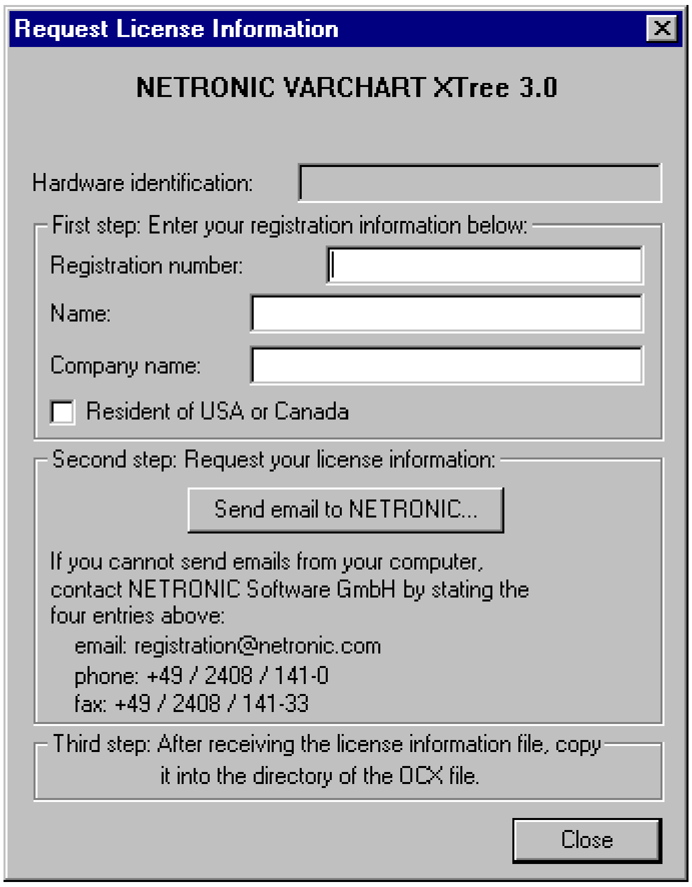
Indicates the modules that have been licenced. If the licencing procedure was successful, the licenced modules are activated.

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

Close

Quits the dialog box.

4.24 The "Request License Information" Dialog


Request License Information

NETRONIC VARCHART XTree 3.0

Hardware identification:

First step: Enter your registration information below:

Registration number:

Name:

Company name:

☐ Resident of USA or Canada

Second step: Request your license information:

Send email to NETRONIC...

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

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

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

Close

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 (vctree.lic) and send it back to you.

After having received this file, please copy it to the directory in which the file vctree.ocx is stored.

After licensing, you have to activate the new license in each of your projects. Therefore you have to open any property page in each of your projects, make any change and store it. Then the new license will be activated.

5 User Interface

5.1 Overview

The following list gives an overview of possible user interactions.

Mouse interactions:

- Generating nodes
- Marking nodes
- Cutting, copying and pasting nodes
- Editing nodes
- Editing the link appearance
- Moving nodes and their subtrees
- Arranging subtrees horizontally or vertically
- Collapsing and expanding subtrees
- Zooming

Context menus (right mouse key):

- Context menu for the diagram
- Context menu for nodes
- Context menu for links

Dialog boxes:

- Edit Data
- Page Setup
- Page Preview

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 via Keyboard

You can use the arrow buttons to move the marking from one node to the other in the selected direction.

You can scroll in the diagram via the arrow buttons while the Ctrl key is pressed.

The following buttons can be used for navigation:

- **Ctrl** + **Pos1:** scrolling to the left upper diagram border
- **Ctrl** + **End:** scrolling to the right lower diagram corner
- **Ctrl** + **screen up/down:** scrolling to the upper/lower diagram corner
- **Ctrl** + **Num +**: zoom in
- **Ctrl** + **Num -**: zoom out
- **Ctrl** + **Num ***: scroll to the next node (scroll to node)
- **Ctrl** + **Num** /: complete view

Via **Ctrl + C**, **Ctrl + X** or **Ctrl + V** respectively you can copy, cut or insert marked nodes. Via the **Del** button you can delete marked nodes.

5.3 Using the Mouse Wheel

You can scroll vertically in the diagram or in the histogram (depending on the cursor position) via turning the mouse wheel.

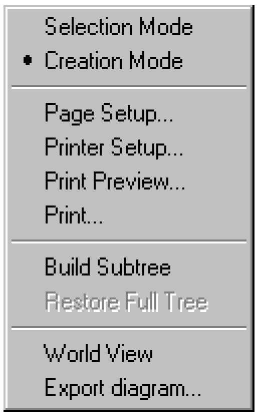
You can zoom by turning the mouse wheel while pressing the Ctrl key. The requirement for zooming is that the **ZoomingPerMouseWheelAllowed** property is activated. This property also can be set on the **General** property page. (By default, it is deactivated.)

You can scroll in any direction of the diagram by pressing the mouse wheel (or the mouse key in the middle) and move the mouse into the appropriate direction.

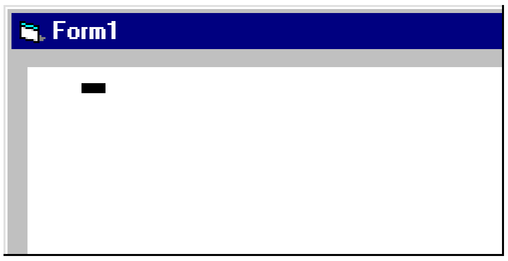
5.4 Creating Nodes

There are two modes that you can toggle between in VARCHART XTree: The **Selection mode** and the **Creation mode**. Nodes 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.

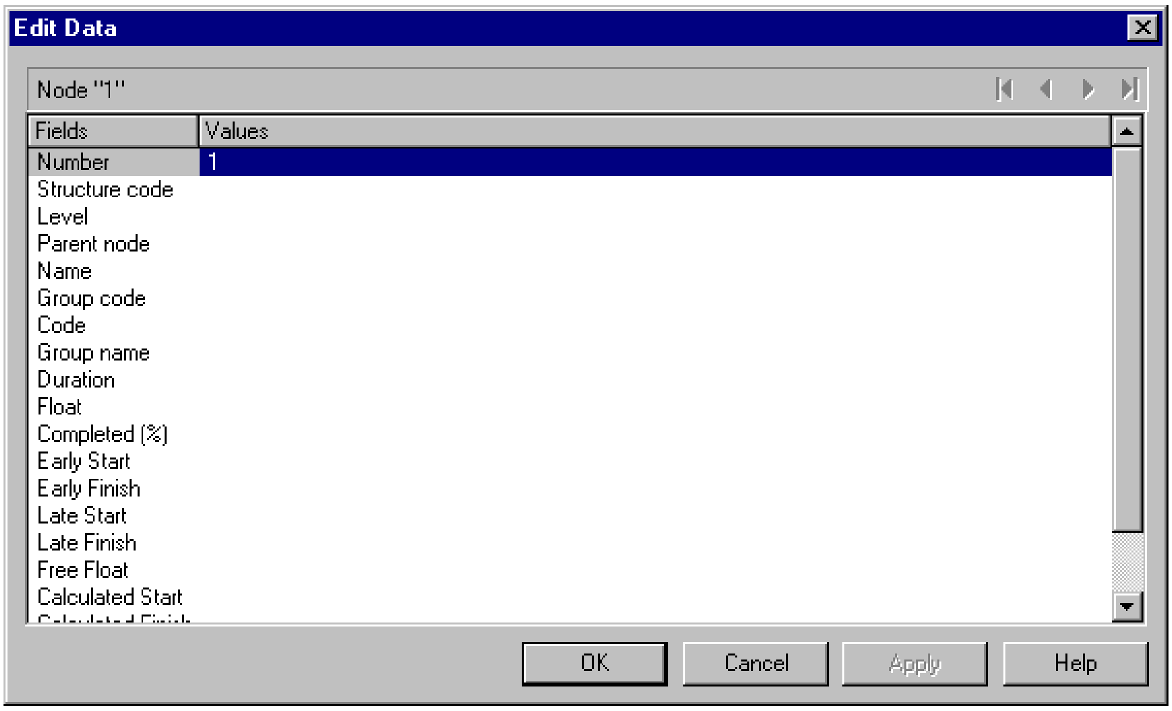
(To be able to create nodes interactively, on the **General** property page the **Allow creation of nodes** option has to be activated.)



In Creation mode the cursor will transform into a small black rectangle.



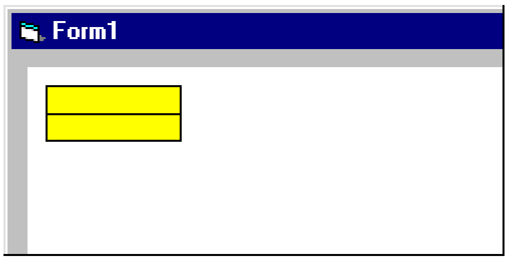
If you click on the left mouse key, two different things may happen, depending on the settings on the **General** property page. If the check box **Edit new node** was ticked, the **Edit Data** dialog will appear that displays the node data.


Edit Data

Node "1"

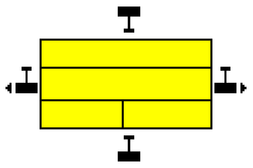
Fields	Values
Number	1
Structure code	
Level	
Parent node	
Name	
Group code	
Code	
Group name	
Duration	
Float	
Completed (%)	
Early Start	
Early Finish	
Late Start	
Late Finish	
Free Float	
Calculated Start	

The left column lists the field names of the node record, whereas the right column displays the corresponding values. Only the field "Number" has a value at this point, which is "1". You can add values, such as dates or a description. As soon as you confirm the data by the **OK** button, the node will be generated. The dialog will disappear and the node will be displayed.



If on the **General** property page the check box **Edit new node** was not ticked, a node will be displayed as soon as you click the left mouse key in an empty place of the diagram. The **Edit Data** dialog will not appear.

Further nodes you can generate from the existing node by placing the cursor near it. The cursor will change its shape according to whether the new node is going to be a parent node, a child node or a left or right brother node.



5.5 Marking nodes

Marking a node

Click the left mouse button on a node to mark it.

Collecting and toggling nodes

To bundle and toggle single nodes, press the Ctrl key and simultaneously click the left mouse button on the appropriate nodes. Each time you click on a node you toggle the marking on or off.

Marking subtrees

To mark a subtree, press the SHIFT key and click the left mouse button on the subtree´s parent node.

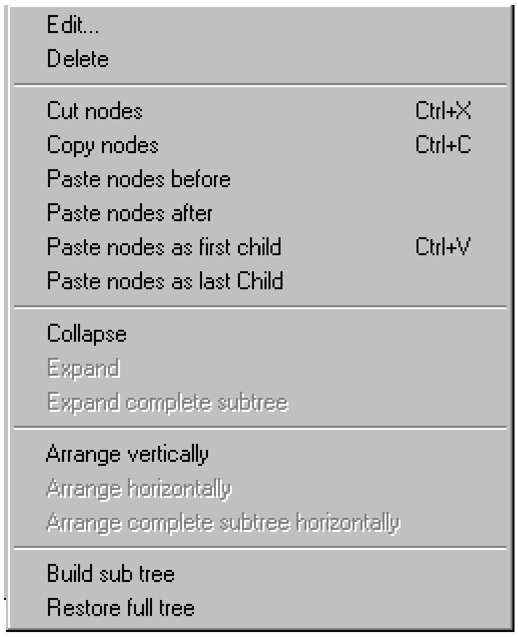
Demarking all nodes

Click the left mouse button in an empty space of the diagram to demark the marked nodes.

On the **Nodes** property page you can specify the appearance of marked nodes. Just select an entry of the **Marking type** combo box.

5.6 Deleting, Cutting, Copying and Pasting Nodes

Menu items of the context menu let you delete, cut, copy or paste nodes.



Context menu of node interactions

To paste a node, you need to mark a node in the diagram in order to place the node to be inserted

- before
- after
- as the first child node
- as the last child node

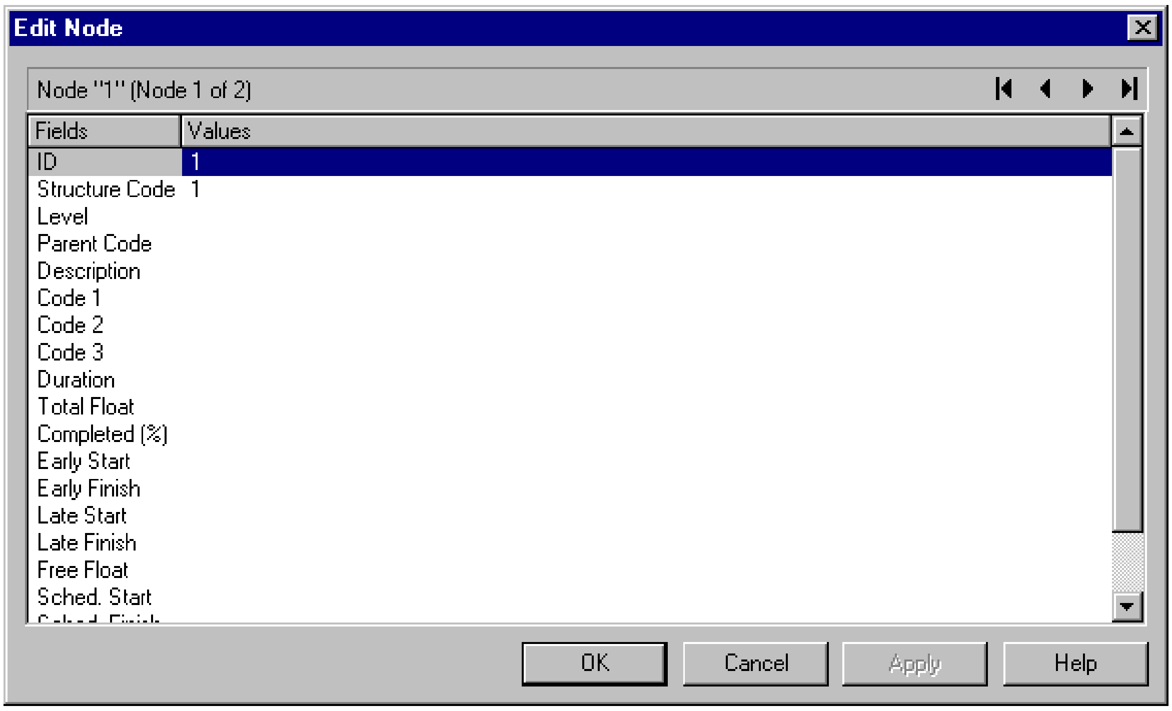
of the node marked.

You also can delete marked nodes via the Del button.

5.7 Editing Nodes

You can edit a node by either clicking on the **Edit** item of the corresponding context menu or by double-clicking on the node. Subsequently, the **Edit Data** dialog will open.

You can edit several nodes by marking them and clicking the right mouse key with the cursor placed on one of the marked nodes. The context menu for nodes will open. Select the **Edit** item to pop up the **Edit Data** dialog.



Node "1" (Node 1 of 2)

Fields	Values
ID	1
Structure Code	1
Level	
Parent Code	
Description	
Code 1	
Code 2	
Code 3	
Duration	
Total Float	
Completed (%)	
Early Start	
Early Finish	
Late Start	
Late Finish	
Free Float	
Sched. Start	
Sched. Finish	

This dialog lets you edit the data of the marked nodes right away.

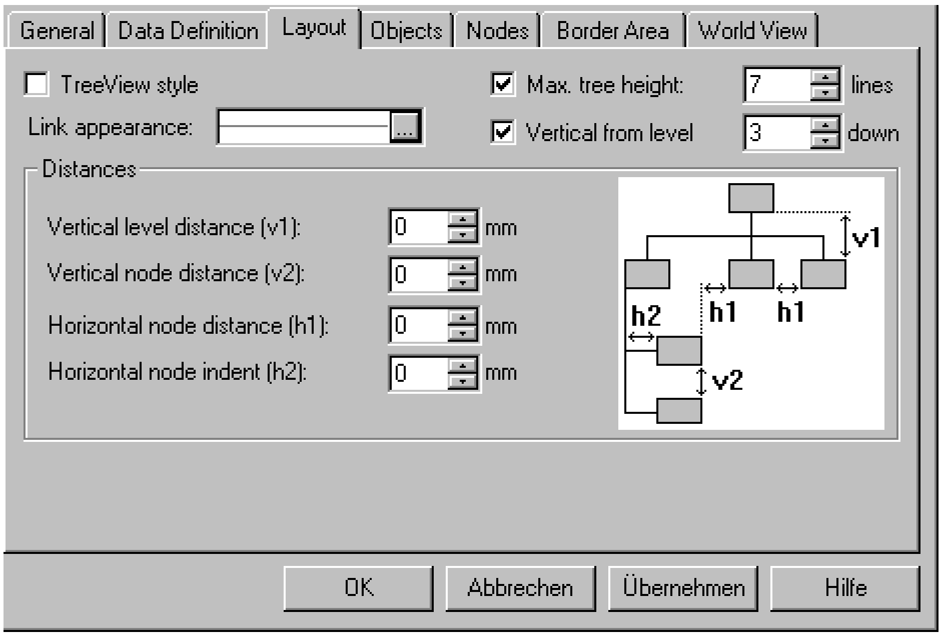
The ID of the node as well as the number of the current node out of the total number of nodes marked is indicated.

The table displays the data and values of the current node and lets you edit them. With the help of the arrow buttons above the table, you can navigate between the nodes. To store the current node data, click the **Apply** button.

5.8 Setting Link Appearances

In a tree diagram, links are generated or deleted automatically with the creation and deletion of nodes.

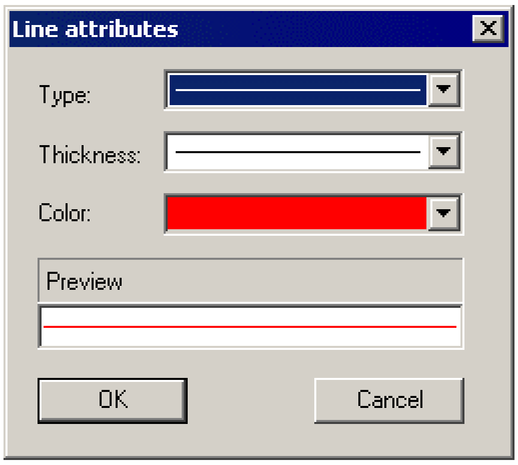
The appearance of nodes you can adjust on the **Layout** property page. The **Link appearance** field displays the type of link applying.



☐ TreeView style
Link appearance: ...
☑ Max. tree height: 7 lines
☑ Vertical from level 3 down

Distances
Vertical level distance (v1): 0 mm
Vertical node distance (v2): 0 mm
Horizontal node distance (h1): 0 mm
Horizontal node indent (h2): 0 mm

To modify the link appearance, please click on the **Edit** button to edit the line attributes. You will get to the **Line attributes** dialog where you can edit the type, thickness and color of the line.



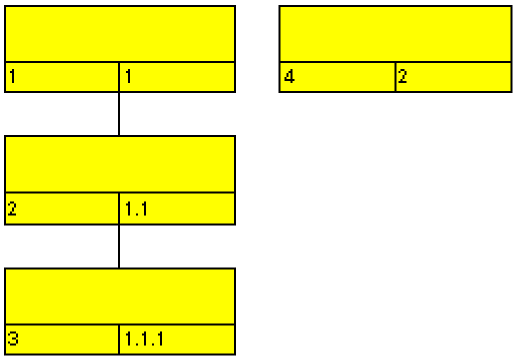
Type:
Thickness:
Color: 🟥
Preview

5.9 Appending a Node and its Associated Subtree

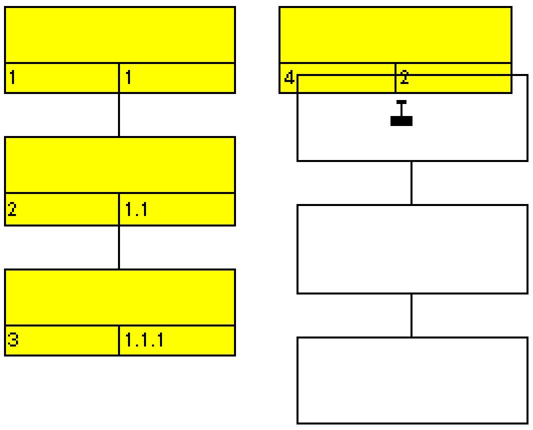
You can move and append a node and its associated subtree in one action using the drag and drop technique. Only one node and its subtree can be moved at a time, even if several nodes are selected.

Press the left mouse key and drag the node you want to move along with its child nodes to the new location. While you move the node a phantom appears for the node and its subtree. Drag the phantom over the node under/beside which you want to append the moved node and its subtree. The phantom of the appended node must at least partly cover the target node. As soon as you release the mouse key, the subtree will be appended.

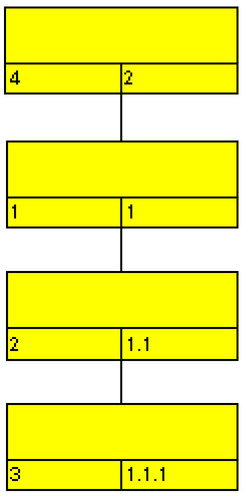
The top node of the subtree shifted will be inserted as the last child node of the target node. The node data of the subtree will automatically adapt to the new position.



You want to append the left parent node and its children below the node on the right.



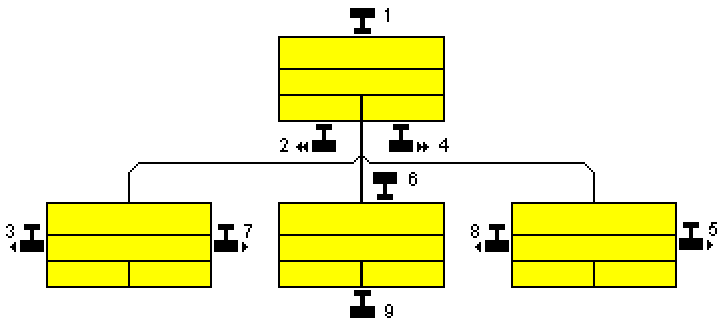
While the node and subtree are being moved a phantom appears and the cursor indicates where the moved node and subtree would be appended when the mouse key is released.



The moved subtree is appended to the node previously located on the right. The hierarchy codes have been modified accordingly.

Note: It is not possible to control the order of the child nodes directly. However, indirectly you can control the order by thoughtfully appending the child nodes to the same parent node.

The next sketch shows the different possibilities to append nodes for the horizontal arrangement:



1: New parent node for the whole branch

2: New child node to the extreme left

4: New child node to the extreme right

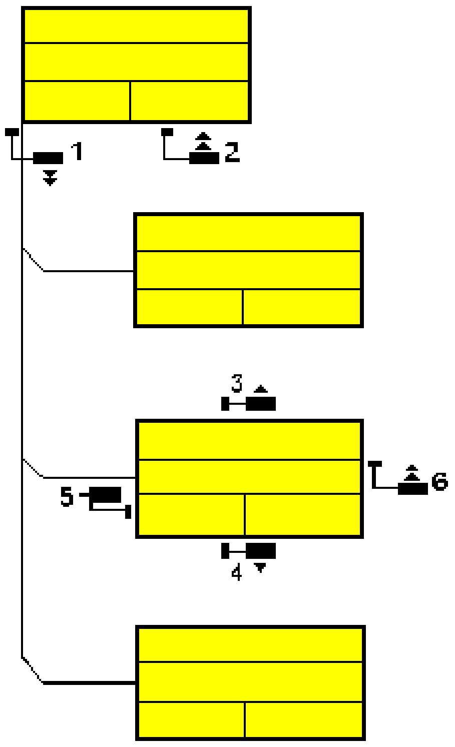
6: New parent for the node

5, 7: New brother to the right of the node

3, 8: New brother to the left of the node

9: New child for the node

The following sketch shows the different possibilities to append nodes for the vertical arrangement:



1: New child at the bottom

2: New child at the top

3: New brother above

4: New brother below

5: New parent node for the node

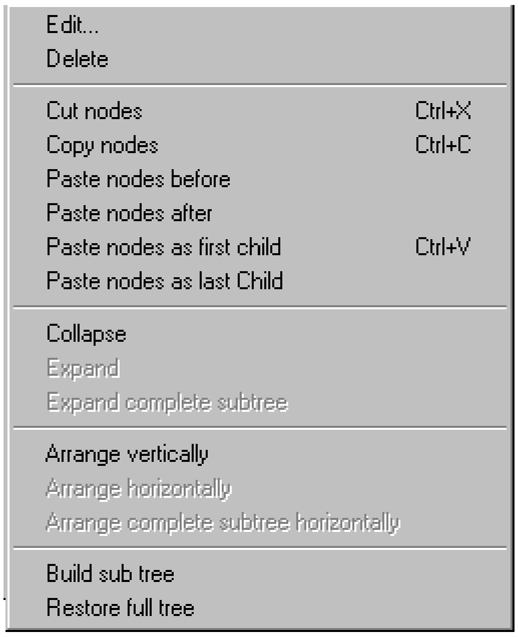
6: New child for the node

5.10 Arranging Subtrees Vertically and Horizontally

Tree structures can be arranged vertically or horizonally, either in parts or completely.

- *Horizontal arrangement:* All nodes of a level will be placed next to each other. The ports, i.e. the places where links join the nodes, will be placed in the center of the bottom line of a parent node, and in the center of the top line of a child node. A horizontal arrangement reduces the height of a tree diagram.
- *Vertical arrangement:* All nodes of a level and its sublevels will be placed beneath each other. The ports will be placed in the bottom left corner of the parent node and in the center of the left line of the child node. Vertically arranged subtrees will reduce the width of a tree diagram.

Menu items to set arrangements will be at your diposition after marking a node and pressing the right mouse key. In the context menu popping up, only the activated commands are available at this time.

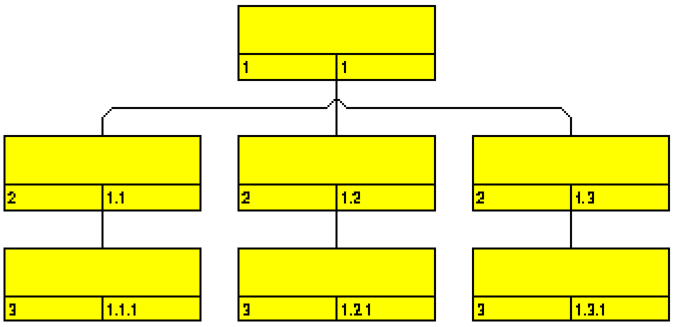


To arrange a subtree horizontally, please mark the top node and select the context menu item **Arrange horizontally**. The first level of the subtree will be arranged horizontally.

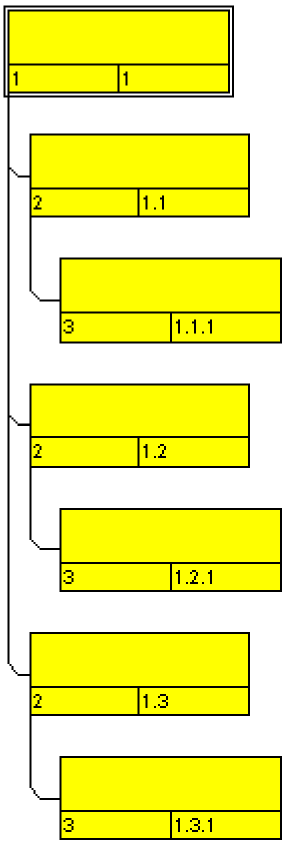
If you wish all levels of the subtree to be arranged horizontally, please select **Arrange complete subtree horizontally**.

By clicking on the command **Arrange vertically** all subtrees will be arranged vertically, starting by the first parent node marked.

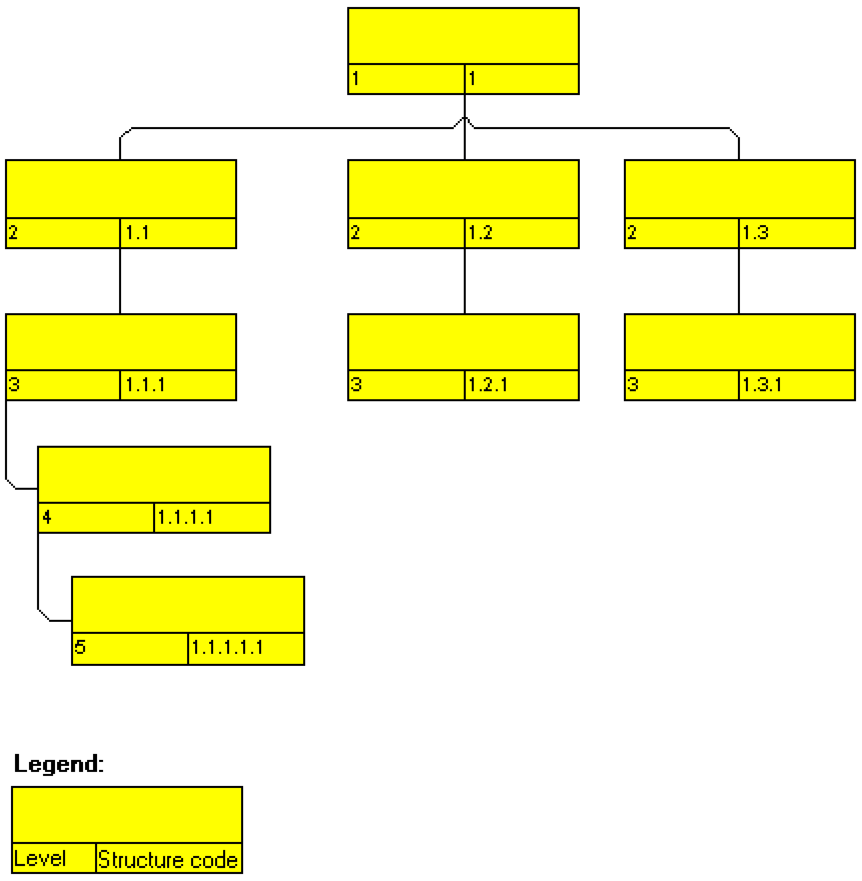
Note: If not all levels of a subtree have been arranged vertically, please check the maximum height of the tree set on the **Layout** property page. The number of levels is limited by the **Max. tree height** check box and field.



All levels arranged horizontally



All levels arranged vertically



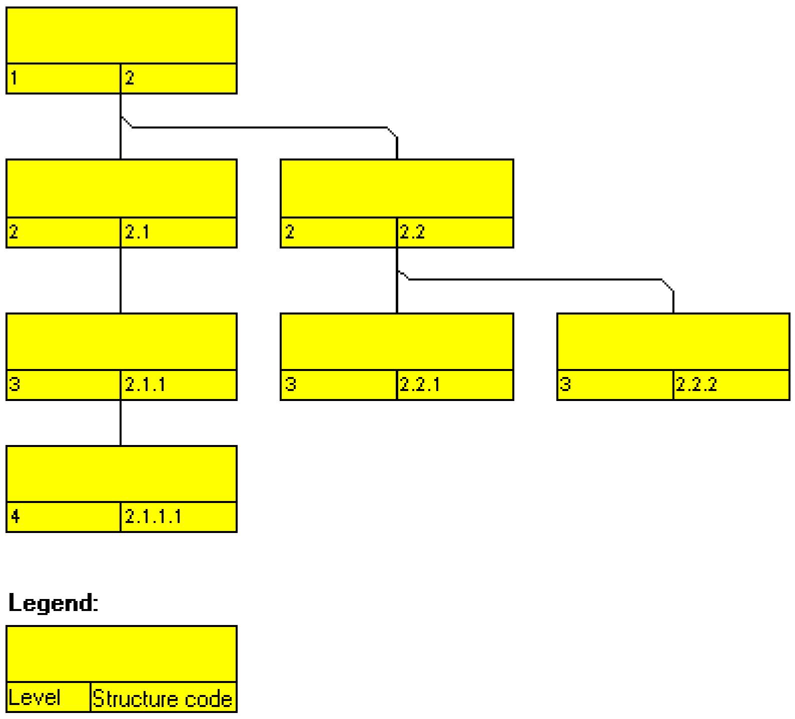
Example of a tree diagram showing vertically and horizontally arranged subtrees.

Note: Arrangement settings will affect collapsed subtrees.

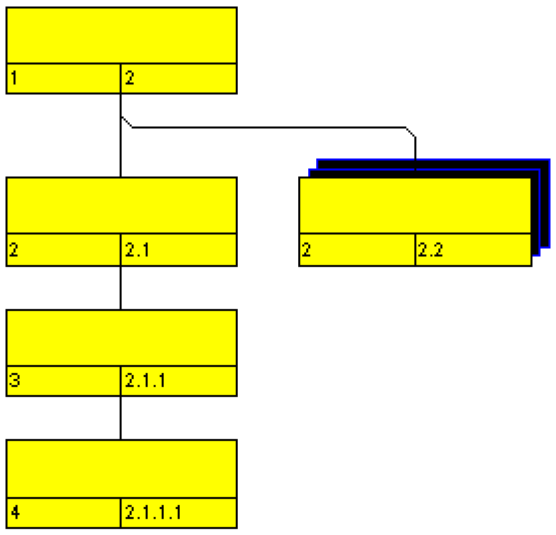
5.11 Collapsing and Expanding Subtrees

Collapsing parts of a tree diagram helps to keep complex trees well structured. It enables you to focus on certain parts of a structure while others can visually disappear. Because the information on the structure of collapsed trees is saved, no part of the total structure will be lost.

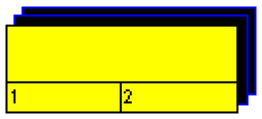
Any subtree can be collapsed, minimizing its extent to the top node of the subtree, to be expanded and displayed in its full size again. The top node of a subtree is called "structure node". It will remain visible while any other node of the subtree disappears when the subtree is collapsed.



Expanded tree



Subtree collapsed to the structure node



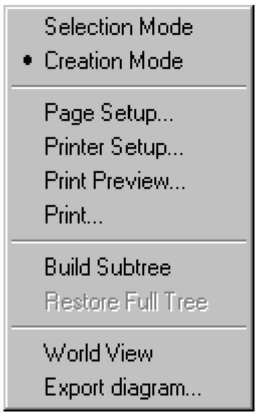
Completely collapsed tree

The menu item **Collapse** of the context menu of a node lets you collapse the subtrees that depend on the marked structure nodes. The structure nodes then represent the hidden subtrees.

The menu item **Expand** lets you expand the subtrees that are represented by the marked structure nodes. Collapsed structure nodes further down the subtree will remain collapsed. You can expand the subtree including all collapsed structure nodes further down by the menu item **Expand complete subtree**.

5.12 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 open:



Selection Mode

The selection mode is the default mode.

Creation Mode

This mode can be switched on only, if on the **General** property page the option **Allow creation of nodes** 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. You can edit all data of the node.

The creation mode can be activated by two 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 VcTree property **InteractionMode** to **vcCreateNodes**.

Page Setup

The dialog **Page Setup** appears.

Printer Setup

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 tree

(only active if nodes are marked) Select this item to display a subtree of the marked nodes.

Restore Full Tree

*(only active if the option **Build Subtree** has been selected before)* Select this item to restore the full tree.

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 shows the diagram section currently displayed in the main window.

Export Diagram

When you select this menu item, you will get to the Windows dialog box **Save as**, that lets you save the diagram as a graphics file. The formats available are:

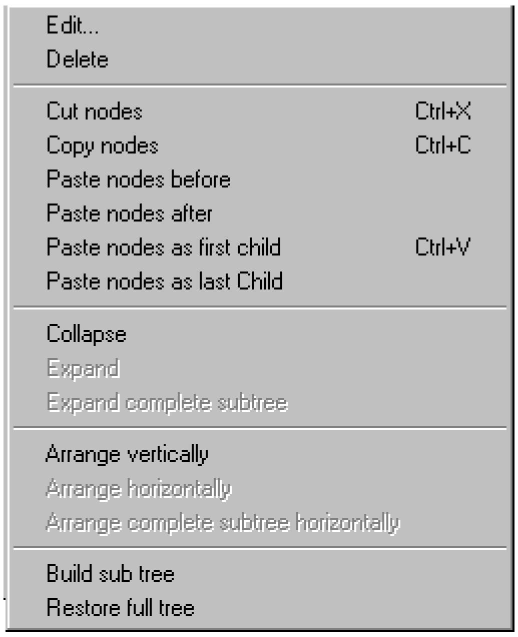
- *.BMP (Microsoft Windows Bitmap)
- *.EPS (Encapsulated Postscript)
- *.GIF (Graphics Interchange Format)
- *.JPG (Joint Photographic Experts Group)
- *.PCX (Microsoft Paint)
- *.PNG (Portable Network Graphics)
- *.TIF (Tagged Image File Format)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile)

EPS, VMF and WMF are vector formats that allow to store a file independently of pixel resolution. All other formats are pixel-orientated and confined to a limited resolution.

Files of the VMF format can be displayed via the NETRONIC WebViewer on any platform by Java compatible internet browsers. For more information please see www.netronic.de.

5.13 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.

Paste nodes before

(only active, if a node has been cut or copied before) The node cut or copied is pasted before the marked one.

Paste nodes after

The cut/copied node is pasted behind the marked one.

Paste nodes as first child

The cut/copied node will be pasted as the first child node to the marked node.

Paste nodes as last child

The cut or copied node will be inserted as the last child node of the marked node.

Collapse

The subtree(s) of the marked node will be collapsed. The marked node is transformed into a structure node, that represents the hidden subtree(s). Because the information on the structure of collapsed subtrees is saved, no part of the total structure will be lost.

Expand

Subtrees which are represented by marked structure nodes will be expanded. Only the first level beneath the structure node will be expanded; the structure nodes of all other levels will remain collapsed.

Expand complete subtree

All levels of a collapsed subtree will be expanded.

Arrange vertically

To limit the width of a tree, you can arrange subtrees vertically. In a vertical arrangement, all nodes of a level are placed beneath each other. The ports to connect a link to a node will be placed in the bottom left corner of the parent node and in the center of the right child node. By clicking on the command **Arrange vertically** all subtrees downwards from the parent node marked will be arranged vertically.

Note: If not all levels of a subtree have been arranged vertically, please check the maximum height of the tree set on the **Layout** property page. The number of levels may have been limited by the **Max. tree height** check box and field.

Arrange horizontally

To limit the height of a tree, you can arrange subtrees horizontally. All nodes of a level will be placed next to each other. The ports to connect a link will be placed in the center of the bottom line of a parent node and in the center of the top line of a child node.

To arrange a subtree horizontally, please mark its top node and select the context menu item **Arrange horizontally**. The first level of the subtree will be arranged horizontally, levels further down will remain as they were.

Arrange complete subtree horizontally

All levels of the subtree(s) of the marked node will be arranged horizontally.

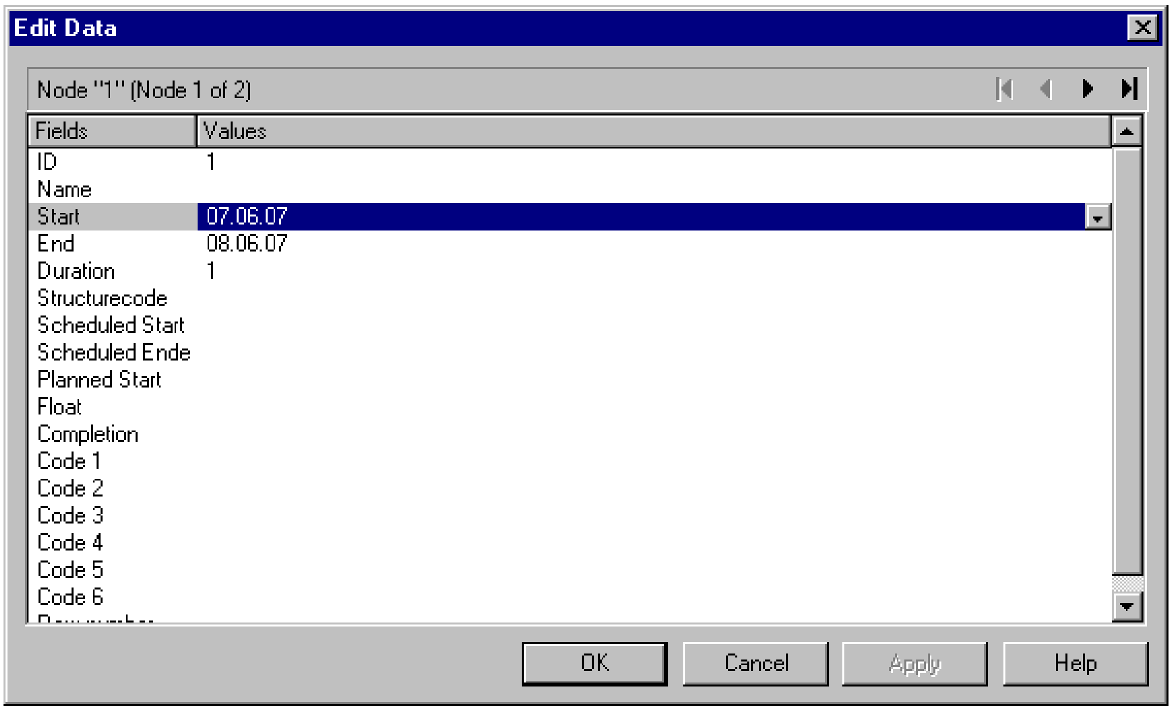
Build sub tree

A subtree of the marked nodes will be displayed.

Restore full tree

(only active if the option ***Build Subtree*** *has been selected before)* The full tree will be restored.

5.14 The "Edit Data" Dialog



Fields	Values
ID	1
Name	
Start	07.06.07
End	08.06.07
Duration	1
Structurecode	
Scheduled Start	
Scheduled Ende	
Planned Start	
Float	
Completion	
Code 1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	

This dialog you can reach via the node context menu or via a double-click on a marked node (event **OnNodeLDblClick**). It lets you edit the data of marked nodes.

The ID of the node as well as the number of the current node out of the total number of nodes marked is indicated.

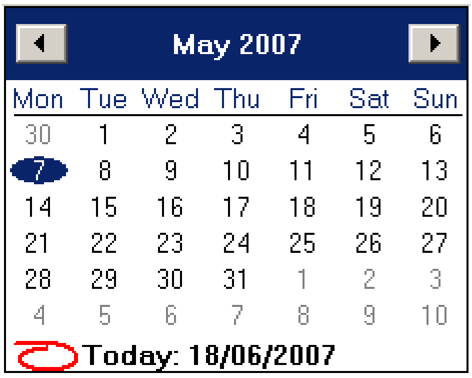
The table displays the data and values of the current node and lets you edit them. With the help of the arrow buttons above the table, you can navigate between the nodes. To store the current node data, click the **Apply** button.

Data fields

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

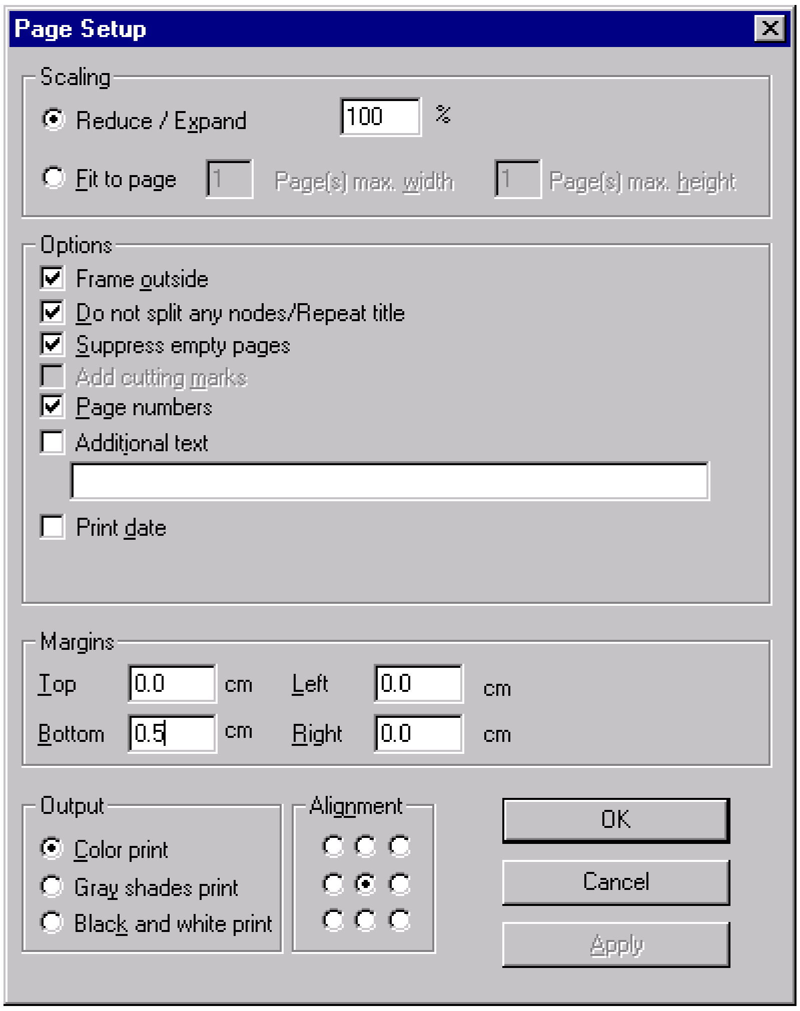
Values

This column lets you edit the values of the nodes marked, if they haven't been defined to be **Read only** on the **DataDefinition** property page. 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.

5.15 The "Page Setup" Dialog


Page Setup

Scaling
- (●) Reduce / Expand: 100 %
- ( ) Fit to page: 1 Page(s) max. width, 1 Page(s) max. height

Options
- ☑ Frame outside
- ☑ Do not split any nodes/Repeat title
- ☑ Suppress empty pages
- ☐ Add cutting marks
- ☑ Page numbers
- ☐ Additional text
- ☐ Print date

Margins
- Top: 0.0 cm
- Left: 0.0 cm
- Bottom: 0.5 cm
- Right: 0.0 cm

Output
- (●) Color print
- ( ) Gray shades print
- ( ) Black and white print

Alignment

○ ○ ○
○ ● ○
○ ○ ○

This dialog lets you set up the page.

Reduce/Expand

This field lets you set a scaling factor to your output. 100% is equivalent to the original size; a smaller value correspondingly reduces the size of the diagram, a greater value increases the size of the diagram.

Fit to page

This option regulates the paging. Specify the maximum number of pages, both width-wise and height-wise, into which the diagram may be split for the output. If necessary, one of the two values may be ignored in order to print the diagram as large as possible while preventing it from being distorted.

Frame outside

*Only activated if the **Do not split any nodes/Repeat title** check box has been ticked.* If you tick this box, each page will be given a frame, otherwise a frame will be drawn around the whole diagram.

Do not split any nodes/Repeat title

By ticking this checkbox 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.

Suppress empty pages

*Only activated if **Do not split any nodes/Repeat title** has been ticked.* If you activate this check box, empty pages will not be output.

Add cutting marks

*Only activated if you have opted to output the diagram on more than a single page and if the check box **Do not split any nodes/Repeat title** has not been ticked.* If you tick this checkbox, cutting marks will be printed on the edges of the diagram that help mounting the pages printed.

Page numbers

If you tick this check box, a page number will be displayed in the bottom left-hand corner of each page.

Additional 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:

Additional text:

{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

Print date

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

Margins

The fields **Top, Botttom, Left** and **Right** let you set the margin between the diagram and the edge of the paper sheet (unit: cm). Minimum margins existing for technical reasons cannot be overridden by the values entered here. Printers that by default print minimum margins will add the values entered here to the default minimum margins, thus resulting in broader margins than visible in these settings.

Color print

If you select this option, the diagram will be printed in color, provided your printer supports color printing.

Gray shades print

Colors will be printed in corresponding gray shades.

Black and white print

All node frames, texts and lines that are not of white color will be printed in black.

Alignment

Select one of the radio buttons to set the alignment on the paper sheet.

OK

The modifications will be saved and you will quit the dialog box.

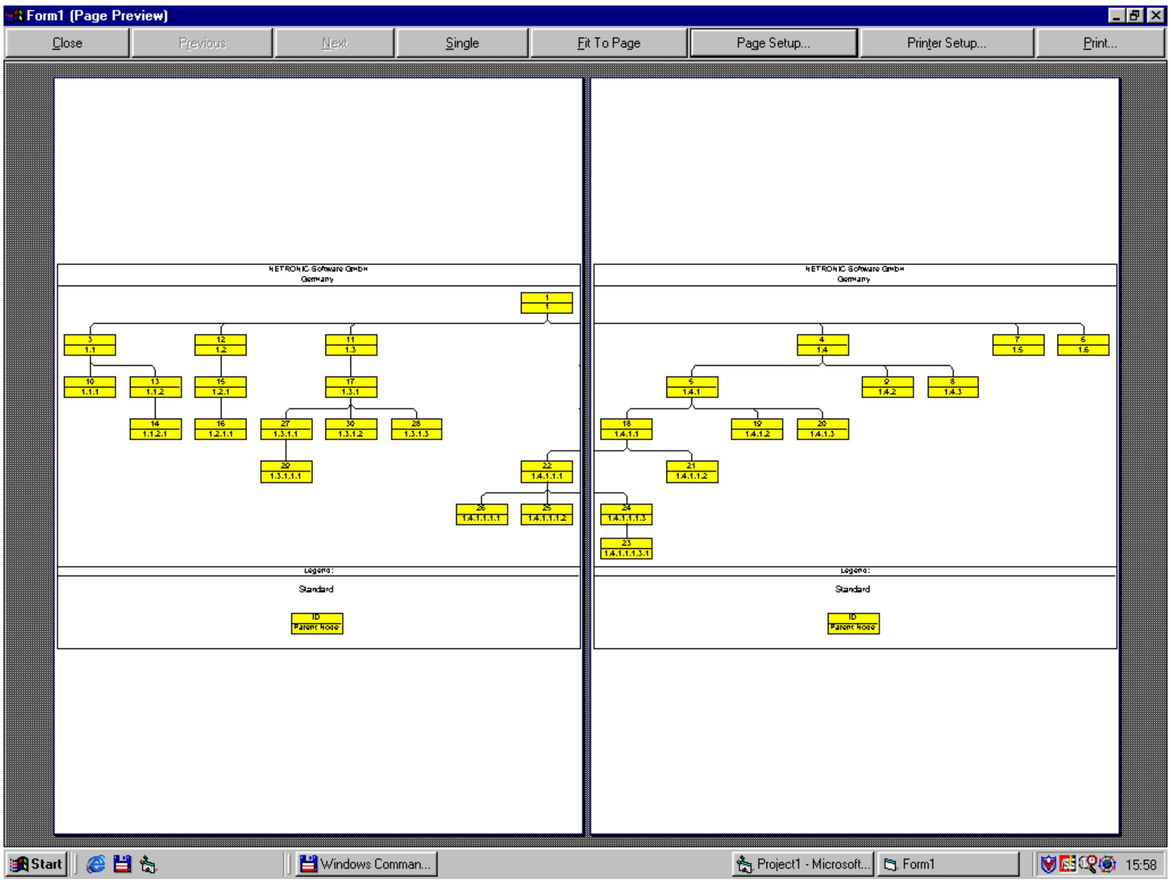
Cancel

You will quit the dialog box without saving your settings.

Apply

Press the **Apply** button to display the settings you have made in the preview. You can still cancel them by clicking on the **Cancel** button. Your settings will be saved if you press **OK**.

5.16 The "Page Preview" Dialog



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

You can view single pages or an overview of all reassembled pages.

You can zoom a section of one page of your diagram interactively by dragging a rectangle around it via the left mouse key. As soon as you release the button, the section framed will be enlarged. You can print the zoomed section by clicking on the **Print area** button.

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

If you zoomed a section from the page preview and you press the **Print area** button, the **Selection** radio button will appear activated in the Windows **Print** dialog. If you click on **OK**, the section displayed on the screeen will be printed.

Close

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

Previous

*Only activated when the **Single** button has been pressed.* If you have opted to print your diagram on more than one page in the **Page Setup** dialog box, you can press this button to view the previous page. You traverse the pages horizontally starting from the bottom right and finishing at the top left page.

Next

*Only activated when the **Single** button has been pressed.* If you have opted to print your diagram on more than one page in the **Page Setup** dialog box, 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.

Single/Overview

If you have opted to print your diagram on more than one page in the **Page Setup** dialog box, you can view each single page. Click on the **Single** button to display the top left page on your screen in full size. Then you can move through the pages by the **Next** and **Previous** buttons. A page you can zoom from the overview by a double-click. A section you can zoom by drawing a rectangle via the mouse while the left mouse button is pressed. As soon as you release the left mouse button, the selected section will be enlarged. You can print the zoomed section by clicking on the **Print area** button.

Fit To Page

If you have opted to print your diagram on more than one page in the **Page Setup** dialog box, you can press this button to re-assemble the individual pages into a complete diagram for zooming purposes.

Page Setup

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

Printer Setup

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

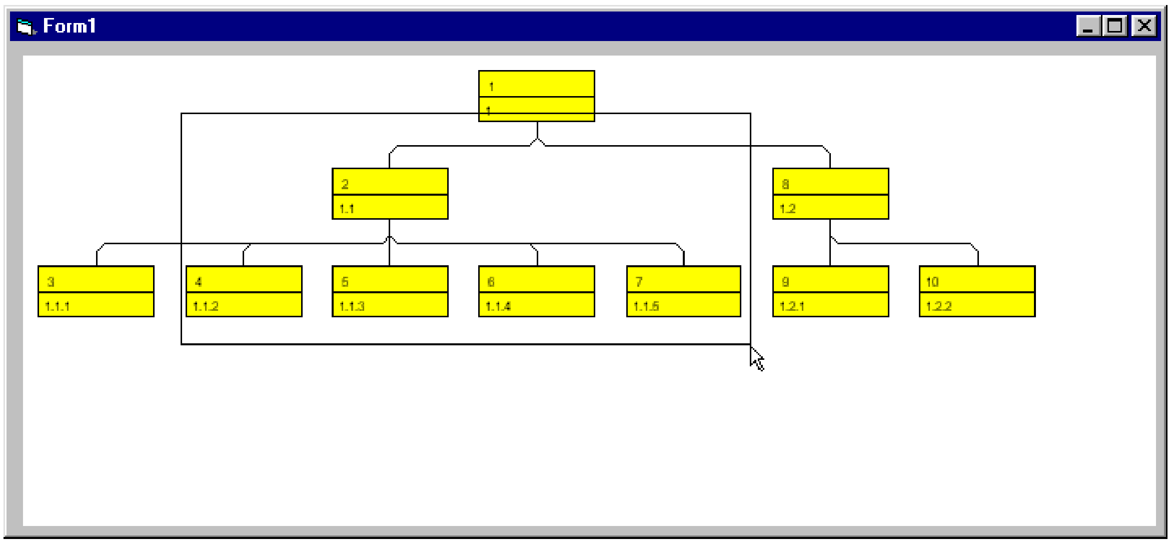
Print/Print Area

Click on this button to reach the Windows **Print** dialog box to start the print procedure. If you press the **Print Area** button for a section zoomed from the page preview, the **Selection** radio button will appear to be activated in the Windows **Print** dialog box. If you click on **OK** in the dialog box, the section displayed on the screen will be printed.

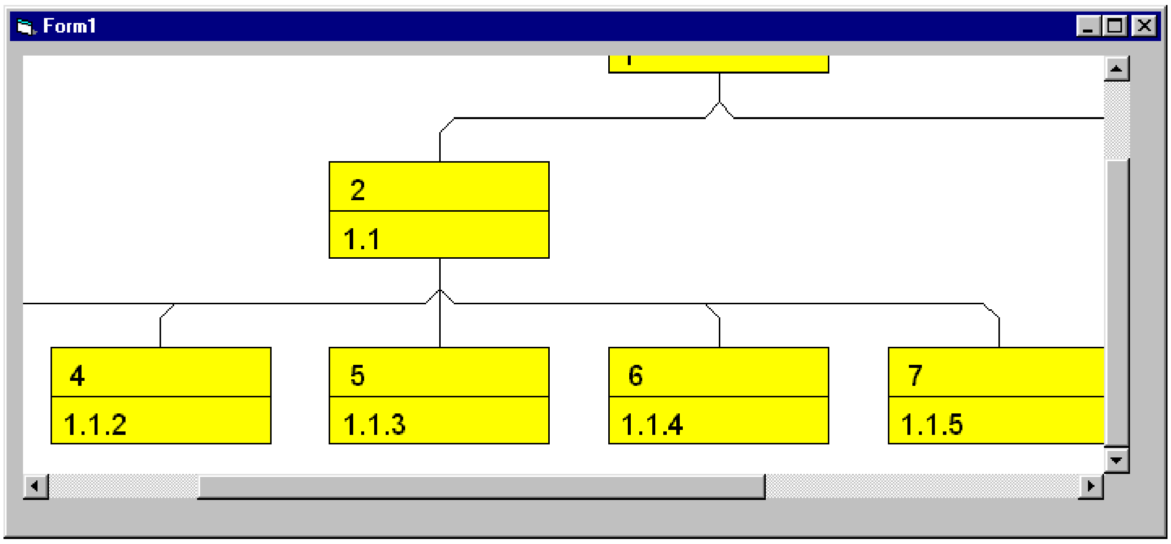
5.17 Zooming a Marked Section

You can mark a section of your diagram and display it full screen. Use the left mouse key to drag a frame around the section to be zoomed, keep the button depressed and in addition press the right mouse key. The marked section will be zoomed to full screen size. 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.



Before zooming



After zooming

6 Frequently Asked Questions

6.1 What can I do if Problems Occur during Licensing?

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

Enter your license number, your name and the name of your company and click on **Send email to NETRONIC**. An email to NETRONIC will be generated automatically. As soon as we have received it, we will generate your license information file (vcgantt.lic) and send it back to you. After having received this file, please copy it to the directory in which the file **vcgantt.ocx** is stored.

After licensing, you need to activate the new license. Please open a property page and make the system store it by making some change. This will activate the new license.

If during licensing of the VARCHART ActiveX 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.

If during licensing of the VARCHART ActiveX you receive an error message "REGSVR32 Error Return: 0X000000b6" and "Number 6877 not found in dynamic link library MFC42U.DLL", the file *mfc42u.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.

6.2 How can I Make VARCHART ActiveX Use a Modified .INI File?

Some of the VARCHART ActiveX 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. You can now make your changements on your *.ini file, that in our example is *project.ini*, by a text file editor and save your changements.
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 effective from now on.

6.3 How can I Activate the License File?

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

Enter your license number, your name and the name of your company and click on **Send email to NETRONIC**. An email to NETRONIC will be generated automatically. As soon as we have received it, we will generate your license information file (vcgantt.lic) and send it back to you. After having received this file, please copy it to the directory in which the file **vcgantt.ocx** is stored.

After licensing, you need to activate the new license. Please open a property page and make the system store it by making some change. This will activate the new license.

If during licensing of the VARCHART ActiveX 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.

If during licensing of the VARCHART ActiveX you receive an error message "REGSVR32 Error Return: 0X000000b6" and "Number 6877 not found in dynamic link library MFC42U.DLL", the file *mfc42u.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.

6.4 What have Borland Delphi Users to do for the Upgrade of a New VARCHART XTree Version?

After the upgrade or update of the VARCHART XTree to a new version it is necessary to install this new version in the Delphi Package Borland User Components. Please proceed as follows:

1. Start Borland Delphi.
2. Click onto **Components** and **ActiveX import**.
3. Select *NETRONIC VARCHART XTree* from the ActiveX Controls list and click onto the **Remove** button to remove the registration. Quit the dialog with **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 be opened.
6. Select one after the other the files *VcTreeLib_TLB.pas* and *VcTreeLib_TLB.dcr* and select them by a right-click from the project.
7. Compile the package and close the dialog. Thus the changes will be saved in the project *dclusrX0.*
8. Now reopen the dialog **Components > ActiveX import**.
9. Click on **Add**, select *vctree.ocx*, and click on **Open**. Now you can see *NETRONIC VARCHART XTree* 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 *dclusrX0*.

6.5 Which DLL Files are Needed for Licensing?

The following shipped DLL files are necessary for registering:

(Please check at first if the file *shlwapi.dll* is available.)

- *COMCTL32.DLL* (Version 4.72)
- *WININET.DLL* (Version 4.72)
- *MFC42U.DLL* (Version 6.0) *
- *MSVCRT.DLL* (Version 6.0)
- *MSVCP60.DLL* (Version 6.0)
- *OLEAUT32.DLL* (Version 2.20) *
- *OLEPRO32.DLL* (Version 2.20) *
- *SHLWAPI.DLL* (Version 4.72)
- *UNICOWS.DLL* (Version 1.0) (only for Windows 95/98/ME)
- *VCPANE32U.DLL* (Version 4.1)
- *VCPRCT32U.DLL* (Version 4.1)
- *VCWIN32U.DLL* (Version 4.1)
- *VXCSV32U.DLL* (Version 1.300)

* These DLL files have to be registered during the installation in the same way as the VARCHART ActiveX.

6.6 Why can I not Create NodesInteractively at Times?

If during runtime you cannot create nodes via the mouse, please verify if the check box **Allow creation of nodes** on the **General** property page has been activated. As soon as you have ticked it, you will be able to create nodes interactively.

Check if the VARCHART VcTree property **AllowNewNodes** has not been set to **False**.

6.7 How can I Disable the Interactive Creation of Nodes?

There are several ways to revoke interactive creating of nodes:

1. You can deactivate the check box **Allow creation of nodes** on the **Nodes** 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
   VcTree1.AllowNewNodes = 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 VcTree1_OnDiagramRClick(ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

'switching off the context menu of nodes
Private Sub VcTree1_OnNodeRClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As
VcTreeLib.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?

For projects with many nodes the updating procedure can spend too much time when a certain action is repeated for many nodes. You can accelerate the updating procedure via the **SuspendUpdate** method. Put the code that describes the repeated action into brackets of **SuspendUpdate (True)** and **SuspendUpdate (False)** like in the following example. Then the nodes will be updated all at once, and the performance will be improved.

Example Code

```
VcTree1.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

VcTree1.SuspendUpdate (False)
```

➤ Graphics

Another reason for a low performance may be graphics in table, node or box fields that are too large or that have 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.

7 API Reference

7.1 Object types

- DataObject
- DataObjectFiles
- VcBorderArea
- VcBorderBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcDataDefinition
- VcDataDefinitionTable
- VcDefinitionField
- VcFilter
- VcFilterCollection
- VcFilterSubCondition
- VcMap
- VcMapCollection
- VcMapEntry
- VcNode
- VcNodeAppearance
- VcNodeAppearanceCollection
- VcNodeCollection
- VcNodeFormat
- VcNodeFormatCollection
- VcNodeFormatField
- VcPrinter
- VcRect
- VcTree
- VcWorldView

7.2 DataObject

DataObject

The OLE Drag & Drop technique allows to move data from a source component to a target component. The components may be a VARCHART ActiveX control or any other application supporting OLE Drag & Drop operations, such as Microsoft Word. A user for example marks a node in a VARCHART ActiveX control (source) and shifts it to a different X-control (target).

The DataObject is a container for data being transferred from the source to the target. For storing, accessing and analysing the data, the object offers five different methods and properties: **Files**, **Clear**, **GetData**, **GetFormat** and **SetData**.

In a DataObject object data can be stored simultaneously in any format known. For example, a square section containing a text that is extracted from a graphics software can be stored as a string and as a bitmap at the same time. The target component will load the format that it can read best. VARCHART ActiveX controls can save and load data in string formats only.

Properties

- DropInsertionPosition
- Files

Methods

- Clear
- GetData
- GetFormat
- SetData

Properties

DropInsertionPosition

Read Only Property of DataObject

This property lets you require the current insertion position in the event **OLEDragOver** or **OLEDragDrop** (OLE Drag & Drop).

This property lets you insert nodes manually. If you require the reference node via **IdentifyObjectAt** at the same time, you can insert a node via **InsertNodeRecordEx** at the insertion position specified.

	Data Type	**Explanation**
Property value	InsertionPositionEnum	insertion position
	Possible Values:	
	vcIPFirstChild 3	Insertion as first child of reference node
	vcIPLastChild 4	Insertion as last child of reference node
	vcIPLeftBrother 31	Insertion as left brother of reference node
	vcIPNone 0	unallowed insertion position (valid only for DataObject.DropInsertionPosition)
	vcIPNormal 1	Insertion without reference node
	vcIPParent 6	Insertion as parent of reference node
	vcIPRightBrother 32	Insertion as right brother of reference node

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 form of a **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 via the **RegisterClipboardFormat()** API function. However, there are a few caveats:

The **GetData** method always returns data in a byte array when it is in a format that it doesn't 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 data's format, and knows only the amount of memory that the operating system has allocated for the data. This allocation of memory often is larger than actually required for the data. Therefore, there may be extraneous bytes near the end of the allocated memory segment. As a result, you must use appropriate functions to interpret the returned 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**, **OLESetData**, **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

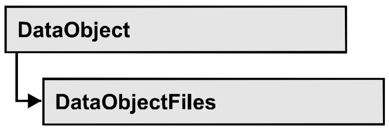
Windows via the **RegisterClipboardFormat()** API function. However, there are a few caveats:

The **SetData** method requires the data to be in the form of a byte array when it does not recognize the data format specified.

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



his object keeps a list of all file names, that are stored in a DataObject, if it contains data of format **15** (list of files).

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. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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

```
Private Sub VcGantt1_OLEDragOver(ByVal data As VcGanttLib.DataObject, effect As
Long, ByVal button As Integer, ByVal Shift As Integer, ByVal x As Long, ByVal y
As Long, ByVal state As VcGanttLib.OLEDragStateEnum)

Dim fileName as String
For Each fileName In DataObject.DataObjectFiles
   Debug.Print fileName
Next

End Sub
```

Count

Read Only Property of DataObjectFiles

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

	Data Type	Explanation
Property value	Long	Number of files

Item

Property of DataObjectFiles

By this property you can assign or retrieve a file name via 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	

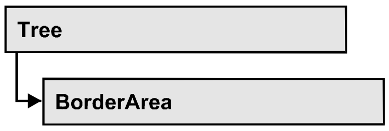
Remove

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

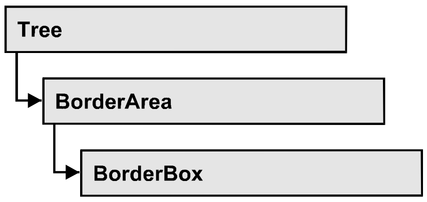
This method gives access to a BorderBox object.

	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 bordArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordArea = VcTree1.BorderArea
Set bBoxBBL = bordArea.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

- GraphicsFileName
- LegendFont
- LegendTitle
- LegendTitleFont
- LegendTitleVisible
- Text
- TextFont
- Type

Properties

GraphicsFileName

Property of VcBorderBox

This property lets you set/retrieve the name of the graphics file used in the VcBorderBox object.

	Data Type	Explanation
Property value	String	name of the graphics file

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxTR As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxTR = bordArea.BorderBox(vcBBXPTopRight)
bBoxTR.Type = vcBBXTGraphics
bBoxTR.GraphicsFilename = "Asterix.jpg"
```

LegendFont

Property of VcBorderBox

This property lets you set/retrieve the font attributes of the legend.

	Data Type	Explanation
Property value	StdFont	font attributes of the legend

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTLegend
logThis (bBoxBBL.LegendFont.Name)
```

LegendTitle

Property of VcBorderBox

This property lets you set/retrieve the legend title.

	Data Type	Explanation
Property value	String	legend title

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

LegendTitleFont

Property of VcBorderBox

This property lets you set/retrieve the font attributes of the legend title.

	Data Type	Explanation
Property value	StdFont	font attributes of the legend title

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTLegend
logThis (bBoxBBL.LegendTitleFont.Name)
```

LegendTitleVisible

Property of VcBorderBox

This property lets you set/retrieve whether the legend title is visible.

	Data Type	Explanation
Property value	Boolean	legend title visible (True)/ not visible (False)

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitleVisible = False
```

Text

Property of VcBorderBox

This property lets you set/retrieve the text of a title line (above or below the diagram). For numbering the pages or displaying the system date 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
{SYSTEMDATE}	= system date

	Data Type	Explanation
Parameter:		
rowIndex	Index	row index {0...5}

Property value	String	text in text boxes

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.BorderBox(vcBBXPBottomBottomLeft)
bBoxBBL.Type = vcBBXTText
bBoxBBL.Text(index) = "Department A"
```

TextFont

Property of VcBorderBox

This property lets you set/retrieve the font attributes of a title line (above or below the diagram).

	Data Type	**Explanation**
Property value	StdFont	font attributes of the text

Example Code

```
Dim bordarea As VcBorderArea
Dim bBoxTL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.BorderBox(vcBBXPBottomBottomLeft)

bBoxTL.TextFont(i).Bold = False
bBoxTL.TextFont(i).Italic = False
bBoxTL.TextFont(i).Name = "Symbol"
```

Type

Property of VcBorderBox

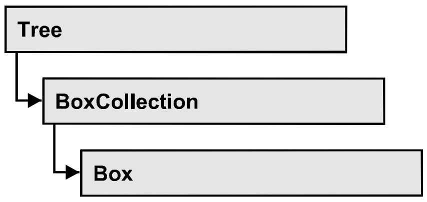
This property lets you set/retrieve the type of the BorderBox object (text, legend, graphics, empty).

	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 bordarea As VcBorderArea
Dim bBoxBBL As VcBorderBox

Set bordarea = VcTree1.BorderArea
Set bBoxBBL = bordArea.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
- Visible

Methods

- GetXYOffset
- GetXYOffsetAsVariant
- SetXYOffset

Properties

FieldText

Property of VcBox

This property lets you specify/enquire 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 boxCollection As VcBoxCollection
Dim box As VcBox

Set boxCollection = VcTree1.boxCollection
Set box = boxCollection.FirstBox
box.FieldText(0) = "User: "
```

FormatName

Property of VcBox

This property lets you specify/enquire the name of the box format.

	Data Type	**Explanation**
Property value	VcBoxFormat	BoxFormat object or **Nothing**

LineColor

Property of VcBox

This property lets you set/enquire the color of the border line of the box.

	Data Type	**Explanation**
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

LineThickness

Property of VcBox

This property lets you specify/enquire the line thickness of the border line of the box.

	Data Type	Explanation
Property value	Integer	Line thickness LineType {vcDashed...vcSolid} allows LineThickness {1...4} (in Pixeln) LineType {vcLineType0... LineType18} allows Integer values {1...1000} (in 1/100 mm) **Default value:** As defined in the dialog

LineType

Property of VcBox

This property lets you set/enquire 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 assigned
vcSolid 2	Line solid

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

Moveable

Property of VcBox

This property lets you specify/enquire whether the box can be moved interactively.

	Data Type	Explanation
Property value	Boolean	moveable (True)/ not moveable (False) **Default value:** True

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 boxCollection As VcBoxCollection
Dim box As VcBox
Dim boxName As String

Set boxCollection = VcTree1.boxCollection
Set box = boxCollection.FirstBox
boxName = box.Name
MsgBox boxName
```

Origin

Property of VcBox

This property lets you set/enquire the origin of the box, i. e. the point of the diagram from which the offset to the reference point of the box will be measured.

With the help of the properties **Origin**, **ReferencePoint** and the method **GetXYOffset** you can position each box in the diagram area. The relative position of the boxes is independent of the current diagram size.

	Data Type	Explanation
Property value	BoxOriginEnum	origin of the box
	Possible Values:	
	vcBOBottomCenter 28	bottom center
	vcBOBottomLeft 27	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

Priority

Property of VcBox

This property lets you specify or enquire the priority of the box.

	Data Type	Explanation
Property value	Integer	Priority value

ReferencePoint

Property of VcBox

This property lets you set/enquire 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	reference point of the box
	Possible Values:	
	vcBRPBottomCenter 28	bottom center
	vcBRPBottomLeft 27	bottom left
	vcBRPBottomRight 29	bottom right
	vcBRPCenterCenter 25	center center
	vcBRPCenterLeft 24	center left
	vcBRPCenterRight 26	center right
	vcBRPTopCenter 22	top center
	vcBRPTopLeft 21	top left
	vcBRPTopRight 23	top right

Specification

Read Only Property of VcBox

This property lets you enquire the specification of a box. The specification is a string that contains only legible ASCII signs in the area 32 to 127 and that thus can be stored in text files or databases without problems. This is necessary for the persistency. Specifications can be used for the recreation of boxes via the method VcBoxCollection.AddBySpecification.

	Data Type	Explanation
Property value	String	Specification of the box

Visible

Property of VcBox

This property lets you specify/enquire 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 boxCollection As VcBoxCollection
Dim box As VcBox

Set boxCollection = VcTree1.BoxCollection
Set box = boxCollection.FirstBox
box.Visible = False
```

Methods

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	Long	x value of the offset
⇦ yOffset	Long	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 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.

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.

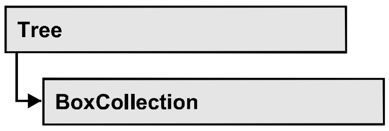
Note: If you use VBScript, you can only use the analogous method **GetXYOffsetAsVariant** because of the parameters by Reference.

	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 = VcTree1.boxCollection.FirstBox.SetXYOffset(100, 100)
```

7.7 VcBoxCollection



The VcBoxCollection object contains all boxes available. You can retrieve a box by the method **BoxByName**. The number of boxes in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- 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. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 box As VcBox

For Each box In VcTree1.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 boxCollection As VcBoxCollection
Dim numberOfBoxes As Long

Set boxCollection = VcTree1.BoxCollection
Dim numberOfBoxes = boxCollection.Count
```

Methods

Add

Method of VcBoxCollection

By this method you can create a box as a member of the BoxCollection. If the name has not been used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ boxName	String	box name
Return value	VcBox	new box object

Example Code

```
Set newBox = VcTree1.BoxCollection.Add("box1")
```

AddBySpecification

Method of VcBoxCollection

By this method you can create a box via a box specification. This is necessary for the persistency of box objects. The specification of a box can be read (see VcBox property **Specification**) and saved. In the next session the box will be created again with the specification that is read again and with its stored name.

	Data Type	Explanation
Parameter:		
⇨ Specification	String	box specification
Return value	VcBox	new box object

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 boxCollection As VcBoxCollection
Dim box As VcBox

Set boxCollection = VcTree1.BoxCollection
Set box = boxCollection.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.

	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

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 boxCollection As VcBoxCollection
Dim box As VcBox

Set boxCollection = VcTree1.BoxCollection
Set box = boxCollection.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	succeeding box

Example Code

```
Dim boxCollection As VcBoxCollection
Dim box As VcBox

Set boxCollection = VcTree1.BoxCollection
Set box = boxCollection.FirstBox

While Not box Is Nothing
    Listbox.AddItem box.Name
    Set box = boxCollection.NextBox
Wend
```

Remove

Method of VcBoxCollection

This method lets you delete a box.

	Data Type	Explanation
Parameter:		
⇨ boxName	String	box name
Return value	Boolean	box deleted (True)/not deleted (False)

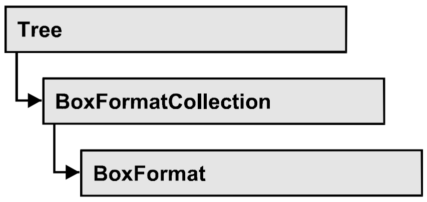
Update

Method of VcBoxCollection

This method lets you update a box collection after having modified it.

	Data Type	Explanation
Return value	Boolean	update successfull (True)/ not successfull (False)

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. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 formatField As VcBoxFormatField

For Each formatField In format
    Debug.Print formatField.Index
Next
```

FieldsSeparatedByLines

Property of VcBoxFormat

This property lets you specify/enquire 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).

FormatField

Read Only Property of VcBoxFormat

This property gives access to a VcBoxFormatField object via index. The index has to be in the range 0 to .FormatFieldCount-1.

Note for users of a version elder than 3.0: The index is **not** in the range 1 to .FormatFieldCount as in elder versions.

	Data Type	Explanation
Parameter:		
index	Integer	index of the box format field
		0 ... .FormatFieldCount-1
Property value	VcBoxFormatField	box format field

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

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

Specification

Read Only Property of VcBoxFormat

This property lets you enquire the specification of the box format. The specification is a string that contains only legible ASCII signs in the area 32 to 127 and that thus can be stored in text files or databases without problems. This is necessary for the persistency. Specifications can be used for the recreation of box formats via 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	position of the new box format field
	Possible Values:	
	vcInnerAbove 1	above
	vcInnerBelow 3	below
	vcInnerLeftOf 0	left of
	vcInnerRightOf 4	right of

⇨ refIndex	Integer	index of the reference box format field
Return value	VcBoxFormatField	box format field object

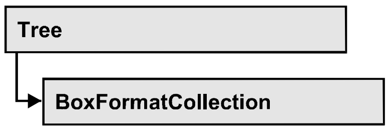
RemoveFormatField

Method of VcBoxFormat

This method lets you remove a box format field via its index. After that, the program will set all box format field indexes newly in order to number them consecutively.

	Data Type	**Explanation**
Parameter:		
⇨ index	Integer	index of the box format field to be deleted

7.9 VcBoxFormatCollection



The VcBoxFormatCollection object contains all box formats available. You can retrieve a box format by the method **FormatByName**. The number of box formats in the collection object can be retrieved by the property **Count**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcBoxFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 format As VcBoxFormat

For Each format In VcTree1.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 boxFormatCollection As VcBoxFormatCollection
Dim numberOfBoxformats As Long

Set boxFormatCollection = VcTree1.BoxFormatCollection
Dim numberOfBoxformats = boxFormatCollection.Count
```

Methods

Add

Method of VcBoxFormatCollection

By this method you can create a box format as a member of the BoxFormatCollection. If the name has not been used before, the new box object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	Explanation
Parameter:		
⇨ FormatName	String	box format name
Return value	VcBoxFormat	new box format object

Example Code

```
Set newBoxFormat = VcTree1.BoxFormatCollection.Add("boxformat1")
```

AddBySpecification

Method of VcBoxFormatCollection

By this method you can create a box format via a box format specification. This is necessary for the persistency of box format objects. The specification of a box can be read (see VcBoxFormat property **Specification**) and saved. In the next session the box format will be created again with the specification that is read again and with its stored 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

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 = VcTree1.BoxFormatCollection.FirstFormat
```

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 formatCollection As VcBoxFormatCollection
Dim format As VcBoxFormat

Set formatCollection = VcTree1.BoxFormatCollection
Set format = formatCollection.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	following box format

Example Code

```
Dim formatCollection As VcBoxFormatCollection
Dim format As VcBoxFormat

Set formatCollection = VcTree1.BoxFormatCollection
Set format = formatCollection.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCollection.NextFormat
Wend
```

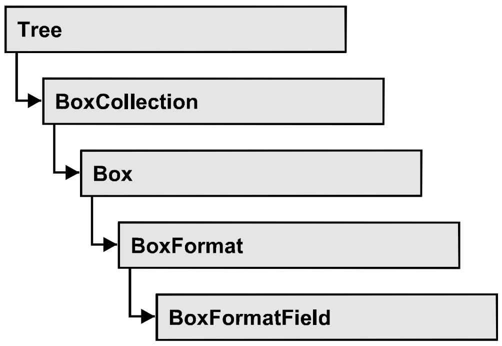
Remove

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)

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
- BackColor
- FormatName
- GraphicsHeight
- Index
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- TextFont
- TextFontColor
- Type

Properties

Alignment

Property of VcBoxFormatField

This property lets you set/enquire the alignment of the content of the box format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum	alignment of the field content
	Possible Values:	

vcFFABottom 28	bottom
vcFFABottomLeft 27	bottom left
vcFFABottomRight 29	bottom right
vcFFACenter 25	center
vcFFALeft 24	left
vcFFARight 26	right
vcFFATop 22	top
vcFFATopLeft 21	top left
vcFFATopRight 23	top right

BackColor

Property of VcBoxFormatField

This property lets you set/enquire the background color of the box format field. If the box format field shall have the background color of the box format, select the value **-1**.

	Data Type	Explanation
Property value	OLE_COLOR	background color of the box format **Default value:** -1

FormatName

Read Only Property of VcBoxFormatField

This property lets you enquire the name of the box format to which this box format field belongs.

	Data Type	Explanation
Property value	String	name of the box format

GraphicsHeight

Property of VcBoxFormatField

This property lets you set/enquire 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

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
Property value	Integer	index of the box format field

MaximumTextLineCount

Property of VcBoxFormatField

only for the type vcFFTText: This property lets you set/enquire the maximum number of lines in the box format field.

	Data Type	Explanation
Property value	Integer	maximum number of lines 0 ... 9

MinimumTextLineCount

Property of VcBoxFormatField

only for the type vcFFTText: This property lets you set/enquire the minimum number of lines in the box format field. If the text of the box format field is too long, it will be enlarged dynamically up to the maximum number of lines.

	Data Type	Explanation
Property value	Integer	minimum number of lines 0 ... 9

MinimumWidth

Property of VcBoxFormatField

This property lets you set/enquire 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

TextFont

Property of VcBoxFormatField

only for the type vcFFTText: This property lets you set/enquire the font of the box format field.

	Data Type	Explanation
Property value	StdFont	font type of the box format

TextFontColor

Property of VcBoxFormatField

only for the type vcFFTText: This property lets you set/enquire the font color of the box format field.

	Data Type	Explanation
Property value	OLE_COLOR	font color of the box format **Default value:** -1

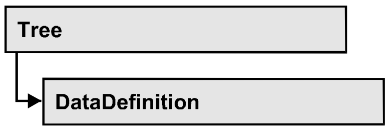
Type

Property of VcBoxFormatField

This property lets you enquire the type of the box format field.

	Data Type	Explanation
Property value	FormatFieldTypeEnum	type of the box format field
	Possible Values:	
	vcFFTGraphics 64	graphics
	vcFFTText 36	text

7.11 VcDataDefinition



The data of nodes can be defined on the **DataDefiniton** property page. It grants access to the names and types of the available fields. The data definition of a VcTree object contains the data definition table vcMaindata.

Properties

- DefinitionTable

Properties

DefinitionTable

Read Only Property of VcDataDefinition

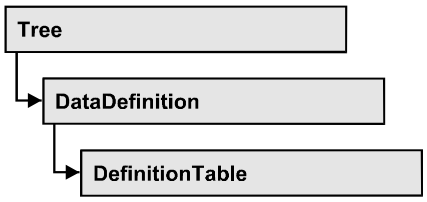
This property allows the access to the table **vcMaindata** of the data definition object that contains the definitions for nodes.

	Data Type	Explanation
Parameter:		
⇨ tableType	DataTableEnum	type of data definition table
	Possible Values: vcMaindata 0	 table type **vcMaindata** (for nodes)
Property value	VcDataDefinitionTable	Data definition table

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable

Set dataDefinition = VcTree1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
```

7.12 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** oder **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 **DataDefinition**.

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. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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

Example Code

```
Dim datdeftable As VcDataDefinitionTable
For Each datdeftable In VcTree1.VcDataDefinition
   Debug.Print datdeftable.Count
Next
```

Count

Read Only Property of VcDataDefinitionTable

This property lets you retrieve the number of fields in the data definition table. You can add fields via the **DataDefinition** property page or at run time via 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 = VcTree1.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 definition table. The data field of the new data field is Integer. You can change the data type via the property **VcDefinitionField.Type**.

	Data Type	Explanation
Parameter:		
⇨ newfieldName	String	name of the new field
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
dataDefinitionTable.CreateDataField ("New data field 1")
```

FieldByIndex

Method of VcDataDefinitionTable

By this method you can access a field of the data definition table by index. A field can be addressed by its name or by its index. You can set data field definitions on the property page **DataDefinition**.

	Data Type	Explanation
Parameter:		
⇨ fieldIndex	Integer	Field index
Return value	VcDefinitionField	Data definition field

Example Code

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDefinitionField

Set dataDefinitionTable =  _
            VcTree1.DataDefinition.DefinitionTable(vcMaindata)
Set definitionField = dataDefinitionTable.FieldByIndex(2)
```

FieldByName

Method of VcDataDefinitionTable

By this method you can get a field of the data definition table by its name. If a field of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic). A field can be addressed by its name or by its index. You can set data field definitions on the property page **DataDefinition**.

	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 = VcTree1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FieldByName("Code 1")
```

FirstField

Method of VcDataDefinitionTable

This method can be used to access the initial value, i.e. the first field of a data definition 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 definition table, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Return value	VcDefinitionField	First Data definition field

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcTree1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
```

NextField

Method of VcDataDefinitionTable

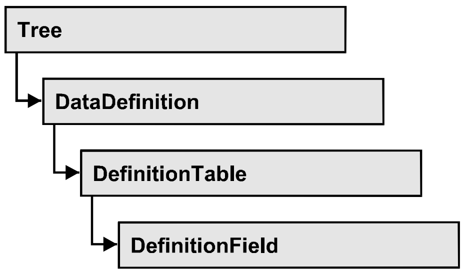
This method can be used in a forward iteration loop to retrieve subsequent fields from a data definition table after initializing the loop by the method **FirstField**. If there is no field left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	**Explanation**
Return value	VcDefinitionField	Data definition field following

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcTree1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstField
While Not dataDefinitionField Is Nothing
      ListBox.AddItem dataDefinitionField.Name
      Set dataDefinitionField = dataDefinitionTable.NextField
Wend
```

7.13 VcDefinitionField



An object of the type VcDefinitionField defines a field of the data definition table. The definition consists of a name, a type (alphanumeric, integer, date/time) and, if necessary, a date format that is used to hand over a date.

Properties

- DateFormat
- Editable
- Hidden
- ID
- Name
- Type

Properties

DateFormat

Property of VcDefinitionField

This property lets you set/retrieve the date format of the field of a data definition table. This property is set only if the field is of the type vcDefFieldDateTime. For all other types the property "" is set.

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 dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Dim dateFormatOfDefinitionField As String

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Start")

dateFormatOfDefinitionField = dataDefinitionField.DateFormat
```

Editable

Property of VcDefinitionField

This property lets you require/set whether a data field is editable at run time.

	Data Type	Explanation
Property value	Boolean	definition field editable/not editable **Default value:** True

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(0)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Start")
dataDefinitionField.Editable = True
```

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 dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(0)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Start")
dataDefinitionField.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 dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Dim idOfDefinitionField As Integer

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Start")

idOfDefinitionField = dataDefinitionField.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 dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Dim nameOfDefinitionField As String

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Start")

nameOfDefinitionField = dataDefinitionField.Name
```

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 be changed!

vcDefFieldAlphanumericType: DateFormat = ""

vcDefFieldDateTimeType: DateFormat = "DD.MM.YYYY hh:mm:ss"

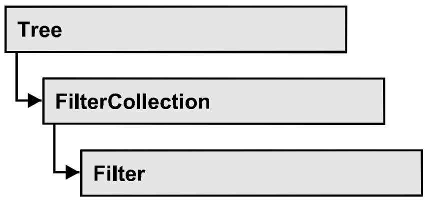
vcDefFieldIntegerType: DateFormat = ""

	Data Type	Explanation
Property value	DefinitionFieldTypeEnum	type of the definition field **Default value:** vcDefFieldIntegerType
	Possible Values:	
	vcDefFieldAlphanumericType 1	data type **alphamumeric**
	vcDefFieldDateTimeType 4	data type **date**
	vcDefFieldIntegerType 2	data type **integer** (32 bits)

Example Code

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim dataDefinitionField As VcDefinitionField
Dim typeOfDefinitionField As String

Set dataDefinition = VcTree1.dataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)
Set dataDefinitionField = dataDefinitionTable.FieldByName("Start")

typeOfDefinitionField = dataDefinitionField.Type
```

7.14 VcFilter



An object of the type VcFilter contains subconditions (VcFilterSubCondition), p.e. permitted values to be compared to the data fields of a node, so that the filter conditions may or may not apply to a node.

Only if the filter is valid after the subconditions have been modified, the modified subconditions will become valid. Otherwise the former filter conditions will remain be valid. This can be controlled via the methods VcFilter.IsValid and VcFilterSubCondition.IsValid.

Properties

- _NewEnum
- DatesWithHourAndMinute
- Name
- Specification
- StringsCaseSensitive
- SubCondition
- SubConditionCount

Methods

- AddSubCondition
- CopySubCondition
- IsValid
- RemoveSubCondition

Properties

_NewEnum

Read Only Property of VcFilter

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 fiSuCo As VcFilterSubCondition

For Each fiSuCo In filter
    Debug.Print fiSuCo.Index
Next
```

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. 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 filterCollection As VcFilterCollection
Dim filter As VcFilter

Set filterCollection = VcTree1.FilterCollection

For Each filter In filterCollection
    ListBox.AddItem filter.name
Next filter
```

Specification

Read Only Property of VcFilter

This property lets you enquire the specification of a filter. The specification is a string that contains only legible ASCII signs in the area 32 to 127 and that thus can be stored in text files or databases without problems. This is necessary for the persistency. Specifications can be used for the recreation of filters via 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 via 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 via 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 via 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

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)

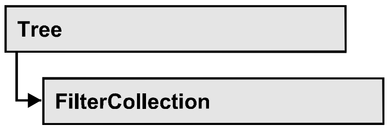
RemoveSubCondition

Method of VcFilter

This method lets you delete a filter subcondition via its index.

	Data Type	**Explanation**
Parameter:		
⇨ index	Integer	index of the filter subcondition to be removed

7.15 VcFilterCollection



An object of the type VcFilterCollection automatically contains all filters available. The property **Count** will return the number of filters contained in the collection.

Properties

- _NewEnum
- Count
- MarkedNodesFilter

Methods

- Add
- AddBySpecification
- Copy
- FilterByName
- FirstFilter
- NextFilter
- Remove

Properties

_NewEnum

Read Only Property of VcFilterCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all filter objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 VcTree1.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 filterCollection As VcFilterCollection
Dim numberOfFilters As Long

Set filterCollection = VcTree1.FilterCollection
numberOfFilters = filterCollection.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 VcTree1.ActiveNodeFilter = VcTree1.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.

	Data Type	Explanation
Parameter:		
⇨ newName	String	filter name
Return value	VcFilter	new filter object

Example Code

```
Set newFilter = VcTree1.FilterCollection.Add("foo")
```

AddBySpecification

Method of VcFilterCollection

By this method you can create a filter via a filter specification. This is necessary for the persistency of filter objects. The specification of a filter can be read (see VcFilter property **Specification**) and saved. In the next session the filter will be created again with the specification that is read again and with its stored 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

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 filterCollection As VcFilterCollection
Dim filter As VcFilter

Set filterCollection = VcTree1.FilterCollection
Set filter = filterCollection.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 filterCollection As VcFilterCollection
Dim filter As VcFilter

Set filterCollection = VcTree1.FilterCollection
Set filter = filterCollection.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	next filter

Example Code

```
Dim filterCollection As VcFilterCollection
Dim filter As VcFilter

Set filterCollection = VcTree1.FilterCollection
Set filter = filterCollection.FirstFilter

While Not filter Is Nothing
    Listbox.AddItem filter.Name
    Set filter = filterCollection.NextFilter
Wend
```

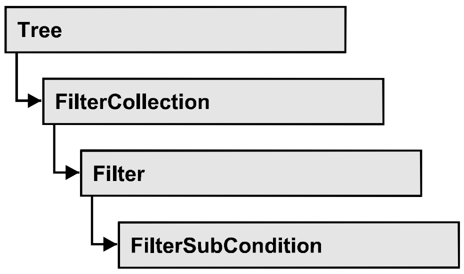
Remove

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)

7.16 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: "#21/03/2005 12:00#". A special date comparison value is "<TODAY>".
- Date field: included by square brackets. Example: "[ID]"
- Number: entered directly. Example: "52076"
- List: for a vc...In operator: included by {} brackets. All values included must have the same type (string, date or number). They may have one of the formats 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	operator for the connection with the following subcondition
	Possible Values:	
	vcAnd 1	And operator
	vcInvalidConnOp 0	invalid operator
	vcOr 2	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	Integer	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 enquire/set the comparison operator. The operators that are available via API correspond to the operators in the **Edit Filter** dialog. The operator type has to match the types of the data field and of the comparison value.

	Data Type	Explanation
Property value	OperatorEnum	comparison operator
	Possible Values:	
	vcDateEarlier 27	date earlier than
	vcDateEarlierOrEqual 28	date earlier than or equal
	vcDateEqual 25	date equal
	vcDateIn 31	date in
	vcDateLater 29	date later than

vcDateLaterOrEqual 30	date later than or equal
vcDateNotEqual 26	date not equal
vcDateNotIn 32	date not in
vcIntEqual 9	integer equal
vcIntGreater 13	integer greater
vcIntGreaterOrEqual 14	integer greater or equal
vcIntIn 15	integer in
vcIntLess 11	integer smaller than
vcIntLessOrEqual 12	integer smaller than or equal
vcIntNotEqual 10	integer not equal
vcIntNotIn 16	integer not in
vcInvalidOp 0	invalid operator
vcStringBeginsWith 3	string begins with
vcStringContains 5	string contains
vcStringEqual 1	string equal
vcStringIn 7	string contains
vcStringNotBeginsWith 4	string does not begin with
vcStringNotContains 6	string does not contain
vcStringNotEqual 2	string is not equal
vcStringNotIn 8	string is not in

Methods

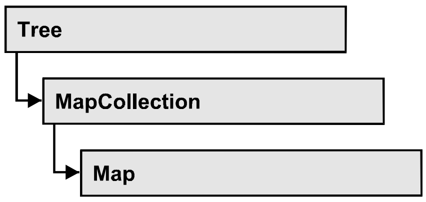
IsValid

Method of VcFilterSubCondition

This property checks whether the filter subcondition is correct.

	Data Type	Explanation
Return value	Boolean	filter subcondition correct (True)/ not correct (False)

7.17 VcMap



Maps specify certain properties of nodes, p.e. their background color, based on the data in the node record.

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. By the call **For Each mapentry In Map** you can retrieve all entries of the map in a loop.

Properties

- _NewEnum
- Count
- Name
- Specification
- Type

Methods

- CreateEntry
- DeleteEntry
- FirstMapEntry
- NextMapEntry

Properties

_NewEnum

Read Only Property of VcMap

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 map As VcMap

For Each map in VcTree1.Map
   Debug.Print.map.Name
Next
```

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 mapCollection As VcMapCollection
Dim map As VcMap
Dim numberOfEntries As Long

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps vcAnyMap
Set map = mapCollection.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 mapCollection As VcMapCollection
Dim map As VcMap
Dim mapName As String

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.FirstMap
mapName = map.Name
```

Specification

Read Only Property of VcMap

This property lets you enquire the specification of a map. The specification is a string that contains only legible ASCII signs in the area 32 to 127 and that thus can be stored in text files or databases without problems. This is necessary for the persistency. Specifications can be used for the recreation of maps via 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	map type
	Possible Values:	
	vcAnyMap 0	**any** (used only for selecting)
	vcColorMap 1	**Colors**
	vcFontMap 8	**Fonts**
	vcGraphicsFileMap 7	**Graphics file**
	vcMillimeterMap 9	**Millimetres**
	vcPatternMap 3	**pattern**

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap

Set mapCollection = VcTree1.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.

	Data Type	Explanation
Return value	VcMapEntry	Map entry

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapEntry = map.CreateEntry
```

DeleteEntry

Method of VcMap

This method lets you delete an entry (a row) of the map.

	Data Type	Explanation
Parameter: ⇨ mapEntry	VcMapEntry	Map entry
Return value	Boolean	Map entry was/was not deleted successfully

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps vcAnyMap
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

map.DeleteEntry mapEntry
```

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 mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)

Set map = mapCollection.FirstMap
Set mapEntry = map.FirstMapEntry
```

NextMapEntry

Method of VcMap

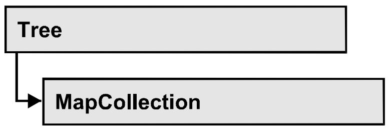
This method can be used in a forward iteration loop to retrieve subsequent entries (rows) from a map object after initializing the loop by the method **FirstMapEntry**. If there is no map entry left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcMapEntry	Succeeding map entry

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)

Set map = mapCollection.FirstMap
Set mapEntry = map.FirstMapEntry

While Not mapEntry Is Nothing
   List1.AddItem (mapEntry.Legend)
   Set mapEntry = map.NextMapEntry
Wend
```

7.18 VcMapCollection



An object of the type VcMapCollection can contain maps, if you assign them to the collection by the method **SelectMaps**. You can retrieve a map by the method **MapByName**.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstMap
- MapByName
- NextMap
- Remove
- SelectMaps
- Update

Properties

_NewEnum

Read Only Property of VcMapCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 map As VcMap

For Each map In VcTree1.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 mapCollection As VcMapCollection
Dim numberOfMaps As Long

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps vcAnyMap
numberOfMaps = mapCollection.Count
```

Methods

Add

Method of VcMapCollection

By this method you can create a map as a member of the MapCollection. If the name has not been used before, the new map object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

	Data Type	**Explanation**
Parameter:		
⇨ mapName	String	map name
Return value	VcMap	new map object

Example Code

```
Set newMap = VcTree1.MapCollection.Add("map1")
```

AddBySpecification

Method of VcMapCollection

By this method you can create a map via a map specification. This is necessary for the persistency of map objects. The specification of a map can be read (see VcMap property **Specification**) and saved. In the next session the map will be created again with the specification that is read again and with its stored 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 mapCollection As VcMapCollection
Dim map As VcMap

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.FirstMap
```

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 mapCollection As VcMapCollection
Dim map As VcMap

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.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	Succeeding map

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.FirstMap

While Not map Is Nothing
   List1.AddItem map.Name
   Set map = mapCollection.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 which map types your map collection should 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	**Millimetres**
	vcPatternMap 3	**pattern**
Return value	Long	Number of maps selected

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps vcAnyMap
```

Update

Method of VcMapCollection

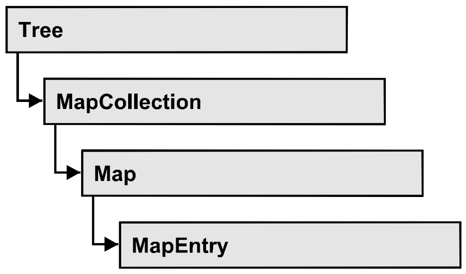
This method has to be used when map modifications have been made. The method **UpdateMaps** updates all objects that are concerned by the maps you have edited. You should call this method at the end of the code that defines the maps and the map collection. Otherwise the update will be processed before all map definitions are processed.

	Data Type	Explanation
Return value	Boolean	update successfull (True)/ not successfull (False)

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps vcAnyMap
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

While Not mapEntry.DataFieldValue = "A"
   Set mapEntry = map.NextMapEntry
Wend

mapEntry.Color = RGB(0, 0, 0)

mapCollection.Update
```

7.19 VcMapEntry



An object of the type VcMapEntry is a map entry and therefore an element of a map. A map entry is defined by the combination of a data field content of the node´s record, a color or graphics file and a legend text.

In each map you can specify up to a maximum of 150 map entries. If you need further map entries, please specify a new map, e. g. as a copy of the current one.

Properties

- Color
- DataFieldValue
- FontBody
- FontName
- FontSize
- GraphicsFileName
- Millimeter

Properties

Color

Property of VcMapEntry

for Color Maps: This property lets you specify/enquire the color value of a map entry.

	Data Type	Explanation
Property value	Color	RGB color values ({0...255},{0...255},{0...255})

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim colorOfMapEntry As OLE_COLOR

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcColorMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

colorOfMapEntry = mapEntry.Color
```

DataFieldValue

Property of VcMapEntry

This property lets you specify/enquire the content of a data of each map entry.

	Data Type	Explanation
Property value	String	Content of the data field

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim dataFieldValue As String

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcAnyMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

FontBody

Property of VcMapEntry

for Font Maps: This property lets you specify/enquire the font body of the map entry.

	Data Type	Explanation
Property value	FontBodyEnum	font body

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontBodyOfMapEntry As FontBodyEnum

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcFontMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontBodyOfMapEntry = vcBold
```

FontName

Property of VcMapEntry

for Font Maps: This property lets you specify/enquire the font name of the map entry.

	Data Type	Explanation
Property value	String	font type

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontNameOfMapEntry As String

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcFontMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontNameOfMapEntry = "Arial"
```

FontSize

Property of VcMapEntry

for Font Maps: This property lets you specify/enquire the font name of he map entry.

	Data Type	Explanation
Property value	Long	font size

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim FontSizeOfMapEntry As Long

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcFontMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

FontSizeOfMapEntry = 12
```

GraphicsFileName

Property of VcMapEntry

for Graphic File Maps: This property lets you specify/enquire the graphics file name of a map entry.

	Data Type	Explanation
Property value	String	name of the graphics file

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcGraphicsFileMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

mapEntry.GraphicsFileName = AppPath & "\picture1.bmp"
```

Millimeter

Property of VcMapEntry

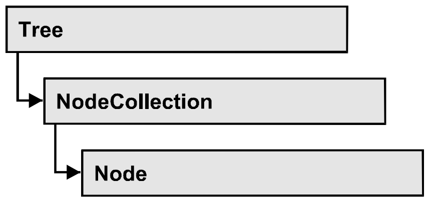
for Millimeter Maps: This property lets you specify/enquire the millimetre value of a map entry.

	Data Type	Explanation
Property value	Long	1/100 units

Example Code

```
Dim mapCollection As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim MillimeterOfMapEntry As Long

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps (vcMillimeterMap)
Set map = mapCollection.MapByName("Map1")
Set mapEntry = map.FirstMapEntry

MillimeterOfMapEntry = 3
```

7.20 VcNode



A node is a basic element of a tree diagram. 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 **VcTree.InsertNodeRecord**.

Properties

- AllData
- Arrangement
- ChildNodeCollection
- Collapsed
- DataField
- InCollapsedSubtree
- LeftBrotherNode
- MarkNode
- ParentNode
- RightBrotherNode
- SubtreeNodeCollection

Methods

- ArrangeSubtree
- Collapse
- DeleteNode
- Expand
- UpdateNode

Properties

AllData

Property of VcNode

This record lets you specify/enquire 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 VcTree1_OnNodeModify(ByVal node As VcTreeLib.VcNode,  _
                                ByVal modificationType As _
                                VcTreeLib.ModificationTypeEnum, _
                                returnStatus As Variant)

    Dim allDataOfNode As String

    returnStatus = vcRetStatFalse

    allDataOfNode = node.AllData
    MsgBox allDataOfNode

End Sub
```

Arrangement

Property of VcNode

By this property you can assign/retrieve whether the subtree descending is to be arranged horizontally or vertically.

	Data Type	Explanation
Property value	ArrangementEnum	Direction of arrangement **Default value:** vcHorizontal
	Possible Values:	
	vcHorizontal 0	horizontal arrangement
	vcVertical 1	Vertical arrangement

Example Code

```
VcNode.Arrangement = vcHorizontal
```

ChildNodeCollection

Read Only Property of VcNode

By this property you can retrieve the immediate child nodes of a node. Please also see the property **SubtreeNodeCollection**.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object containing child nodes

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim noOfChildren As Integer

    noOfChildren = node.ChildNodeCollection.Count
    MsgBox (noOfChildren)
    returnStatus = vcRetStatFalse

End Sub
```

Collapsed

Read Only Property of VcNode

By this property you can retrieve, whether (True) or not (False) a node is collapsed.

	Data Type	Explanation
Property value	Boolean	Node collapsed/expanded

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim collapseState As Boolean

    collapseState = node.collapsed
    MsgBox (collapseState)
    returnStatus = vcRetStatFalse

End Sub
```

DataField

Property of VcNode

This property lets you assign/retieve data to/from the data field of a node. If the data field has been 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 VcTree1_OnNodeRClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.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
```

InCollapsedSubtree

Read Only Property of VcNode

By this property you can retrieve, whether a node forms a part of a collapsed subtree (True) and therefore is invisible.

	Data Type	Explanation
Property value	Boolean	Node is/is not located in a collapsed subtree

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim inCollapsedSubtree As Boolean

    inCollapsedSubtree = node.inCollapsedSubtree
    MsgBox (inCollapsedSubtree)
    returnStatus = vcRetStatFalse

End Sub
```

LeftBrotherNode

Read Only Property of VcNode

The left brother of the node is returned.

	Data Type	Explanation
Property value	VcNode	left brother node

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                              ByVal location As VcTreeLib.LocationEnum, _
                              ByVal x As Long, ByVal y As Long, _
                              returnStatus As Variant)
    If node.LeftBrotherNode Is Nothing Then
        MsgBox "This node doesn´t have a left brother."
    Else
        MsgBox (node.LeftBrotherNode.AllData)
    End If
    returnStatus = vcRetStatFalse
End Sub
```

MarkNode

Property of VcNode

This property lets you specify/enquire 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 VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.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
```

ParentNode

Property of VcNode

By this property you can insert a node as a child node/retrieve its parent node.

	Data Type	Explanation
Property value	VcNode	Parent node

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    Dim parentNodeID As Integer

    parentNodeID = node.parentNode.DataField(0)
    MsgBox (parentNodeID)
    returnStatus = vcRetStatFalse

End Sub
```

RightBrotherNode

Read Only Property of VcNode

The right brother of the node is returned.

	Data Type	Explanation
Property value	VcNode	right brother node

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                             ByVal location As VcTreeLib.LocationEnum, _
                             ByVal x As Long, ByVal y As Long, _
                             returnStatus As Variant)
    If node.RightBrotherNode Is Nothing Then
        MsgBox "This node doesn´t have a right brother."
    Else
        MsgBox (node.RightBrotherNode.AllData)
    End If
    returnStatus = vcRetStatFalse
End Sub
```

SubtreeNodeCollection

Read Only Property of VcNode

By this property you can retrieve the subtree of the reference node (the reference node itself and all immediate or indirect child nodes of the reference node). Also see **ChildNodeCollection**.

	Data Type	Explanation
Property value	VcNodeCollection	Collection of Nodes that form the subtree

Example Code

```
Private Sub VcTree1_OnNodeRClick(ByVal node As VcTreeLib.VcNode,_
                              ByVal location As VcTreeLib.LocationEnum, _
                              ByVal x As Long, ByVal y As Long, _
                              returnStatus As Variant)

    Dim noOfNodes As Integer

    noOfNodes = node.SubtreeNodeCollection.Count
    MsgBox (noOfNodes)
    returnStatus = vcRetStatNoPopup

End Sub
```

Methods

ArrangeSubtree

Method of VcNode

By this method you can arrange a subtree horizontally or vertically. In contrast to the property **Arrangement** by this method the properties of the nodes in the subtree in addition are set.

	Data Type	Explanation
Parameter:		
⇨ arrangement	ArrangementEnum	Direction of arrangement
	Possible Values:	
	vcHorizontal 0	horizontal arrangement
	vcVertical 1	Vertical arrangement
Return value	Void	

Example Code

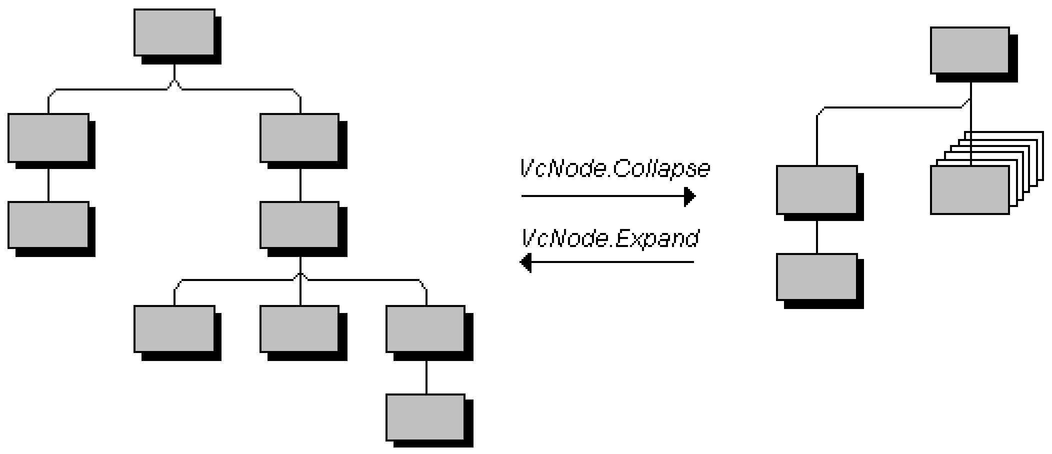
```
Dim nodecol As VcNodeCollection
Dim node As VcNode

Set nodecol = VcTree1.NodeCollection
Set node = VcTree1.GetNodeByID("8")
node.ArrangeSubtree vcVertical
```

Collapse

Method of VcNode

By this method you can collapse a tree including its subtree. If you use **vcSelf** when collapsing the subtree, all nodes will disappear from the screen. If you use **vcComplete** when collapsing the subtree, each node that is no leave node in addition will be collapsed itself. When expanding these nodes later, each one of them will need to be expanded separately.

	Data Type	Explanation
Parameter:		
⇨ action	CollapseExpandEnum	Type of Collapsing
	Possible Values:	
	vcComplete 1	Nodes in subtree included
	vcSelf 0	Nodes in subtree excluded
Return value	Void	



Collapsing and expanding a subtree

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    If MsgBox("Collapse Node No." & node.DataField(0) & "? ", vbYesNo, _
                                "Collapse node") =  vbYes Then
    node.Collapse vcComplete
    returnStatus = vcRetStatFalse

End Sub
```

DeleteNode

Method of VcNode

This method lets you delete a node.

	Data Type	Explanation
Return value	Boolean	Node was/was not deleted successfully

Example Code

```
Private Sub VcTree1_OnNodeRClick(ByVal node As VcTreeLib.VcNode, _
                                  ByVal location As _
                                  VcTreeLib.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
```

Expand

Method of VcNode

By this method you can expand a tree. If you use **vcSelf** when expanding the subtree, the node itself will be expanded only, but not its collapsed subtrees. If you use **vcComplete**, all nodes of the subtree that are no leaf nodes, will be expanded.

	Data Type	Explanation
Parameter:		
⇨ action	CollapseExpandEnum	Type of Expanding
	Possible Values:	
	vcComplete 1	Nodes in subtree included
	vcSelf 0	Nodes in subtree excluded
Return value	Void	

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                              ByVal location As VcTreeLib.LocationEnum, _
                              ByVal x As Long, ByVal y As Long, _
                              returnStatus As Variant)

    If MsgBox("Expand Node No." & node.DataField(0) & "? ", vbYesNo, _
                              "Expand node") =  vbYes Then
    node.Expand vcComplete
    returnStatus = vcRetStatFalse

End Sub
```

UpdateNode

Method of VcNode

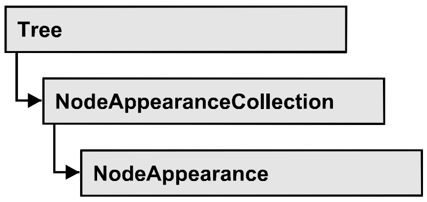
If data fields of a node have been modified by the **DataField** property, the diagram needs to be updated by the **UpdateNode** method.

	Data Type	Explanation
Return value	Boolean	Node was/was not updated successfully

Example Code

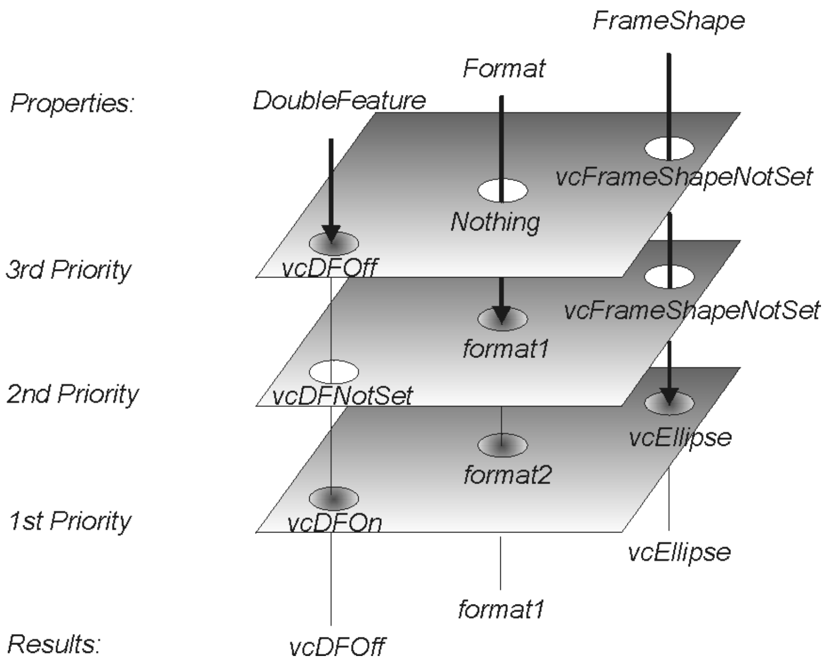
```
Dim nodeCollection As VcNodeCollection
Dim node As VcNode

Set nodeCollection = VcTree1.NodeCollection
Set node = nodeCollection.FirstNode

node.DataField(12) = "Group A"
node.UpdateNode
```

7.21 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

- BackColor
- BackColorDataFieldIndex
- BackColorMapName
- DoubleFeature
- FilterName
- FormatName
- FrameShape

- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- Piles
- Shadow
- Specification
- StrikeThrough
- StrikeThroughColor
- ThreeDEffect
- VisibleInLegend

Methods

- PutInOrderAfter

Properties

BackColor

Property of VcNodeAppearance

This property lets you set/enquire the background color of a node. 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 ({0...255},{0...255},{0...255})

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.BackColor = RGB(100, 100, 100)
```

BackColorDataFieldIndex

Property of VcNodeAppearance

This property lets you set/enquire the data field index to be used together with a map specified via 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/enquire 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/enquire 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	settings of double feature
	Possible Values:	

	vcDFNotSet -1 vcDFOff 0 vcDFOn 1	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 = VcTree1.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

	Data Type	Explanation
Property value	String	name of filter

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim filtername As String

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

filtername = nodeAppearance.filtername
```

FormatName

Property of VcNodeAppearance

This property lets you assign/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	NodeFormat object or empty

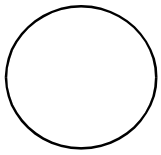
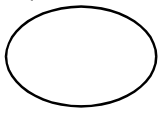
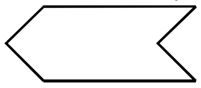
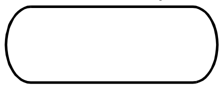
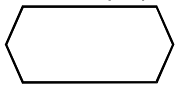
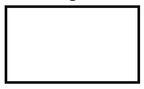
Example Code

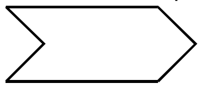
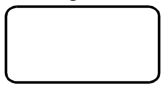
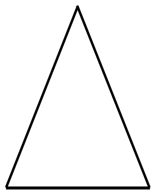
```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim format1 As VcNodeFormat

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

Set format1 = nodeAppearance.format
MsgBox (format1.name)
```

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 
	vcFrameShapeNotSet -1	frame shape not set
	vcLeftArrow 17	frame arrow shaped, pointing left 
	vcNoFrameShape 1	no frame shape
	vcOval 4	oval frame shape 
	vcParallelogram 9	frame shape parallelogram
	vcPointed 7	frame shape pointed at vertical sides 
	vcRectangle 2	rectangular frame shape 

vcRightArrow 18	frame arrow shaped, pointing right 
vcRounded 3	rectangular frame shape, rounded 
vcTriangleLeft 10 vcTriangleUp 13	triangular frame shape, pointing left triangular frame shape, pointing up 

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.FrameShape = vcEllipse
```

LegendText

Property of VcNodeAppearance

This property lets you set/enquire 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 = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.LineColor = RGB(256, 0, 100)
```

LineColorDataFieldIndex

Property of VcNodeAppearance

This property lets you set/enquire the data field index to be used together with a map specified via 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/enquire 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 specify/enquire the line thickness of a NodeAppearance object. 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	line thickness LineType {vcDashed...vcSolid} allows LineThickness {-1 ...} (in Pixeln) LineType {vcLineType0... LineType18} allows Integer values {1...1000} (in 1/100 mm) **Default value:** As defined on property page

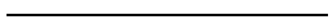
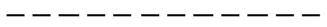
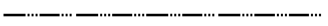
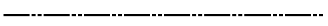
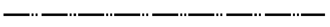
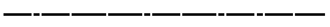
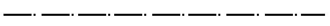
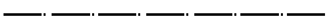
Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.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. When set to **vcNone**, 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 **vcNone** (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 assigned
vcSolid 2	Line solid

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.LineType = vcDotted
```

Name

Property of VcNodeAppearance

This property lets you specify/enquire 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 = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppName = nodeAppearance.name
```

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 = VcTree1.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 = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Shadow = vcShOn
```

Specification

Read Only Property of VcNodeAppearance

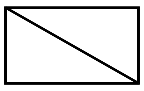
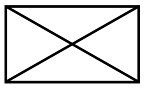
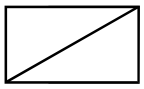
This property lets you enquire the specification of the node appearance. The specification is a string that contains only legible ASCII signs in the area 32 to 127 and that thus can be stored in text files or databases without problems. This is necessary for the persistency. Specifications can be used for the recreation of node appearances via the method VcNodeAppearanceCollection.AddBySpecification.

	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	strike through pattern or **-1**
	Possible Values:	
	vcBackslashed 3	Backslashed strike through 
	vcCrossed 4	Crossed 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 = VcTree1.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 = VcTree1.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	3DEffect setting
	Possible Values: vc3DNotSet -1	Flag of 3D appearance not set

	vc3DOff 0	Flag of 3D appearance set off
	vc3DOn 1	Flag of 3D appearance set on

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.ThreeDEffect = vc3DOn
```

VisibleInLegend

Property of VcNodeAppearance

This property lets you set/require whether a node appearance object is to be visible in the legend. This property also can be set via 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 = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.NodeAppearanceByName("Standard")

nodeAppearance.VisibleInLegend = False
```

Methods

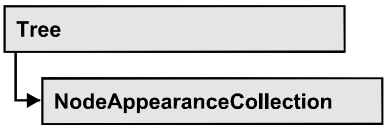
PutInOrderAfter

Method of VcNodeAppearance

This method lets you set the current node appearance in the NodeAppearanceCollection behind the node appearance specified by name. If you select the name "", then the current node appearence will be set to the first place. The order of the node appearances determines the order in which they will be assigned to the nodes.

	Data Type	Explanation
Parameter:		
refNodeAppearanceName	String	name of the node appearance behind which the current node appearance is to be set

7.22 VcNodeAppearanceCollection



An object of the type VcNodeAppearanceCollection automatically contains all available node appearances. You can retrieve a node appearances by the method **NodeAppearanceByName**. The **Count** property lets you enquire the number of node appearances that exist in the collection.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstNodeAppearance
- NextNodeAppearance
- NodeAppearanceByName
- Remove

Properties

_NewEnum

Read Only Property of VcNodeAppearanceCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all node appearance objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 nodeApp As VcNodeAppearance

For Each nodeApp In VcTree1.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 = VcTree1.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 has not been 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 are set to transparent by default.

	Data Type	Explanation
Parameter: ⇨ newName	String	node apearance name
Return value	VcNodeAppearance	new node apearance object

Example Code

```
Set newNodeAppearance = VcTree1.NodeAppearanceCollection.Add("nodeapp1")
```

AddBySpecification

Method of VcNodeAppearanceCollection

By this method you can create a node appearance via a node appearance specification. This is necessary for the persistency of node appearance objects. The specification of a node appearance can be read (see VcNodeAppearance property **Specification**) and saved. In the next session the node appearance will be created again with the specification that is read again and with its stored 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. If the node appearance that is to be copied exists, and if the name for the new node appearance does not yet exist, the new node appearance 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 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 = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance
```

NextNodeAppearance

Method of VcNodeAppearanceCollection

This method can be used in a forward iteration loop to retrieve subsequent node appearance from a collection after initializing the loop by the method **FirstNodeAppearance**. If there is no node appearance left, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Return value	VcNodeAppearance	succeeding node appearance object

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

While Not nodeAppearance Is Nothing
     Listbox.AddItem nodeAppearance.Name
     Set nodeAppearance = nodeAppearanceCollection.NextNodeAppearance
Wend
```

NodeAppearanceByName

Method of VcNodeAppearanceCollection

This method lets you retrieve a NodeAppearance object by its name. If a NodeAppearance object of the specified name does not exist, a **none** object will be returned (**Nothing** in Visual Basic).

	Data Type	Explanation
Parameter:		
⇨ nodeAppearanceName	String	name of the node apearance object
Return value	VcNodeAppearance	node appearance object

Example Code

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.NodeAppearanceByName("Standard")
```

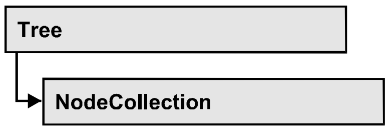
Remove

Method of VcNodeAppearanceCollection

This method lets you delete a node appearance. If the node appearance is used in another object, it cannot be deleted. Then False will be returned, otherwise True.

	Data Type	Explanation
Parameter:		
⇨ name	String	node appearance name
Return value	Boolean	node appearance deleted (True)/not deleted (False)

7.23 VcNodeCollection



An object of the type VcNodeCollection comprises all nodes available in the diagram. You can select a part of them by using the method **SelectNodes**.

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. With the help of this object the iteration over all filter subcondition objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 node As VcNode

For Each node In VcTree1.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 nodeCollection As VcNodeCollection

Set nodeCollection = VcTree1.NodeCollection
MsgBox "Number of nodes: " & nodeCollection.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 nodeCollection As VcNodeCollection
Dim node As VcNode

Set nodeCollection = VcTree1.NodeCollection
Set node = nodeCollection.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	Succeeding node

Example Code

```
Dim nodeCollection As VcNodeCollection
Dim node As VcNode

Set nodeCollection = VcTree1.NodeCollection
Set node = nodeCollection.FirstNode

While Not node Is Nothing
     node.MarkNode = False
     Set node = nodeCollection.NextNode
Wend
```

SelectNodes

Method of VcNodeCollection

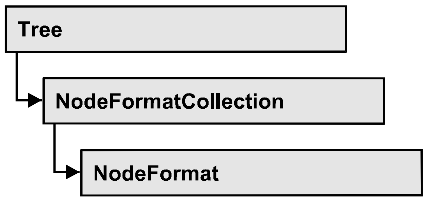
This method lets you specify the nodes to be collected by the NodeCollection object.

	Data Type	Explanation
Parameter:		
⇨ selType	SelectionTypeEnum	Nodes to be selected
	Possible Values:	
	vcAll 0	All objects in the diagram will be selected
	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 nodeCollection As VcNodeCollection
Dim node As VcNode

Set nodeCollection = VcTree1.NodeCollection
nodeCollection.SelectNodes vcSelected
```

7.24 VcNodeFormat



An object of the type VcNodeFormat defines the contents and the format of nodes. At run time, node formats are administered and edited in the **Administrate Node Formats** dialog box that you reach via the **Nodes** property page.

Properties

- _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. With the help of this object the iteration over all node format field objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 formatField As VcNodeFormatField

For Each formatField In format
    Debug.Print formatField.Index
Next
```

FieldsSeparatedByLines

Property of VcNodeFormat

This property lets you specify/enquire 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 = VcTree1.NodeFormatCollection.FormatByName("format1")
format.FieldsSeparatedByLines = True
```

FormatField

Read Only Property of VcNodeFormat

This property gives access to a VcNodeFormatField object via index. The index has to be in the range 0 to .FormatFieldCount-1.

Note for users of a version elder than 3.0: The index is **not** in the range 1 to .FormatFieldCount as in elder 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 = VcTree1.NodeFormatCollection
Set format = formatCollection.FormatByName("Standard")

numberofFormatField = format.FormatFieldCount
```

Name

Property of VcNodeFormat

This property lets you specify/enquire 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 = VcTree1.NodeFormatCollection.FirstFormat
formatName = format.Name
```

Specification

Read Only Property of VcNodeFormat

This property lets you enquire the specification of the node format. The specification is a string that contains only legible ASCII signs in the area 32 to 127 and that thus can be stored in text files or databases without problems. This is necessary for the persistency. Specifications can be used for the recreation of node formats via the method VcNodeFormatCollection.AddBySpecification.

	Data Type	Explanation
Property value	String	specification of the node format

WidthOfExteriorSurrounding

Property of VcNodeFormat

This property lets you set/enquire 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	above
	vcBelow 3	below
	vcLeftOf 0	left of
	vcOutsideAbove 9	outside, above
	vcOutsideBelow 11	outside, below
	vcOutsideLeftOf 8	outsite, left of
	vcOutsideRightOf 12	outside, right of
	vcRightOf 4	right of
⇨ refIndex	Integer	index of the reference node format field
Return value	VcNodeFormatField	node format field object

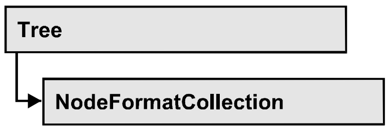
RemoveFormatField

Method of VcNodeFormat

This method lets you remove a node format field via its index. After that, the program will set all node format field indexes newly in order to number them consecutively.

	Data Type	Explanation
Parameter:		
⇨ index	Integer	index of the node format field to be deleted

7.25 VcNodeFormatCollection



An object of the type VcNodeFormatCollection automatically contains all node formats available to a node. You can retrieve a node format by the method **FormatByName**. The property **Count** will return the number of node formats contained in the collection.

Properties

- _NewEnum
- Count

Methods

- Add
- AddBySpecification
- Copy
- FirstFormat
- FormatByName
- NextFormat
- Remove

Properties

_NewEnum

Read Only Property of VcNodeFormatCollection

This property returns an Enumerator object that implements the OLE Interface IEnumVariant. With the help of this object the iteration over all ndoe format objects is possible. In Visual Basic this property never will be displayed, but it can be used via 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 format As VcNodeFormat

For Each format In VcTree1.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 formatCollection As VcNodeFormatCollection
Dim numberOfFormats As Long

Set formatCollection = VcTree1.NodeFormatCollection
numberOfFormats = formatCollection.Count
```

Methods

Add

Method of VcNodeFormatCollection

By this method you can create a node format as a member of the NodeFormatCollection. If the name has not been used before, the new VcNodeFormat object will be returned. Otherwise "Nothing" (in Visual Basic) or "0" (other languages) will be returned.

The node format has the following properties by default:

- only one field
- WidthOfExteriorSurrounding: 3 mm

The field has the following properties:

- Type: vcFFTText
- TextDataFieldIndex: IDMinimumWidth specified on the **General** property page: 3000
- Alignment: vcFFACenter
- BackColor: -1 (transparent)
- TextFontColor: RGB(0,0,0) (black)
- TextFont: Arial, 10, normal
- LeftMargin, RightMargin, TopMargin, BottomMargin: 0,3 mm
- MinimumTextLineCount, MaximumTextLineCount: 1

	Data Type	Explanation
Parameter:		
⇨ newName	String	name of the node format
Return value	VcNodeFormat	node format object

Example Code

```
Set newNodeFormat = VcTree1.NodeFormatCollection.Add("nodeformat1")
```

AddBySpecification

Method of VcNodeFormatCollection

By this method you can create a node format via a node format specification. This is necessary for the persistency of node format objects. The specification of a node format can be read (see VcNodeFormat property **Specification**) and saved. In the next session the node format will be created again with the specification that is read again and with its stored 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 = VcTree1.NodeFormatCollection.FirstFormat
```

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 = VcTree1.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	following node format

Example Code

```
Dim formatCollection As VcNodeFormatCollection
Dim format As VcNodeFormat

Set formatCollection = VcTree1.NodeFormatCollection
Set format = formatCollection.FirstFormat

While Not format Is Nothing
     List1.AddItem format.Name
     Set format = formatCollection.NextFormat
Wend
```

Remove

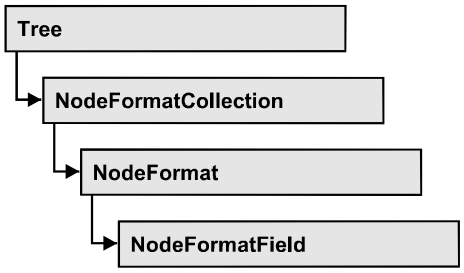
Method of VcNodeFormatCollection

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)

7.26 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
- BackColor
- BackColorDataFieldIndex
- BackColorMapName
- BottomMargin
- CombiField
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- RightMargin
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TopMargin

- Type

Properties

Alignment

Property of VcNodeFormatField

This property lets you set/enquire the alignment of the content of the node format field.

	Data Type	Explanation
Property value	FormatFieldAlignmentEnum	alignment of the field content
	Possible Values:	
	vcFFABottom 28	bottom
	vcFFABottomLeft 27	bottom left
	vcFFABottomRight 29	bottom right
	vcFFACenter 25	center
	vcFFALeft 24	left
	vcFFARight 26	right
	vcFFATop 22	top
	vcFFATopLeft 21	top left
	vcFFATopRight 23	top right

BackColor

Property of VcNodeFormatField

This property lets you set/enquire the background color of the node format field. If the node format field shall have the background color of the node format, select the value **-1**. If in the property **BackColorMapName** a map is specified, this map will control the background color dependent on the data.

	Data Type	Explanation
Property value	OLE_COLOR	background color of the node format **Default value:** -1

BackColorDataFieldIndex

Property of VcNodeFormatField

This property lets you set/enquire the data field index to be used together with a color map specified via 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 VcNodeFormatField

This property lets you set/enquire the name of a color map (type vcColorMap) for the background color. If set to "", no map will be used. If the name of a map and additionally a data field index is specified in the property **BackColorDataFieldIndex**, then the background color is controlled by the map. If no data field entry applies, the background color that is specified in the property **BackColor** will be used.

	Data Type	Explanation
Property value	String	name of the color map

BottomMargin

Property of VcNodeFormatField

This property lets you set/enquire 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/enquire 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

only for the type ***vcFFTText***: If the property **TextDataFieldIndex** is set to **-1**, this property lets you set a constant text in the node format field.

	Data Type	Explanation
Property value	String	constant text

FormatName

Read Only Property of VcNodeFormatField

This property lets you enquire 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/enquire 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/enquire 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/enquire 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/enquire 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/enquire 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

only for the type vcFFTText: This property lets you set/enquire the maximum number of lines in the node format field. Also see the property **MinimumTextLineCount**.

	Data Type	Explanation
Property value	Integer	maximum number of lines 0 ... 9

MinimumTextLineCount

Property of VcNodeFormatField

only for the type vcFFTText: This property lets you set/enquire the minimum number of lines in the node format field. If the text of the node format field is too long, it will be enlarged dynamically up to the maximum number of lines. Also see the property **MaximumTextLineCount**.

	Data Type	Explanation
Property value	Integer	minimum number of lines 0 ... 9

MinimumWidth

Property of VcNodeFormatField

This property lets you set/enquire 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

RightMargin

Property of VcNodeFormatField

This property lets you set/enquire 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

only for the type ***vcFFTText***: This property lets you set/enquire the index of the data field the content of which is to be displayed in the node format field. If its value is **-1**, the content of the property **ConstantText** will be returned.

	Data Type	Explanation
Property value	Integer	index of the data field

TextFont

Property of VcNodeFormatField

only for the type vcFFTText: This property lets you set/enquire the font color of the node format field. If in the property **TextFontMapName** a map is specified, this map will control the text font color dependent on the data.

	Data Type	Explanation
Property value	StdFont	font type of the node format

TextFontColor

Property of VcNodeFormatField

only for the type vcFFTText: This property lets you set/enquire the font color of the node format field. If in the property **TextFontColorMapName** a map is specified, this map will control the font color dependent on the data.

	Data Type	Explanation
Property value	OLE_COLOR	font color of the node format **Default value:** -1

TextFontColorDataFieldIndex

Property of VcNodeFormatField

This property lets you set/enquire the data field index to be used together with a font color map specified via the property **TextFontColorMapName**. If you set this property to **-1**, no map will be used.

	Data Type	Explanation
Property value	Integer	data field index

TextFontColorMapName

Property of VcNodeFormatField

only for the type vcFFTText: This property lets you set/enquire the name of a color map (type vcColorMap) for the font color. If set to "", no map will be used. If a map name and additionally a data field index is specified in the property **TextFontColorDataFieldIndex**, then the font color is controlled by the map. If no data field entry applies, the font color that is specified in the property **TextFontColor** will be used.

	Data Type	Explanation
Property value	String	name of the font color map

TextFontDataFieldIndex

Property of VcNodeFormatField

This property lets you set/enquire the data field index to be used together with a font map specified via the property **TextFontMapName**. 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/enquire 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/enquire 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	type of the node format field
	Possible Values:	
	vcFFTGraphics 64	graphics
	vcFFTText 36	text

7.27 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

- Alignment
- BottomMargin
- ColorMode
- CuttingMarks
- FitToPage
- LeftMargin
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation
- PageDescription
- PageDescriptionString
- PageFrame
- PageNumberMode
- PageNumbers
- PaperSize
- PrintDate
- PrinterName
- RepeatTitleAndLegend
- RightMargin
- StartUpSinglePage
- TopMargin
- ZoomFactor

Properties

Alignment

Property of VcPrinter

This property lets you set/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
	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

```
VcTree1.Printer.Alignment = vcPTopLeft
```

BottomMargin

Property of VcPrinter

This property lets you set/retrieve the height of the bottom margin of the pages to be printed. The BottomMargin values will be processed as follows:

x= BottomMargin	actual margin width
x < 100	x cm
x >= 100	x/1000 cm

	Data Type	Explanation
Property value	Long	Height of the bottom margin of the page

Example Code

```
VcTree1.Printer.BottomMargin = 2  ' 2 cm
VcTree1.Printer.BottomMargin = 12  ' 12 cm
VcTree1.Printer.BottomMargin = 100  ' 0,1 cm
VcTree1.Printer.BottomMargin = 200  ' 0,2 cm
VcTree1.Printer.BottomMargin = 1200  ' 1,2 cm
```

ColorMode

Property of VcPrinter

This property lets you set/retrieve the color mode of the output.

	Data Type	Explanation
Property value	ColorModeEnum	Color mode of the output
	Possible Values:	
	vcBlackAndWhite 15	Black and white print mode
	vcColor 6	Color print mode
	vcGrayShades 0	Gray shades print mode

Example Code

```
VcTree1.Printer.ColorMode = vcGrayShade
```

CuttingMarks

Property of VcPrinter

This property lets you set/enquire, whether (True) or not (False) cutting marks are to printed onto a page.

	Data Type	Explanation
Property value	Boolean	Cutting marks are/are not printed

Example Code

```
VcTree1.Printer.CuttingMarks = True
```

FitToPage

Property of VcPrinter

This property lets you set/enquire, 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

```
VcTree1.Printer.FitToPage = True
```

LeftMargin

Property of VcPrinter

This property lets you set/retrieve the width of the left margin of the pages to be printed. The LeftMargin values will be processed as follows:

x= LeftMargin	actual margin width
x < 100	x cm
x >= 100	x/1000 cm

	Data Type	Explanation
Property value	Long	Width of the left margin of the page

Example Code

```
VcTree1.Printer.LeftMargin = 2   ' 2 cm
VcTree1.Printer.LeftMargin = 12  ' 12 cm
VcTree1.Printer.LeftMargin = 100  ' 0,1 cm
VcTree1.Printer.LeftMargin = 200  ' 0,2 cm
VcTree1.Printer.LeftMargin = 1200  ' 1,2 cm
```

MaxHorizontalPagesCount

Property of VcPrinter

After setting the **FitToPage** property to **True**, you need to define the number of pages for the horizontal and the vertical direction each, the product of which equals the total number of pages that the diagram is printed on. This property lets you define/retrieve the horizontal number of pages. Also see **MaxVerticalPagesCount**.

	Data Type	Explanation
Property value	Long	Maximum number of pages counted in horizontal direction

Example Code

```
VcTree1.Printer.MaxHorizontalPagesCount = 4
```

MaxVerticalPagesCount

Property of VcPrinter

After setting the **FitToPage** property to **True**, you need to define the number of pages for the horizontal and the vertical direction each, the product of which equals the total number of pages that the diagram is printed on. This property lets you define/retrieve the vertical number of pages. Also see **MaxHorizontalPagesCount**.

	Data Type	Explanation
Property value	Long	Maximum number of pages counted in vertical direction

Example Code

```
VcTree1.Printer.MaxVerticalPagesCount = 4
```

Orientation

Property of VcPrinter

This property lets you set/enquire the orientation of the output.

	Data Type	Explanation
Property value	OrientationEnum	Orientation
	Possible Values:	
	vcLandscape 42	Printing orientation **landscape**
	vcPortrait 41	Printing orientation **portrait**

Example Code

```
VcTree1.Printer.Orientation = vcLandScape
```

PageDescription

Property of VcPrinter

This property lets you set/enquire 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/is not printed

Example Code

```
VcTree1.Printer.PageDescription = True
```

PageDescriptionString

Property of VcPrinter

This property lets you set/enquire a page description in the bottom left corner of each page. Whether or not the page description string is printed you can control via 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

Example Code

```
VcTree1.Printer.PageDescriptionString = "Gantt-Graphics"
```

PageFrame

Property of VcPrinter

This property lets you set/enquire, whether (True) or not (False) a frame is to be drawn around the ouput. If the **RepeatTableTimeScale** property has been 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/is not displayed

Example Code

```
VcTree1.Printer.PageFrame = True
```

PageNumberMode

Property of VcPrinter

This property lets you set/enquire in which way the page numbers are to be displayed: "Page N of M pages" or "x.y" (row no./column no.).

	Data Type	Explanation
Property value	pageNumberModeEnum	mode of page numbering
	Possible Values:	
	vcPageNOfM 1597	"Page N of M pages"
	vcPRowColumn 1596	"x.y" (row no./column no.).

Example Code

```
Dim printer As VcPrinter

Set printer = VcTree1.printer

With printer
    .Orientation = vcLandscape
    .PageNumberMode = vcPageNOfM
    .PageNumbers = True
    .FitToPage = False
End With

VcTree1.PrintPreview
```

PageNumbers

Property of VcPrinter

This property lets you set/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/are not printed

Example Code

```
VcTree1.Printer.PageNumbers = True
```

PaperSize

Property of VcPrinter

This property lets you set/retrieve the paper size to be used.

	Data Type	Explanation
Property value	PaperSizeEnum	Paper size
	Possible Values:	
	vcDIN A2 66	DIN A2
	vcDIN_A3 8	DIN A3
	vcDIN_A4 9	DIN A4
	vcISO C 24	ISO C
	vcISO D 25	ISO D
	vcISO E 26	ISO E
	vcUS_LEGAL 5	US LEGAL
	vcUS_LETTER 1	US LETTER

Example Code

```
VcTree1.Printer.PaperSize = vcDIN_A3
```

PrintDate

Property of VcPrinter

This property lets you set/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

```
VcTree1.Printer.PrintDate = True
```

PrinterName

Read Only Property of VcPrinter

This property lets you set/retrieve the name of the currently selected printer. You can use this property for saving and restoring the state of the printer object.

If you transfer an empty string when setting the property, the system printer will be used.

	Data Type	Explanation
Property value	String	Printer name

RepeatTitleAndLegend

Property of VcPrinter

This property lets you set/retrieve, whether (True) or not (False) the title and the legend should appear on each page. Furthermore it specifies whether the pages will be automatically splitted so that the nodes will not be cutted.

	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

```
VcTree1.Printer.RepeatTitleAndLegend = True
```

RightMargin

Property of VcPrinter

This property lets you set/retrieve the width of the right margin of the pages to be printed. The RightMargin values will be processed as follows:

x= RightMargin	actual margin width
x < 100	x cm
x >= 100	x/1000 cm

	Data Type	Explanation
Property value	Long	Width of the right margin of the page

Example Code

```
VcTree1.Printer.RightMargin = 2  ' 2 cm
VcTree1.Printer.RightMargin = 12  ' 12 cm
VcTree1.Printer.RightMargin = 100  ' 0,1 cm
VcTree1.Printer.RightMargin = 200  ' 0,2 cm
VcTree1.Printer.RightMargin = 1200  ' 1,2 cm
```

StartUpSinglePage

Property of VcPrinter

This property lets you set/enquire 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 = VcTree1.printer

With printer
    .Orientation = vcLandscape
    .StartUpSinglePage = True
    .FitToPage = False
End With

VcTree1.PrintPreview
```

TopMargin

Property of VcPrinter

This property lets you set/retrieve the height of the top margin of the pages to be printed. The TopMargin values will be processed as follows:

x= TopMargin	actual margin width
x < 100	x cm
x >= 100	x/1000 cm

	Data Type	Explanation
Property value	Long	Height of the top margin of the page

Example Code

```
VcTree1.Printer.TopMargin = 2  ' 2 cm
VcTree1.Printer.TopMargin = 12  ' 12 cm
VcTree1.Printer.TopMargin = 100  ' 0,1 cm
VcTree1.Printer.TopMargin = 200  ' 0,2 cm
VcTree1.Printer.TopMargin = 1200  ' 1,2 cm
```

ZoomFactor

Property of VcPrinter

This property lets you set or enquire the zoomfactor to enlarge or downsize the output (zoom factor = 100: original size, zoom factor > 100: enlargement, zoom factor < 100: reduction).

	Data Type	**Explanation**
Property value	Long	Zoom factor of the diagram

Example Code

```
VcTree1.Printer.ZoomFactor = 150
```

7.28 VcRect

Rect

An object of the type **VcRect** designates a rectangle object and is only available in VcGantt_VcInPlaceEditorShowing.

Properties

- Bottom
- Height
- Left
- Right
- Top
- Width

Properties

Bottom

Property of VcRect

This property returns/sets the bottom coordinate of the VcRect object.

	Data Type	Explanation
Property value	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 VcTree1_OnShowInPlaceEditor(ByVal editObject As Object, _
                           ByVal editObjectType As _
                           VcTreeLib.VcObjectTypeEnum, _
                           ByVal fieldIndex As Long, ByVal objRectComplete As _
                           VcTreeLib.VcRect, ByVal objRectVisible As _
                           VcTreeLib.VcRect, ByVal fldRectComplete As _
                           VcTreeLib.VcRect, ByVal fldRectVisible As _
                           VcTreeLib.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 + VcTree1.Left
            Text1.Top = fldRectVisible.Top + VcTree1.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 + VcTree1.Left
            MonthView1.Top = fldRectVisible.Top + VcTree1.Top

            MonthView1.Value = editObject.DataField(fieldIndex)
            MonthView1.Visible = True
            MonthView1.SetFocus

         Case 13   'Employee
            Combo1.Left = fldRectVisible.Left + VcTree1.Left
            Combo1.Top = fldRectVisible.Top + VcTree1.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 + VcTree1.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
```

7.29 VcTree

Tree

An object of the type **VcTree** is the VARCHART XTree control. You use events to control interactions with the VcTree object. It can be customized by a number of properties and methods to meet your demands.

Properties

- ActiveNodeFilter
- AllowNewNodes
- ArrangementField
- BackColor
- BorderArea
- BoxCollection
- BoxFormatCollection
- CollapseField
- ConfigurationName
- CtrlCXVProcessing
- DataDefinition
- DateOutputFormat
- DialogFont
- EditNewNode
- Enabled
- EnableSupplyTextEntryEvent
- FilePath
- FilterCollection
- FirstVerticalLevel
- HorizontalNodeDistance
- HorizontalNodeIndent
- hWnd
- InPlaceEditingAllowed
- InteractionMode
- LevelField
- MapCollection
- MouseProcessingEnabled
- NodeAppearanceCollection
- NodeCollection
- NodeFormatCollection
- NodeTooltipTextField

- OLEDragMode
- OLEDragWithOwnMouseCursor
- OLEDragWithPhantom
- OLEDropMode
- Printer
- RowLimit
- ScrollOffsetX
- ScrollOffsetY
- ShowToolTip
- TreeViewStyle
- VerticalLevelDistance
- VerticalNodeDistance
- WorldView
- ZoomFactor
- ZoomingPerMouseWheelAllowed

Methods

- AboutBox
- Arrange
- Clear
- CopyNodesIntoClipboard
- CutNodesIntoClipboard
- DeleteNodeRecord
- EditNode
- EndLoading
- GetNodeByID
- GraphicExport
- GraphicExportToFile
- IdentifyFormatField
- IdentifyFormatFieldAsVariant
- IdentifyObjectAt
- IdentifyObjectAtAsVariant
- InsertNodeRecord
- InsertNodeRecordEx
- Open
- PageLayout
- PasteNodesFromClipboard
- PrintDirect
- PrinterSetup
- PrintIt

- PrintPreview
- SaveAs
- ScrollToNodePosition
- ShowAlwaysCompleteView
- SuspendUpdate
- UpdateNodeRecord
- Zoom

Events

- Error
- KeyDown
- KeyPress
- KeyUp
- OLECompleteDrag
- OLEDragDrop
- OLEDragOver
- OLEGiveFeedback
- OLESetData
- OLEStartDrag
- OnBoxLClick
- OnBoxLDblClick
- OnBoxModify
- OnBoxModifyComplete
- OnBoxRClick
- OnDiagramLClick
- OnDiagramLDblClick
- OnDiagramRClick
- OnModifyComplete
- OnMouseDblClk
- OnMouseDown
- OnMouseMove
- OnMouseUp
- OnNodeCollapse
- OnNodeCreate
- OnNodeCreateCompleteEx
- OnNodeDelete
- OnNodeDeleteComplete
- OnNodeExpand
- OnNodeLClick
- OnNodeLDblClick

- OnNodeModifyComplete
- OnNodeModifyCompleteEx
- OnNodeModifyEx
- OnNodeRClick
- OnNodesMarkComplete
- OnNodesMarkEx
- OnSelectField
- OnShowInPlaceEditor
- OnStatusLineText
- OnSupplyTextEntry
- OnSupplyTextEntryAsVariant
- OnToolTipText
- OnToolTipTextAsVariant
- OnWorldViewClosed
- OnZoomFactorModifyComplete

Properties

ActiveNodeFilter

Property of VcTree

This property lets you specify/retrieve a filter that selects the nodes to be displayed. The nodes selected by the filter and their subtrees will 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 = VcTree1.ActiveNodeFilter

If Not Filter Is Nothing Then
    filterName = filter.Name
End If

Set VcTree1.ActiveNodeFilter = VcTree1.FilterCollection. _
                    FilterByName("Filter_1")
```

AllowNewNodes

Property of VcTree

This property permits (True) or prohibits (False) the user to create new nodes. If this property is set to False, the user cannot activate the **Create nodes** mode. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Generation of new nodes allowed/not allowed

Example Code

```
VcTree1.AllowNewNodes = False
```

ArrangementField

Property of VcTree

This property allows to trace the arrangement type of a subtree in a data field. The content of the data field may be **0** (subtree horizontally arranged) or **1** (subtree vertically arranged). A horizontal arrangement is visible only if the immediate or mediate parent nodes are arranged horizontally. This property can also be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Integer	Index of definition field or "-1". When set to "-1", the arrangement type will not be kept in a field. **Default value:** -1

Example Code

```
Dim subTreeArrangement As Integer

subTreeArrangement = VcTree1.ArrangementField
```

BackColor

Property of VcTree

This property lets you assign/retrieve a background color to your tree diagram. The default is white (RGB=(255,255,255)).

	Data Type	Explanation
Property value	Color	RGB color values

Example Code

```
VcTree1.BackColor = RGB(200, 100, 150)
```

BorderArea

Read Only Property of VcTree

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 = VcTree1.BorderArea
```

BoxCollection

Read Only Property of VcTree

This property gives access to the BoxCollection object that contains all boxes available.

	Data Type	Explanation
Property value	VcBoxCollection	BoxCollection object

Example Code

```
Dim boxCollection As VcBoxCollection

Set boxCollection = VcTree1.BoxCollection
```

BoxFormatCollection

Read Only Property of VcTree

This property gives access to the BoxFormatCollection object that contains all box formats available.

	Data Type	Explanation
Property value	VcBoxFormatCollection	BoxFormatCollection object

CollapseField

Property of VcTree

This property allows to trace the state of collapsing in a data field. The content of the data field may be **0** (node expanded) or **1** (node collapsed). The node is visible only if the immediate or mediate parent nodes are expanded. This property can also be set on the **Nodes** property page.

	Data Type	Explanation
Property value	Integer	Index of definition field or "-1". When set to "-1" the collapse status will not be kept in a field. **Default value:** -1

Example Code

```
Dim nodeCollapsed As Integer

nodeCollapsed = VcTree1.CollapseField
```

ConfigurationName

Property of VcTree

This property enables a configuration file (*.ini) to be loaded, that all settings are adopted from, including the corresponding data interface.

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

- *local file:* The default configuration file *vcgantt.ini* should be stored in the directory where the *vctree.ocx* is registered. If you specify the file name without path, *vctree.ini* will be expected to exist in the installation directory. If the specified file does not exist, the default configuration will be loaded, which does not necessarily exist at the end user.

- *URL:* An URL should be used as configuration file only if the configuration is specified during runtime via API because only then the *ini* and *ifd* files will be loaded from the URL specified. (Otherwise, i. e. if you specify an URL as configuration file during design time, the *ini* and *ifd* files will be downloaded too, but they will be stored in the Structured Storage (VB: *frx* file). Then this storing will be used during runtime instead of loading the files directly.) If you embed the VARCHART ActiveX into an HTML page, you can specify the *ini* and *ifd* files directly. Other mechanisms to create a temporary local file are not necessary. Thus browser problems will be avoided.

	Data Type	Explanation
Property value	String	Name of File **Default value:** vctree.ini

Example Code

```
VcTree1.ConfigurationName = "c:\VARCHART\XGantt\sample.ini"
' or:
VcTree1.ConfigurationName = "http://members.tripod.de/netronic_te/ _
                                                   xgantt_sample.ini"
```

CtrlCXVProcessing

Property of VcTree

This property automatically translates the key combination <Ctrl>+<C>, <Ctrl>+<X> and <Ctrl>+<V> into the clipboard commands **CopyNodesTo-Clipboard**, **CutNodesToClipboard** and **PasteNodesFromClipboard**, respectively. You can suppress this feature by setting the property to **False**, in order to avoid conflicts with menu commands in Visual Basic. This property can also be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Key combinations will/will not be translated into clipboard commands **Default value:** True

Example Code

```
VcTree1.CtrlCXVProcessing = True
```

DataDefinition

Read Only Property of VcTree

This property gives access to the current data definition object. The data definition of VcTree contains the data definition table **vcMaindata**.

	Data Type	Explanation
Property value	VcDataDefinition	DataDefinition object

Example Code

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataField As VcDefinitionField

Set dataDefTable = VcTree1.DataDefinition.DefinitionTable(vcMaindata)
Set dataField = dataDefTable.FieldByName("Start")
index = dataField.ID
```

DateOutputFormat

Property of VcTree

This property lets you specify the date output format. The format is a combination of the letters **D** (Day), **M** (Month), **Y** (Year), **h** (hour), **m** (minute), **s** (second) using the separators **:;/-.**.

Different combinations of the the sequences DD, MM, MMM, YY, YYYY, hh, mm and ss are permitted, such as **DD.MMM.YYYY**, **MM/DD/YY**, **DD.MMM.YYYY; hh:mm:ss** etc.

	Data Type	Explanation
Parameter: ⇨ dateFormat	String	Date format MYhms:;/}
Property value	Date/Time	Date {DMYhms:;/}

Example Code

```
VcTree1.DateOutputFormat = "DD.MM.YY"
```

DialogFont

Property of VcTree

This property specifies/retrieves the font name and size in the dialogs of the VARCHART XTree 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 newFont As New StdFont

newFont.Size = 14
newFont.Name = "Arial"
VcTree1.DialogFont = newFont
```

EditNewNode

Property of VcTree

This property specifies whether or not the **Edit Data** dialog box appears when a new node is created. The **AllowNewNodes** property must be set to **True** to enable the user to create new nodes. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	**Edit Data** dialog appears/does not appear.

Example Code

```
VcTree1.EditNewNode = False
```

Enabled

Property of VcTree

This property lets you disable the VARCHART XTree 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

```
VcTree1.Enabled = False
```

EnableSupplyTextEntryEvent

Property of VcTree

This property lets you activate the **OnSupplyTextEntry** event. This event lets you modify the texts of context menus, dialog boxes, error messages, months' and days' names etc. that occur during run time, for example for translation into different languages. This property also can be set on the **General** property page.

	Data Type	Explanation
Property value	Boolean	Property active/not active

Example Code

```
VcTree1.EnableSupplyTextEntryEvent = True
```

FilePath

Property of VcTree

This property lets you set the file path so that graphics files will be found in the directory specified, even if only a relative file name has been 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"
VcTree1.FilePath = graphicsPath
```

FilterCollection

Read Only Property of VcTree

This property gives access to the filter collection object that contains all filters available.

	Data Type	Explanation
Property value	VcFilterCollection	FilterCollection object

Example Code

```
Dim filterCollection As VcFilterCollection

Set filterCollection = VcTree1.FilterCollection
```

FirstVerticalLevel

Property of VcTree

This property lets you set/enquire the level, from that on the nodes are arranged vertically. If set to **-1**, the property is disabled. The arrangement of nodes is to be performed by the **Arrange** method. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	Number of the level, from that on the subtree is arranged vertically **Default value:** -1

Example Code

```
VcTree1.FirstVerticalLevel = 3
VcTree1.Arrange
```

HorizontalNodeDistance

Property of VcTree

This property lets you set/enquire the horizontal distance between two horizontally arranged nodes. Unit: mm. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	Distance (mm) **Default value:** 0

Example Code

```
VcTree1.HorizontalNodeDistance = 10
```

HorizontalNodeIndent

Property of VcTree

This property lets you set/enquire the horizontal indent of vertically arranged nodes. Unit: mm. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	Distance (mm) **Default value:** 0

Example Code

```
VcTree1.HorizontalNodeIndent = 30
```

hWnd

Read Only Property of VcTree

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)
```

InPlaceEditingAllowed

Property of VcTree

This property lets you specify/enquire whether inline editing in node fields is possible or not. You also can set this property on the **General** property page.

Note: If some data fields are to be not editable, specify them as "ReadOnly" in the data definition.

	Data Type	Explanation
Property value	Boolean	Inline editing in node fields possible/ not possible **Default value:** True

Example Code

```
VcTree1.InPlaceEditingAllowed = True
```

InteractionMode

Property of VcTree

This property activates/retrieves one of the available modes of interaction.

	Data Type	Explanation
Property value	InteractionModeEnum	Interaction mode **Default value:** vcPointer
	Possible Values:	
	vcCreateNode 2	Node creating mode
	vcPointer 0	Select mode

Example Code

```
VcTree1.InteractionMode = vcCreateNode
```

LevelField

Property of VcTree

This property lets you specify/enquire the data field that contains the level number of nodes. The level numbers are counted from 0 upwards.

This property also can be set on the **Nodes** property page.

Note: At run time it is not possible to modify the level of a node by modifying the value of the level number data field.

	Data Type	Explanation
Property value	Integer	Level number **Default value:** -1

Example Code

```
VcTree1.LevelField = 4
```

MapCollection

Read Only Property of VcTree

This property gives access to the MapCollection object that contains a defined number of maps. The number of maps is defined by the method **VcMapCollection.SelectMaps**.

	Data Type	Explanation
Property value	VcMapCollection	MapCollection object

Example Code

```
Dim mapCollection As VcMapCollection

Set mapCollection = VcTree1.MapCollection
mapCollection.SelectMaps vcAnyMap
```

MouseProcessingEnabled

Property of VcTree

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 XTree 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 VcTree

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 = VcTree1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance
nodeAppearance.BackColor = RGB(200, 200, 200)
```

NodeCollection

Read Only Property of VcTree

This property gives access to the NodeCollection object, that contains either all nodes (vcAll) or only the marked nodes (vcMarked) or only the visible nodes (vcAllVisible), depending on **NodesSelected**.

	Data Type	Explanation
Property value	VcNodeCollection	NodeCollection object

Example Code

```
Dim numberOfNodes As Long

numberOfNodes = VcTree1.NodeCollection.Count
```

NodeFormatCollection

Read Only Property of VcTree

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 = VcTree1.NodeFormatCollection
```

NodeTooltipTextField

Property of VcTree

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

```
VcTree1.NodeTooltipTextField = 1
```

OLEDragMode

Property of VcTree

By this property you can set/retrieve, whether dragging a node beyond the limits of the current VARCHART XTree control is allowed. This property can also be set on the **General** 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 XTree 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 XTree 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 XTree control **Default value:** vcOLEDragManual
	Possible Values:	
	vcOLEDragAutomatic 1	Method OLEDrag is invoked automatically
	vcOLEDragManual 0	Method OLEDrag needs to be invoked separately.

Example Code

```
VcTree1.OLEDragMode = vcOLEDragAutomatic
```

OLEDragWithOwnMouseCursor

Read Only Property of VcTree

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 via 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 via 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 **General** property page.

	Data Type	Explanation
Property value	Boolean	Cursor does/does not occur in the target control **Default value:** True

Example Code

```
VcTree1.OLEDragWithOwnMouseCursor = False
```

OLEDragWithPhantom

Property of VcTree

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 **General** property page.

	Data Type	Explanation
Property value	Boolean	Phantom does/does not occur **Default value:** True

Example Code

```
VcTree1.OLEDragWithPhantom = False
```

OLEDropMode

Property of VcTree

By this property you can set/retrieve, whether a node from a different VARCHART XTree 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 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 XTree 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 **General** 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

```
VcTree1.OLEDropMode = vcOLEDropAutomatic
```

Printer

Property of VcTree

This method gives access to the printer object. This object lets you set/enquire 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 = VcTree1.Printer.ZoomFactor
printerCuttingMarks = VcTree1.Printer.CuttingMarks
```

RowLimit

Property of VcTree

This property allows to set/retrieve the height of a tree structure. This property also can be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	Maximum number of tree structure rows. Zero indicates that no limit is set. **Default value:** 0

Example Code

```
Dim rowLimit As Integer

rowLimit = VcTree1.RowLimit
```

ScrollOffsetX

Property of VcTree

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 VcTree

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

ShowToolTip

Property of VcTree

This property lets you activate/deactivate the event **OnToolTipText**. This property also can be set on the **General** property page. The event **OnToolTipText** lets you edit the tooltip texts.

	Data Type	Explanation
Property value	Boolean	Property active/not active **Default value:** False

Example Code

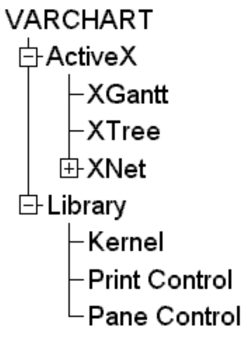
```
VcTree1.ShowToolTip = True
```

TreeViewStyle

Property of VcTree

This property lets you add a **plus** or a **minus** symbol to vertically arranged node levels. The **plus** symbol indicates that the subtree of this node is collapsed, the **minus** symbol indicates that it is expanded. The symbols are set to those nodes only that do have child nodes. Clicking on a plus symbol will expand a subtree and transform the symbol into a minus. Clicking on a minus symbol will collapse the subtree and transform the symbol into a plus.

	Data Type	Explanation
Property value	Boolean	Property active/not active



TreeViewStyle: a collapsed subtree is represented by a ***plus*** *symbol, an expanded one by a* ***minus*** *symbol*

Example Code

```
VcTree1.TreeViewStyle = True
```

VerticalLevelDistance

Property of VcTree

This property lets you set/enquire the distance between two horizontally arranged levels of nodes. Unit: mm. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	Distance (mm) **Default value:** 0

Example Code

```
VcTree1.VerticalLevelDistance = 10
```

VerticalNodeDistance

Property of VcTree

This property lets you set/enquire the distance between two vertically arranged nodes. Unit: mm. This property can also be set on the **Layout** property page.

	Data Type	Explanation
Property value	Integer	Distance (mm) **Default value:** 0

Example Code

```
VcTree1.VerticalNodeDistance = 10
```

WorldView

Read Only Property of VcTree

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 = VcTree1.WorldView
worldview.Visible = True
```

ZoomFactor

Property of VcTree

This property lets you specify/enquire 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 {1...10000}	Zoom factor (%)

Example Code

```
VcTree1.ZoomFactor = 200
```

ZoomingPerMouseWheelAllowed

Property of VcTree

This property lets you set/retrieve whether zooming per Ctrl + mouse wheel is allowed for the user.

	Data Type	Explanation
Property value	Boolean	Zooming allowed (true)/not allowed (false)

Example Code

```
VcTree1.ZoomingPerMouseWheelAllowed = False
```

Methods

AboutBox

Method of VcTree

This method lets you open the **About** box. It contains an overview of the program and the library files currently used with the absolute path and version numbers. This feature makes the hotline support more comfortable. The overview can be selected with the help of the mouse and copied via Ctrl+C and inserted via Ctrl+V into a mail.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcTree1.AboutBox
```

Arrange

Method of VcTree

By this method you can arrange the nodes as set by the property **FirstVerticalLevel**.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcTree1.FirstVerticalLevel = 2
VcTree1.Arrange
```

Clear

Method of VcTree

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

```
VcTree1.Clear
```

CopyNodesIntoClipboard

Method of VcTree

This method lets you copy the selected nodes to the clipboard. Also see methods **CutNodesIntoClipboard** and **PasteNodesFromClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcTree1.CopyNodesIntoClipboard
```

CutNodesIntoClipboard

Method of VcTree

This method lets you cut the marked nodes from the tree diagram and store them to the clipboard. Also see **CopyNodesIntoClipboard** and **PasteNodes-FromClipboard**.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcTree1.CutNodesIntoClipboard
```

DeleteNodeRecord

Method of VcTree

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/was not deleted successfully

Example Code

```
VcTree1.DeleteNodeRecord "A100;;;;;;"
```

EditNode

Method of VcTree

This property invokes the **Edit Data** dialog box for the node passed.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	node whose data are to be edited
Return value	Boolean	Node data were edited/editing was cancelled.

Example Code

```
VcTree1.EditNode node
```

EndLoading

Method of VcTree

This methods shows the end of loading via **InsertNodeRecord**. Thereby an update of the diagram will be triggered.

	Data Type	Explanation
Return value	Boolean	Loading finished {True}

Example Code

```
VcTree1.EndLoading
```

GetNodeByID

Method of VcTree

This method gives access to a node via its ID.

	Data Type	Explanation
Parameter: ⇨ nodeID	Variant	Node identification
Return value	VcNode	Node

Example Code

```
Dim node As VcNode

Set node = VcTree1.GetNodeByID("10")
```

GraphicExport

Method of VcTree

This method lets you invoke a **Save As** dialog box for saving the diagram. Possible formats for saving: WMF (Windows Metafile), EPS (Encapsulated Postscript), VMF (Viewer Metafile). VMF is used by the NETRONIC WebViewer.

	Data Type	Explanation
Return value	Boolean	Graphics successfully exported {True}

Example Code

```
VcTree1.GraphicExport
```

GraphicExportToFile

Method of VcTree

This method lets you store a tree diagram to a file without generating a **Save as** dialog box. Possible formats for saving:

- *.BMP (Microsoft Windows Bitmap)
- *.EPS (Encapsulated Postscript)
- *.GIF (Graphics Interchange Format) (because of the used export library file names with any unicode characters whatever are not possible)
- *.JPG (Joint Photographic Experts Group) (because of the used export library file names with any unicode characters whatever are not possible)
- *.PCX (Microsoft Paint) (because of the used export library file names with any unicode characters whatever are not possible)
- *.PNG (Portable Network Graphics) (because of the used export library file names with any unicode characters whatever are not possible)
- *.TIF (Tagged Image File Format) (because of the used export library file names with any unicode characters whatever are not possible)
- *.VMF (Viewer Metafile)
- *.WMF (Microsoft Windows Metafile)

EPS, VMF and WMF are vector formats that allow to store the file independent of pixel resolution. All other formats are pixel-orientated and confined to a limited resolution. Files of the VMF format can be displayed via the GRANEDA WebViewer on any platform by Java compatible internet browsers.

	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 is put out in the format BMP.
	vcEPS 3	File is put out in the format EPS.
	vcGIF 4	File is put out in the format GIF.
	vcJPG 5	File is put out in the format JPG.
	vcPCX 6	File is put out in the format PCX.
	vcPNG 7	File is put out in the format PNG.
	vcTIF 8	File is put out in the format TIF.
	vcVMF 0	File is put out in the format VMF.
	vcWMF 1	File is put out in the format WMF.
Return value	Boolean	File was/was not stored successfully.

Example Code

```
VcTree1.GraphicExportToFile "C:\xtree1.vmf", vcVMF
```

IdentifyFormatField

Method of VcTree

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 one, **False** will be returned.

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 Vctree1_OnNodeLClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.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 = VcTree1.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 VcTree

This method is identical with the method **IdentifyFormatField** except 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 VcTree

This method lets you identify the object that is located at any position of the diagram. The object type will be returned. Only nodes can be identified.

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 object identified
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeNode 2	object type **node**
	vcObjTypeNone 0	no object
Return value	Boolean	Object identified/no object identified

Example Code

```
Dim x, y As Long
Dim obj As Object
Dim vcObjType As VcObjectTypeEnum
If VcTree1.IdentifyObjectAt(x, y, obj, vcObjType) Then
    If vcObjType = vcObjTypeNode Then
        obj.DataField(1) = "XXX"
    End If
End If
```

IdentifyObjectAtAsVariant

Method of VcTree

This method is identical with the method **IdentifyObjectAt** except 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.

InsertNodeRecord

Method of VcTree

This method lets you load node data. The data will be passed as CSV string in accordance with the structure defined on the **DataDefinition** property page. **EndLoading** should be called when the process of loading nodes is 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"
VcTree1.InsertNodeRecord "A100;Node 1;12.09.02;17.09.02;5;Planning"
VcTree1.InsertNodeRecord "A105;Node 5;13.09.02;18.09.02;7;Testing"

VcTree1.EndLoading
```

InsertNodeRecordEx

Method of VcTree

This method lets you insert nodes by specifying the insertion position and a reference 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 has been completed.

	Data Type	Explanation
Parameter:		
⇨ nodeRecord	Variant	Node record
⇨ position	InsertionPositionEnum	Possible insertion positions
	Possible Values:	
	vcIPFirstChild 3	Insertion as first child of reference node
	vcIPLastChild 4	Insertion as last child of reference node
	vcIPLeftBrother 31	Insertion as left brother of reference node
	vcIPNone 0	unallowed insertion position (valid only for DataObject.DropInsertionPosition)
	vcIPNormal 1	Insertion without reference node
	vcIPParent 6	Insertion as parent of reference node
	vcIPRightBrother 32	Insertion as right brother of reference node
⇨ referenceNode	VcNode	Reference node
Return value	VcNode	Node

Example Code

```
Dim node1 As VcNode

Set node1 = VcTree1.InsertNodeRecordEx("A2;Node 1;12.09.02;17.09.02;5;Planning",
vcIPFirstChild, VcTree1.GetNodeByID("A1"))
VcTree1.EndLoading
```

Open

Method of VcTree

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 settings on the **DataDefinition** property page.

	Data Type	Explanation
Parameter:		
⇨ fileName	String	File name
Return value	Boolean	No meaning {True}

Example Code

```
VcTree1.Open "C:\Data\project1.wbs"
```

PageLayout

Method of VcTree

This method lets you invoke the **Page Setup** dialog box.

	Data Type	Explanation
Return value	Boolean	No meaning {True}

Example Code

```
VcTree1.PageLayout
```

PasteNodesFromClipboard

Method of VcTree

This method lets you paste the nodes from the clipboard into the diagram at a defined position.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Reference node
⇨ pastePosition	PastePositionEnum	Position of the nodes pasted from the clipboard.
	Possible Values:	
	vcPasteAfter 1	In horizontal arrangements, objects are pasted right of the reference node. In vertical arrangements they are pasted below the reference node.
	vcPasteAsFirstChild 2	Objects are pasted as first child nodes below the reference node.
	vcPasteAsLastChild 3	Objects are pasted as last child nodes below the reference node.
	vcPasteBefore 0	In horizontal arrangements, objects are pasted left of the reference node. In vertical arrangemants they are pasted above the reference node.
Return value	Void	

Example Code

```
Dim nodecollection As VcNodeCollection

Set nodecollection = VcTree1.nodecollection
nodecollection.SelectNodes (vcMarked)

If nodecollection.Count = 1 Then
    VcTree1.PasteNodesFromClipboard nodecollection.FirstNode, vcPasteAsLastChild
End If
```

PrintDirect

Method of VcTree

This method lets you print the diagram directly. A dialog box will not be displayed.

	Data Type	Explanation
Return value	Void	

Example Code

```
VcTree1.PrintDirect
```

PrinterSetup

Method of VcTree

This method lets you invoke the Windows **Print Setup** dialog box.

	Data Type	Explanation
Return value	Boolean	No meaning {True}

Example Code

```
VcTree1.PrinterSetup
```

PrintIt

Method of VcTree

This method triggers the printing of the diagram. The parameters defined in the **PageLayout** will be used.

	Data Type	Explanation
Return value	Boolean	No meaning {True}

Example Code

```
VcTree1.PrintIt
```

PrintPreview

Method of VcTree

This method invokes the print preview.

	Data Type	Explanation
Return value	Boolean	No meaning {True}

Example Code

```
VcTree1.PrintPreview
```

SaveAs

Method of VcTree

This method lets you save the current data as a file specified in CSV format. Thereby the structure defined on the **DataDefinition** property page will be used. If an empty string "" is specified, the file last saved with the **Open** method will be overwritten (in accordance with the default **Save** function).

	Data Type	Explanation
Parameter: ⇨ fileName	String	File name
Return value	Boolean	Storing was/was not performed successfully

Example Code

```
VcTree1.SaveAs "C:\Data\project1.wbs"

' or

VcTree1.SaveAs ""
```

ScrollToNodePosition

Method of VcTree

This method allows you to scroll to the row containing a particular node.

	Data Type	Explanation
Parameter:		
⇨ Node	VcNode	Node to be scrolled to
Return value	Boolean	Scrolling was/was not performed successfully.

Example Code

```
Private Sub Vctree1_OnNodeLClick(ByVal node As VcTreeLib.VcNode, _
                            ByVal location As VcTreeLib.LocationEnum, _
                            ByVal x As Long, ByVal y As Long, _
                            returnStatus As Variant)
    'scroll the diagram so that the node is completely on screen
    VcTree1.ScrollToNodePosition node

End Sub
```

ShowAlwaysCompleteView

Method of VcTree

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

```
VcTree1.ShowAlwaysCompleteView
```

SuspendUpdate

Method of VcTree

For projects with many nodes the updating procedure can spend too much time when a certain action is repeated for many nodes. You can accelerate the updating procedure via the **SuspendUpdate** method. Put the code that describes the repeated action into brackets of **SuspendUpdate (True)** and **SuspendUpdate (False)** like in the following example. Then the nodes will be updated all at once, and the performance will be improved.

	Data Type	Explanation
Return value	Boolean	SuspendUpdate(True): Start of the SuspendUpdate method/ SuspendUpdate(False): end of the SuspendUpdate method

Example Code

```
VcTree1.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

VcTree1.SuspendUpdate (False)
```

UpdateNodeRecord

Method of VcTree

This method lets you modify the data of an existing node record. The node 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: ⇨ nodeRecord	Variant	Node record
Return value	VcNode	Node updated

Example Code

```
VcTree1.UpdateNodeRecord "A100;Activity 1;12.09.02;18.09.02;6;Planning"
```

Zoom

Method of VcTree

This method lets you enlarge/reduce the diagram on the display by the specified percentage factor (enlarging the diagram: zoom factor > 100, reducing the diagram: zoom factor < 100).

	Data Type	**Explanation**
Parameter: ⇨ zoomFactor	Integer {1...1000}	Zoom factor
Return value	Boolean	Zooming was performed successfully {True}

Example Code

```
VcTree1.Zoom 120
```

Events

Error

Event of VcTree

This event occurs when an unforeseen error is found in the code of VARCHART XTree. 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**, an error of number 71 (which can be reacted to by an On-Error-Go-To call) will not be output.

Example Code

```
Private Sub VcTree1_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
```

KeyDown

Event of VcTree

This event occurs when the user presses a key while VARCHART XTree has the focus. With the help of the key events you can trigger VARCHART ActiveX functions via 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 VcTree1_KeyDown(KeyCode As Integer, Shift As Integer)
    MsgBox "key pressed"
End Sub
```

KeyPress

Event of VcTree

This event occurs when the user presses and releases an ANSI key while VARCHART XTree has the focus. With the help of the key events you can trigger VARCHART ActiveX functions via the keyboard.

	Data Type	Explanation
Parameter:		
⇨ KeyAscii	Integer	An integer value that returns the numerical key code of a default ANSI key. KeyAscii is returned as reference. If the parameter is changed, a different symbol will be returned to the object. If KeyAscii is set to 0, pressing a key will have no effect, i.e. no symbol will be passed to the object.

Example Code

```
Private Sub VcTree1_KeyPress(KeyAscii As Integer)
    MsgBox "Key pressed and released."
End Sub
```

KeyUp

Event of VcTree

This event occurs when the user releases a key while VARCHART XTree has the focus. With the help of the key events you can trigger VARCHART ActiveX functions via 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 VcTree1_KeyUp(KeyCode As Integer, Shift As Integer)
    MsgBox "key released"
End Sub
```

OLECompleteDrag

Event of VcTree

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	Long	A long integer value that identifies the action performed when dropping the data to 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.
	Possible Values:	
	vcDropEffectCopy 1	Dropping results in a copy of data from the source to the target. The original data have remained unmodified by the drag operation.
	vcDropEffectLink 4	A shortcut was generated for data from the drag source to the drop target.
	vcDropEffectMove 2	Dropping results in data being moved from the drag source to the drop target. The drag source should remove the data after the move.
	vcDropEffectNone 0	Drop target cannot accept the data

OLEDragDrop

Event of VcTree

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**.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	An OLE Drag & Drop-DataObject by which the data is imported.
⇨ effect	Long	A long integer value that describes the action performed when dropping the data to the drop target.
	Possible Values:	

	vcDropEffectCopy 1	Dropping results in a copy of data from the source to the target. The original data have remained unmodified by the drag operation.
	vcDropEffectLink 4	A shortcut was generated for data from the drag source to the drop target.
	vcDropEffectMove 2	Dropping results in data being moved from the drag source to the drop target. The drag source should remove the data after the move.
	vcDropEffectNone 0	Drop target cannot accept the data
⇨ button	Integer	Indicates the mouse button(s) pressed: **1** represents the left button, **2** is the right button, and the middle button is represented by **4**.
⇨ 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 position

OLEDragOver

Event of VcTree

This event occurs when the data is dragged over a drop target and the **OLEDropMode** property of the drop target has been set to **vcOLEDropManual**.

	Data Type	Explanation
Parameter:		
⇨ data	DataObject	An OLE Drag & Drop-DataObject by which the data is imported.
⇔ effect	Long	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.

OLEGiveFeedback

Event of VcTree

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	Long	A long integer value that describes the action performed when dropping the data to the drop target. It is initially set by the source object and identifies all supported OLE Drag & Drop operations. The target component should verify the value and finally set it when the drop procedure actually is taking place.
	Possible Values:	
	vcDropEffectCopy 1	Dropping results in a copy of data from the source to the target. The original data have remained unmodified by the drag operation.
	vcDropEffectLink 4	A shortcut was generated for data from the drag source to the drop target.
	vcDropEffectMove 2	Dropping results in data being moved from the drag source to the drop target. The drag source should remove the data after the move.
	vcDropEffectNone 0	Drop target cannot accept the data
⇦ defaultCursors	Boolean	Determines whether (true) the default mouse cursor or (false) a user-defined mouse cursor is used.

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 VcTree

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 store the data to. The component will invoke the SetData method to load the required 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 VcTree

This event occurs when the **OLEDrag** method is performed, or when the VARCHART XTree 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	Long	A long integer containing the effects that the source component supports. The programmer should provide the values for this parameter in this event.
	Possible Values:	
	vcDropEffectCopy 1	Dropping results in a copy of data from the source to the target. The original data have remained unmodified by the drag operation.
	vcDropEffectLink 4	A shortcut was generated for data from the drag source to the drop target.
	vcDropEffectMove 2	Dropping results in data being moved from the drag source to the drop target. The drag source should remove the data after the move.
	vcDropEffectNone 0	Drop target cannot accept the data

Example Code

```
Private Sub VcTree1_OLEStartDrag(ByVal data As VcTreeLib.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 VcTree

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 position
⇨ y	Long	Y-coordinate of the mouse cursor
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnBoxLClick(ByVal box As VcTreeLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)

    Text1.Text = box.FieldText(1)

End Sub
```

OnBoxLDblClick

Event of VcTree

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 position
⇨ y	Long	Y-coordinate of the mouse position
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnBoxLDblClick(ByVal box As VcTreeLib.VcBox, _
                  ByVal x As Long, ByVal y As Long, returnStatus As Variant)
    box.FieldText(0) = Text1.Text
End Sub
```

OnBoxModify

Event of VcTree

This event occurs when the user has modified a box interactively. The modified VcBox object and the modification type are returned. If you set the return status to **vcRetStatFalse**, the modification will be revoked.

This event should be used only for reading data of the current box, but not for modifying them. For modifying them please use **OnBoxModifyComplete**.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box modified
⇨ modificationType	BoxModificationTypeEnum	Modification type
	Possible Values:	
	vcBMTAnything 1	any modification
	vcBMTNothing 0	no modification
	vcBMTTextModified 4	text modified
	vcBMTXYOffsetModified 2	Offset modified
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnBoxModify(ByVal box As VcTreeLib.VcBox, _
                  ByVal modificationType As _
                  VcTreeLib.BoxModificationTypeEnum, _
                  returnStatus As Variant)

Select Case modificationType
    Case vcBMTAnything: MsgBox "Box modification"
    Case vcBMTXYOffsetModified: MsgBox "Offset changed"
    Case vcBMTTextModified: MsgBox "Box field text changed"
End Select

End Sub
```

OnBoxModifyComplete

Event of VcTree

This event occurs when the modification of the box is finished.

	Data Type	Explanation
Parameter:		
⇨ box	VcBox	Box modified

Example Code

```
Private Sub VcTree1_OnBoxModifyComplete(ByVal box As _
                  VcTreeLib.VcBox)

    MsgBox "The box has been modified."

End Sub
```

OnBoxRClick

Event of VcTree

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 position
⇨ y	Long	Y-coordinate of the mouse position
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnBoxRClick(ByVal box As VcTreeLib.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
```

OnDiagramLClick

Event of VcTree

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 position
⇨ y	Long	Y-coordinate of the mouse position
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnDiagramLClick(ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)
    Dim zoomfactor As Integer

    zoomfactor = VcTree1.Zoomfactor + 10
    VcTree1.Zoomfactor = zoomfactor
End Sub
```

OnDiagramLDblClick

Event of VcTree

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 position
⇨ y	Long	Y-coordinate of the mouse position
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnDiagramLDblClick(ByVal x As Long, _
                                ByVal y As Long, returnStatus As Variant)

    Dim zoomfactor As Integer

    zoomfactor = VcTree1.Zoomfactor - 10
    VcTree1.Zoomfactor = zoomfactor

End Sub
```

OnDiagramRClick

Event of VcTree

This event occurs when the user clicks the right mouse button on the diagram (neither on the date line nor on a node). The position of the mouse (x,y-coordinates) is captured, so that you can for example display your own context menu at the appropriate location. If you set the returnStatus to **vcRetStatNoPopup**, the integrated context menu will be revoked.

	Data Type	Explanation
Parameter:		
⇨ x	Long	X-value
⇨ y	Long	Y-value

⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnDiagramRClick(ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    If MsgBox("Vertical from level 1?", vbYesNo, "First vertical level?") _
                              = vbYes Then
        VcTree1.FirstVerticalLevel = 1
        VcTree1.Arrange
    End If

    returnStatus = vcRetStatNoPopup

End Sub
```

OnModifyComplete

Event of VcTree

This event occurs when data have been modified interactively in the chart, that means it occurs after the following events:

- OnBoxModifyComplete
- 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 VcTree

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 VcTree

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 VcTree

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 VcTree

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

OnNodeCollapse

Event of VcTree

This event is invoked when a user has collapsed a node. If the parameter **action** is **vcSelf**, the node collapsed simultaneously will be the selected one. The operation will be performed by the context menu. If the parameter **action** is **vcComplete**, the selected node will be in the subtree below the node collapsed.

	Data Type	Explanation
Parameter:		
⇨ action	CollapseExpandEnum	Type of collapsing
	Possible Values:	
	vcComplete 1	Nodes in subtree included
	vcSelf 0	Nodes in subtree excluded
⇨ node	VcNode	Node object
⇔ returnStatus	Variant	Return status

OnNodeCreate

Event of VcTree

This event occurs when the user creates a node. The node object is passed by a parameter, so that a validation can be made. (For the validation, the **Edit Data** dialog has to be activated.) If you set the returnStatus to **vcRetStat-False**, the node will not be generated.

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 object to be created
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeCreate(ByVal node As VcTreeLib.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 VcTree

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:	
	vcNodeCreated 1	node created via mouse-click (only for **OnNode-CreateComplete**)
	vcNodesCloned 4	selected nodes were copied via dragging the mouse and pressing the the Ctrl button
⇨ isLastNodeInSeries	Boolean	The node created is/is not the only one or/nor the last one of a node collection.

Example Code

```
Private Sub VcTree1_OnNodeCreateCompleteEx(ByVal node As _
                                 VcTreeLib.VcNode, ByVal creationType As _
                                 VcTreeLib.CreationTypeEnum, _
                                 ByVal isLastNodeInSeries As Boolean)
   'create a record in the underlying database of the application
   addDataRecord node.AllData
End Sub
```

OnNodeDelete

Event of VcTree

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 VcTree1_OnNodeDelete(ByVal node As VcTreeLib.VcNode, _
                                returnStatus As Variant)
   'deny the deletion of the last node in the chart
    If VcTree1.nodecollection.Count = 1 Then
        returnStatus = vcRetStatFalse
        MsgBox ("The last node in the chart cannot be deleted.")
    End If
End Sub
```

OnNodeDeleteComplete

Event of VcTree

This event occurs when the interactive deletion of a node is completed. The node object is returned, so that a validation can be made.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node deleted

OnNodeExpand

Event of VcTree

This property is to be invoked when a user has collapsed a node. If the parameter **action** is **vcSelf**, the node collapsed simultaneously will be the selected one. The operation will be performed by the context menu. If the parameter **action** is **vcComplete**, the node is a child node of the node selected. If you set the return status to **vcRetStatFalse**, the action will be interrupted and the node(s) selected will not be expanded.

	Data Type	Explanation
Parameter:		
⇨ action	CollapseExpandEnum	Type of collapsing
	Possible Values:	
	vcComplete 1	Nodes in subtree included
	vcSelf 0	Nodes in subtree excluded
⇨ node	VcNode	Node object
⇔ returnStatus	Variant	Return status

OnNodeLClick

Event of VcTree

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 VcTree1_OnNodeLClick(ByVal node As VcTreeLib.VcNode, _
                              ByVal location As VcTreeLib.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 VcTree

This event occurs when the user double-clicks the left mouse button on a node. The node object and the mouse position (x,y-coordinates) are captured and passed. If you set the returnStatus to **vcRetStatFalse**, the integrated **Edit data** dialog will be revoked.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node object
⇨ x	Long	x-value
⇨ y	Long	Y-value
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnNodeLDblClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.LocationEnum, _
                                ByVal x As Long, ByVal y As Long, _
                                returnStatus As Variant)

    If node.RightBrotherNode Is Nothing Then
        MsgBox "No right brother"
    Else
        MsgBox (node.RightBrotherNode.AllData)
    End If

    returnStatus = vcRetStatFalse

End Sub
```

OnNodeModifyComplete

Event of VcTree

This event occurs when the modification of the node specified is finished.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	node created

⇨ isLastNodeInSeries	Boolean	The node created is/is not the only one or/nor the last one of a node collection.

Example Code

```
Private Sub VcTree1_OnNodeModifyComplete(ByVal node As VcTreeLib.VcNode, _
                                 ByVal isLastNodeInSeries As Boolean)
    'modify a record in the underlying database of the application
    modifyDataRecord node.AllData
End Sub
```

OnNodeModifyCompleteEx

Event of VcTree

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
	vcMoved 8	Node was moved
	vcNothing 0	no modification

OnNodeModifyEx

Event of VcTree

This event occurs when the user modifies a node. In the course of this, the length or the position of the node or a value in the **Edit Data** dialog may have been changed. The data of the node before and after the modification are passed. By the **modificationType** parameter you get further information of the kind of modification. If you set the returnStatus to **vcRetStatFalse**, the modification will 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 **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
	vcMoved 8	Node was moved
	vcNothing 0	no modification
⬄ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnNodeModifyEx(ByVal oldNode As _
                                 VcTreeLib.VcNode, ByVal node As _
                                 VcTreeLib.VcNode, ByVal modificationType As _
                                 VcTreeLib.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 VcTree

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 context menu will be revoked.

	Data Type	Explanation
Parameter:		
⇨ node	VcNode	Node object
⇨ location	LocationEnum	Section of the chart
	Possible Values:	
	vcInDiagram 1	Located in the node area
⇨ x	Long	y-value
⇨ y	Long	Y-value
⬄ returnStatus	Variant	Return status

Example Code

```
Private Sub VcNet1_OnNodeRClick(ByVal node As VcTreeLib.VcNode, _
                                ByVal location As VcTreeLib.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 VcTree

This event occurs when the operation of marking/demarking a node is finished.

	Data Type	Explanation
Parameter:		
⇦ (no parameter)		No parameter

Example Code

```
Private Sub VcTree1_OnNodesMarkComplete()
    MsgBox "Nodes have been successfully marked."
End Sub
```

OnNodesMarkEx

Event of VcTree

This event occurs when the user has selected one node or several nodes for marking/demarking. The nodeCollection contains these nodes. The parameters **button** and **shift** indicate which control and mouse buttons have been pressed. If you set the return status to **vcRetStatFalse**, you have to mark/demark nodes yourself.

	Data Type	Explanation
Parameter:		
⇨ nodeCollection	VcNodeCollection	NodeCollection that contains the nodes selected by the user. If the user clicked in the diagram, the collection will be 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 VcTree1_OnNodesMarkEx(ByVal NodeCollection As _
                               VcTreeLib.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 VcTree

This event occurs, when in the table area a field (cell) is selected.

	Data Type	Explanation
Parameter:		
editObject	Object	
editObjectType	VcObjectTypeEnum	
	Possible Values:	
	vcObjTypeBox 15	object type **box**
	vcObjTypeNode 2	object type **node**
	vcObjTypeNone 0	no object
fieldIndex	Long	
objRectComplete	VcRect	
objRectVisible	VcRect	
fldRectComplete	VcRect	
fldRectVisible	VcRect	
returnStatus	Variant	

OnShowInPlaceEditor

Event of VcTree

This event occurs when the the implemented editor is started.

The event will be activated only if the property **InPlaceEditingAllowed** is set to True.

Via retstat FALSE this event can be prohibited 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	object type **box**
	vcObjTypeNode 2	object type **node**
	vcObjTypeNone 0	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 VcTree1_OnShowInPlaceEditor(ByVal editObject As Object, _
                  ByVal editObjectType As VcTreeLib.VcObjectTypeEnum, _
                  ByVal fieldIndex As Long, ByVal objRectComplete As _
                  VcTreeLib.VcRect, ByVal objRectVisible As _
                  VcTreeLib.VcRect, ByVal fldRectComplete As _
                  VcTreeLib.VcRect, ByVal fldRectVisible As _
                  VcTreeLib.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 + VcTree1.Left
            Text1.Top = fldRectVisible.Top + VcTree1.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 + VcTree1.Left
            MonthView1.Top = fldRectVisible.Top + VcTree1.Top
            MonthView1.Value = editObject.DataField(fieldIndex)
            MonthView1.Visible = True
            MonthView1.SetFocus

         Case 13   'Employee
            Combo1.Left = fldRectVisible.Left + VcTree1.Left
            Combo1.Top = fldRectVisible.Top + VcTree1.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 VcTree

This event always occurs when messages of general interest are displayed, e.g. a functional note during loading, or the ID of the node the cursor is just positioned.

	Data Type	Explanation
Parameter:		
➪ text	String	Text
➪ paneNo	Integer	Pane index

Example Code

```
Private Sub VcNet1_OnStatusLineText(ByVal Text As String)
    'show text on status bar
    txtStatusBar.Text = Text
End Sub
```

OnSupplyTextEntry

Event of VcTree

This event only occurs when the VcTree property **EnableSupplyText-EntryEvent** is set to **True**. It occurs when a text is displayed. You can use this event for editing the texts of context menus, dialog boxes, info boxes, error messages and the names of days and months.

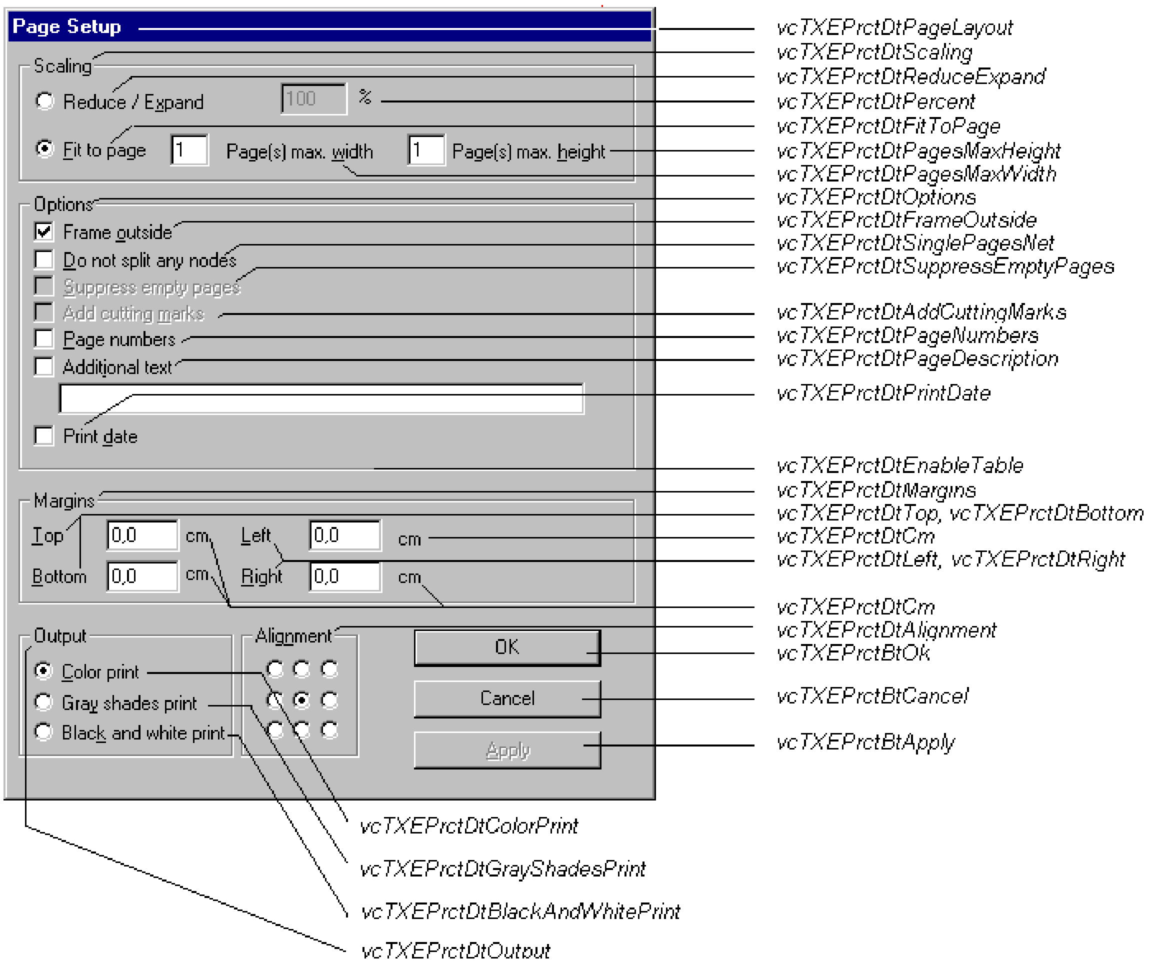
	Data Type	Explanation
Parameter:		
➪ controlIndex	TextEntryIndexEnum	Text to be replaced
	Possible Values:	
	vcTXECtxmenArrowMode 2116	Text in context menu: Cursor in pointer mode
	vcTXECtxmenCollapse 2184	Text in context menu: **Collapse**
	vcTXECtxmenCopyNodes 2152	Text in context menu: **Copy nodes**
	vcTXECtxmenCreateNodeMode 2117	Text in context menu: **Create node mode**
	vcTXECtxmenCutNodes 2151	Text in context menu: **Cut nodes**
	vcTXECtxmenDeleteNode 2101	Text in context menu: **Delete node**
	vcTXECtxmenEditNode 2100	Text in context menu: **Edit node**
	vcTXECtxmenExpand 2185	Text in context menu: **Expand**
	vcTXECtxmenExpandComplete 2186	Text in context menu: **Expand completely**
	vcTXECtxmenFilePrint 2122	Text in context menu: **Print file**
	vcTXECtxmenFilePrintPreview 2121	Text in context menu: **Print file proview**
	vcTXECtxmenFilePrintSetup 2120	Text in context menu: **Setup print file**
	vcTXECtxmenFirstChild 2182	Text in context menu: **Paste node as first child node**
	vcTXECtxmenFullDiagram 2156	Text in context menu: **Restore full tree**
	vcTXECtxmenGraphicExport 2123	Text in context menu: **Export graphics**
	vcTXECtxmenHorizontal 2187	Text in context menu: **Horizontally**
	vcTXECtxmenHorizontalComplete 2188	Text in context menu: **Horizontally completely**

vcTXECtxmenLastChild 2182	Text in context menu: **Paste node as last child node**
vcTXECtxmenPageLayout 2119	Text in context menu: **Page layout**
vcTXECtxmenPasteNodesAfter 2180	Text in context menu: **Paste nodes after**
vcTXECtxmenPasteNodesBefore 2181	Text in context menu: **Paste nodes before**
vcTXECtxmenPasteNodesFirstChild 2182	Text in context menu: **Paste nodes as first child node**
vcTXECtxmenPasteNodesLastChild 2182	Text in context menu: **Paste nodes as last child node**
vcTXECtxmenShowWorldView 2157	Text in context menu: **Show world view**
vcTXECtxmenSubDiagram 2155	Text in context menu: **Create subtree**
vcTXECtxmenVertical 2189	Text in context menu: **Vertically!**
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**
vcTXEDlgNedCaptionPrefix 2024	**Edit data** dialog, text element 1 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 marked node**
vcTXEDlgNedTTGotoLast 2035	**Edit data** dialog, Tooltip “Show last selected node“
vcTXEDlgNedTTGotoNext 2034	**Edit data** dialog, tooltip text **Show next selected node**
vcTXEDlgNedTTGotoPrev 2033	**Edit data** dialog: tooltip text **Show previous marked node**
vcTXEDlgNedValuesColStr 2019	Text in the **Edit data** dialog: **Values**

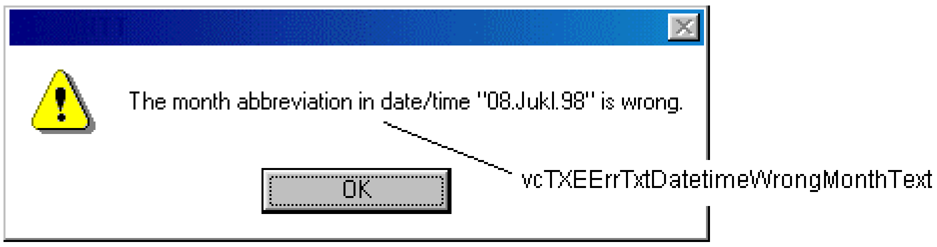
vcTXEErrTxtDatetimeCapitalDay 2705	Message text: "Day" (capitalized) of the date
vcTXEErrTxtDatetimeCapitalMonth 2704	Message text: "Month" (capitalized) of the date
vcTXEErrTxtDatetimeCapitalYear 2703	Message text: "Year" (capitalized) of the date
vcTXEErrTxtDatetimeChar 2716	Message text: "character"
vcTXEErrTxtDatetimeDay 2712	Message text: "day"
vcTXEErrTxtDatetimeHour 2713	Message text: "hour"
vcTXEErrTxtDatetimeMinute 2714	Message text: "minute"
vcTXEErrTxtDatetimeMonth 2711	Message text: "month"
vcTXEErrTxtDatetimeNotFound 2702	Message text: "\n\n%s was not found."
vcTXEErrTxtDatetimeSecond 2715	Message text: "second"
vcTXEErrTxtDatetimeSyntaxCorrect 2701	Message text: Chr(10) & "The correct syntax is ""%s""."
vcTXEErrTxtDatetimeSyntaxError 2700	Message text: "Syntax error in date/time ""%s""."
vcTXEErrTxtDatetimeTooManyParameters 2706	Message text: "\n\nDate/time includes too many parameters."
vcTXEErrTxtDatetimeWrong 2710	Message text: "Date/time ""%s"" includes a wrong %s."
vcTXEErrTxtDatetimeWrongFormat 2717	Message text: "%s cannot be read because the date/time format (%s) is wrong."
vcTXEErrTxtDatetimeWrongMonthText 2709	Message text: "The month abbreviation in date/time ""%s"" is wrong."
vcTXEErrTxtDatetimeWrongYearMax 2708	Message text: "Date/time ""%s"" is too far in the future.\nThe largest possible year number is %s."
vcTXEErrTxtDatetimeWrongYearMin 2707	Message text: "Date/time ""%s"" is too old.\nThe smallest possible year number is %s."
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 page preview dialog: **Overview**
vcTXEPrctBtApply 2318	Button text in page layout dialog: **Apply**
vcTXEPrctBtCancel 2302	Button text in Print Busy box: **Cancel**
vcTXEPrctBtClipboard 2316	Text **Clipboard** button
vcTXEPrctBtClose 2303	Button text in page preview dialog: **Close**
vcTXEPrctBtExport 2317	Text **Export** button
vcTXEPrctBtFitToPage 2308	Button text in page preview dialog: **Fit To Page**
vcTXEPrctBtNext 2305	Button text in page preview dialog: **Next**
vcTXEPrctBtOk 2301	Button text in page layout dialog: **OK**
vcTXEPrctBtOptions 2314	Text **Options** button
vcTXEPrctBtPageLayout 2311	Button text in page preview dialog: **page layout**
vcTXEPrctBtPrevious 2304	Button text in page preview dialog: **Previous**
vcTXEPrctBtPrint 2313	Button text in page preview dialog: **Print**
vcTXEPrctBtPrinterSetup 2312	Button text in page preview dialog: **Setup printer**
vcTXEPrctBtSingle 2307	Button text in page preview dialog: **Single**

vcTXEPrctBtSpool 2315	Text **Spool** button
vcTXEPrctBtZoomIn 2309	Button text for future use
vcTXEPrctBtZoomOut 2310	Button text for future use
vcTXEPrctBtZoomPrint 2319	
vcTXEPrctDtAddCuttingMarks 2514	Text in the **Page Setup** dialog: **Add cutting marks**
vcTXEPrctDtAlignment 2526	Text in the **Page Setup** dialog: **Alignment**
vcTXEPrctDtApplicationName 2501	Text in Print Busy box: **Name of application**
vcTXEPrctDtBlackAndWhitePrint 2525	Text in the **Page Setup** dialog: **Black and white print**
vcTXEPrctDtBottom 2521	Text in the **Page Setup** dialog: **Bottom**
vcTXEPrctDtCm 2530	Text in the **Page Setup** dialog: **cm**
vcTXEPrctDtColorPrint 2523	Text in the **Page Setup** dialog: **Color print**
vcTXEPrctDtDeviceName 2504	Text in Print Busy Box: **Device name**
vcTXEPrctDtDriver 2541	Text **Driver name** dialog
vcTXEPrctDtEnableDiagram 2559	Text in **Page Setup** dialog: **Diagram only**
vcTXEPrctDtExport 2567	
vcTXEPrctDtExportPage 2568	
vcTXEPrctDtFitToPage 2508	Text in the **Page Setup** dialog: **Fit to page**
vcTXEPrctDtFrameOutside 2515	Text in the **Page Setup** dialog: **Frame outside**
vcTXEPrctDtGrayShadesPrint 2524	Text in the **Page Setup** dialog: **Gray shades print**
vcTXEPrctDtLeft 2520	Text in the **Page Setup** dialog: **Left**
vcTXEPrctDtMargins 2529	Text in the **Page Setup** dialog: **Margins**
vcTXEPrctDtMessage 2555	Text **Message** dialog
vcTXEPrctDtOff 2557	Text **Off** dialog
vcTXEPrctDtOptions 2528	Text in the **Page Setup** dialog: **Options**
vcTXEPrctDtOutputFormat 2554	Text **Output Format** dialog
vcTXEPrctDtPageDescription 2562	Text in **Page Setup** dialog: **Additional Text**
vcTXEPrctDtPageLayout 2532	Window title of the **Page Setup** dialog
vcTXEPrctDtPageNumbers 2518	Text in the **Page Setup** dialog: **Page numbers**
vcTXEPrctDtPagePreview 2533	Window title of the **Page Setup** dialog
vcTXEPrctDtPagesMaxHeight 2511	Text in the **Page Setup** dialog: **Page(s) max. height**
vcTXEPrctDtPagesMaxWidth 2510	Text in the **Page Setup** dialog: **Page(s) max. width**
vcTXEPrctDtPercent 2509	Text in the **Page Setup** dialog: **Reduce/Expand**
vcTXEPrctDtPrint 2506	Text in Print Busy Box: **Print**
vcTXEPrctDtPrintDate 2564	Text in **Page Setup** dialog: **Print date**
vcTXEPrctDtPrinterSetup 2536	Text in the **Page Preview** dialog: **Printer Setup**
vcTXEPrctDtPrintingPage 2556	Text in Print Busy Box: **Print page(s)**
vcTXEPrctDtProjectName 2502	Text in Print Busy Box: **project name**
vcTXEPrctDtReduceExpand 2507	Text in the **Page Setup** dialog: **Reduce/Expand**

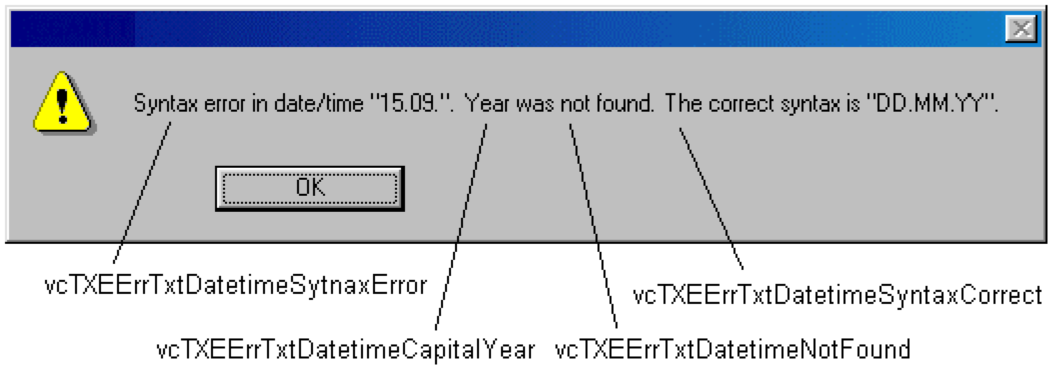
	vcTXEPrctDtRepeatTable 2565	Text in **Page Setup** dialog: **Repeat table**
	vcTXEPrctDtRight 2522	Text in the **Page Setup** dialog: **Right**
	vcTXEPrctDtRollmedia 2542	Text in **Page Setup** dialog: **Roll media**
	vcTXEPrctDtScaling 2527	Text in the **Page Setup** dialog: **Scaling**
	vcTXEPrctDtSinglePagesNet 2563	Text in **Page Setup** dialog: **Do/do not split nodes**
	vcTXEPrctDtSuppressEmptyPages 2517	Text in the **Page Setup** dialog: **Suppress empty pages**
	vcTXEPrctDtTableCols 2566	
	vcTXEPrctDtTo 2505	Text in Print Busy Box: **To**
	vcTXEPrctDtTop 2519	Text in the **Page Setup** dialog: **Top**
	vcTXEPrctDtWmf 2552	Text **Wmf** dialog
	vcTXEPrctMtBottomMargin 2409	Message text: **Bottom margin...**
	vcTXEPrctMtExportNotPossible 2415	Message text: **Export not possible**
	vcTXEPrctMtIncompatibleVcVersion 2414	Message text: **VcVersion incompatible**
	vcTXEPrctMtLeftMargin 2406	Message text: **Left margin...**
	vcTXEPrctMtModifyPageLayout 2412	Message text: **Page Layout settings will be modified**
	vcTXEPrctMtPaperHeightToLarge 2417	Message text: **Paper height too large**
	vcTXEPrctMtPaperWidthToLarge 2416	Message text: **Paper width too large**
	vcTXEPrctMtPrinterBusy 2403	**Message text: Printer Busy**
	vcTXEPrctMtPrinterNotInstalled 2411	Message text: **Printer not installed**
	vcTXEPrctMtPrintingNotPossible 2402	Message text: **Printing not possible at time**
	vcTXEPrctMtRightMargin 2408	Message text: **Right margin...**
	vcTXEPrctMtSelectPaperSize 2413	Message text: **Selected paper size too small**
	vcTXEPrctMtTopMargin 2407	Message text: **Top margin...**
	vcTXEPrctMtValueOutOfRange 2404	Message text: **Value out of range**
	vcTXEPrctMtVersion 2401	Message text: **Version**
	vcTXEPrctMtWarning 2418	Message text: **Warning**
	vcTXEPrctMtWillBeAdjustedTo 2410	Message text: **Will be adjusted to...**
	vcTXEPrctMtWrongCharacter 2405	Message text: **Invalid character**
⇨ textEntry	String	Text replacing the default
⇔ returnStatus	Variant	Return status



Constants of the button texts of the **Page Setup** dialog


The month abbreviation in date/time "08.Jukl.98" is wrong.

Constants of the error message **Date error, wrong month**

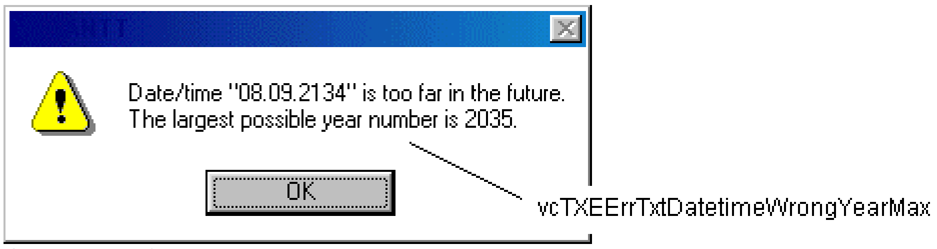


Syntax error in date/time "15.09.". Year was not found. The correct syntax is "DD.MM.YY".

OK

vcTXEErrTxtDatetimeSytnaxError
vcTXEErrTxtDatetimeSyntaxCorrect
vcTXEErrTxtDatetimeCapitalYear
vcTXEErrTxtDatetimeNotFound

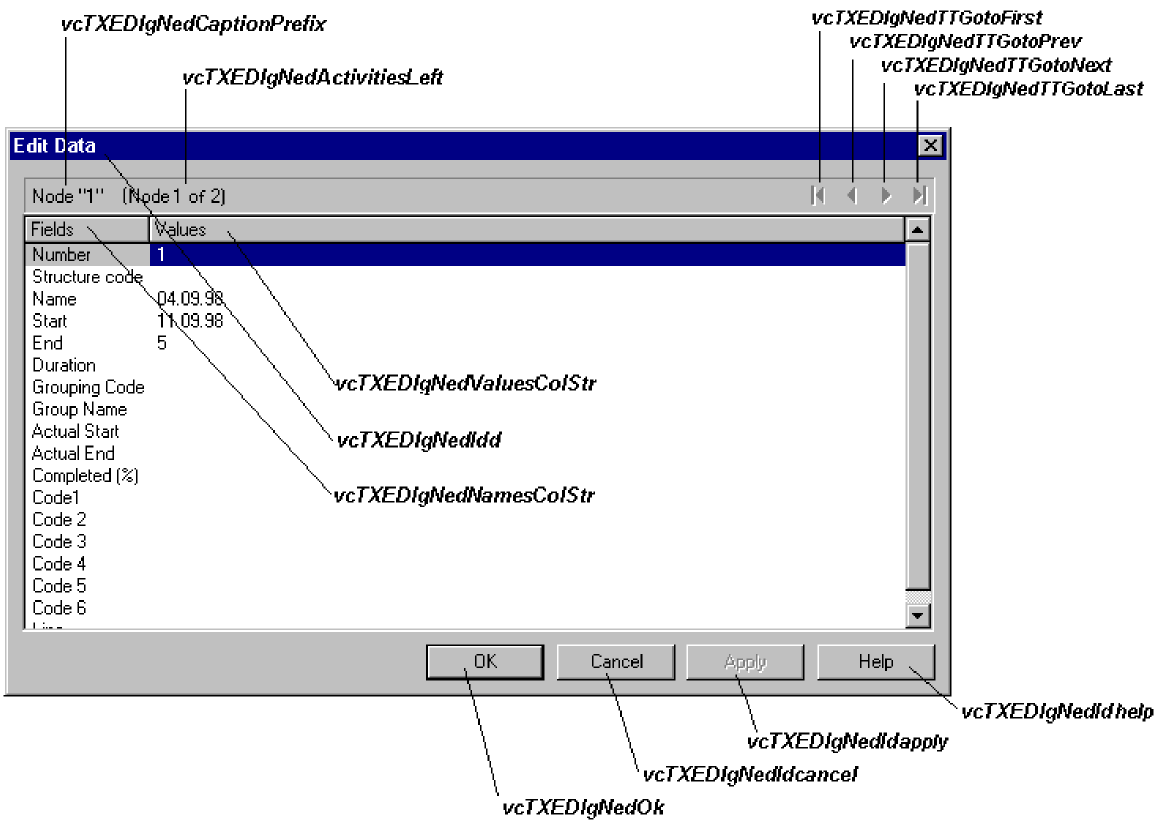
Constants of the error message ***Syntax error***



Date/time "08.09.2134" is too far in the future.
The largest possible year number is 2035.

OK

Constants of the error message ***Date error, maximum year exceeded***



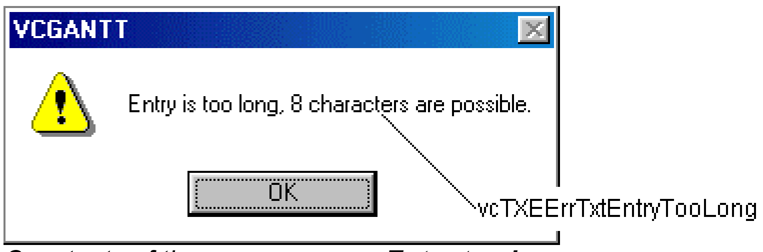
Edit Data

Node "1" (Node 1 of 2)

Fields	Values
Number	1
Structure code	
Name	04.09.98
Start	11.09.98
End	5
Duration	
Grouping Code	
Group Name	
Actual Start	
Actual End	
Completed (%)	
Code1	
Code 2	
Code 3	
Code 4	
Code 5	
Code 6	

OK Cancel Apply Help

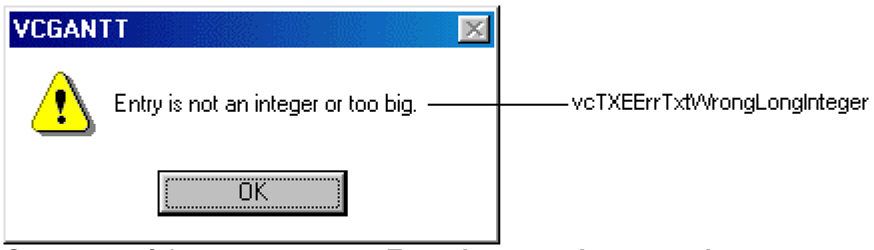
Constants of the dialog ***Edit data***


VCGANTT

Entry is too long, 8 characters are possible.

OK

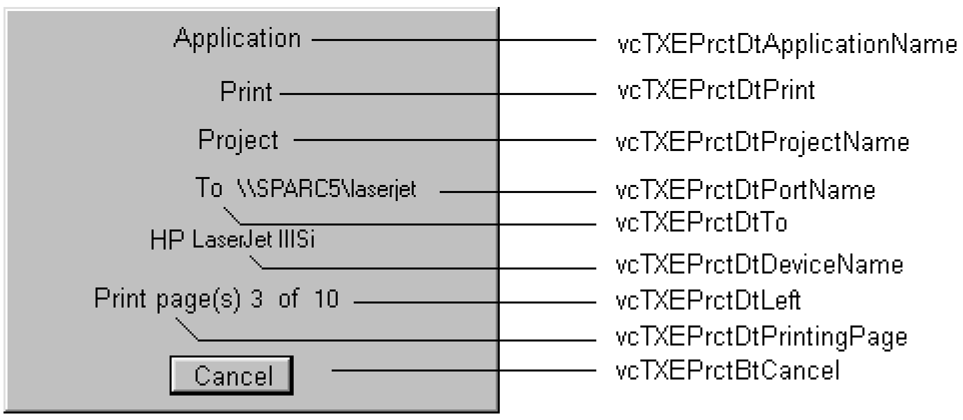
Constants of the error message **Entry too large**


VCGANTT

Entry is not an integer or too big.

OK

Constants of the error message **Entry is not an integer value**


Application

Print

Project

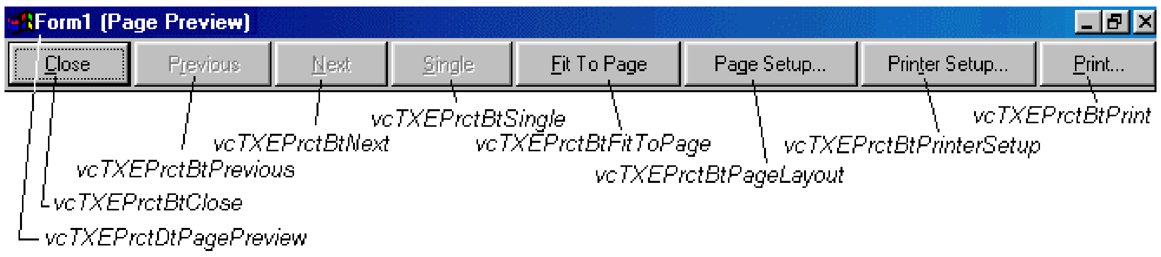
To \\SPARC5\laserjet

HP LaserJet IIISi

Print page(s) 3 of 10

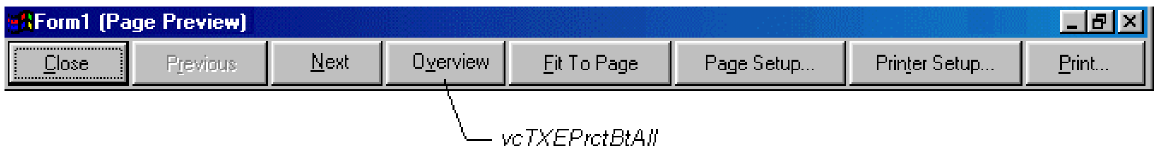
Cancel

Constants of the info box **Printing**


Form1 (Page Preview)

Close | Previous | Next | Single | Fit To Page | Page Setup... | Printer Setup... | Print...

Constants of the button texts of the **Page Preview** dialog


Form1 (Page Preview)

Close | Previous | Next | Overview | Fit To Page | Page Setup... | Printer Setup... | Print...

Constants of the button texts of the **Page Preview, Overview** dialog

Example Code

```
Private Sub VcTree1_OnSupplyTextEntry(ByVal controlIndex As _
                                VcTreeLib.TextEntryIndexEnum, _
                                TextEntry As String, _
                                returnStatus As Variant)
    Select Case controlIndex
        Case vcTXECtxmenCollapse
            TextEntry = "Collapse nodes"
        Case vcTXECtxmenExpand
            TextEntry = "Expand nodes"
        End Select
End Sub
```

OnSupplyTextEntryAsVariant

Event of VcTree

This event is identical with the event **OnSupplyTextEntry** except 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 VcTree

This event only occurs when the VcGantt property **ShowToolTip** is set to **True**. It occurs when a tooltip for an object should be displayed. 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	object type **box**
	vcObjTypeNode 2	object type **node**
	vcObjTypeNone 0	no object
⇨ x	Long	X-value
⇨ y	Long	Y-value
⇨ ToolTipText	String	Text to be displayed
⇔ returnStatus	Variant	Return status

Example Code

```
Private Sub VcTree1_OnToolTipText(ByVal hitObject As Object, _
                                  ByVal hitObjectType As _
                                  VcTreeLib.VcObjectTypeEnum, _
                                  ByVal x As Long, ByVal y As Long, _
                                  toolTipText As String, _
                                  returnStatus As Variant)
    If hitObjectType = vcObjTypeNode Then
        toolTipText = "The cursor has been moved over a node!"
    End If
End Sub
```

OnToolTipTextAsVariant

Event of VcTree

This event is identical with the event **OnToolTipText** except 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 VcTree

This event occurs when the worldview popup window is closed.

	Data Type	Explanation
Parameter:		
⇨ (no parameter)		

Example Code

```
Private Sub VcTree1_OnWorldViewClosed()
    MsgBox "Do you want to close the worldview window?", vbOKCancel
End Sub
```

OnZoomFactorModifyComplete

Event of VcTree

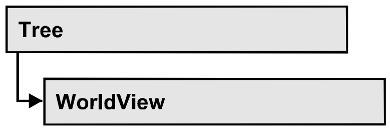
This events occurs when the user has modified the size of the rectangle in the Worldview or when he has zoomed onto objects marked by drawing a rectangle and then clicking the right mouse button. You also can zoom by turning the mouse wheel while you press the Ctrl button or via the Plus or Minus keys of the number block of the keyboard while you press the Ctrl button.

	Data Type	Explanation
Parameter:		
⇨ (no parameter)		

Example Code

```
Private Sub VcTree1_OnZoomFactorModifyComplete()
     MsgBox "Zoomfactor: " & VcTree1.ZoomFactor
End Sub
```

7.30 VcWorldView



An object of the type **VcWorldView** designates the world view window.

Properties

- Border
- Height
- Left
- MarkingColor
- Mode
- ParentHWnd
- Top
- Visible
- Width

Properties

Border

Property of VcWorldView

This property lets you set/enquire whether the world view has a frame (not in **vcPopupWindow** mode). This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Boolean	World view with a border line (True)/without border line (False) **Default value:** True

Example Code

```
VcTree1.WorldView.Mode = vcNotFixed
VcTree1.WorldView.Border = True
```

Height

Property of VcWorldView

This property lets you enquire the vertical extension of the world view. In the modi **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** and **vcPopupWindow** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a calculation from/in Twips via the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Long	Height of the world view {0, ...} **Default value:** 100

Example Code

```
VcTree1.WorldView.Height = 100
```

Left

Property of VcWorldView

This property lets you enquire the left position of the world view. In the modi **vcNotFixed** und **vcPopupWindow** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a calculation from/in Twips via the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Long	Left position of the world view **Default value:** 0

Example Code

```
VcTree1.WorldView.Left = 200
```

MarkingColor

Property of VcWorldView

This property lets you enquire/set the line color of the rectangle that indicates in the World View the currently selected section. This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Color	RGB color values **Default value:** RGB(0, 255, 0)

Example Code

```
VcTree1.WorldView.MarkingColor = RGB(255, 0, 0)
```

Mode

Property of VcWorldView

This property lets you enquire/set the world view mode. This property also can be set on the **World View** 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 VARCHART ActiveX control window. Then the height can be specified, whereas the position and the width are fixed.
	vcFixedAtLeft 1	The world view is displayed on the left side of the VARCHART ActiveX control window. Then the width can be specified, whereas the position and the height are fixed.
	vcFixedAtRight 2	The world view is displayed on the right side of the VARCHART ActiveX control window. Then 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 VARCHART ActiveX control window. Then the height can be specified, whereas the position and the width are fixed.
	vcNotFixed 5	The world view is a child window of the current parent window of the VARCHART 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 world 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

```
VcTree1.WorldView.Mode = vcNotFixed
```

ParentHWnd

Property of VcWorldView

This property lets you set in the **vcNotFixed** mode the HWnd handle of the parent window, if, e. g., the world view is to appear in a frame window implemented by your own. Per default, this 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 (VcTree1.worldview.ParentHWnd)
```

Top

Property of VcWorldView

This property lets you enquire the top position of the world view. In the modi **vcNotFixed** und **vcPopupWindow** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a calculation from/in Twips via the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Long	Top position of the world view **Default value:** 0

Example Code

```
VcTree1.WorldView.Top = 20
```

Visible

Property of VcWorldView

This property lets you enquire/set whether the world view is visible or not. This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Boolean	World view visible (True)/not visible (False) **Default value:** False

Example Code

```
VcTree1.WorldView.Visible = True
```

Width

Property of VcWorldView

This property lets you enquire the horizontal extension of the world view. In the modi **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** and **vcPopupWindow** it also can be set.

Please note that the pixel coordinates are system coordinates, i. e. in Visual Basic you have to perform a calculation from/in Twips via the properties **App.TwipsPerPixelX** and **App.TwipsPerPixelY**.

This property also can be set on the **World View** property page.

	Data Type	Explanation
Property value	Long	Horizontal extension of the world view {0, ...} **Default value:** 100

Example Code

```
VcTree1.WorldView.Width = 200
```

8 Index