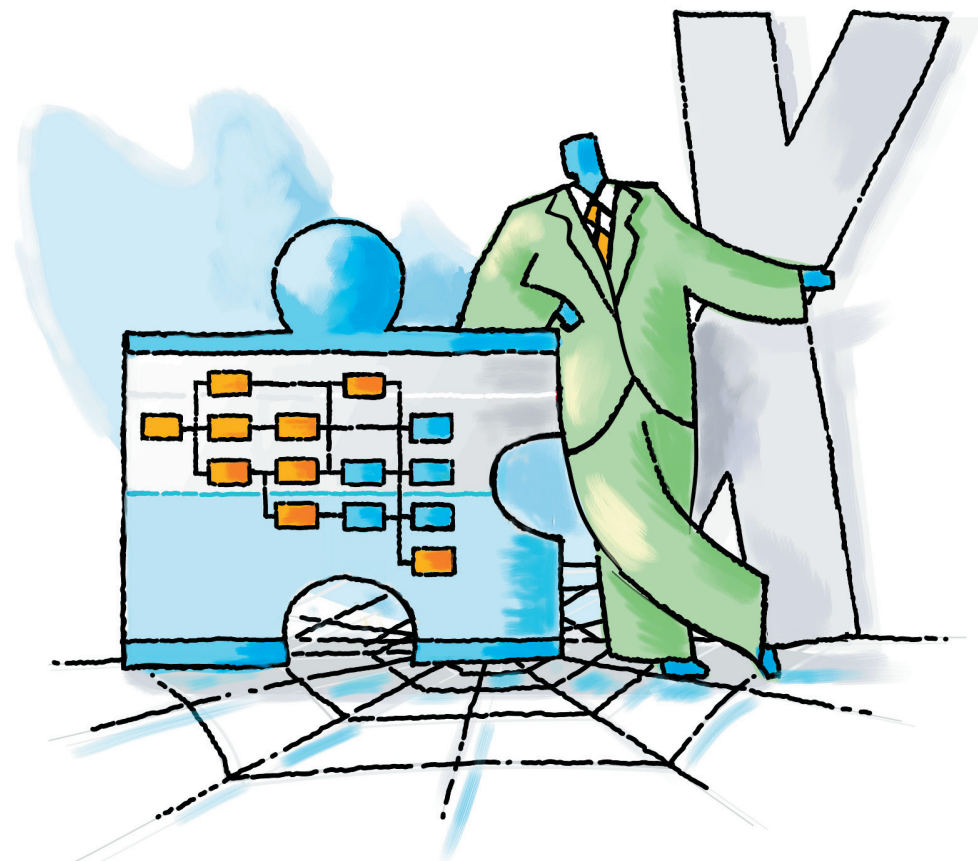


# VARCHART XNet

.NET Edition 5.2  
Benutzer- und  
Referenzhandbuch



# **VARCHART XNet .NET Edition**

**Version 5.2**

**Benutzerhandbuch**

NETRONIC Software GmbH  
Pascalstraße 15  
52076 Aachen  
Deutschland  
Tel: +49 (0) 2408 141-0  
Fax: +49 (0) 2408 141-33  
E-Mail [sales@netronic.de](mailto:sales@netronic.de)  
[www.netronic.de](http://www.netronic.de)

© Copyright 2020 NETRONIC Software GmbH  
Alle Rechte vorbehalten.

Die Informationen im vorliegenden Benutzerhandbuch werden ohne Rücksicht auf einen eventuellen Patentschutz veröffentlicht. Warennamen werden ohne Gewährleistung der freien Verwendbarkeit benutzt. Bei der Zusammenstellung von Texten und Abbildungen wurde mit größter Sorgfalt vorgegangen. Trotzdem können Fehler nicht vollständig ausgeschlossen werden. Herausgeber und Autoren können für fehlerhafte Angaben und deren Folgen weder eine juristische Verantwortung noch irgendeine Haftung übernehmen.

Das illegale Kopieren und Vertreiben dieses Produktes stellt einen Diebstahl geistigen Eigentums dar und wird von NETRONIC Software GmbH strafrechtlich verfolgt.

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Die gewerbliche Nutzung der in diesem Buch gezeigten Abbildungen ist nicht zulässig.

Microsoft Windows, Microsoft Explorer, Microsoft Visual Basic und Microsoft Visual Studio sind Warenzeichen der MICROSOFT Corp.

Bearbeitungsstand: 27 April 2020

# Inhaltsverzeichnis

---

<b>1</b>	<b>Einleitung</b>	<b>9</b>
1.1	VARCHART XNet auf einen Blick	9
1.2	Installation	12
1.3	Lizenzierung	14
1.4	Auslieferung	15
1.5	Verwendung der deutschen Version	17
1.6	Unterstützung und Beratung	19

---

<b>2</b>	<b>Tutorium</b>	<b>21</b>
2.1	Überblick	21
2.2	Das Steuerelement in einem Formular platzieren	22
2.3	VARCHART XNet automatisch skalieren	23
2.4	Schnittstelle einrichten	24
2.5	Der erste Lauf	28
2.6	Daten aus einer Datei einlesen	32
2.7	Flußrichtung des Netzdiagramms festlegen	35
2.8	Knoten und Verbindungen erzeugen und bearbeiten	37
2.9	Knoten und Verbindungen markieren	39
2.10	Filter für Knoten festlegen	40
2.11	Knotenaussehen festlegen	43
2.12	Knotenformate festlegen	47
2.13	Das Aussehen von Verbindungen festlegen	50
2.14	Positionen von Knoten und Verbindungsbeschriftungen speichern	55
2.15	Hilfsknoten übersichtlich anordnen	58
2.16	Knoten gruppieren	63
2.17	Zeitrechnung des VARCHART XNet festlegen	65
2.18	Diagramm drucken	71
2.19	Diagramm exportieren	72
2.20	Konfigurationseinstellungen speichern	73

## 4 Inhaltsverzeichnis

---

<b>3</b>	<b>Wichtige Konzepte</b>	<b>75</b>
3.1	Boxen	75
3.2	Datentabellen	79
3.3	Datumsangaben und Zeitumstellung	88
3.4	Drag & Drop	90
3.5	Ereignisse	92
3.6	Filter	93
3.7	Grafikformate	95
3.8	Gruppierung	100
3.9	In-Flow-Gruppierung	103
3.10	Knoten	105
3.11	Knotenaussehen	111
3.12	Knotenformat	113
3.13	Komplettansicht (World View)	116
3.14	Laufzeitsicherheitsrichtlinien für den Einsatz im Internet Explorer	118
3.15	Legendenansicht (Legend View)	120
3.16	Plattformen x86 und x64	122
3.17	Schreiben von PDF-Dateien	126
3.18	Sprachanpassung von Textausgaben	129
3.19	Statuszeilentext	131
3.20	Tooltips zur Laufzeit	132
3.21	Verbindungen	133
3.22	Verbindungsaussehen	139
3.23	Viewer Metafile (*.vmf)	141
3.24	Zeitrechnung	142
3.25	Zuordnungstabellen	144

---

<b>4</b>	<b>Eigenschaftenseiten und Dialogfelder</b>	<b>149</b>
4.1	Allgemeines	149
4.2	Eigenschaftenseite "Allgemeines"	151
4.3	Eigenschaftenseite "Außenbereich"	161
4.4	Eigenschaftenseite "Gruppierung"	163

4.5	Eigenschaftenseite "Knoten"	166
4.6	Eigenschaftenseite "Zusätzliche Ansichten"	169
4.7	Eigenschaftenseite "Objekte"	173
4.8	Eigenschaftenseite "Verbindungen"	175
4.9	Eigenschaftenseite "Zeitrechnung"	177
4.10	Dialogfeld "Datentabellen verwalten"	179
4.11	Dialogfeld "Filter verwalten"	182
4.12	Dialogfeld "Filter bearbeiten"	184
4.13	Dialogfeld "Zuordnungstabellen verwalten"	188
4.14	Dialogfeld "Zuordnungstabelle bearbeiten"	190
4.15	Dialogfeld "Zuordnung einstellen"	192
4.16	Dialogfeld "Knotenaussehen verwalten"	194
4.17	Dialogfeld "Knotenaussehen bearbeiten"	198
4.18	Dialogfeld "Boxen verwalten"	202
4.19	Dialogfeld "Box bearbeiten"	205
4.20	Dialogfeld "Boxformate/Knotenformate verwalten"	207
4.21	Dialogfeld "Boxformat bearbeiten"	209
4.22	Dialogfeld "Knotenformat bearbeiten"	212
4.23	Dialogfeld "Verbindungsformate verwalten"	217
4.24	Dialogfeld "Verbindungsformat bearbeiten"	219
4.25	Dialogfeld "Verbindungsaussehen verwalten"	222
4.26	Dialogfeld "In-Flow-Gruppierung bearbeiten"	226
4.27	Dialogfeld "Linienattribute bearbeiten"	229
4.28	Dialogfeld "Musterattribute bearbeiten"	230
4.29	Dialogfeld "Kalender festlegen"	231
4.30	Dialogfeld "Intervalle verwalten" (Kalender)	233
4.31	Dialogfeld "Kalenderprofile verwalten"	235
4.32	Dialogfeld "Intervalle verwalten" für Tagesprofil	236
4.33	Dialogfeld "Intervalle verwalten" für Wochenprofil	238
4.34	Dialogfeld "Intervalle verwalten" für Variables Profil	239
4.35	Dialogfeld "Intervalle verwalten" für Jahresprofil	241
4.36	Dialogfeld "Texte, Grafiken und Legende festlegen"	243
4.37	Dialogfeld "Legendenattribute"	247
4.38	Dialogfeld "Lizenzierung"	249

## 6 Inhaltsverzeichnis

4.39	Dialogfeld "Lizenzinformationen anfordern"	251
------	--	-----

---

<b>5</b>	<b>Benutzerschnittstelle</b>	<b>253</b>
5.1	Übersicht	253
5.2	Navigation im Diagramm	254
5.3	Zoomen	255
5.4	Knotendaten bearbeiten	257
5.5	Verbindungen bearbeiten	259
5.6	Navigation mit Hilfe der Tastatur	260
5.7	Knoten und Verbindungen erzeugen	261
5.8	Knoten und Verbindungen markieren, löschen oder verschieben	263
5.9	Seite einrichten	264
5.10	Druckvorschau	268
5.11	Kontextmenü für das Diagramm	271
5.12	Kontextmenü für Knoten	275
5.13	Kontextmenü für Verbindungen	276
5.14	Kontextmenü für die Legende	277

---

<b>6</b>	<b>Häufig gestellte Fragen</b>	<b>279</b>
6.1	Was muss ich tun, wenn ich von VARCHART XGantt 4.4 für .NET nach XGantt 5.0 umsteigen möchte?	279
6.2	Was muss ich tun, wenn ich im Rahmen eines Service Releases auf einen neuen Build von VARCHART XGantt umsteigen möchte?	280
6.3	Warum erscheint eine Fehlermeldung, wenn man bei Visual Studio 2010 ein neues Projekt erzeugt und versucht, das Steuerelement auf die Form zu ziehen?	282
6.4	Wie kann das VARCHART Windows Forms neu lizenziert werden?	283
6.5	Wieso können Knoten u. U. nicht interaktiv erzeugt werden?	284
6.6	Wieso können Verbindungen u. U. nicht interaktiv erzeugt werden?	285
6.7	Wie verhindert man das interaktive Erzeugen von Knoten?	286
6.8	Wie lassen sich die Standard-Kontextmenüs abschalten?	287
6.9	Wie lässt sich die Performance verbessern?	288
6.10	Fehlermeldungen	289

6.11	Was tun, wenn das Steuerelement unerwartet nicht in allen Benutzerkonten eines Rechners funktioniert?	290
6.12	Sind alle Fonts verwendbar?	291

---

## **7 API Referenz 293**

7.1	Objekttypen	293
7.2	VcBorderArea	295
7.3	VcBorderBox	297
7.4	VcBox	306
7.5	VcBoxCollection	318
7.6	VcBoxFormat	325
7.7	VcBoxFormatCollection	331
7.8	VcBoxFormatField	338
7.9	VcCalendar	347
7.10	VcCalendarCollection	356
7.11	VcCalendarProfile	363
7.12	VcCalendarProfileCollection	366
7.13	VcDataRecord	372
7.14	VcDataRecordCollection	378
7.15	VcDataTable	386
7.16	VcDataTableCollection	390
7.17	VcDataTableField	397
7.18	VcDataTableFieldCollection	404
7.19	VcFilter	410
7.20	VcFilterCollection	417
7.21	VcFilterSubCondition	424
7.22	VcGroup	428
7.23	VcGroupCollection	435
7.24	VcInterval	439
7.25	VcIntervalCollection	447
7.26	VcLegendView	452
7.27	VcLink	460
7.28	VcLinkAppearance	466
7.29	VcLinkAppearanceCollection	476



## 8 Inhaltsverzeichnis

7.30	VcLinkCollection	483
7.31	VcLinkFormat	487
7.32	VcLinkFormatCollection	491
7.33	VcLinkFormatField	498
7.34	VcMap	502
7.35	VcMapCollection	509
7.36	VcMapEntry	517
7.37	VcNet	526
7.38	VcNode	675
7.39	VcNodeAppearance	683
7.40	VcNodeAppearanceCollection	706
7.41	VcNodeCollection	712
7.42	VcNodeFormat	716
7.43	VcNodeFormatCollection	721
7.44	VcNodeFormatField	728
7.45	VcPrinter	742
7.46	VcRect	761
7.47	VcScheduler	765
7.48	VcWorldView	773

---

## 8 Index

783

---

---

# 1 Einleitung

---

## 1.1 VARCHART XNet auf einen Blick

VARCHART XNet ist eine interaktive Diagrammkomponente zur Visualisierung von grafikbezogenen Daten. Mit Hilfe von VARCHART XNet können Sie Ihre Daten in Szenarios wie Geschäftsprozessmodellierung und Workflow-Design anzeigen, bearbeiten und drucken. Der leistungsfähige eingebaute Layout-Algorithmus und das integrierte Zeitrechnungsmodul eignen sich hervorragend für Vorgangs-Netzdiagramme im Projektmanagement. Auch Klassen- und Entity-Relationship-Diagramme lassen sich mit VARCHART XNet erstellen. Nutzen Sie die umfangreichen Gestaltungsmöglichkeiten und erzeugen Sie innerhalb weniger Minuten eine erste grafische Darstellung Ihrer Daten .

### > Kurzer Überblick über die Leistungsmerkmale

- **Beschriftungsboxen**

Zusätzlich zum Diagramm lassen sich Informationsboxen, die Beschriftungen und Bilder enthalten können, frei platzieren.

- **Automatisches Layout**

Das automatische Layout von Knoten sorgt für Übersichtlichkeit bei großen und komplizierten Strukturen.

- **Kalender**

Kalender bestehen aus einer lückenlosen Abfolge von Arbeitszeiten und arbeitsfreien Zeiten und werden in Zeitberechnungen verwendet, um arbeitsfreie Zeiten zu berücksichtigen. Sie können individuelle Kalender für Knoten verwenden.

- **Clustering**

Um größere Diagrammstrukturen komfortabel zu handhaben, können Knoten in Clustern gruppiert werden. Sie können kollabiert und expandiert werden. Sie können das Aussehen und die Information des Stellvertreterknotens, der dargestellt werden soll, bestimmen.

- **Datenschnittstelle**

Nutzen Sie die flexible Datenschnittstelle um bestehende Datenstrukturen einfach anzupassen. Wie in einer relationalen Datenbank können unterschiedliche Tabellen mit bestimmten Datenfeldern definiert und

## 10 Einleitung

miteinander verbunden werden. Es steht ein CSV- Importfilter zur Verfügung, um die Anwendungsdaten einzulesen. Die Datenfelder einer Datentabelle können in Filtern und Zuordnungstabellen sowie zur Beschriftung von Knoten verwendet werden.

- **Flussrichtung**

Bestimmen Sie die Flussrichtung des Diagramms: von oben nach unten oder von links nach rechts.

- **Filter**

Mit Hilfe von Filtern können Sie Knoten oder Verbindungen auswählen, die bestimmte Kriterien erfüllen, z.B. um Knoten im Diagramm hervorzuheben.

- **Grafik-Export**

Speichern Sie das Diagramm im gewünschten Grafikformat: PNG, BMP, EMF, GIF, TIF, JPG.

- **In-Flow Gruppierung**

Mit Hilfe der In-Flow Gruppierung werden Knoten in chronologischer oder in einer anders definierten Reihenfolge angeordnet. Sie kann sowohl für die x- als auch die y-Richtung angewendet werden.

- **Intuitive Interaktionen**

Überarbeiten Sie die Darstellung am Bildschirm und verändern die Basisdaten der Benutzeraktionen. So können z. B. Knoten und Verbindungen mit Drag & Drop verschoben oder kopiert werden.

- **Sprachunterstützung**

Das Produkt und die Dokumentation sind in Englisch und Deutsch verfügbar. darüber hinaus kann jedes Textelement im Diagramm zur Laufzeit durch einen Begriff Ihrer Wahl in beliebiger Sprache ersetzt werden. Unicodezeichen werden unterstützt. Die Zeichen aller Sprachen können gleichzeitig und unabhängig vom eingesetzten Betriebssystem verwendet werden.

- **Legende**

Die Legende kann im Außenbereich des Diagramms positioniert werden und ist im Ausdruck und in den exportierten Grafiken sichtbar. Sie kann in einem definierbaren Matrixlayout übersichtlich gestaltet werden.

- **Verbindungen**

Beschriften Sie Verbindungen, wählen Sie Port- Symbole und legen Sie das passende Aussehen für Ihre Verbindungen fest. darüber hinaus können die

vier verschiedenen Verbindungstypen (Start-Start, Start-Ende, Ende-Start, Ende-Ende) grafisch dargestellt werden.

- **Navigationfenster**

Aufgrund der integrierten Komplettansicht ist die Navigation in der Grafik sehr einfach.

- **Knotenaussehen**

Knoten können in verschiedenen Formen, Farben, Farbverläufen und eigenen Bitmaps dargestellt werden. Durch individuelle Tooltiptexte lassen sie sich hervorheben und ergänzen. Das Aussehen von Knoten kann durch den Einsatz von Filtern und Zuordnungstabellen dynamisch an Ihre Daten gebunden werden. Die Knotenbeschriftung kann aus verschiedenen Datenfeldern bestehen, die sich innerhalb oder außerhalb der Knotengrenzen platzieren lassen.

- **Knotenposition**

Die Knotenposition kann auch manuell eingestellt werden, falls gewünscht. Ebenso lassen sich automatisch eingestellte Knotenpositionen aus Datenfeldern auslesen.

- **Drucken**

Wählen Sie das Seitenlayout und lassen es sich in der integrierten Seitenvorschau anzeigen. Wählen Sie bestimmte Bereiche des Diagramms, die auf jeder Seite wiederholt werden sollen und geben Sie die Anzahl der Seiten vor, auf die das Diagramm bei der Ausgabe verteilt werden soll.

- **Eigenschaftenseiten**

Für jedes wichtige Objekt gibt es eine Eigenschaftenseite, die den Programmieraufwand erheblich reduziert. Auf den Eigenschaftenseiten lässt sich beinahe jede Einzelheit der Komponente intuitiv anpassen. Die leistungsfähige API bietet zur Laufzeit weitere Möglichkeiten. Mit Hilfe von Ereignissen kann die Anwendung auf Benutzeraktionen in einer bestimmten Art und Weise reagieren (z.B. um eingegebene Daten zu überprüfen).

- **Zeitrechnung**

Mit der integrierten Zeitrechnung können frühester Start, frühestes Ende, spätester Start, freier Puffer und Gesamtpuffer berechnet werden. Die Grundlage der Berechnung bilden die Dauer der Vorgänge, ihre logischen Abhängigkeiten und Projektstart oder Projektende.

**Hinweis:** Alle Codebeispiele in dieser Dokumentation wurden in VB.NET und C# geschrieben.

---

## 1.2 Installation

Für die Entwicklung von Anwendungen auf der Basis von .NET benötigen Sie eine Entwicklungsumgebung wie zum Beispiel Microsoft Visual Studio 2010 oder neuer. Die Entwicklungsumgebung muss mindestens .NET Framework 2.0 unterstützen und mit Mixed-Mode-.NET-Komponenten kompatibel sein. Als Betriebssystem kann nur Windows ab XP Service Pack 3 aufwärts verwendet werden und zwar in 32bit- oder 64bit-(x64)-Editionen.

Um das Steuerelement VARCHART XNet .NET auf Ihrem Rechner zu installieren, führen Sie bitte das Installationsprogramm aus.

Standardmäßig wird das Steuerelement mit allen zugehörigen Dateien unterhalb des Ordners

**c:\Programme\NETRONIC** (32bit-Windows) bzw.

**c:\Programme (x86)\NETRONIC** (64bit-Windows)

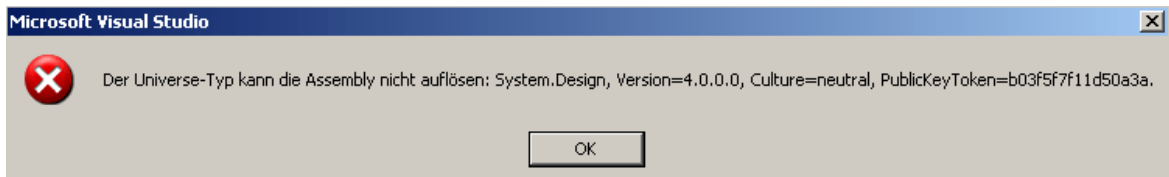
abgelegt.

Nach der Installation sollten Sie das Steuerelement in die Toolbox Ihrer Entwicklungsumgebung aufnehmen.

Wir stellen Ihnen die nötigen Schritte am Beispiel von Microsoft Visual Studio vor; andere Entwicklungsumgebungen arbeiten ähnlich:

1. Legen Sie in Visual Studio ein neues Projekt vom Typ **Windows-Anwendung** an. Es spielt dabei keine Rolle, welche Sprache Sie wählen. Wichtig ist, dass die Toolbox sichtbar ist. Sollte dies nicht der Fall sein, so können Sie sie unter **Ansicht** den Menüpunkt **Toolbox** einschalten.
2. Klicken Sie mit der rechten Maustaste auf die Toolbox und wählen im Kontextmenü **Elemente auswählen**.
3. Mit der Schaltfläche **Durchsuchen** des Registers **.NET Framework-Komponenten** können Sie die Assembly **NETRONIC.Xnet.dll** aus dem entsprechenden Installationsordner auswählen. Nach dem Bestätigen mit **OK** erscheint das Symbol für VARCHART Xnet .NET  als letzter Eintrag in der Toolbox.
4. Wichtig für die Nutzer von **Visual Studio 2010**: **Bevor** Sie das Steuerelement auf die Form ziehen, müssen Sie unter **Projekteigenschaften/Anwendung** (C#) bzw. **Erweiterte Kompilereinstellungen** (VB) das Zielframework von **.NET Framework 4 Client Profile** auf **.NET Framework 4** ändern, da ersteres nicht die von den Eigenschaftenseiten zur Designzeit benötigte System.Design.dll enthält. Falls Sie das Framework nicht umstellen,

erscheint folgende Fehlermeldung, wenn Sie das Steuerelement auf die Form ziehen möchten:



Sie können VARCHART XNet auch bedienerlos installieren. Dazu geben Sie ein:

```
start/wait (NameDerSetupDatei).exe /L1031 /s /V"/qn ADDLOCAL=ALL"
```

Damit läuft die Installation ohne jegliche Benutzereingabe und ohne Fortschrittmeldungen auf dem Bildschirm ab. Bitte beachten Sie Folgendes:

1. Der aufrufende Prozess, z.B. eine DOS-Box, muss mit Administratorrechten gestartet werden; andernfalls erscheint eine UAC-Abfrage, die eine Benutzereingabe notwendig macht.
2. Sprachparameter: /L1031: Installation in deutsch; /L1033: Installation in englisch
3. Fortschrittmeldungen: /qb: Es erscheinen Fortschrittmeldungen; /qn: Es erscheinen keine Fortschrittmeldungen, d.h. Sie sehen am Bildschirm nichts.
4. Start/wait sollte man benutzen, wenn die Installation über eine Batch-Datei abläuft; andernfalls liefere die Batch-Datei während der Installation parallel weiter.

---

## 1.3 Lizenzierung

### 1.3.1 Entwickler-Lizenzen

Zum Lizenzieren von VARCHART XNet .NET ziehen Sie nun das Steuerelement  von der Toolbox auf die Form.

Öffnen Sie dann die **Eigenschaftenseiten**, indem Sie mit der rechten Maustaste auf das Steuerelement klicken.

Auf der Registerkarte **Allgemeines** gelangen Sie über die Schaltfläche **Lizenzierung** in den Lizenzierungsdialog.

Die Schaltfläche **Lizenzinformation von NETRONIC anfordern** führt zu einem weiteren Dialogfeld, das den Eintrag der Lizenzierungsinformationen ermöglicht.

Für die Registrierung benötigen wir drei Informationen:

- die Lizenznummer
- Ihren Namen
- den Firmennamen

Tragen Sie die erforderlichen Daten in die vorgesehenen Felder ein. Die Lizenznummer "NXnnnn" finden Sie auf dem Lieferschein zu Ihrer Bestellung.

Durch Klick auf die Schaltfläche **E-Mail an NETRONIC senden...** wird eine E-Mail erstellt, die Sie uns nur noch schicken müssen. Sie können aber auch selber eine E-Mail verfassen, in der Sie uns die benötigten Daten mitteilen. Richten Sie bitte alle Lizenzierungsanfragen an [license@netronic.com](mailto:license@netronic.com).

Danach erhalten Sie von uns umgehend eine passende Lizenzdatei, die Sie bitte im Installationsordner (Ordner mit der Datei NETRONIC.XNet.dll) ablegen. Der Lizenzierungsvorgang ist nun abgeschlossen.

---

## 1.4 Auslieferung

Wenn Sie eine mit VARCHART XNet .NET entwickelte Anwendung an Ihre Kunden weitergeben möchten, müssen Sie die im folgenden aufgeführten Dateien mit ausliefern. Alle anderen Dateien, die zum Produkt VARCHART XNet .NET gehören, werden nur für die Entwicklungsphase benötigt und dürfen **nicht** an Ihre Kunden weitergegeben werden.

### > Framework .NET 2.0/3.0/3.5

- **In der entsprechenden Prozessorvariante für x86 oder x64**

*NETRONIC.XNet.dll*

*NETRONIC.XNetd.dll* (wenn Sie die deutsche Version nutzen möchten)

*NETRONIC.XNetc.dll* (wenn Sie die chinesische Version nutzen möchten)

*mfc80u.dll*

*mfc80u.dll*

*msvcp80.dll*

*msvcr80.dll*

Die Installation der Bibliotheken *mfc80u.dll*, *msvcp80.dll*, *mfc80u.dll* und *msvcr80.dll* muss über die im Unterverzeichnis **redist** mitgelieferten Setup-Dateien *vc\_redist\_vs2005sp1\_x86.exe* bzw. *vc\_redist\_vs2005sp1\_x64.exe* erfolgen.

Nähere Informationen dazu bietet folgende Seite:

[msdn2.microsoft.com/en-us/library/ms235285\(VS.80\).aspx](https://msdn2.microsoft.com/en-us/library/ms235285(VS.80).aspx).

### > Framework .NET 4.0/4.5

- **In der entsprechenden Prozessorvariante für x86 oder x64**

*NETRONIC.XNet.dll*

*NETRONIC.XNetd.dll* (wenn Sie die deutsche Version nutzen möchten)

*NETRONIC.XNetc.dll* (wenn Sie die vereinfachte chinesische Version nutzen möchten)

*mfc100u.dll*

*mfc100u.dll*

*msvcp100.dll*

*msvcr100.dll*



## 16 Einleitung

Die Installation der Bibliotheken *mfc100u.dll*, *msvcp100.dll*, *mfc100u.dll* und *msvcr100.dll* kann entweder direkt durch Kopieren in das Windows-Systemverzeichnis oder über die im Unterverzeichnis **redist** mitgelieferten Setup-Dateien *vcredist\_vs2010\_x86.exe* bzw. *vcredist\_vs2010\_x64.exe* erfolgen.

VARCHART XNet.NET wird für folgende Plattformen angeboten:

- Windows 8
- Windows 7
- Windows Server 2008
- Windows Vista
- Windows Server 2003
- Windows XP ab SP3

unter Verwendung des .NET Frameworks ab Version 2.0 (nähere Informationen dazu finden Sie unter

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

### **Tipp**

So überprüfen Sie, welches .NET Framework bereits installiert ist:

In der **Systemsteuerung** wählen Sie **Software** und suchen in der Programmliste nach dem **Microsoft .NET Framework** Eintrag.

---

## 1.5 Verwendung der deutschen Version

Die VARCHART XNet .NET Edition ist in Deutsch und Englisch verfügbar. Bei der deutschen Version wird während der Installation zusätzlich zur Control-Assembly NETRONIC.XNet.dll die Ressource-Assembly NETRONIC.XNetd.dll ins Installationsverzeichnis kopiert.

### Verwendung zur Designzeit

Wenn in den **Regionalen Einstellungen** (Systemsteuerung, Regions- und Sprachoptionen, Regionale Einstellungen) **Deutsch** eingestellt ist, wird die Ressource-Assembly aus dem Installationsverzeichnis geladen und die deutschen Dialoge und Eigenschaftenseiten stehen zur Designzeit zur Verfügung.

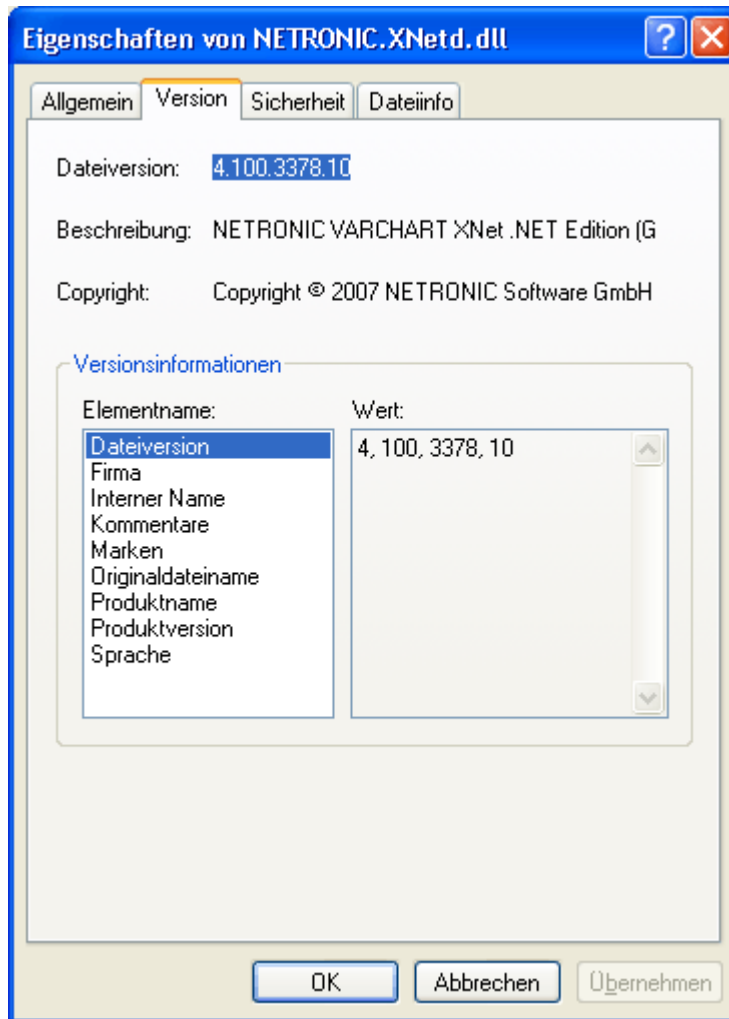
### Verwendung zur Laufzeit

Um sicher zu stellen, dass auch zur Laufzeit auf die Ressource-Assembly zugegriffen wird und deutsche Dialoge zur Verfügung stehen, muss die Ressource-Assembly ins Applikationsverzeichnis kopiert werden. Dazu muss im Projekt ein Verweis auf die Assembly eingetragen werden ("Verweis hinzufügen").

**Tipp:** Da die Entwicklungsumgebung standardmäßig den "Copy-Local"-Parameter auf **False** setzt, muss dieser noch manuell auf **True** gesetzt werden. Wenn das Projekt dann neu erstellt wird, wird auch die Ressource-Assembly automatisch in das passende Applikationsverzeichnis kopiert und von dort geladen.

Bei eventuell auftretenden Problemen sollte überprüft werden, ob die Dateiversionsnummern der beiden Assemblies übereinstimmen (im Windows-Explorer das Kontextmenü der Datei aufrufen, Menüpunkt **Eigenschaften**, Registerkarte **Version**).

## 18 Einleitung



---

## 1.6 Unterstützung und Beratung

Sie wissen nicht, ob VARCHART XNet die speziellen Anforderungen Ihres Netzdiagramms erfüllt?

Sie wollen sich eine Vorstellung davon machen, wie viel Aufwand es ist, die eine oder andere Anforderung Ihres Netz-Diagramms zu realisieren?

Sie machen gerade den Anfangstest mit VARCHART XNet und möchten eine ganz spezifische Anforderung Ihres Netzplans programmieren?

Bei der Beantwortung dieser und anderer Fragen helfen wir Ihnen gerne weiter:

NETRONIC Software GmbH

Pascalstraße 15

52076 Aachen

Deutschland

Tel.: +49-2408-141-0

Fax: +49-2408-141-33

E-Mail: [support@netronic.de](mailto:support@netronic.de)

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

... übrigens, Sie können mit uns einen Wartungs- und Supportservice vereinbaren. Dann kommen Sie in den Genuss sofortiger Hilfe auch über den kostenfreien Support während der 30-tägigen Testphase hinaus. Der Wartungs- und Supportservice umfasst folgende Leistungen:

- Support-Hotline
- Qualifizierte und ausführliche Beratung in allen Anwendungsfragen
- Beseitigung von möglichen Fehlern in der gelieferten Software
- Upgrade auf ein neues Release von VARCHART XNet für Entwicklungs- und Laufzeitversionen

Auf Wunsch können wir Ihnen auch Schulungskurse und Workshops (in unserem Hause oder bei Ihnen vor Ort) anbieten.



---

---

## 2 Tutorium

---

### 2.1 Überblick

In diesem Kapitel machen wir Sie mit den Grundlagen und Grundbegriffen von VARCHART XNet vertraut, die notwendig sind, um die Netzplan-Komponente in eigene Anwendungen integrieren zu können.

Wir werden Schritt für Schritt die bedeutsamen Aspekte von VARCHART XNet für die Anwendungsentwicklung erläutern und auf die vielfältigen Gestaltungsmöglichkeiten näher eingehen. Aus diesem Grund empfiehlt es sich, dieses Kapitel sequenziell durchzuarbeiten. Die anderen Kapitel des Handbuchs sind eher zum Nachschlagen gedacht:

- **Eigenschaftenseiten und Dialogfelder**

Hier finden Sie umfassende Informationen zu den Eigenschaftenseiten und Dialogen, die Ihnen zur Entwurfszeit erlauben, VARCHART XNet nach Wunsch zu konfigurieren, ohne dass Sie dafür Programm-Code schreiben müssen.

- **Elemente der Benutzerschnittstelle**

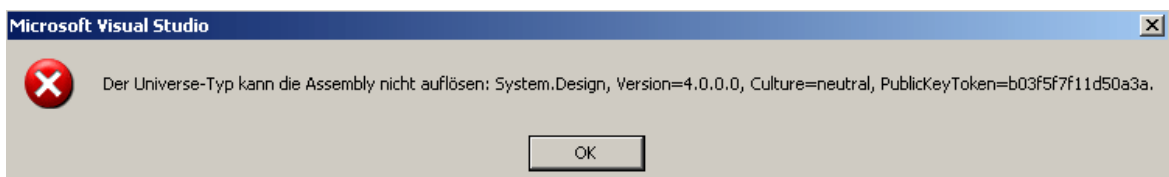
In diesem Kapitel werden die bereits im Diagramm vorhandenen Interaktionen beschrieben. Details der Benutzerschnittstelle können individuell angepasst bzw. abgewandelt werden.


- **API-Referenz**

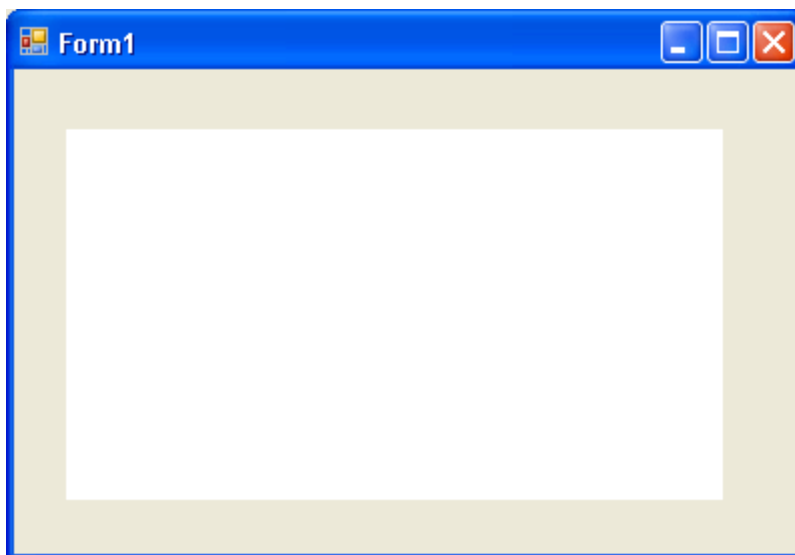
In diesem Kapitel finden Sie ausführliche Informationen zu allen Objekten, Eigenschaften, Methoden und Ereignissen, die Ihnen VARCHART XNet bietet.

## 2.2 Das Steuerelement in einem Formular platzieren

Wichtig für die Nutzer von **Visual Studio 2010**: **Bevor** Sie das Steuerelement auf die Form ziehen, müssen Sie unter **Projekteigenschaften/Anwendung (C#)** bzw. **Erweiterte Kompilereinstellungen (VB)** das Zielframework von **.NET Framework 4 Client Profile** auf **.NET Framework 4** ändern, da ersteres nicht die von den Eigenschaftenseiten zur Designzeit benötigte System.Design.dll enthält. Falls Sie das Framework nicht umstellen, erscheint folgende Fehlermeldung, wenn Sie das Steuerelement auf die Form ziehen möchten:



Um **VARCHART XNet** in ein Formular einzufügen, klicken Sie auf das entsprechende Symbol in der Toolbox  und ziehen im Formular an der gewünschten Stelle mit der Maus einen Rahmen auf. Das **VARCHART-XNet**-Steuerelement erscheint (zunächst als weiße Fläche) in der von Ihnen bestimmten Größe. Diese können Sie selbstverständlich mit Hilfe der Maus noch modifizieren.



## 2.3 VARCHART XNet automatisch skalieren

Wenn der rechte und der untere Rand des VARCHART-XNet-Steuerelements zur Laufzeit immer auf die Größe des gesamten Fensters angepasst werden sollen, können Sie das z. B. mit Hilfe des folgenden Codes erreichen:

### Code-Beispiel VB.NET

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

### Code-Beispiel C#

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

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

### Hinweis:

Eine Namespace-Anweisung am Anfang des Programms erspart die vollständige und mitunter umständliche Referenzangabe bei der Benutzung von Datentypen und Enum-Elementen.

VB: Imports NETRONIC.XNet

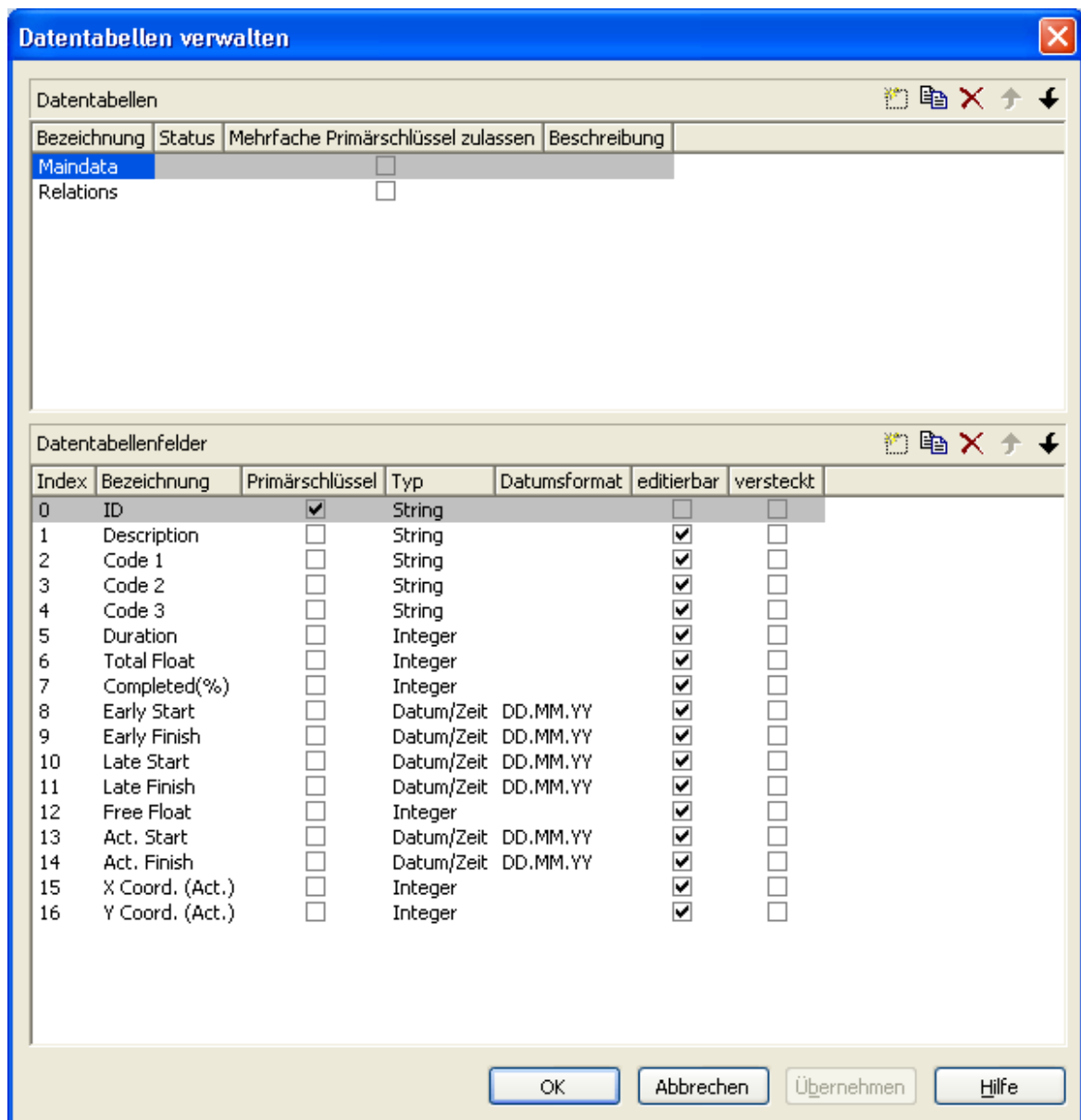
C#: using NETRONIC.XNet;

Zum Beispiel ist dann statt **NETRONIC.XNet.VcNodeCollection** nur noch **VcNodeCollection** erforderlich.



## 2.4 Schnittstelle einrichten


Richten Sie nun die Schnittstelle ein, indem Sie die Datenfelder der Tabellen anpassen. Öffnen Sie dazu den Dialog **Datentabellen verwalten**:



Wählen Sie zuerst die Tabelle "Maindata" (Knotendaten) und vereinbaren Sie deren Datenfelder.

Um der Beispiel-Schnittstelle gerecht zu werden, überschreiben Sie "ID" durch "Nummer" und wählen Sie den Datentyp "Integer". Deaktivieren Sie das Kontrollkästchen **editierbar**, damit die Nummer im Standarddialog **Vorgänge bearbeiten** nicht überschrieben werden kann.

Ein Name lässt sich editieren, indem Sie ihn doppelt anklicken bzw. ihn markieren und dann einmal anklicken. Der Datentyp kann aus einer Kombobox ausgewählt werden.

Ein neues Datenfeld können Sie anlegen, indem Sie im Bereich **Datentabellenfelder** die Schaltfläche  anklicken und das neue Feld in der letzten Zeile editieren.

### Felder der Maindata-Tabelle:

Feld	Name	Typ
0	Nummer	Integer
1	Gliederungsnummer	String
2	Ebene	Integer
3	Vaterknoten	String
4	Name	String
5	Gruppencode	String
6	Code	Integer
7	Gruppenname	String
8	Dauer	Integer
9	Pufferzeit	Integer
10	fertig (%)	Integer
11	Frühester Start	Datum/Zeit
12	Frühestes Ende	Datum/Zeit
13	Spätester Start	Datum/Zeit
14	Spätestes Ende	Datum/Zeit
15	Freier Puffer	Integer
16	Berechneter Start	Datum/Zeit
17	Berechnetes Ende	Datum/Zeit
18	X-Koordinate (Knoten)	Integer
19	Y-Koordinate (Knoten)	Integer
20	Hilfsknoten	String

Aktivieren Sie für die Felder "Berechneter Start" und "Berechnetes Ende" das Kontrollkästchen **versteckt**, damit der Anwender deren Werte im Standarddialog **Vorgänge bearbeiten** nicht sieht.

Bei den **Datum/Zeit**-Feldern können Sie angeben, in welchem Format das Datum übergeben wird. Wählen Sie "DD.MM.YY".

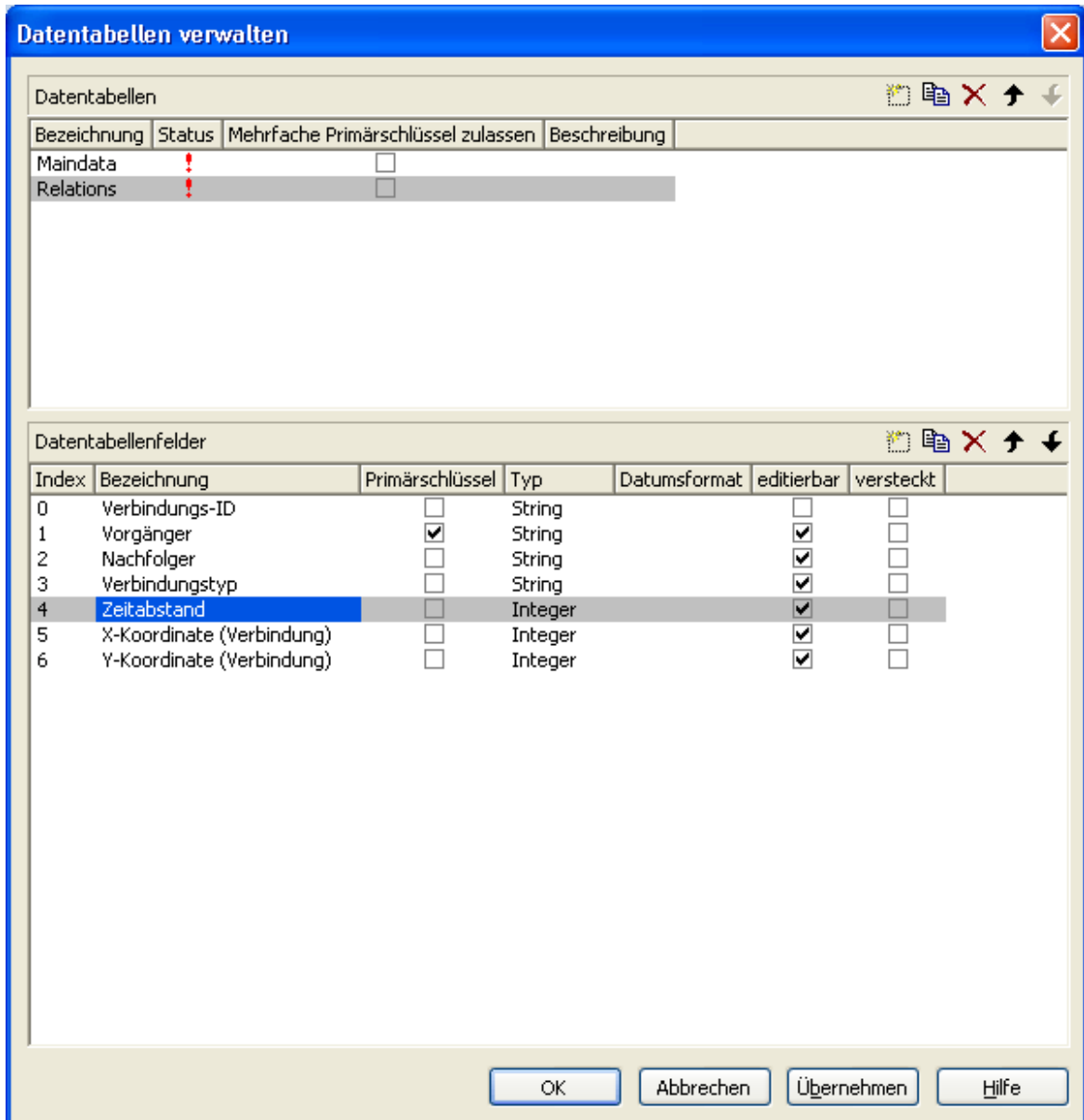
Legen Sie jetzt noch einen Primärschlüssel für die Knoten fest. Wählen Sie unter **Primärschlüssel** das Feld "Nummer". Aus diesem Feld wird dann die Identifikation der Knoten genommen.

Wechseln Sie nun zur Tabelle "Relations", die für die logischen Verbindungen verwendet wird.

**Felder der Relations-Tabelle:**

Index	Name	Typ
0	Verbindungs-ID	String
1	Vorgänger	String
2	Nachfolger	String
3	Verbindungstyp	String
4	Zeitabstand	Integer
5	X-Koordinate (Verbindung)	Integer
6	Y-Koordinate (Verbindung)	Integer

Legen Sie jetzt noch einen Primärschlüssel für die Verbindungen fest. Wählen Sie unter **Primärschlüssel** das Feld "Verbindungs-ID". Aus diesem Feld wird dann die Identifikation der Verbindungen genommen.



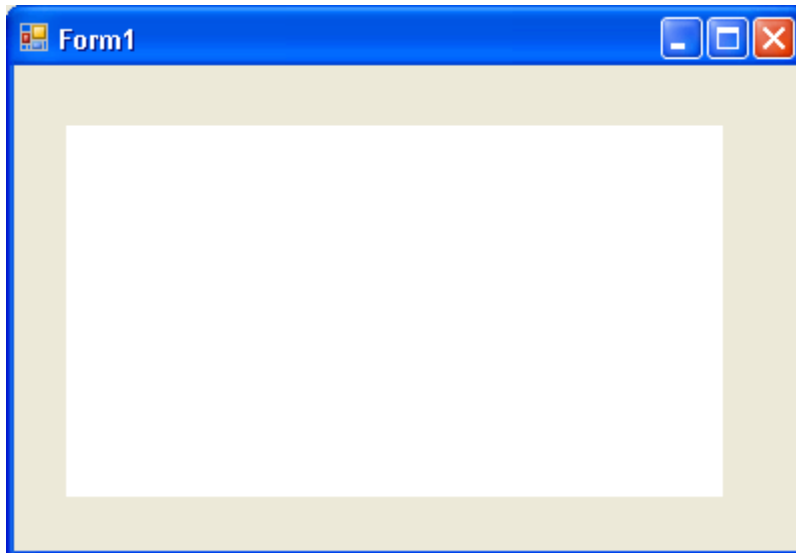
Deaktivieren Sie für das Feld "Verbindungs-ID" das Kontrollkästchen **editierbar**.

**Hinweis:** Wenn Sie einen Namen eingeben, der bereits für ein anderes Feld vergeben ist, erscheint eine Fehlermeldung und Sie werden aufgefordert, einen anderen Namen zu vergeben.

Durch Anklicken der Schaltfläche **Übernehmen** werden die Änderungen übernommen. Alternativ werden die Einstellungen auch bei **OK** und beim Wechsel der Eigenschaftenseite übernommen und stehen damit direkt in den anderen Eigenschaftenseiten zur Verfügung.

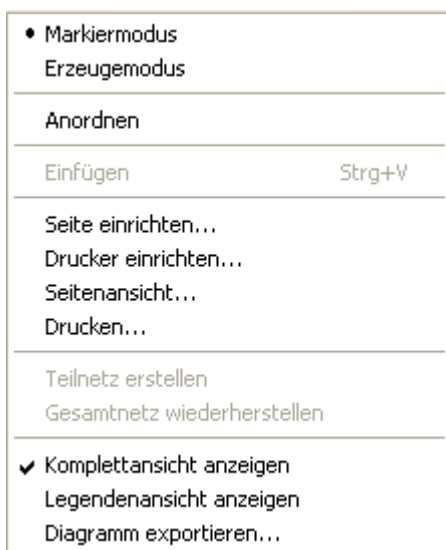
## 2.5 Der erste Lauf

Über **Ausführen – Starten**, die Funktionstaste **F5** oder die entsprechende Schaltfläche (▶) starten Sie nun das Programm. Das angelegte Formular erscheint mit dem leeren Diagramm.



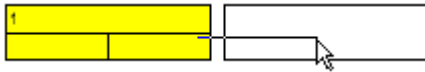
### > Knoten und Verbindungen anlegen

Zur Laufzeit stehen Ihnen zwei grundlegende Modi zur Verfügung: der Markiermodus und der Erzeugemodus. Knoten und Verbindungen können Sie nur im Erzeugemodus anlegen. Um in den Erzeugemodus zu wechseln, klicken Sie mit der rechten Maustaste in den freien Diagrammbereich und wählen im Kontextmenü **Erzeugemodus**.



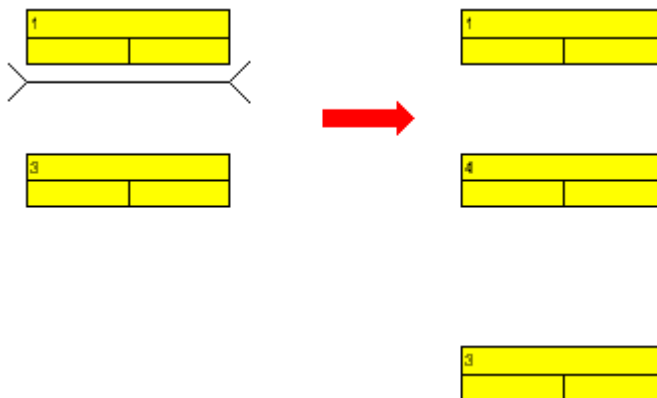
Der Mauszeiger wird nun im leeren Diagrammbereich zu einem Knotenphantom in Form eines Rechtecks. Nun können Sie Knoten erzeugen, indem Sie mit der linken Maustaste in den Diagrammbereich klicken.

Verbindungen erzeugen Sie, indem Sie die Maus mit gedrückter linker Maustaste von dem einem Knoten zu einem anderen ziehen. Dabei wird der Mauszeiger zu einem Pfeil-Symbol mit einem Knoten-Phantom.



Die Verbindung wird erzeugt, sobald Sie die Maustaste loslassen. Lassen Sie die Maustaste im leeren Diagrammbereich los, wird dort ein neuer Knoten zusammen mit einer Verbindung vom Ausgangsknoten erzeugt. Auf diese Weise können Sie Knoten und Verbindungen gemeinsam erzeugen.

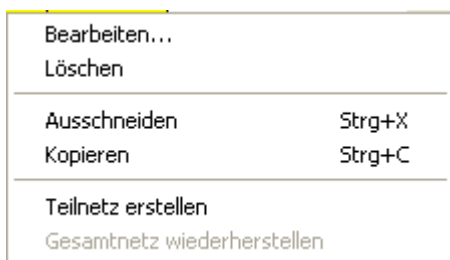
Wenn Sie die Maus zwischen zwei dicht übereinander oder nebeneinander stehende Vorgänge führen, um dazwischen einen neuen Vorgang anzulegen, verändert das Phantom seine Form und wird zu einer waagerechten Linie mit zwei nach innen gerichteten Pfeilspitzen ("Knochen"). Klicken Sie nun mit der linken Maustaste, so wird der neue Vorgang zwischen den beiden vorhandenen Vorgängen eingefügt.



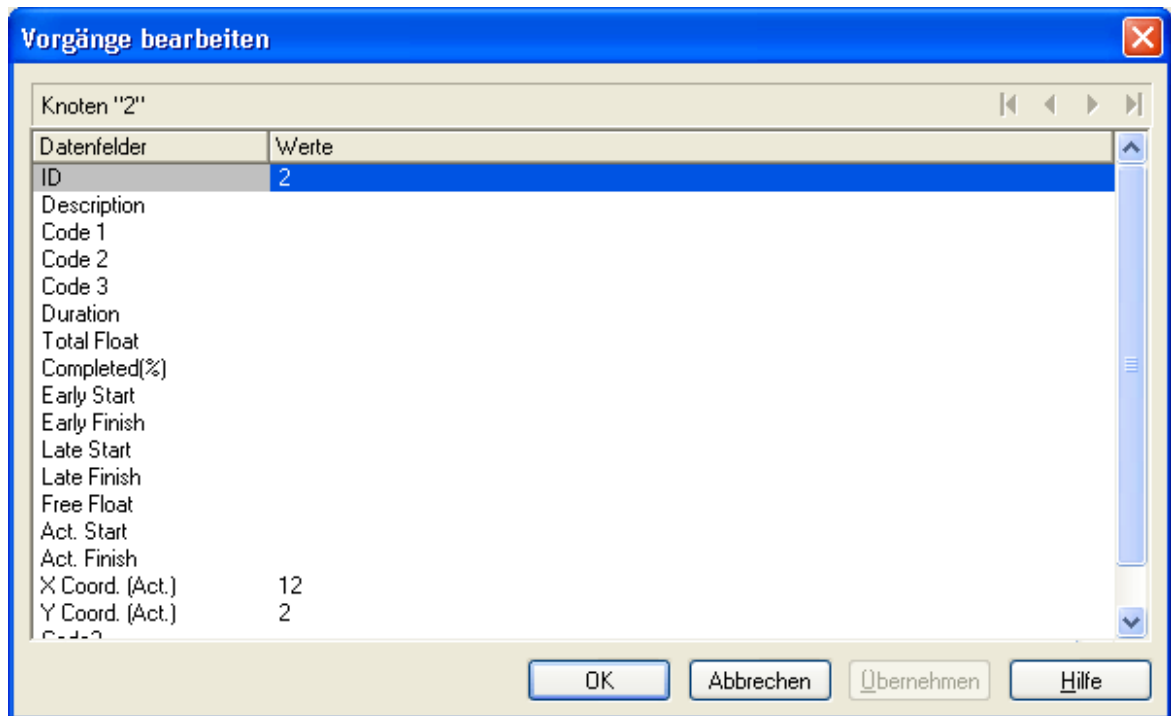
### > Knoten bearbeiten

Um einen Knoten zu bearbeiten, wechseln Sie zum Markiermodus und doppelklicken Sie auf den Knoten. Dann erscheint das Dialogfeld **Vorgänge bearbeiten**.

Oder klicken Sie mit der rechten Maustaste auf den Knoten. Dann erscheint ein Kontextmenü, über das Sie den Knoten bearbeiten, kopieren, ausschneiden oder löschen können.

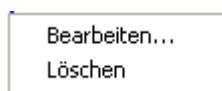


Wählen Sie im Kontextmenü die Option **Bearbeiten**, um das Dialogfeld **Vorgänge bearbeiten** für einen Knoten aufzurufen. Sie finden hier die Datenfelder wieder, die Sie im Dialog **Datentabellen verwalten** vereinbart haben. (Die Datenfelder, die dort als **versteckt** definiert worden sind, erscheinen nicht in diesem Dialogfeld. Die als **nicht editierbar** definierten Datenfelder können in diesem Dialogfeld nicht verändert werden.)

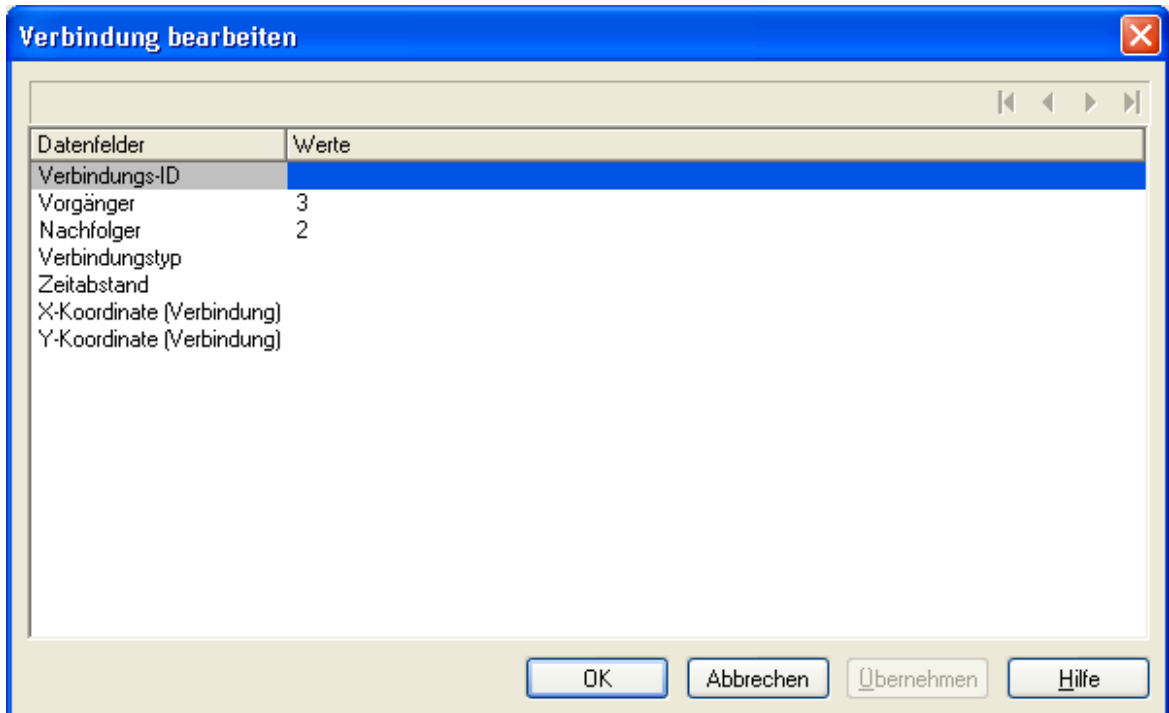


### > Verbindung bearbeiten

Eine Verbindung können Sie bearbeiten, indem Sie entweder über den Befehl **Bearbeiten** des entsprechenden Kontextmenüs oder mit einem Doppelklick auf die Verbindung das Dialogfeld **Verbindung bearbeiten** öffnen.



*Kontextmenü für Verbindungen*



In diesem Dialogfeld können Sie alle Daten der Verbindung bearbeiten.

### > **Knoten und Verbindungen interaktiv verschieben**

Knoten und Verbindungen können Sie mit der Maus beliebig verschieben. Positionieren Sie dazu im Selektiermodus den Mauszeiger auf einem Knoten oder einer Verbindung. Der Zeiger wird dann zu einem kleinen Quadrat mit vier Pfeilen. Sie können den gewünschten Knoten bzw. die gewünschte Verbindung beliebig mit gedrückter linker Maustaste verschieben. Beim Verschieben eines Knotens werden die zugehörigen Verbindungen automatisch angepasst.

### > **Zurück zum Design-Modus**

Beenden Sie nun den ersten Lauf, indem Sie das Formular schließen.



## 2.6 Daten aus einer Datei einlesen

Um das VARCHART XNet für die nächsten Schritte mit Daten zu füllen, können Sie die mitgelieferte Datei *tutorial.net* beim Start automatisch laden. (*tutorial.net* ist eine CSV-Datei entsprechend Ihrer eingestellten Schnittstelle. Zu Veränderungen der Schnittstelle siehe "Tutorium: Schnittstelle einrichten".)

Dazu reagieren Sie auf das Ereignis **Form\_Load**:

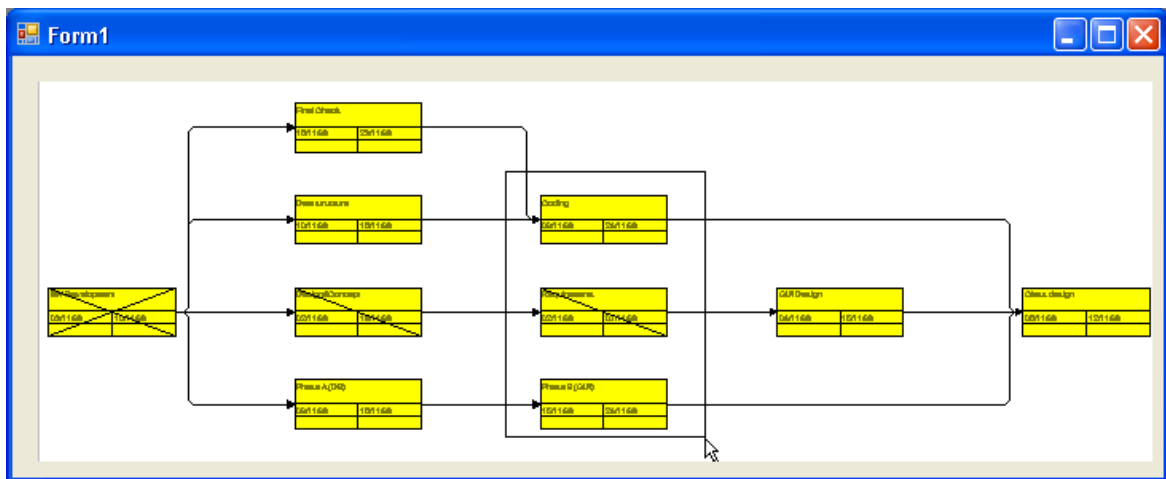
### Code-Beispiel

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

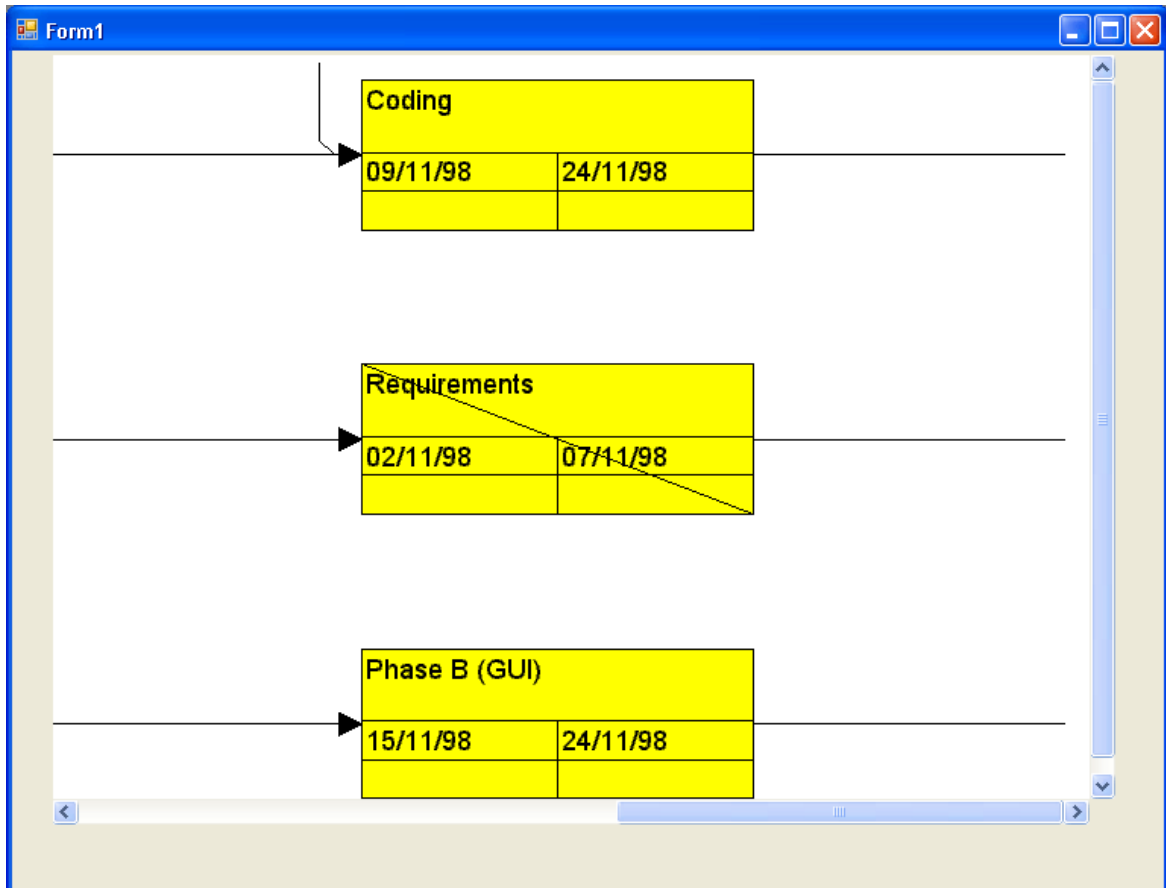
Die Pfadangabe ist abhängig von Ihrer Installation. Speichern Sie nun das Projekt. Wenn Sie nun das Programm starten, werden die Knoten und Verbindungen des Projektes angezeigt.

VARCHART XNet stellt ein Netz-Diagramm komplett dar.

Sie können einen Ausschnitt Ihres Diagramms bildschirmfüllend darstellen lassen, indem Sie mit gedrückter linker Maustaste ein Gummirechteck um den zu vergrößernden Ausschnitt aufziehen und dann (bei noch gedrückter linker Maustaste) auf die rechte Maustaste klicken.



Nun wird der gewählte Ausschnitt bildschirmfüllend dargestellt. Mit Hilfe der Bildlaufleisten können Sie das Fenster wie eine Lupe über der Darstellung verschieben und so auch die anderen Bereiche der Darstellung in derselben Vergrößerung betrachten.

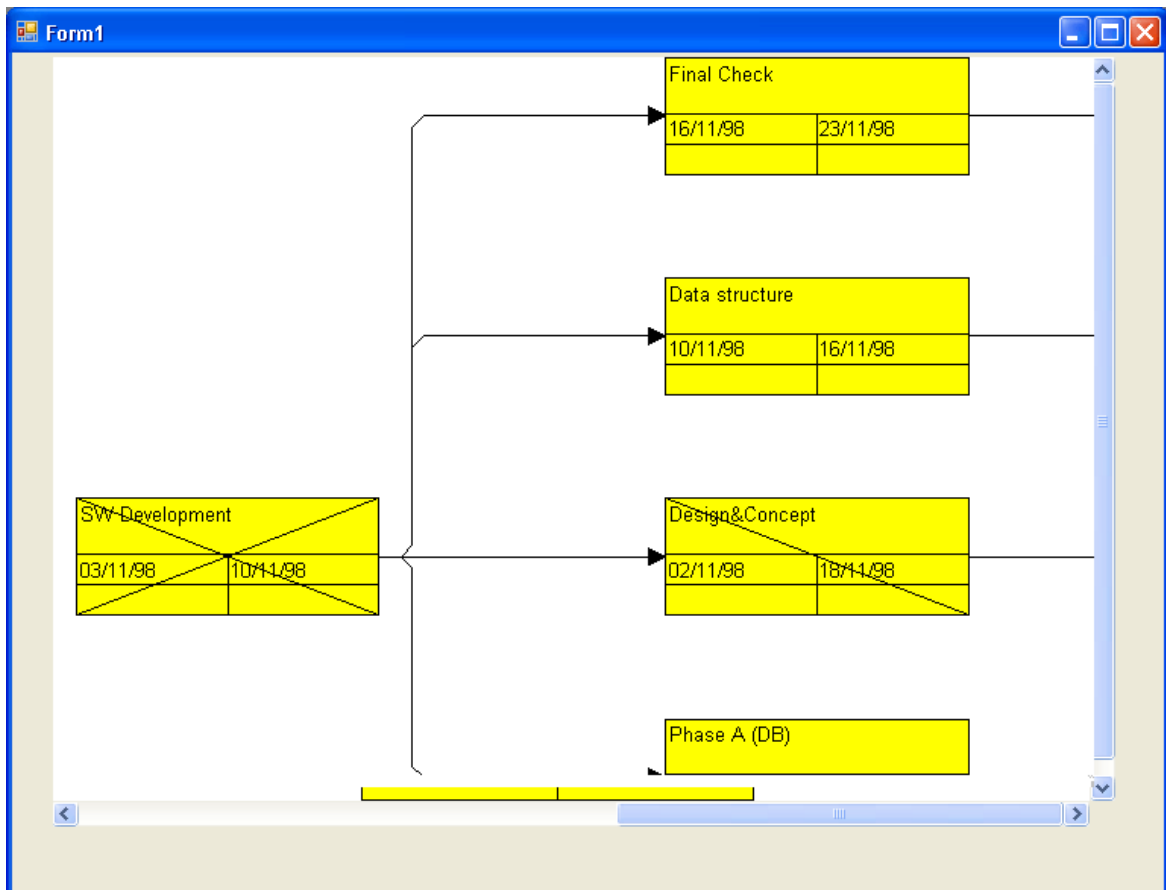


Kehren Sie nun wieder zum Design-Modus zurück. Ergänzen Sie ggf. den folgenden Code, damit Scrollbars in X- und Y-Richtung ausgegeben werden. (Die Ausgabe von Scrollbars hängt von dem gewählten Zoomfaktor ab.)

#### Code-Beispiel

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

## 34 Daten aus einer Datei einlesen



Kehren Sie wieder zum Design-Modus zurück.

Soll VARCHART XNet das gesamte Formular ausfüllen, dann muss folgendes eingestellt werden:

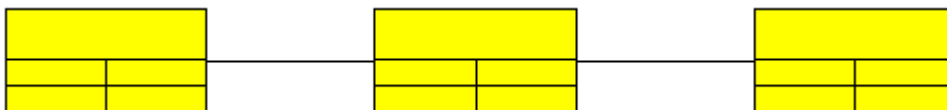
- Stellen Sie sicher, dass die Eigenschaften **Top** und **Left** den Wert 0 haben. Damit wird VARCHART XNet oben links im Formular positioniert.
- Tragen Sie für die VARCHART-XNet-Eigenschaften **Width** und **Height** die Formularwerte von **ScaleWidth** und **ScaleHeight** ein. (Dieser Schritt ist überflüssig, wenn Sie das VARCHART XNet wie oben beschrieben automatisch reskalieren lassen.)

## 2.7 Flußrichtung des Netzdiagramms festlegen

Auf der Eigenschaftenseite **Allgemeines** können Sie die wichtigsten allgemeinen Einstellungen für das VARCHART XNet festlegen.

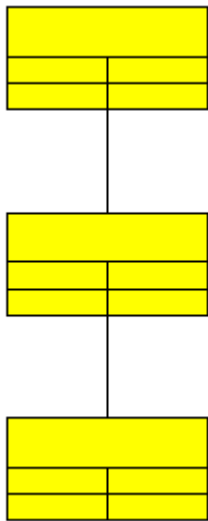


Legen Sie zunächst unter **Flussrichtung** fest, ob die Knoten von links nach rechts oder von oben nach unten angeordnet werden sollen. Probieren Sie beide Einstellungen mit den Daten der Beispieldatei *tutorial.net* aus. Sie können das schon in der Design-Phase tun. Wählen Sie dazu unter **Temporäre Datendatei** die Datei *tutorial.net* aus.



*Flussrichtung von links nach rechts*

## 36 Flußrichtung des Netzdiagramms festlegen



*Flussrichtung von oben nach unten*

Damit Sie Knoten anlegen können, muss die Option **Neue Knoten und Verbindungen zulassen** auf der Eigenschaftenseite **Allgemeines** aktiviert sein.

Um das Programm nun mit einem leeren Diagramm zu starten, setzen Sie die entsprechende Code-Zeile auf Kommentar:

### Code-Beispiel

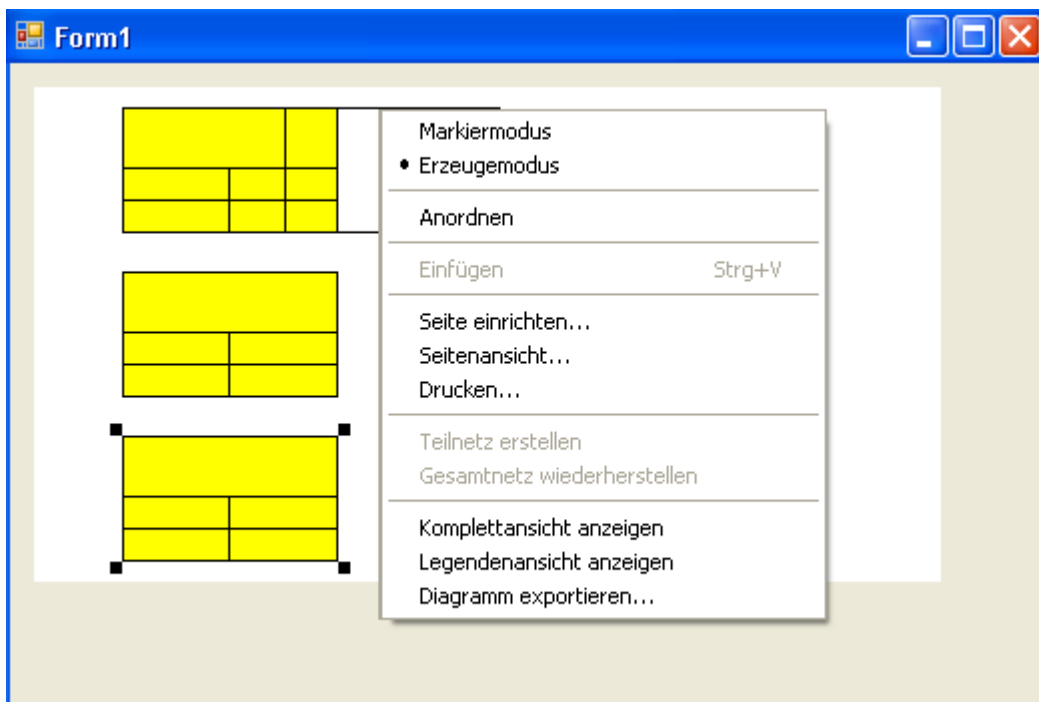
```
Private Sub Form_Load()  
    ' VcNet1.Open "C:\Programme\Varchart\xnet\tutorial.net"  
    VcNet1.Zoomfactor = 100  
End Sub
```

Das Formular erscheint mit dem leeren Netzdiagramm. Öffnen Sie mit der rechten Maustaste das Kontextmenü und wählen Sie den Erzeugemodus. Legen Sie per Mausklick einige Knoten neben- und untereinander an sowie einige Verbindungen. Klicken Sie nun mit der rechten Maustaste in den leeren Diagrammbereich und wählen Sie im Kontextmenü den Befehl **Anordnen**. VARCHART XNet ordnet die Knoten nun automatisch gemäß der gewählten Einstellung für die gewählte Flussrichtung an.

## 2.8 Knoten und Verbindungen erzeugen und bearbeiten

Auf der Eigenschaftenseite **Allgemeines** können Sie festlegen, ob zur Laufzeit neue Knoten und Verbindungen erzeugt werden können (**Erzeugung neuer Knoten und Verbindungen zulassen**), und ob das Dialogfeld **Vorgänge bearbeiten** erscheint, sobald Sie einen neuen Knoten bzw. eine neue Verbindung erzeugt haben (**Erzeugung neuer Knoten mit Dialog** bzw. **Erzeugung neuer Verbindungen mit Dialog**).

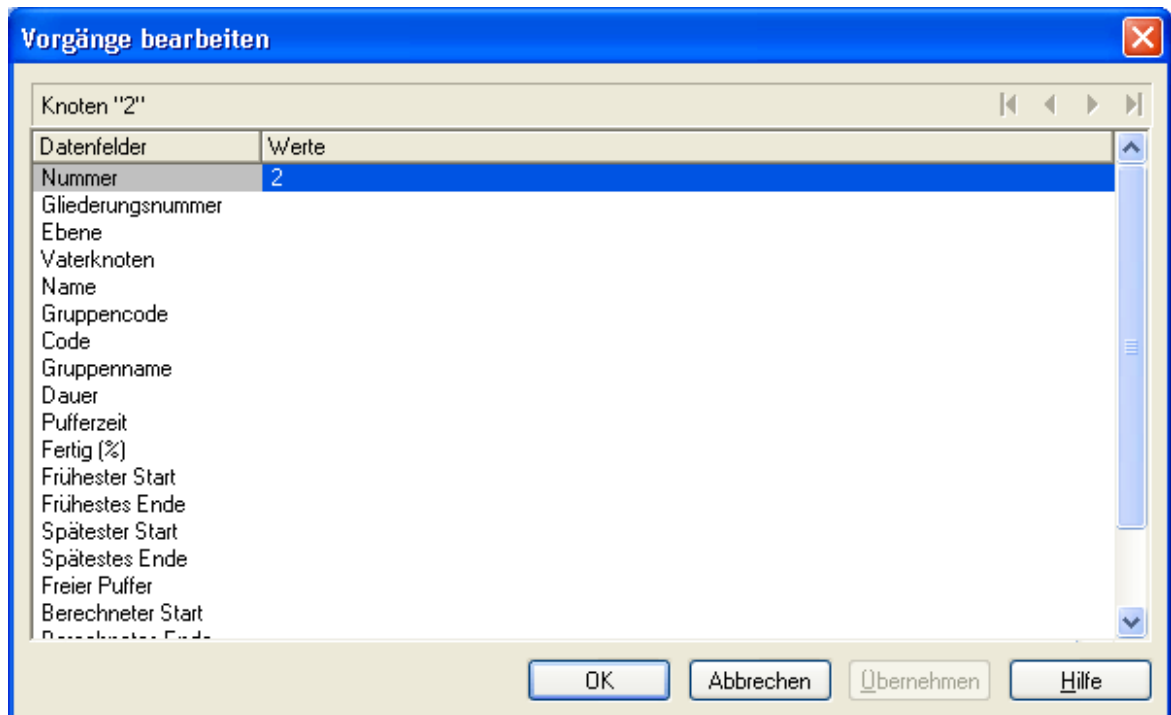
Aktivieren Sie zunächst nur die Option **Erzeugung neuer Knoten und Verbindungen zulassen** und deaktivieren Sie die Optionen **Erzeugung neuer Knoten mit Dialog** und **Erzeugung neuer Verbindungen mit Dialog**. Starten Sie die Applikation mit F5, öffnen Sie durch einen Klick auf die rechte Maustaste das Kontextmenü und wählen Sie hier den **Erzeugemodus**. Klicken Sie nun mit der linken Maustaste in das leere Diagramm. Mit jedem Mausklick wird ein neuer Knoten angelegt.



Wenn die Option **Erzeugung neuer Knoten und Verbindungen zulassen** deaktiviert ist, kann der Anwender zur Laufzeit auch im Erzeugemodus keine neuen Knoten und Verbindungen anlegen. Knoten und Verbindungen können dann nur über die API geladen werden.

Aktivieren Sie nun die Optionen **Erzeugung neuer Knoten mit Dialog** und **Erzeugung neuer Verbindungen mit Dialog**.

Starten Sie die Applikation, wechseln Sie in den Erzeugemodus und klicken Sie mit der rechten Maustaste auf einen Knoten. Das Kontextmenü für Knoten erscheint. Wählen Sie hier die Option **Bearbeiten**. Das Dialogfeld **Vorgänge bearbeiten** öffnet sich. Hier werden Ihnen alle Datenfelder des neuen Vorgangs, die Sie im Dialog **Datentabellen verwalten** vereinbart haben, mit ihren aktuellen Werten angezeigt. Sobald Sie die Daten mit **OK** bestätigen, wird der neue Vorgang an der Mauszeigerposition angelegt.



Erzeugen Sie nun einige Knoten und Verbindungen (vgl. "Tutorium: Der erste Lauf"). Sie können nun mehrere Knoten unmittelbar hintereinander bearbeiten. Markieren Sie diese, indem Sie sie bei gedrückter Strg-Taste mit der linken Maustaste anklicken. Klicken Sie dann mit der rechten Maustaste auf eines der markierten Objekte. Es öffnet sich das Kontextmenü für Knoten. Wählen Sie den Befehl **Bearbeiten**. Das Dialogfeld **Vorgänge bearbeiten** erscheint. Sie können hier die Daten der markierten Vorgänge sukzessive editieren.

In der Kopfzeile dieses Dialogfeldes wird die Identifikationsnummer des Vorgangs angezeigt, den Sie in der **Datenfelder/Werte**-Tabelle bearbeiten können. In Klammern hinter der Identifikationsnummer wird angegeben, um den wievielten der markierten Vorgänge es sich handelt.

Beim Öffnen dieses Dialogfeldes werden die Datenfelder und Werte des ersten markierten Knotens angezeigt. Sie können die Werte aller Datenfelder dieses Vorgangs bearbeiten. Mit den Pfeil-Schaltflächen können Sie zwischen den Knoten navigieren.

Schließen Sie nun das Formular und kehren Sie zum Design-Modus zurück.

---

## 2.9 Knoten und Verbindungen markieren

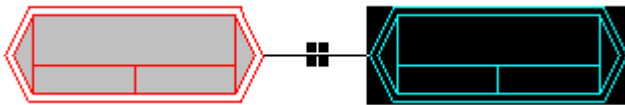
Auf den Eigenschaftenseiten **Knoten** bzw. **Verbindungen** können Sie aus der Kombobox **Markierungstyp** auswählen, in welcher Weise Knoten und Verbindungen markiert werden sollen.

Starten Sie das Programm und legen Sie im Erzeugemodus einige Knoten und Verbindungen an. Klicken Sie nun im Markiermodus einmal auf einen Knoten oder eine Verbindung, um sie zu markieren.

Mehrere Objekte (Knoten oder Verbindungen) können Sie markieren und toggeln, indem Sie sie bei gedrückter Strg-Taste mit der linken Maustaste anklicken. Mehrere Knoten oder Verbindungen können Sie markieren und sammeln, indem Sie diese bei gedrückter Shift-Taste anklicken. Dabei werden die zwischen den markierten Knoten liegenden Knoten und Verbindungen ebenfalls markiert.

Probieren Sie verschiedene Alternativen für den Knotenmarkierungstyp und den Verbindungsmarkierungstyp aus.

In der Abbildung sehen Sie einen durch Invertierung markierten Knoten sowie eine durch Pickmarks markierte Verbindung:



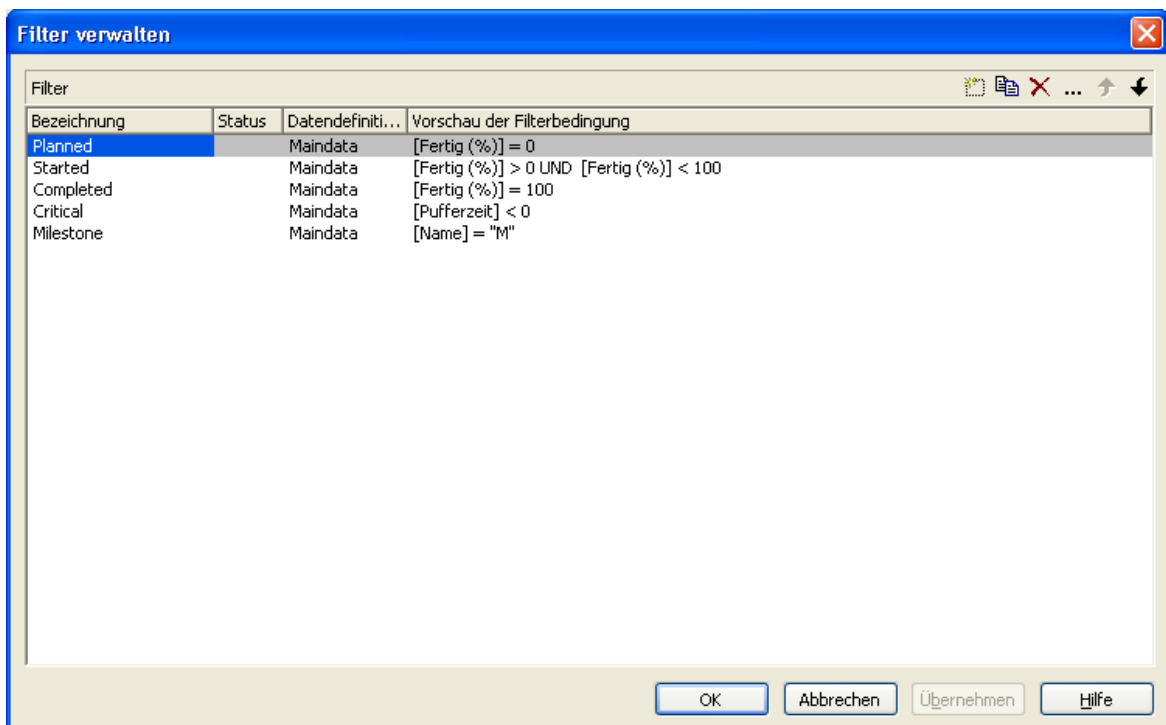


## 2.10 Filter für Knoten festlegen





Ein Filter enthält Kriterien zur Auswahl von bestimmten Daten, beispielsweise von Knoten und Verbindungen.

Wenn Sie Filter für das Knotenaussehen verwenden, erhalten genau die Knoten, die die Filterkriterien eines bestimmten Knotenaussehens erfüllen, dieses Aussehen.

Um einen Filter für ein Knotenaussehen zu bearbeiten, klicken Sie auf der Eigenschaftenseite **Objekte** auf die Schaltfläche **Filter**. Sie gelangen dann in das Dialogfeld **Filter verwalten**, in dem Sie Filter umbenennen, bearbeiten, neu definieren, kopieren und löschen können.



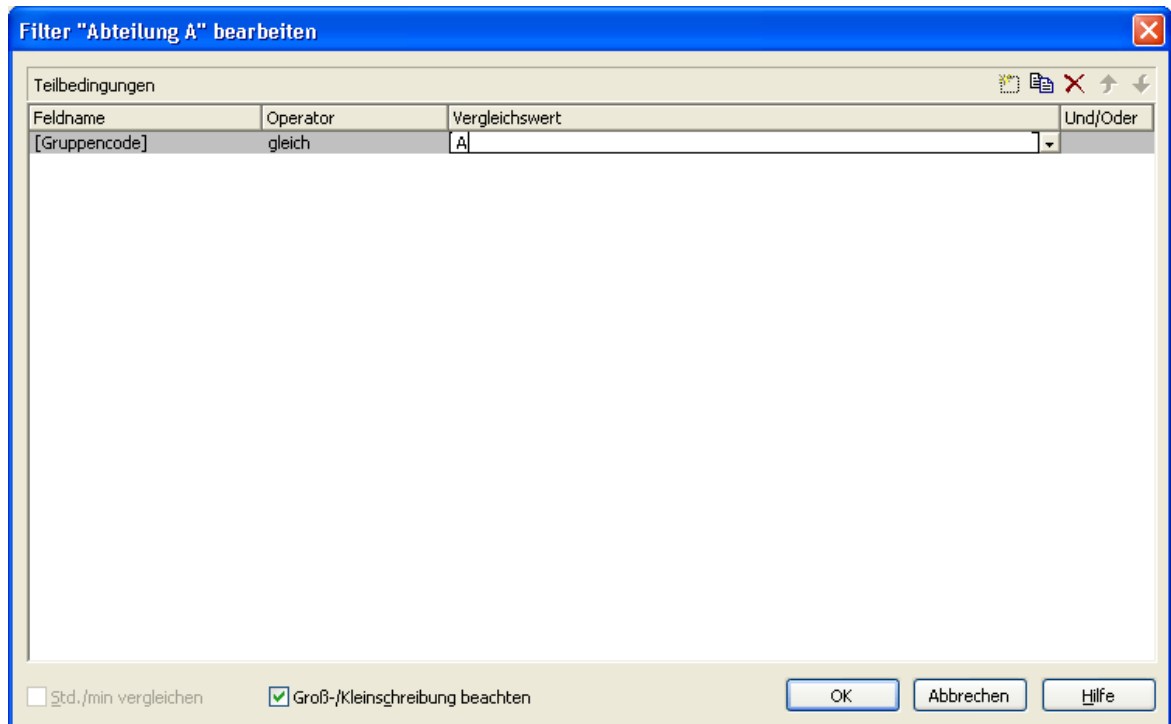
### > Schaltflächen im Dialogfeld "Filter verwalten"

-  Filter hinzufügen
-  Filter kopieren
-  Filter löschen
-  Filter bearbeiten

## > Filter erstellen und bearbeiten

Lernen Sie nun, wie Sie neue Filter erstellen und bearbeiten können. Klicken Sie dazu auf die Schaltfläche **Filter hinzufügen**. Der neue Filter wird am Ende der Liste angefügt. Ändern Sie seinen Namen in "Abteilung A" um.

Bearbeiten Sie nun den neuen Filter. Klicken Sie dazu auf die Schaltfläche **Filter bearbeiten**. Sie gelangen dann in das gleichnamige Dialogfeld. Nehmen Sie hier die folgenden Einstellungen vor:



The screenshot shows a dialog box titled "Filter 'Abteilung A' bearbeiten". It contains a table with the following structure:

Feldname	Operator	Vergleichswert	Und/Oder
[Gruppencode]	gleich	A	

At the bottom of the dialog, there are two checkboxes: "Std./min vergleichen" (unchecked) and "Groß-/Kleinschreibung beachten" (checked). There are also three buttons: "OK", "Abbrechen", and "Hilfe".

In der Kopfzeile wird der aktuelle Filter angezeigt, den Sie hier bearbeiten können.

Unter **Feldname** wird das Datenfeld angezeigt, dessen Inhalt mit dem Vergleichswert verglichen werden soll. Wählen Sie hier das Datenfeld "Gruppenname".

Unter **Operator** wird der aktuelle Vergleichsoperator angezeigt. Welche Operatoren möglich sind, hängt vom Typ des gewählten Datenfeldes ab. Wählen Sie hier "gleich".

Im Feld **Vergleichswert** können Sie angeben, mit welchem Wert der Eintrag des Datenfeldes verglichen werden soll. Der Typ des Vergleichswertes muss mit dem Typ des Datenfeldes übereinstimmen, das unter **Feldname** angegeben wurde. Wählen Sie hier "A".

Unter **Und/Oder** können Sie eine Verknüpfung auswählen, wenn Sie mehrere Kriterien definieren möchten.

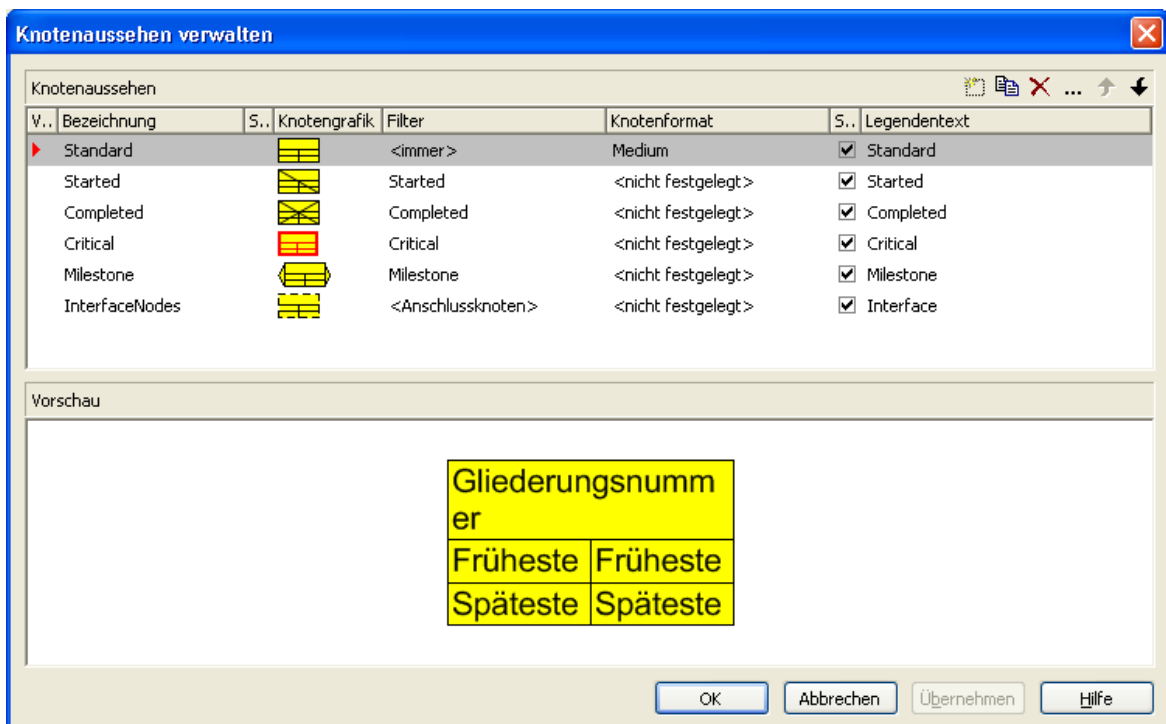
## 42 Filter für Knoten festlegen

Definieren Sie den Filter wie in der Abbildung. Verlassen Sie dann das Dialogfeld **Filter bearbeiten** mit **OK**. Sie kehren in das Dialogfeld **Filter verwalten** zurück.

## 2.11 Knotenaussehen festlegen

VARCHART XNet bietet eine Vielzahl von Gestaltungsmöglichkeiten für das Aussehen von Knoten. Sie können das Aussehen von Knoten in Abhängigkeit von deren Daten festlegen. Beispielsweise können alle kritischen Knoten durch einen roten Hintergrund und eine doppelte schwarze Rahmenlinie gekennzeichnet werden, alle beendeten Knoten durch gekreuzte Linien etc. Diese grafischen Attribute werden als Knotenaussehen bezeichnet.

Öffnen Sie nun die Eigenschaftenseite **Objekte** und klicken Sie auf die **Knotenaussehen**-Schaltfläche. Sie gelangen in das Dialogfeld **Knotenaussehen verwalten**.





In der **Knotenaussehen**-Tabelle werden alle aktuell vorhandenen Knotenaussehen angezeigt. Wenn Sie ein Aussehen anklicken, wird es im Vorschaufenster angezeigt.

Jedes Knotenaussehen ist mit einem Filter und einem Knotenformat verbunden. (Ausnahme: Das Knotenaussehen "Standard" ist nicht mit einem Filter verbunden.)

Der Filter gibt die Bedingungen an, unter denen ein Knoten dieses Knotenaussehen erhält. Beispielsweise ist das Knotenaussehen "Markiert" mit dem Filter "Markiert" verbunden. Dieser Filter selektiert alle markierten Knoten.

Erfüllt ein Vorgang die Filterkriterien mehrerer Knotenaussehen, werden diese Knotenaussehen für den Knoten grafisch überlagert. Begonnen wird dabei jeweils mit dem Knotenaussehen, das in der Liste ganz oben steht. Das Knotenaussehen, das ganz unten steht, wird als letztes zugewiesen und überlagert daher alle anderen.

Die niedrigste Position hat i. d. R. das Knotenaussehen "Standard", das normalerweise ganz oben in der Liste steht. Das Knotenaussehen "Standard" hat keinen Filter und wird auf alle Knoten angewendet.

  Sie können die Abarbeitungsreihenfolge der Knotenaussehen mit Hilfe der Pfeil-Schaltflächen verändern.

### > **Knotenaussehen hinzufügen, kopieren, löschen und bearbeiten**

Sie können im Dialogfeld **Knotenaussehen verwalten** mit Hilfe der folgenden Schaltflächen Knotenaussehen hinzufügen, kopieren, löschen und bearbeiten:

 **Knotenaussehen hinzufügen**

 **Knotenaussehen kopieren**

 **Knotenaussehen löschen**

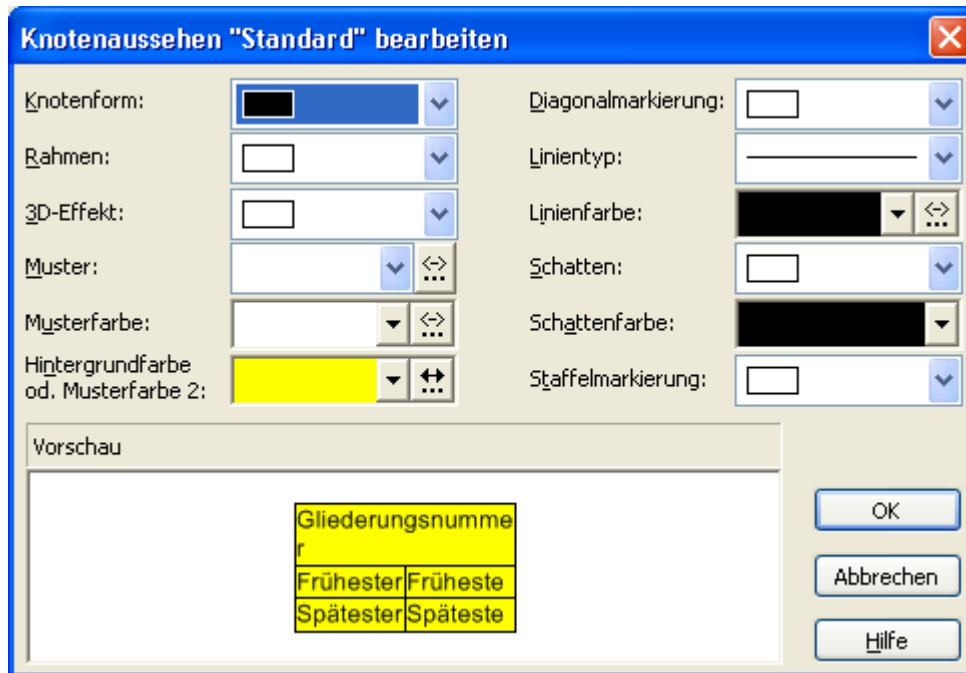
 **Knotenaussehen bearbeiten**

**Hinweis:** Alle Knotenaussehen außer den Standard-Knotenaussehen können gelöscht werden. Bevor das markierte Knotenaussehen tatsächlich gelöscht wird, erfolgt eine Rückfrage des Programms.

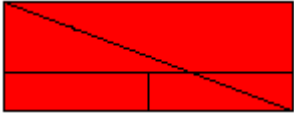
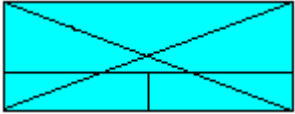
### > **Knotenaussehen und Filter verwenden**

Lernen Sie nun den Umgang mit Knotenaussehen und Filtern an einem Beispiel kennen. Weisen Sie dazu dem Knotenaussehen "Abgeschlossen" die höchste und dem Knotenaussehen "Begonnen" die zweithöchste Priorität zu.

Bearbeiten Sie nun diese beiden Knotenaussehen. Markieren Sie das zu bearbeitende Knotenaussehen im Dialogfeld **Knotenaussehen verwalten** und klicken Sie auf die **Knotenaussehen bearbeiten**-Schaltfläche. Sie gelangen in das Dialogfeld **Knotenaussehen bearbeiten**. In der Kopfzeile wird der Name des aktuellen Knotenaussehens angezeigt. In diesem Dialogfeld können Sie die grafischen Attribute der Knoten festlegen.



Legen Sie nun folgendes fest:

Knotenaussehen	Started	Completed
Filter	Started	Completed
Filterbedingung	completed (%) größer als 0 und kleiner als 100	completed (%) = 100
Hintergrundfarbe	rot	blau
Diagonalmarkierung	abwärts	gekreuzt
Aussehen		

Bestätigen Sie die Einstellungen mit **OK** und starten Sie das Programm. Erzeugen Sie einen Knoten, klicken Sie ihn doppelt an und bearbeiten Sie seine Daten im Dialogfeld **Vorgänge bearbeiten** in den folgenden drei Schritten:

- Geben Sie für "completed (%)" den Wert "0" an: Der Knoten erscheint mit dem Standard-Knotenaussehen.
- Geben Sie als nächstes für "completed (%)" eine Zahl größer als Null, aber kleiner als 100 an: Der Knoten erscheint mit dem "Started"-Knotenaussehen, also in rot und abwärts durchgestrichen.
- Geben Sie schließlich für "completed (%)" den Wert "100" an: Der Knoten erscheint mit dem Knotenaussehen "completed", also in blau und durchgestrichen mit gekreuzten Linien.

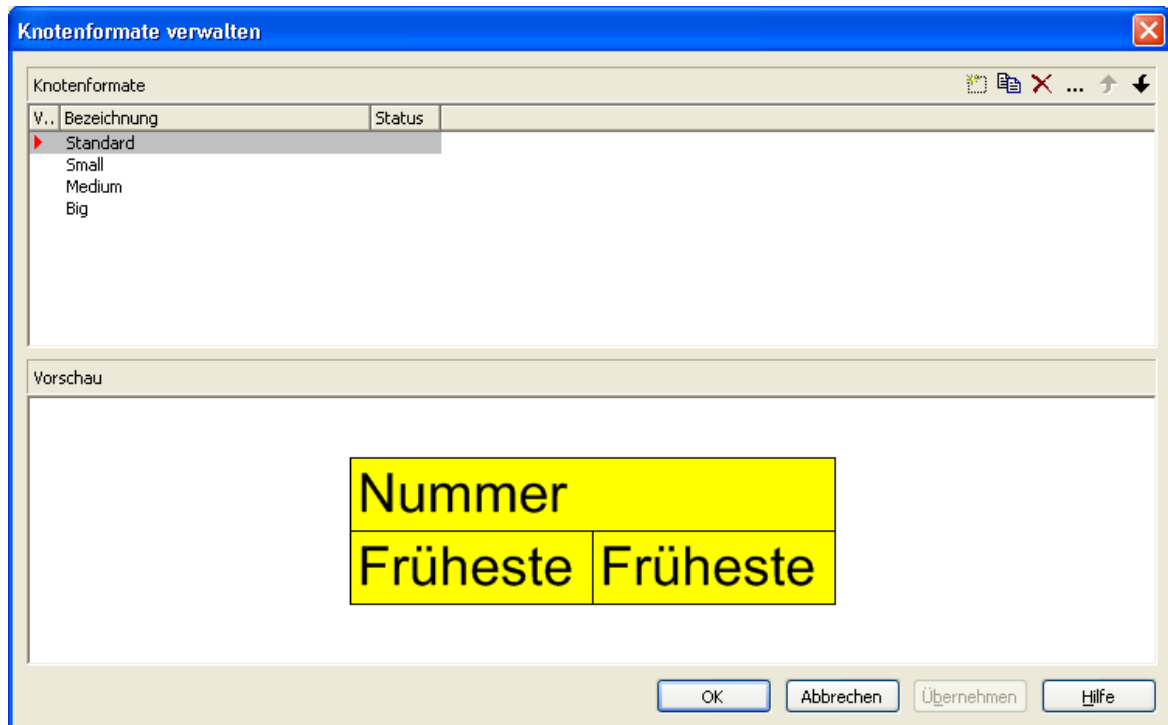
> **Knotenaussehen datenabhängig festlegen**

Muster, Musterfarbe, Hintergrundfarbe und die Linienfarbe eines Knotenaussehens können mit Hilfe von Zuordnungstabellen in Abhängigkeit von den Daten der Knoten festgelegt werden. Siehe hierzu das Kapitel "Wichtige Begriffe: Zuordnungstabellen".

## 2.12 Knotenformate festlegen





Jedes Knotenaussehen ist mit einem Knotenformat verbunden. Sie können die Knotenformate selbst festlegen.

Klicken Sie auf der Eigenschaftenseite **Objekte** auf die Schaltfläche **Knotenformate**. Sie gelangen dann in das Dialogfeld **Knotenformate verwalten**.



In der **Knotenformate**-Tabelle werden alle vorhandenen Knotenformate aufgeführt. Gehen Sie die Tabelle durch, um im Vorschaufenster das Aussehen des jeweils markierten Knotenformats zu betrachten.

Sie können im Dialogfeld **Knotenformate verwalten** mit Hilfe der folgenden Schaltflächen Knotenformate hinzufügen, kopieren, löschen und bearbeiten:

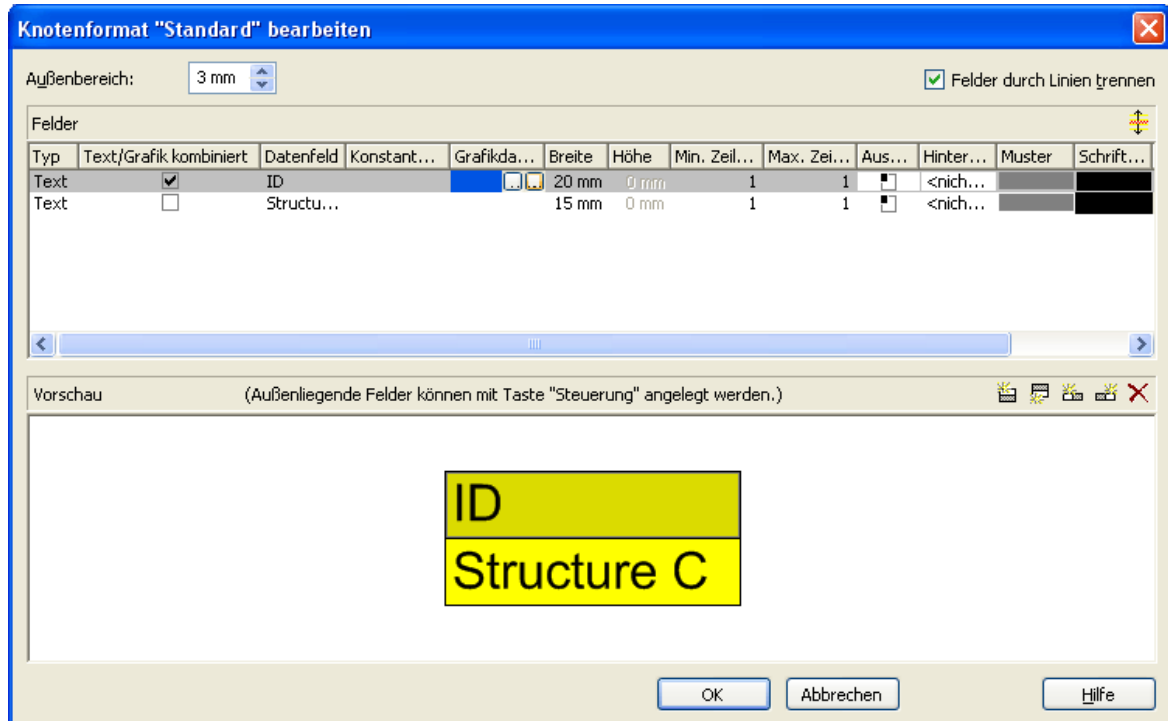
-  **Knotenformat hinzufügen**
-  **Knotenformat kopieren**
-  **Knotenformat löschen**
-  **Knotenformat bearbeiten**

**Hinweis:** Das Knotenformat "Standard" sowie jedes in einem Knotenaussehen verwendete Knotenformat können nicht gelöscht werden.



### > Knotenformat bearbeiten


Um ein Knotenformat zu bearbeiten, markieren Sie es und klicken Sie auf die Schaltfläche **Knotenformat bearbeiten**. Das folgende Dialogfeld erscheint:





In diesem Dialog können Sie für das gewählte Knotenformat Folgendes festlegen:

- ob die Knotenfelder durch Linien getrennt werden sollen
- den Außenbereich (Abstand in Millimetern, den Knoten mit diesem Knotenformat zu benachbarten Knoten und zum Rand der Darstellung halten)
- den Typ des aktuellen Knotenfeldes (Text oder Grafik)
- für den Typ Text: das Datenfeld, dessen Inhalt in dem aktuellen Knotenfeldes ausgegeben werden soll, oder einen konstanten Text
- für den Typ Grafik: Name und Pfad der Grafikdatei, die in dem gewählten Knotenfeld dargestellt wird
- die Breite und Höhe des markierten Knotenfeldes
- die maximale Anzahl von Textzeilen im aktuellen Knotenfeld
- die Ausrichtung des Textes bzw. der Grafik des markierten Knotenfeldes
- die Hintergrundfarbe des Knotenfeldes
- das Muster des Knotenfeldes
- die Schriftart und -farbe des Knotenfeldes

### > **Darstellung von Grafiken in Knotenfeldern**

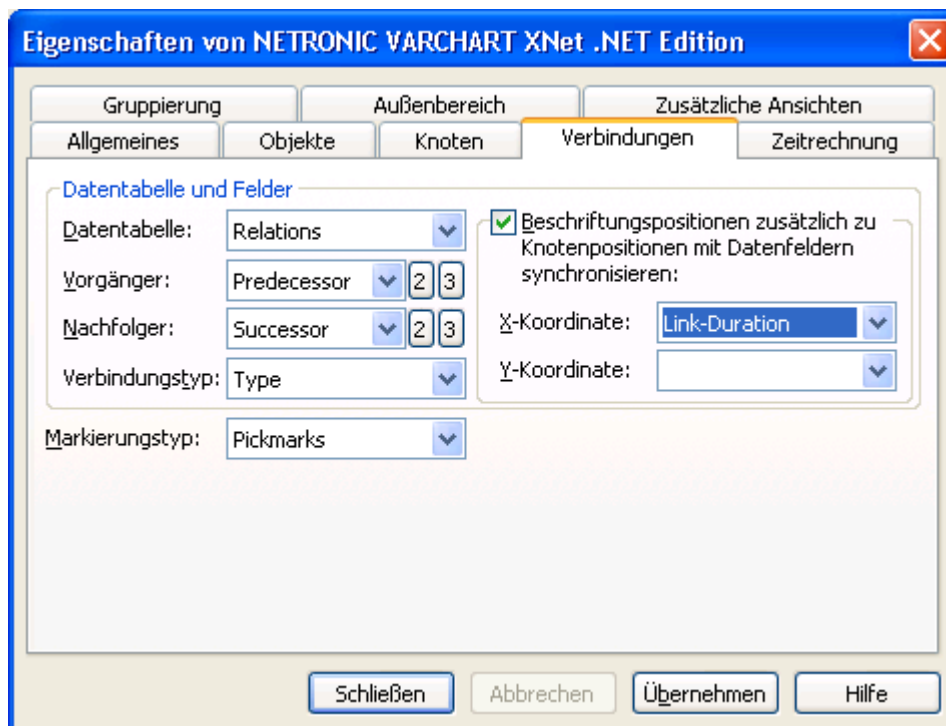
Sie können für ein Knotenfeld des Typs Grafik eine Grafikdatei wählen, indem Sie auf die Schaltfläche **Grafikdatei auswählen** () klicken und dann im gleichnamigen Windows-Dialog eine Datei wählen.

Sie können aber auch eine Zuordnung zwischen den Einträgen eines Datenfeldes und Grafikdateien herstellen. Klicken Sie dazu auf die Schaltfläche **Zuordnungen einstellen** () , um den gleichnamigen Dialog zu öffnen. Wenn eine Zuordnung vorgenommen worden ist, wird das durch ein Symbol neben dem Grafikdateinamen dargestellt () . Einzelheiten hierzu finden Sie in den Kapiteln "Eigenschaftenseiten und Dialogfelder" und "Wichtige Begriffe: Zuordnungstabellen".

## 2.13 Das Aussehen von Verbindungen festlegen

Sie haben in den letzten Abschnitten schon die Verwendung von Filtern für Knoten kennengelernt. Filter werden nicht nur für die Festlegung des Knotenaussehens, sondern auch des Verbindungsaussehens verwendet. Sie können beispielsweise ein jeweils unterschiedliches Aussehen der Verbindungen für unterschiedliche Verbindungstypen (z.B. Start-Start-, Ende-Ende-Verbindungen etc.) festlegen.

Um das Aussehen von Verbindungen festzulegen, öffnen Sie die Eigenschaftenseite **Verbindungen**.



In der Tabelle im unteren Bereich der Seite können Sie neue Verbindungsaussehen definieren bzw. bestehende bearbeiten.

In der letzten Zeile finden Sie den Eintrag "Neu...". Klicken Sie hier zweimal und definieren Sie das neue Verbindungsaussehen "FF-Verbindung". Sobald Sie in ein anderes Feld klicken, wird dieses Verbindungsaussehen der Liste hinzugefügt und ein neuer "Neu"-Eintrag erzeugt. Für das neu definierte Verbindungsaussehen werden zunächst derselbe Filter und dieselben Linienattribute wie für das letzte Verbindungsaussehen verwendet.

Klicken Sie nun auf das **Filter**-Feld des zu bearbeitenden Verbindungsaussehens. Eine Pfeil- und eine **Bearbeiten**-Schaltfläche erscheinen. Klicken Sie auf die Pfeil-Schaltfläche neben dem Filternamen, damit alle verfügbaren

Filter für Verbindungsaussehen in der Kombobox angezeigt werden. Bis jetzt existiert nur der Standardfilter ("immer").

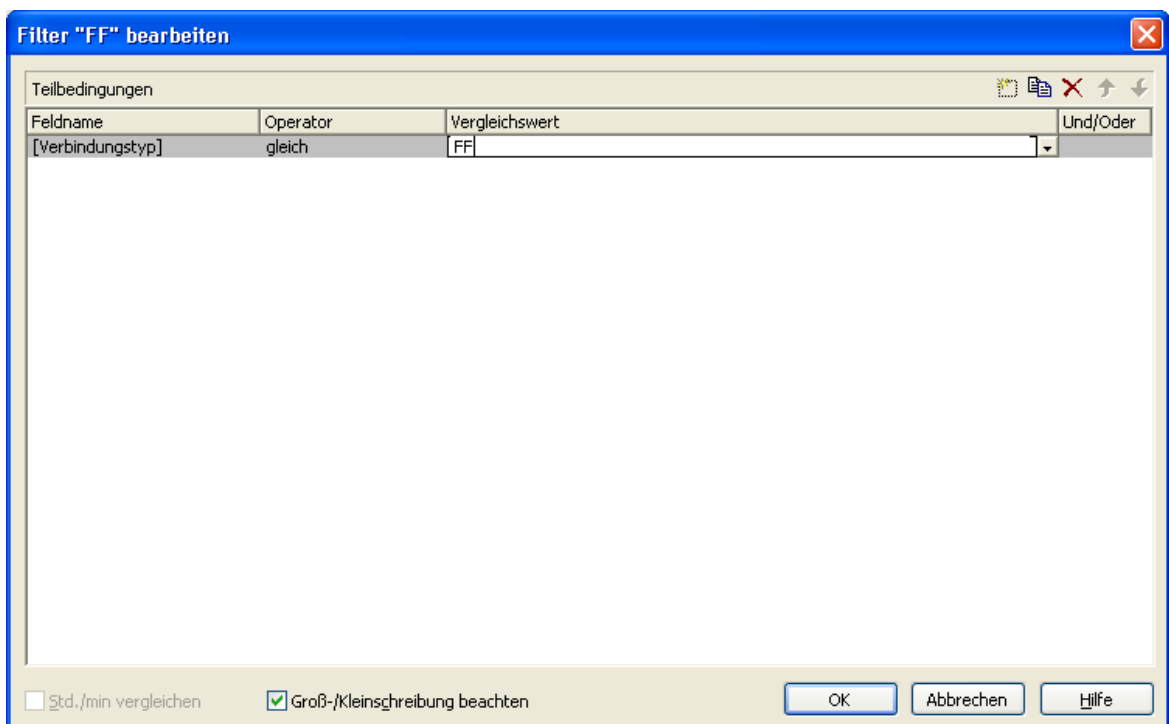
Definieren Sie nun einen neuen Filter für Verbindungsaussehen. Klicken Sie dazu auf die **Filter**--Schaltfläche um in das Dialogfeld **Filter verwalten** zu gelangen. Hier können Sie Filter hinzufügen, kopieren, bearbeiten und löschen. Klicken Sie auf die **Filter hinzufügen**-Schaltfläche, um einen neuen Filter anzulegen. Benennen Sie diesen Filter um in "FF". Dieser Filter soll alle Ende-Ende-Verbindungen (FF) auswählen.

Bearbeiten Sie nun den Filter "FF". Klicken Sie dazu auf die Schaltfläche **Filter bearbeiten**, um das gleichnamige Dialogfeld zu öffnen.

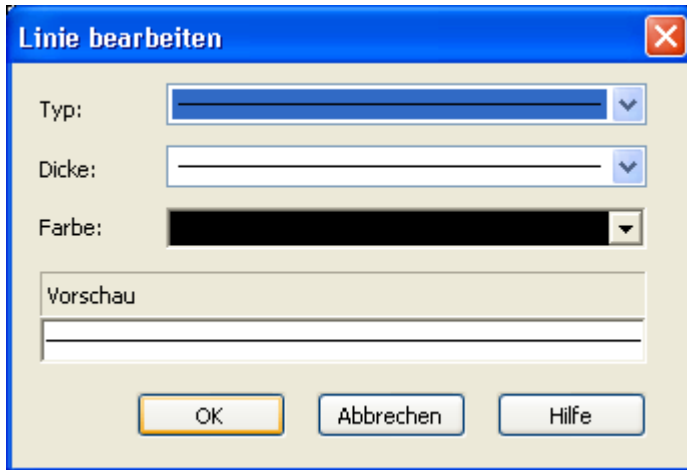
Definieren Sie die Filterbedingung "Verbindungstyp gleich FF". Damit werden alle Verbindungen vom Typ Ende-Ende (FF) mit dem als "FF-Verbindung" definierten Verbindungsaussehen dargestellt.

Beachten Sie bitte, dass alle Änderungen, die Sie an einem Filter durchführen, für diesen Filter allgemein und nicht nur für das aktuelle Verbindungsaussehen gelten.

Klicken Sie auf **OK** und anschließend im Dialog **Filter verwalten** nochmals auf **OK**.

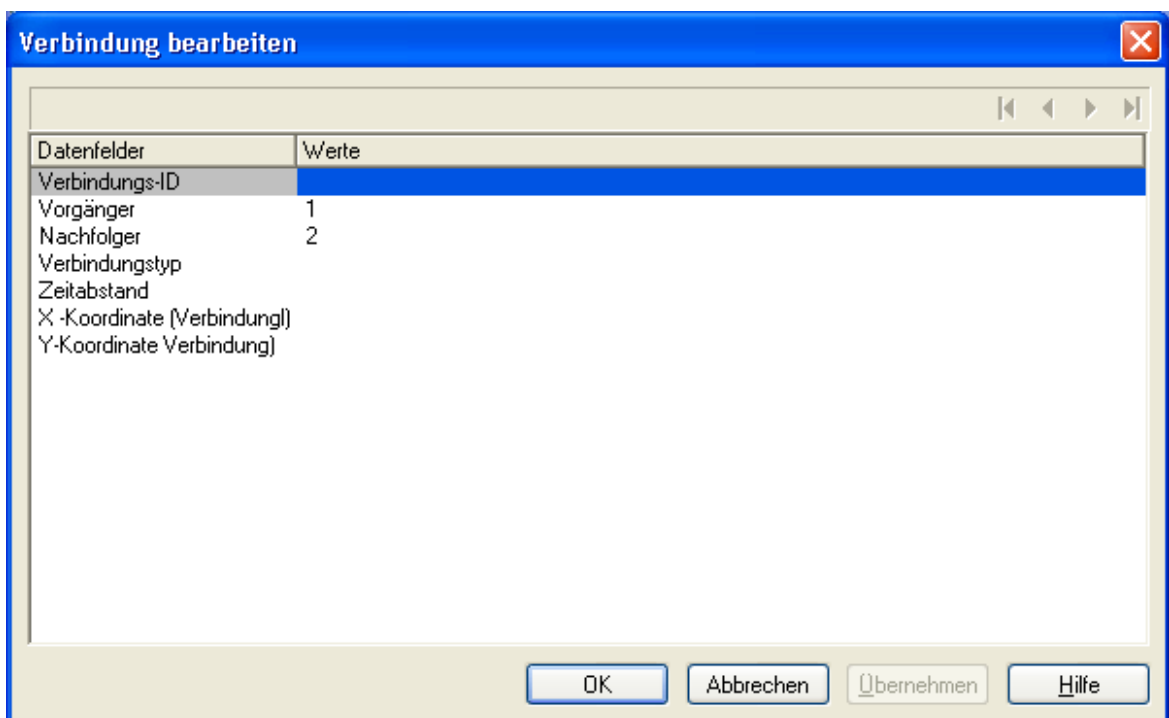


Legen Sie schließlich noch die Linienattribute für das Verbindungsaussehen "FF-Verbindung" fest. Klicken Sie dazu auf die **Bearbeiten**-Schaltfläche neben der Linienart. Es öffnet sich das Dialogfeld **Linie bearbeiten**, in dem Sie das Aussehen der Verbindungslinien festlegen können.

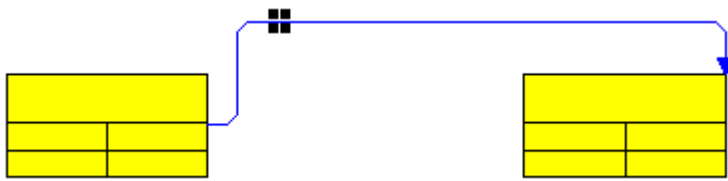


Wählen Sie eine blaue Linie für die Verbindungen vom Typ "FF-Verbindung", also für die Ende-Ende-Verbindungen.

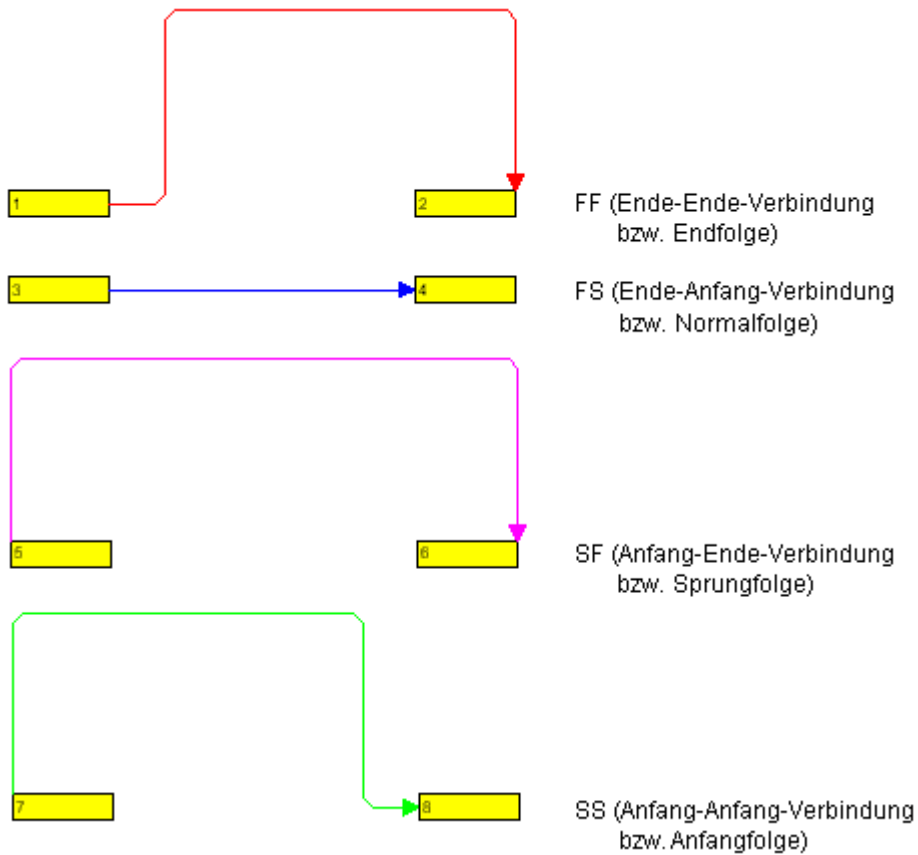
Starten Sie nun das Programm mit F5 und erzeugen Sie zwei Knoten mit einer Verbindung. Klicken Sie mit der rechten Maustaste auf die Verbindung und wählen Sie den Befehl **Bearbeiten** aus dem Kontextmenü, um das Dialogfeld **Verbindung bearbeiten** aufzurufen.



Geben Sie unter **Typ** nun "FF" ein und bestätigen Sie mit **OK**. Dann wird die markierte Verbindung zu einer Ende-Ende-Verbindung, die mit einer blauen Linie dargestellt wird.

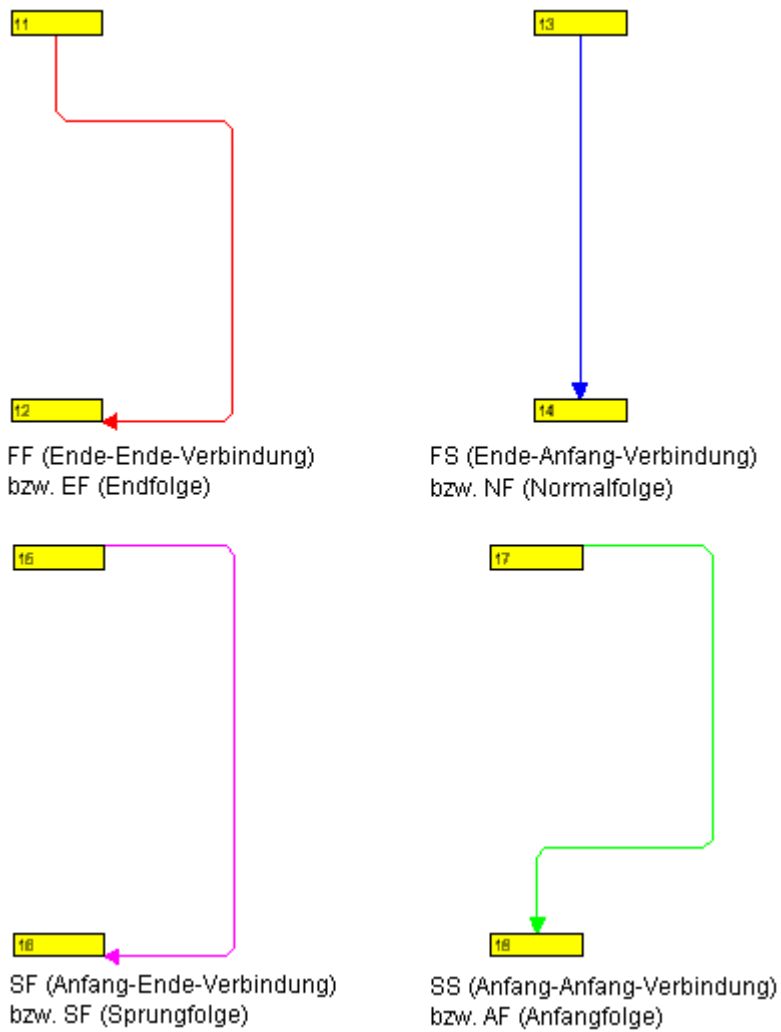


Eine Übersicht über das Aussehen der verschiedenen Verbindungstypen bei den beiden Flussrichtungen geben die folgenden Abbildungen:



*Flussrichtung von links nach rechts*

## 54 Das Aussehen von Verbindungen festlegen



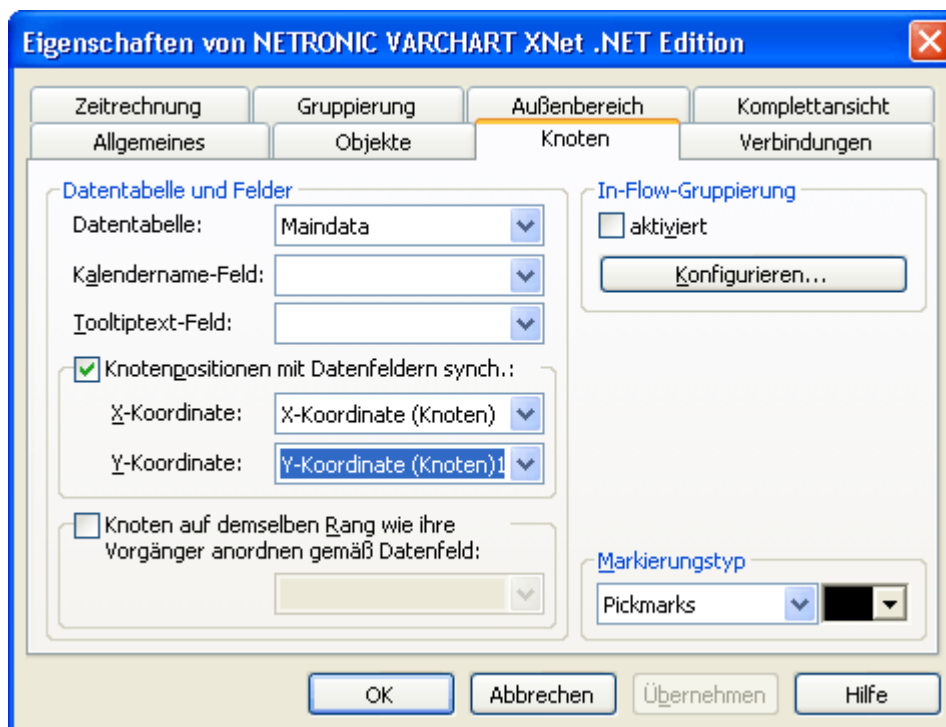
*Flussrichtung von oben nach unten*

## 2.14 Positionen von Knoten und Verbindungsbeschriftungen speichern

Damit die Positionen von Knoten und Verbindungsbeschriftungen nach dem Schließen Ihres Projekts wiederhergestellt werden können, müssen diese in bestimmten Datenfeldern gespeichert werden.

Um die Positionen der Knoten mit den entsprechenden Datenfeldern zu synchronisieren, aktivieren Sie auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Knotenpositionen mit Datenfeldern synchronisieren** und geben Sie folgende Datenfelder an:

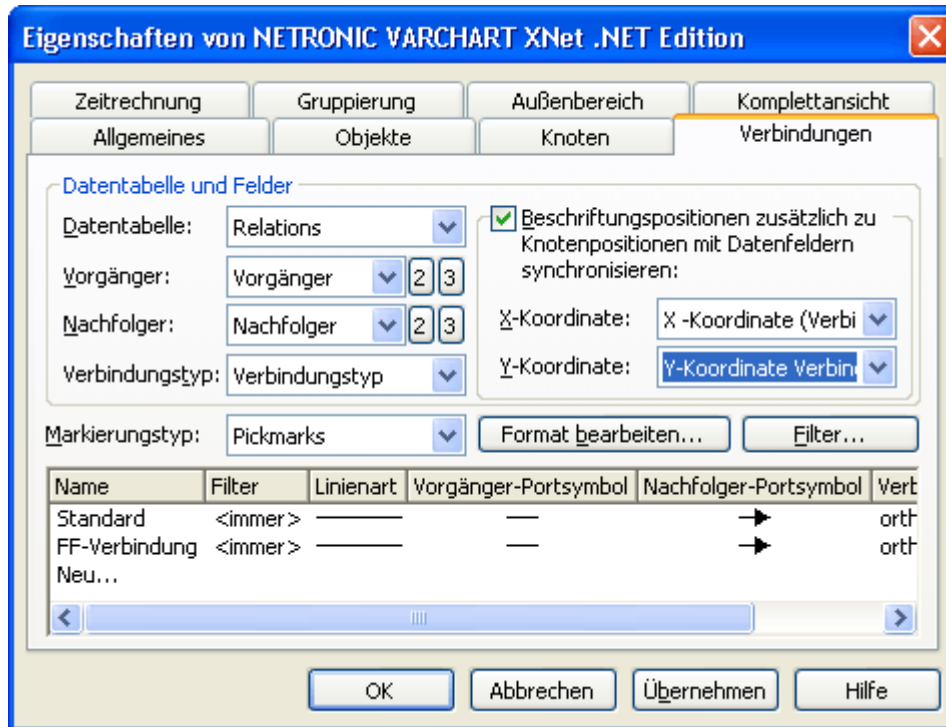
- für die X-Koordinate: "X-Koordinate (Knoten)"
- für die Y-Koordinate: "Y-Koordinate (Knoten)"



Und um die Positionen der Verbindungsbeschriftungen mit den entsprechenden Datenfeldern zu synchronisieren, aktivieren Sie auf der Eigenschaftenseite **Verbindungen** das Kontrollkästchen **Positionen der Beschriftungen mit Datenfeldern synchronisieren** und geben Sie folgende Datenfelder an:

- für die X-Koordinate: "X-Koordinate (Verbindung)"
- für die Y-Koordinate: "Y-Koordinate (Verbindung)"

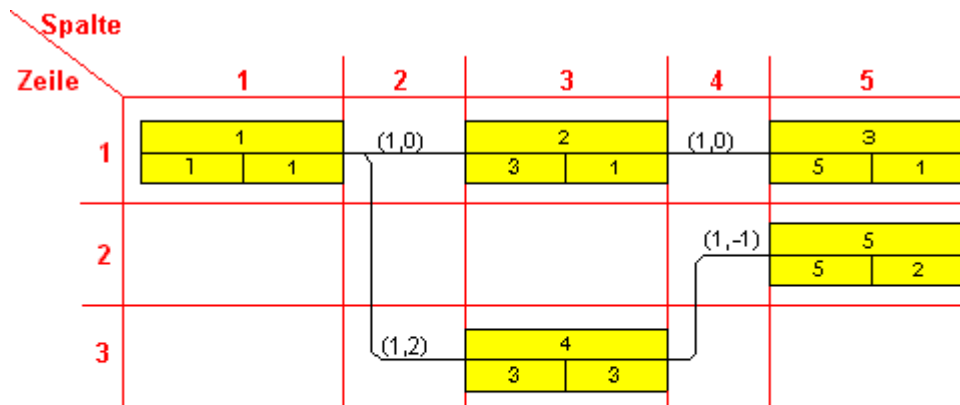




Die Positionen von Knoten und Verbindungsbeschriftungen im Diagramm werden als deren Koordinaten in einem Raster festgelegt.

- Die X- und Y-Koordinate eines Knotens geben die absolute Position dieses Knotens im Raster an.
- Die X- und Y-Koordinate einer Verbindungsbeschriftung geben die Position dieser Verbindungsbeschriftung relativ zu ihrem Vorgängerknoten an.

Das erste Feld des Rasters (ganz links oben) ist definiert als  $(X,Y) = (1,1)$ . Es ist für Knoten reserviert. Von links nach rechts und von oben nach unten wird jeweils in Schritten von 1 weitergezählt. In allen weiteren Feldern können Knoten oder Verbindungsbeschriftungen stehen. Knoten haben daher immer positive X- und Y-Koordinaten. Bei Verbindungsbeschriftungen sind alle Wertepaare außer  $(0,0)$  möglich.

**Legende:**

Nummer		Knotenfeld
X-Position	Y-Position	

(x,y) Verbindungsbeschriftungsposition

Damit Sie die angelegten Knoten und Verbindungen sichern und wieder laden können, ergänzen Sie für das Ereignis FormClosing:

**Code-Beispiel**

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

**Code-Beispiel**

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

## 2.15 Hilfsknoten übersichtlich anordnen

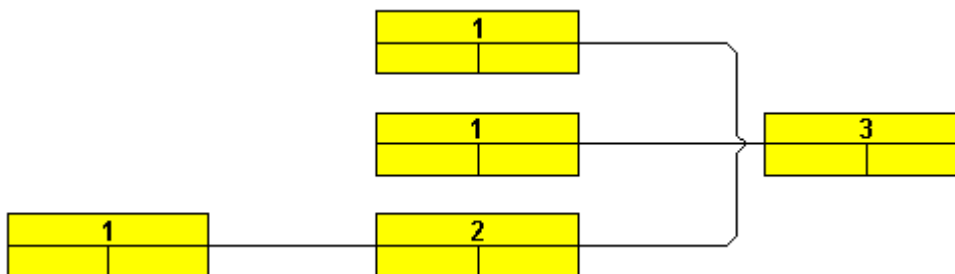
In vielen Fällen ist es sinnvoll, Hilfsknoten nicht in Flussrichtung anzuordnen, sondern entgegen der Flussrichtung unter- bzw. oberhalb ihrer Vorgänger (bei Flussrichtung von links nach rechts) bzw. neben ihren Vorgängern (bei Flussrichtung von oben nach unten). Dies ist in VARCHART XNet möglich, da es erlaubt, den Rang von Knoten zu reduzieren.

Unter dem Rang eines Knotens versteht man eine Zahl, die folgendermaßen definiert ist: Der Rang eines Knotens ohne Vorgänger ist 1. Der Rang eines Knotens mit Vorgängern ist gleich 1 plus Rang desjenigen seiner Vorgänger, der den höchsten Rang besitzt.

Durch diese Definition wird verhindert, dass in Netzdiagrammen Zyklen vorkommen.

### Einige Beispiele:

- Der Rang eines Knotens mit einem Vorgänger, der keinen Vorgänger besitzt, ist  $1+1=2$ .
- Der Rang eines Knotens, der drei Vorgänger mit den Rängen 1, 1 bzw. 2 hat, ist  $1+2=3$  (siehe Abbildung).



### Ränge der Knoten bei Flussrichtung von links nach rechts

Den Rang von Knoten kann man sich folgendermaßen veranschaulichen:

- Bei Flussrichtung von links nach rechts entspricht der höchste Rang von allen Knoten in einer Diagrammspalte der Nummer dieser Diagrammspalte (gezählt werden hierbei nur die Diagrammspalten für Knoten, nicht die für Verbindungsbeschriftungen).
- Bei Flussrichtung von oben nach unten entspricht der höchste Rang von allen Knoten in einer Diagrammzeile der Nummer dieser Diagrammzeile (gezählt werden hierbei nur die Diagrammzeilen für Knoten, nicht die für Verbindungsbeschriftungen).

Ränge werden nur beim Aufruf von **Anordnen** (Kontextmenü für das Diagramm) berechnet. Sie dienen dem Layout-Algorithmus als Vorgabe für

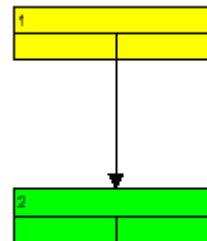
die Positionierung der Knoten in Flussrichtung. Bei Ringbeziehungen (Schleifen) zwischen Knoten wählt VARCHART XNet über einen eigenen Algorithmus Verbindungen aus, die bei der Rangberechnung vorübergehend ignoriert werden. Die ignorierten Verbindungen sind nach dem Layout als rückführende Verbindungen sichtbar. Dabei wird darauf geachtet, dass das Layout möglichst wenige Veränderungen gegenüber einem Layout ohne diese rückführenden Verbindungen aufweist.

In manchen Anwendungsfällen ist es sinnvoll, einen Knoten auf demselben Rang wie seinen Vorgänger zu plazieren, z. B. wenn dieser Knoten einen Hilfsknoten zu seinem Vorgänger darstellt. Der Rang eines solchen Knotens kann dazu um 1 reduziert werden.

Bei Flussrichtung von links nach rechts steht der Hilfsknoten, dessen Rang um 1 reduziert wurde, unter oder über seinem Vorgänger statt daneben.

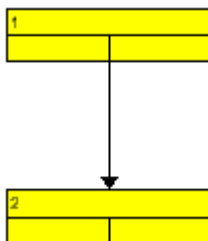


*Zwei Knoten mit Rang 1 bzw. 2*

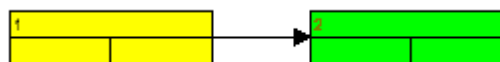


*Der Rang des 2. Knotens wurde um 1 reduziert. Anschließend wurde der Befehl **Anordnen** aufgerufen.*

Bei Flussrichtung von oben nach unten steht der Hilfsknoten, dessen Rang um 1 reduziert wurde, links oder rechts neben seinem Vorgänger statt darunter.



*Zwei Knoten mit Rang 1 bzw. 2*



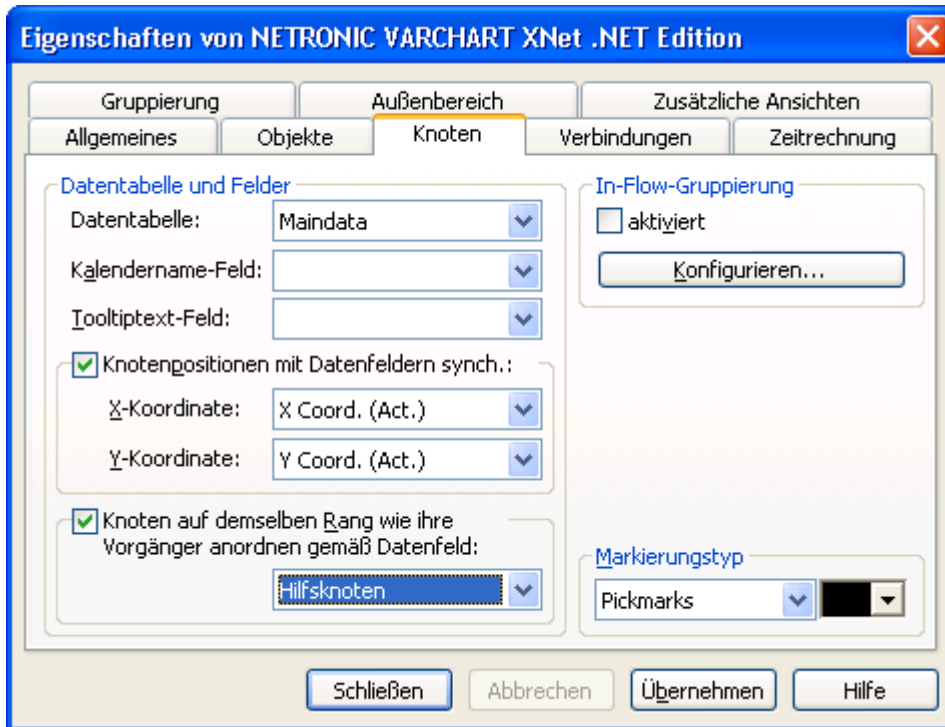
*Der Rang des 2. Knotens wurde um 1 reduziert. Anschließend wurde der Befehl **Anordnen** aufgerufen.*

Um den Rang eines Knotens reduzieren zu können, ist das Datenfeld "Hilfsknoten" vorgesehen. Der Eintrag in diesem Datenfeld legt fest, wie der Hilfsknoten positioniert wird. Mögliche Werte sind 0, 1, 2, 3.

Wert im Feld "Hilfsknoten"	Flussrichtung von oben nach unten	Flussrichtung von links nach rechts
0	Der Rang des Hilfsknotens wird nicht reduziert.	Der Rang des Hilfsknotens wird nicht reduziert.
1	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann links oder rechts neben seinem Vorgänger statt darunter.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann über oder unter seinem Vorgänger statt daneben.
2	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann links von seinem Vorgänger.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann über seinem Vorgänger.
3	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann rechts von seinem Vorgänger.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann unter seinem Vorgänger.

Lernen Sie nun an einem praktischen Beispiel kennen, wie Sie Hilfsknoten auf demselben Rang wie ihre Vorgänger anordnen können.

Wählen Sie auf der Eigenschaftenseite **Allgemeines** die **Flussrichtung** von links nach rechts. Kreuzen Sie dann auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Knoten auf demselben Rang wie ihre Vorgänger anordnen gemäß Datenfeld** an. Wählen Sie aus der Kombobox das Datenfeld "Hilfsknoten" aus. Von dem Wert eines Knotens im Datenfeld "Hilfsknoten" hängt dann ab, ob dieser Knoten auf demselben Rang positioniert wird wie sein Vorgänger.



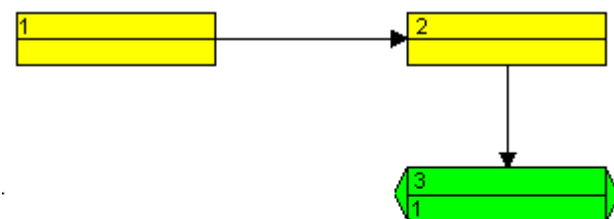
Starten Sie mit diesen Einstellungen das Programm, erzeugen Sie einige Knoten und verbinden Sie diese miteinander wie in der folgenden Abbildung:




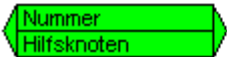
Doppelklicken Sie nun auf den 3. Knoten und geben Sie in dem **Vorgänge bearbeiten**-Dialogfeld unter "Hilfsknoten" den Wert 1 an. Sie erhalten dann die folgende Darstellung:



Öffnen Sie das Kontextmenü für das Diagramm und wählen Sie den Befehl **Anordnen**. Das Resultat sieht folgendermaßen aus:



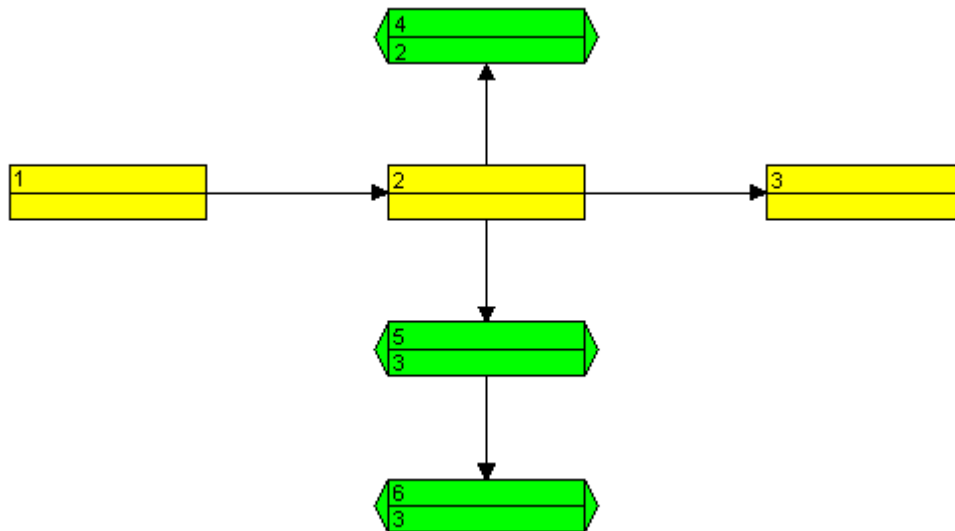
**Legende:**

-  Standard
-  Hilfsknoten

## 62 Hilfsknoten übersichtlich anordnen

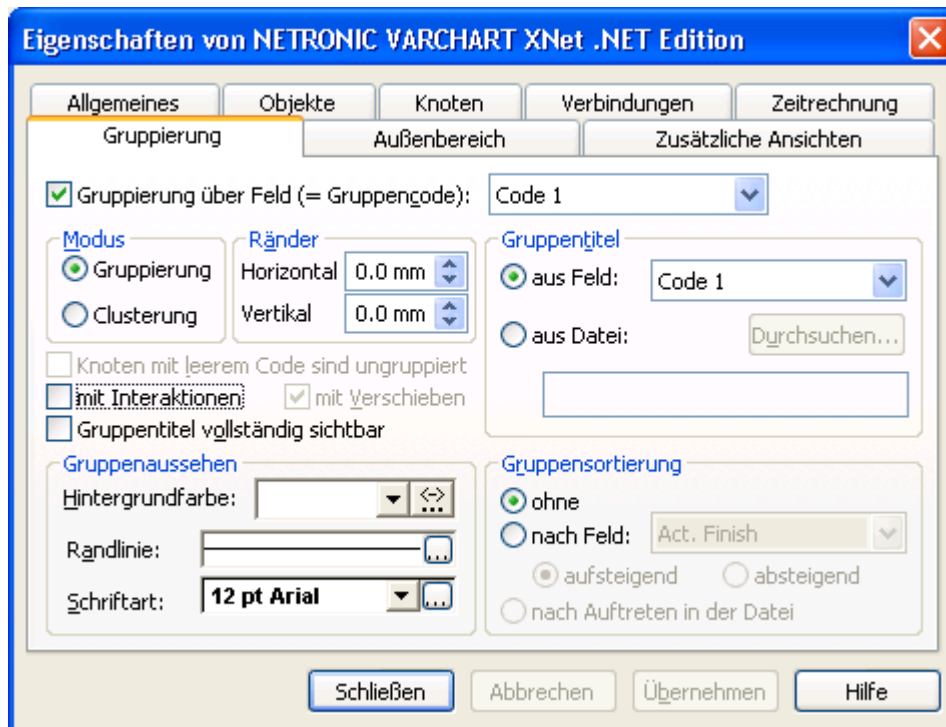
Der Knoten, dessen Rang reduziert worden ist, wird nun unterhalb statt rechts von seinem Vorgänger angeordnet.

Die folgende Abbildung zeigt eine Anordnung mit unterschiedlichen Hilfsknoten (Flussrichtung: von links nach rechts):



## 2.16 Knoten gruppieren

In vielen Fällen lassen sich Netzpläne übersichtlicher gestalten, indem Knoten gruppiert angeordnet werden. Die Gruppierung können Sie auf der Eigenschaftenseite **Gruppierung** festlegen.



Legen Sie zunächst unter **Gruppierung über Feld (= Gruppencode)** fest, nach welchem Feld die Gruppierung vorgenommen werden soll. Das Feld, das Sie hier auswählen, wird als **Gruppencode** bezeichnet. Wählen Sie für das Tutorium das Feld "Gruppencode". Alle Knoten, die denselben Eintrag für "Gruppencode" haben, werden so in einer Gruppe zusammengefasst.

Für die Beschriftung der Gruppen können Sie zwischen zwei Alternativen wählen: Entweder werden die Gruppentitel aus einem Feld Ihrer Wahl (**aus Feld**) oder aus einer Datei (**aus Datei**) genommen.

Aktivieren Sie für das Tutorium das Kontrollkästchen **aus Feld** und wählen Sie das Feld "Gruppenname". Das Feld, das Sie für den Gruppentitel wählen, muss nicht notwendigerweise mit dem Gruppencode (= Feld, nach dem gruppiert wird) identisch sein. Doch damit die Gruppen sinnvoll beschriftet werden, sollten die Einträge im **Gruppencode**- und im **Gruppentitel**-Feld miteinander korrespondieren.

Verwenden Sie für das Tutorium die folgende Tabelle, wenn Sie zur Laufzeit Knoten erzeugen und bearbeiten:



## 64 Knoten gruppieren

Gruppencode	Gruppenname
A	Anlagenplanung
B	Berechnung Gesamtanlage
C	Detailkonstruktion

Legen Sie nun fest, ob und ggf. nach welchem Kriterium die Gruppen sortiert werden. Gruppen können unsortiert bleiben oder sortiert werden, entweder nach einem Datenfeld Ihrer Wahl oder nach ihrem Auftreten in der Datei.

Aktivieren Sie unter **Gruppensortierung** die Option **nach Feld**, wählen Sie das Datenfeld "Gruppencode" und die Option **absteigend**. Dadurch werden die Gruppen in absteigender Reihenfolge nach ihrem Gruppencode sortiert.

Im Bereich **Gruppenaussehen** können Sie die Farbe, die Dicke und den Typ der Linien, mit denen die einzelnen Gruppen umrandet werden, sowie die Hintergrundfarbe der Gruppen und die Schriftattribute der Gruppentitel festlegen. Gestalten Sie das Gruppenaussehen nach Ihren Vorstellungen und starten Sie anschließend das Programm.

Legen Sie nun einige Knoten an und tragen Sie dafür jeweils Werte in die Datenfelder "Gruppencode" und "Gruppenname" ein. Beachten Sie dabei die Korrespondenz zwischen diesen Datenfeldern.

Sie erhalten eine Darstellung wie die folgende:

Anlagenplanung	
4	5
A	A
Berechnung Gesamtanlage	
1	2
B	B
Detail konstruktion	
3	
C	

Verschieben Sie nun einen Vorgang von einer Gruppe in eine andere. Der Wert des als Gruppencode vereinbarten Datenfeldes (hier "Gruppencode") wird dabei automatisch angepasst. (Sie können das überprüfen, indem Sie das **Vorgänge bearbeiten**-Dialogfeld für den verschobenen Vorgang aufrufen.)

## 2.17 Zeitrechnung des VARCHART XNet festlegen

Mit dem Scheduler von VARCHART XNet können Sie einfache Terminberechnungen durchführen. Die gewünschten Projektstart- und Projektende-Termine werden dabei als Parameter übergeben.

Mit Hilfe der Eigenschaftenseite **Zeitrechnung** können Sie die Zeitrechnung des VARCHART XNet an Ihre Schnittstelle anpassen, indem Sie festlegen, welche Datenfelder für die Eingabe (**Zeitrechnungseingabe**) und die Ausgabe (**Zeitrechnungsergebnis**) des Schedulers verwendet werden sollen. Außerdem können Sie die Zeiteinheit festlegen, die für die Berechnung der Dauer in den entsprechenden Datenfeldern in Knoten und Verbindungen verwendet werden soll.



Wählen Sie unter **Zeitrechnungseingabe** für jede Eingabe aus, aus welchem Feld sie entnommen werden soll. Verwenden Sie für dieses Tutorium dieselben Festlegungen wie in der Abbildung.

Als Eingaben für die Zeitrechnung verwendet der Scheduler Datenfelder der Maindata- und der Relations-Tabelle. Die Grundlage der Berechnung sind die Dauer der einzelnen Vorgänge, deren logische Abhängigkeiten und der Projektanfang. Die Felder **Vorgänger**, **Nachfolger** und **Verbindungstyp** können in der **Zeitrechnungseingabe**-Tabelle nicht bearbeitet werden. Sie geben nur die auf den Eigenschaftenseiten **Verbindungen** vorgenommenen Einstellungen wieder.

Wählen Sie unter **Zeitrechnungsergebnis** für jede Ausgabe aus, in welches Feld sie geschrieben werden soll. Die Ausgabe des Schedulers erfolgt nur in Datenfelder der Maindata-Tabelle. Die Ausgaben werden wiederum in Datenfelder der Schnittstelle geschrieben. Als Ausgaben stehen zur Verfügung: **früh. Start, früh. Ende, spät. Start, spät. Ende, Freier Puffer** und **Gesamtpuffer**. Weisen Sie jeder dieser Ausgaben ein Datenfeld aus der in der Datendefinition vereinbarten Liste von Feldern zu (wie in der Abbildung).

Es gibt folgende Möglichkeiten, die Zeitrechnung zu beeinflussen:

1. Sie können einen Projektstart vorgeben. Das erreichen Sie über die API mit der VcNet-Methode **ScheduleProject**:

```
VcNet1.ScheduleProject "04.05.2000", 0
```

Mit der Methode **ScheduleProject** können Sie eine Vorwärts- und Rückwärtsberechnung des aktuellen Projekts durchführen. Bei Übergabe des Starttermins wird zunächst eine Vorwärtsberechnung, dann eine Rückwärtsberechnung durchgeführt. Bei Übergabe des Endtermins wird zunächst eine Rückwärtsberechnung, dann eine Vorwärtsberechnung durchgeführt. Es können auch Anfangs- und Enddatum übergeben werden, die Vorgänge erhalten dann entsprechende Pufferzeiten.

#### Mögliche Wahl der Parameter für die Methode **ScheduleProject**:

Anfang	Ende
Termin 1	0
0	Termin 2
Termin 1	Termin 2

2. Sie können aktuelle Start- bzw. Endtermine angeben. Die Knoten sind dann unverrückbar.
3. Sie können für die Bedingungen "Start nicht früher als" und "Ende nicht später als" Referenztermine angeben. Dazu wird auf der Eigenschaftenseite **Zeitrechnung** in der linken Tabelle für die entsprechenden Werte auch jeweils ein Feld aus der Datendefinition festgelegt. Dann liegt der früheste Start eines Knoten nicht vor dem angegebenen Termin bzw. das späteste Ende nicht nach dem angegebenen Termin.

Probieren Sie nun die Zeitrechnung aus. Definieren Sie dazu drei Schaltflächen ("Command1", "Command2" und "Command3"), um die Zeitrechnung zur Laufzeit ausführen zu lassen. Benennen Sie die

Schaltflächen mit "Projektstart", "Projektende" und "Projektanfang und -ende" und schreiben Sie dafür den folgenden Code:

#### Code-Beispiel

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

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

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

Geben Sie nun die folgenden Codezeilen ein, damit beim Starten des Programms einige Knoten und Verbindungen geladen werden.

#### Code-Beispiel

```
Private Sub Form_Load()

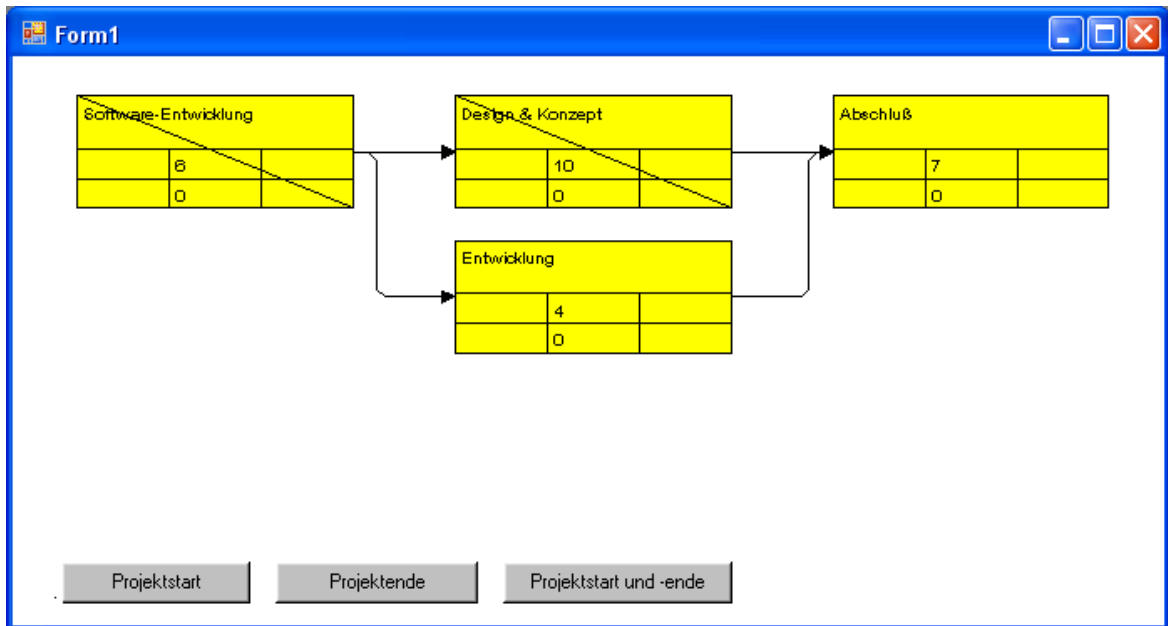
    VcNet1.InsertNodeRecord ("1;1.;;;Software-Entwicklung;A;;
                             Gruppe A;6;0;10;;;0;")
    VcNet1.InsertNodeRecord ("2;1.2;;;Design & Konzept;C;;
                             Gruppe A;10;0;50;;;0;")
    VcNet1.InsertNodeRecord ("3;1.2.2;;;Abschluss;B;;
                             Gruppe A;7;0;0;;;0;")
    VcNet1.InsertNodeRecord ("4;1.2.4;;;Entwicklung;B;;
                             Gruppe A;4;0;0;;;0;")

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

    VcNet1.EndLoading

End Sub
```

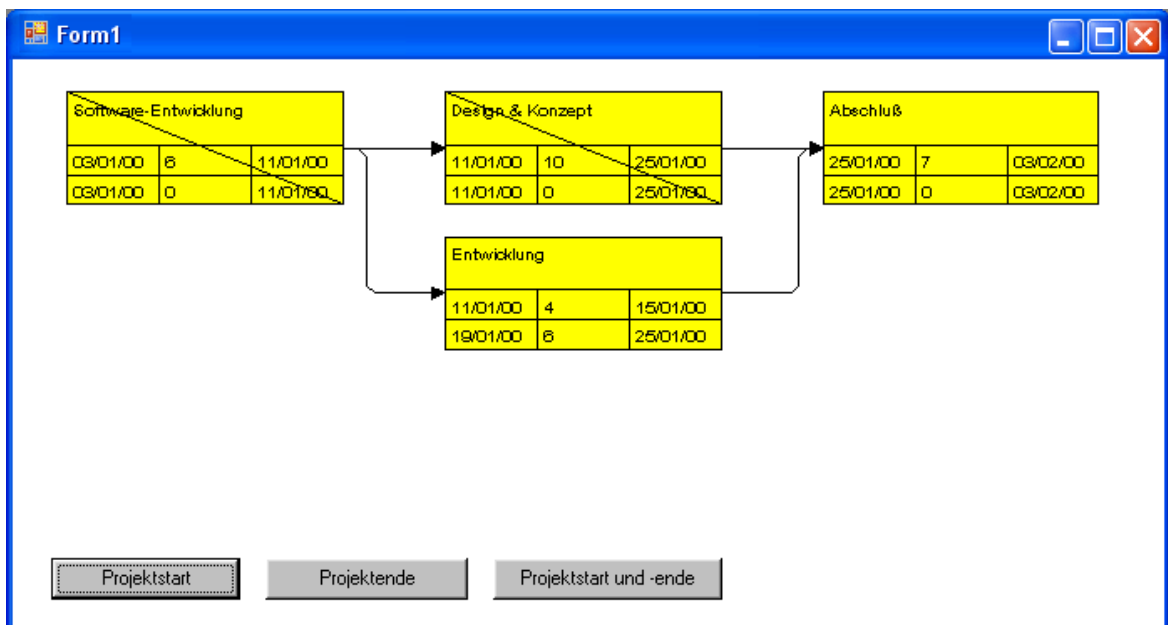
Starten Sie nun das Programm. Die Knoten und Verbindungen, die Sie über die API geladen haben, werden angezeigt:



verwendetes Knotenformat: "Groß"

Name		
Frühester Start	Dauer	Frühestes Ende
Spätester Start	Pufferzeit	Spätestes Ende

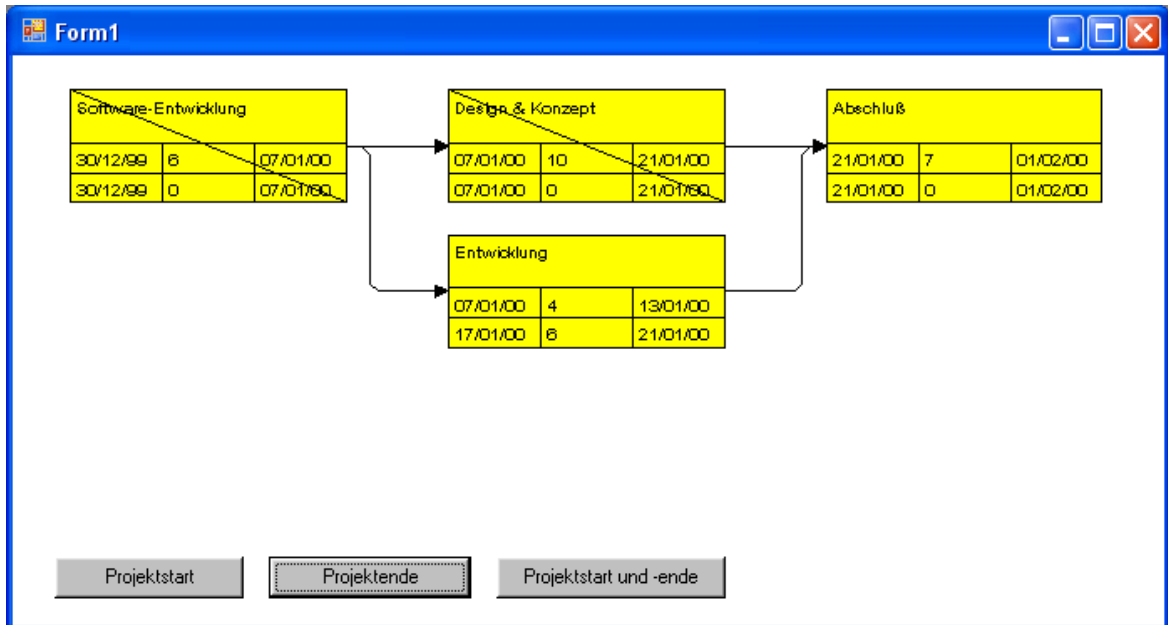
Klicken Sie dann auf die Schaltfläche "Projektstart". Nun werden die Termine so berechnet, dass der Starttermin berücksichtigt wird.



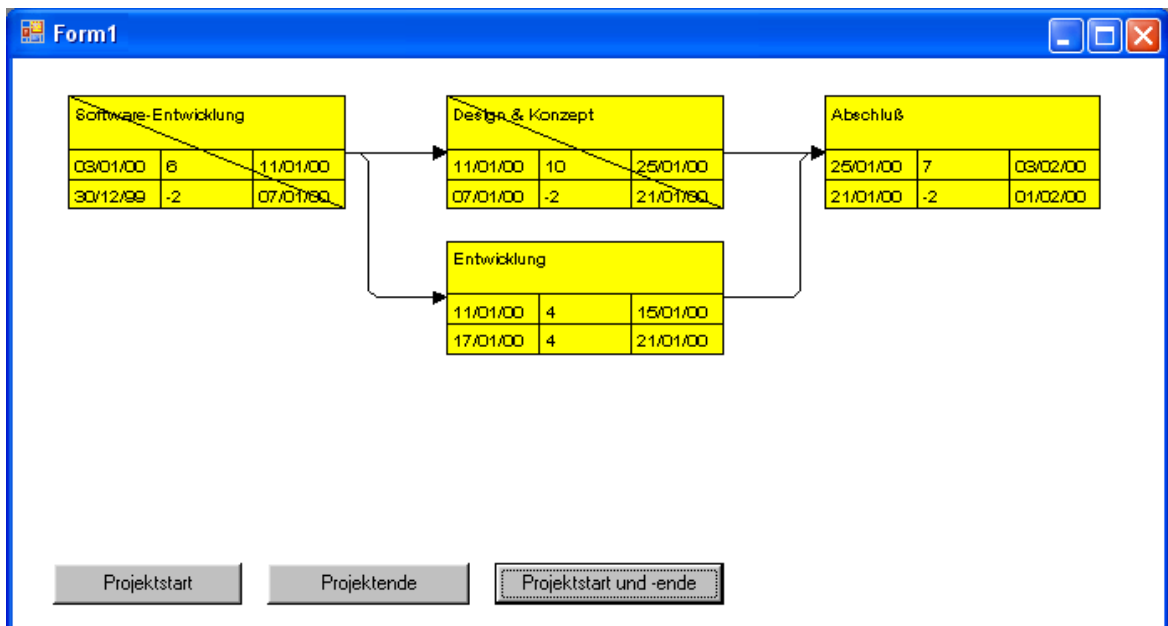
Beachten Sie bitte, dass bei der Berechnung der Termine der interne Kalender berücksichtigt wurde, in dem die Wochenenden als arbeitsfreie Zeiten festgelegt sind. (Der interne Kalender wird verwendet, wenn auf der

Eigenschaftenseite **Allgemeines** das Kontrollkästchen **Scheduler benutzt internen Kalender** aktiviert ist.)

Klicken Sie nun auf die Schaltfläche "Projektende". Nun werden die Termine so berechnet, dass der Endtermin berücksichtigt wird.



Klicken Sie schließlich auf die Schaltfläche "Projektstart und -ende". Nun werden die Termine so berechnet, dass der Start- und der Endtermin berücksichtigt werden.



## 70 Zeitrechnung des VARCHART XNet festlegen

Sie sehen, dass bei der Berücksichtigung von Start- und Endtermin negative Pufferzeiten auftreten.

---

## 2.18 Diagramm drucken

Wenn Sie Ihr Diagramm nach Ihren Vorstellungen gestaltet haben, können Sie es schließlich ausdrucken. Wählen Sie dazu zur Laufzeit den Befehl **Drucken** des Kontextmenüs (rechter Mausklick im freien Diagrammbereich).

Alternativ können Sie auch die Methode **ShowPrintDialog** des Objektes VcNet verwenden.

Sie gelangen dann in das Windows-Dialogfeld **Drucken**. Gegebenenfalls können Sie zur Laufzeit auch die Druckereinstellungen bearbeiten, indem Sie den Befehl **Drucker einrichten** des Kontextmenüs aufrufen und damit das entsprechende Windows-Dialogfeld aufrufen.

Mit der Methode **Print** des Objektes VcNet können Sie das Diagramm direkt ausdrucken, ohne dass zuvor ein Dialogfeld erscheint.

Wenn Sie zur Laufzeit die Seiteneinstellungen verändern möchten, können Sie aus dem Kontextmenü den Befehl **Seite einrichten** auswählen, oder auf **Seitenansicht** klicken und dort auf die Schaltfläche **Seite einrichten**.

Alternativ können Sie auch die Methode **ShowPageSetupDialog** des Objektes VcNet verwenden, um das entsprechende Dialogfeld aufzurufen.

Im **Seite einrichten**-Dialogfeld können Sie Einstellungen zur Skalierung, zur Seitenaufteilung, zur Seitennummerierung, zu den Seitenrändern etc. vornehmen. Weitere Informationen hierzu finden Sie im Kapitel 5.14, "Seite einrichten".



---

## 2.19 Diagramm exportieren

Sie können Ihr Diagramm auch als Grafikdatei exportieren. Dazu gibt es folgende Möglichkeiten:

- Wählen Sie bitte den Befehl **Grafik exportieren** des Standard-Kontextmenüs. Dann gelangen Sie in das Windows-Dialogfeld **Speichern unter**, in dem Sie das dargestellte Diagramm als Grafikdatei speichern können.
- Verwenden Sie die API-Methode **ShowExportGraphicsDialog** bzw. **ExportGraphicsToFile**

Detaillierte Erläuterungen zu den Grafikformaten finden Sie im Kapitel: **Wichtige Konzepte: Grafikformate**.

---

## 2.20 Konfigurationseinstellungen speichern

Alle Einstellungen, die Sie auf den Eigenschaftenseiten zur Design-Zeit vornehmen, werden als Ressource Ihrem Projekt hinzugefügt. Vorgenommene Änderungen werden erst wirksam, wenn Sie Ihr Projekt speichern, denn durch das Speichern wird die eingebettete Ressource aktualisiert.

**Tipp:** Aus diesem Grunde bietet es sich an, dass Sie in Microsoft Visual Studio .NET 2005 unter **Extras > Optionen > Umgebung > Projekte und Projektmappen > Erstellen und Ausführen** (diese Auswahl ist nur verfügbar, wenn **Alle Einstellungen anzeigen** ausgewählt ist) die Option **Alle Änderungen speichern** aktivieren, damit Ihre Einstellungen vor dem Kompilieren automatisch gespeichert werden.

Sollten Sie diese Option nicht aktiviert haben, dann müssen Sie Ihr Projekt immer manuell speichern, wenn Sie möchten, dass die in den Eigenschaftenseiten vorgenommenen Änderungen im Programm verwendet werden.

Alle Einstellungen der Eigenschaftenseiten können Sie auch jederzeit in Form einer Konfiguration außerhalb Ihres Projektes speichern und nach Bedarf wieder einlesen. Dies ist sehr praktisch, wenn Sie zu einem früheren Stand der Einstellungen zurückkehren oder die gleichen Einstellungen für andere Projekte verwenden möchten.

Eine gespeicherte Konfiguration besteht aus zwei Dateien mit gleichem Namen aber unterschiedlichen Dateiendungen. Zu einer Konfiguration gehört jeweils eine ini- und eine ifd-Datei, die beide zwingend benötigt werden.

### **So speichern Sie Ihre aktuelle Konfiguration:**

Klicken Sie auf der Eigenschaftenseite **Allgemeines** auf die Schaltfläche **Exportieren als...** und vergeben im folgenden Dialog den gewünschten Dateinamen für die ini-Datei. Die ifd-Datei wird automatisch erzeugt.

### **So lesen Sie eine bereits gespeicherte Konfiguration wieder ein:**

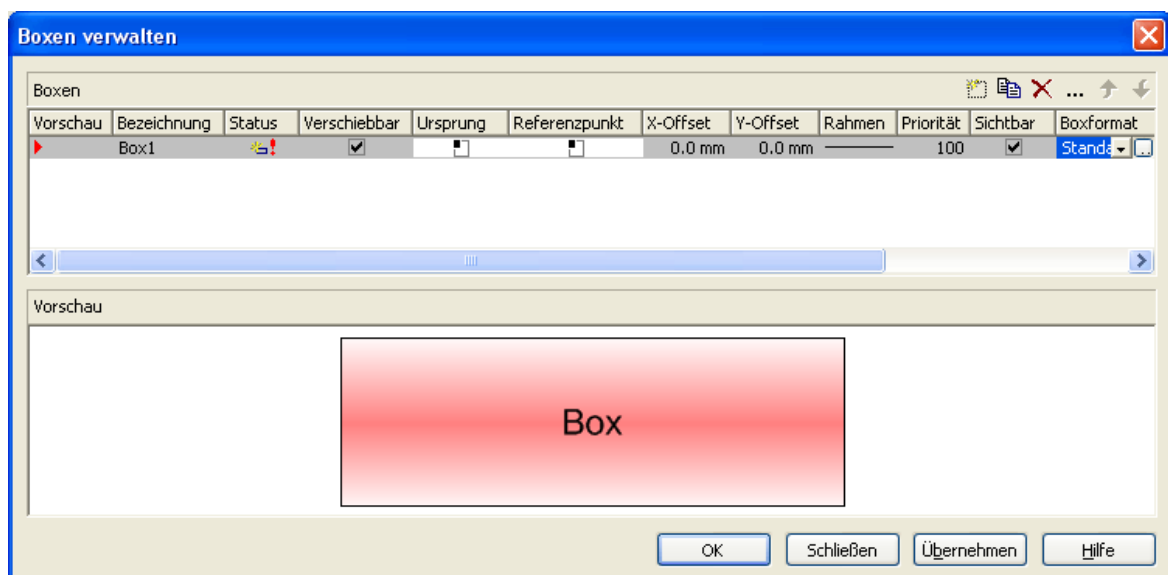
Klicken Sie auf der Eigenschaftenseite **Allgemeines** auf die Schaltfläche **Import...** und wählen die gewünschte Datei aus.



## 3 Wichtige Konzepte

### 3.1 Boxen

Im Diagrammbereich können beliebig viele Boxen, die Text oder Grafiken enthalten können, dargestellt werden. Über die Eigenschaftenseite **Objekte** und die Schaltfläche **Boxen...** gelangen Sie zum Dialog **Boxen verwalten**, in dem Sie Boxen neu anlegen, kopieren, bearbeiten und löschen können.



Mithilfe der Eigenschaften **Ursprung**, **Referenzpunkt**, **X-Offset** und **Y-Offset** können Sie jede einzelne Box im Diagrammbereich positionieren, wobei die relative Position der Box zum Diagramm unabhängig von der Diagrammgröße ist.

Für jede Box können Sie hier folgendes festlegen:

- ihre Bezeichnung
- ob die Box zur Laufzeit frei im Diagrammbereich verschiebbar sein soll
- ihren Ursprung (den Diagrammpunkt, von dem aus der Abstand zum Referenzpunkt der Box in x- bzw. y-Richtung angegeben wird)
- ihren Referenzpunkt (Punkt der Box, von dem aus der Abstand zum Ursprung in x- bzw. y-Richtung angegeben wird)
- ihren X- bzw. Y-Offset (Abstand zwischen Ursprung und Referenzpunkt in x- bzw. y-Richtung)

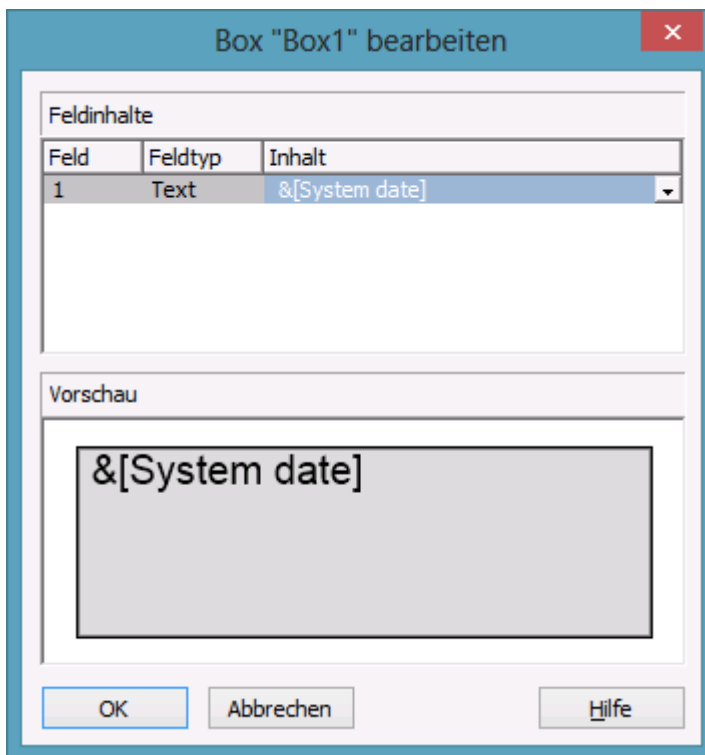
## 76 Wichtige Konzepte: Boxen

- Typ, Dicke und Farbe der Umrandungslinie der Box
- ihre Priorität gegenüber anderen Objekten im Diagramm
- ob die Box sichtbar ist
- das Boxformat

### > **Boxen bearbeiten**

Im Dialogfeld **Box bearbeiten** können Sie den Inhalt der Felder festlegen.

Zur Designzeit erreichen Sie dieses Dialogfeld, indem Sie im Dialogfeld **Boxen verwalten** auf die **Box bearbeiten**-Schaltfläche klicken. Zur Laufzeit kann ein Benutzer durch Doppelklick mit der linken Maustaste auf die jeweilige Box in das Dialogfeld gelangen. Wenn die Option **In-Place-Editieren zulassen** auf der Eigenschaftsseite **Allgemeines** aktiviert wird, können Texte zur Laufzeit auch direkt bearbeitet werden.



In der Spalte **Feld** werden die Nummern aller Felder der Box aufgeführt. (Die Anzahl der Felder hängt vom gewählten Boxformat ab.)

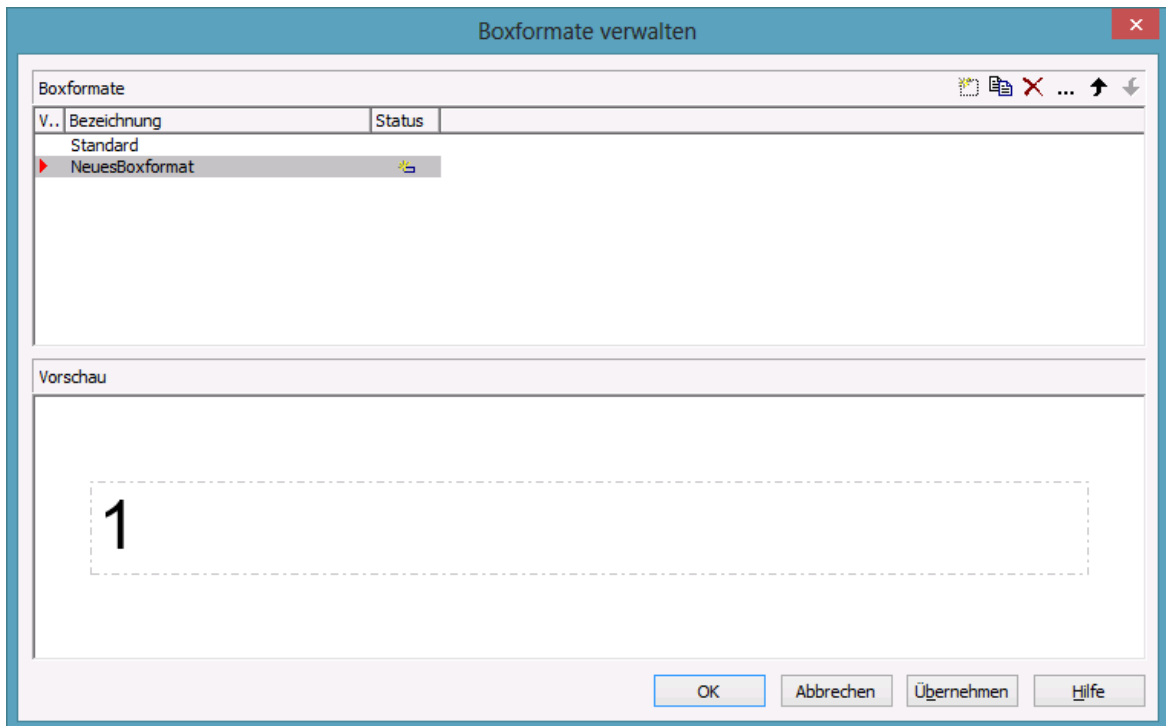
In der Spalte **Typ** wird der Feldtyp jedes Feldes angezeigt (Text oder Grafik).


In der Spalte **Inhalt** können Sie den Inhalt des Feldes bzw. den Namen einer Grafikdatei eingeben. Bei mehrzeiligen Textfeldern müssen für einen erzwungenen Umbruch die einzelnen Zeilen des Textfelds mit "\n" im String getrennt sein (Beispiel: "Zeile1\nZeile2"). Ohne erzwungenen Umbruch wird automatisch an Leerzeichen umgebrochen.

## > Boxformate

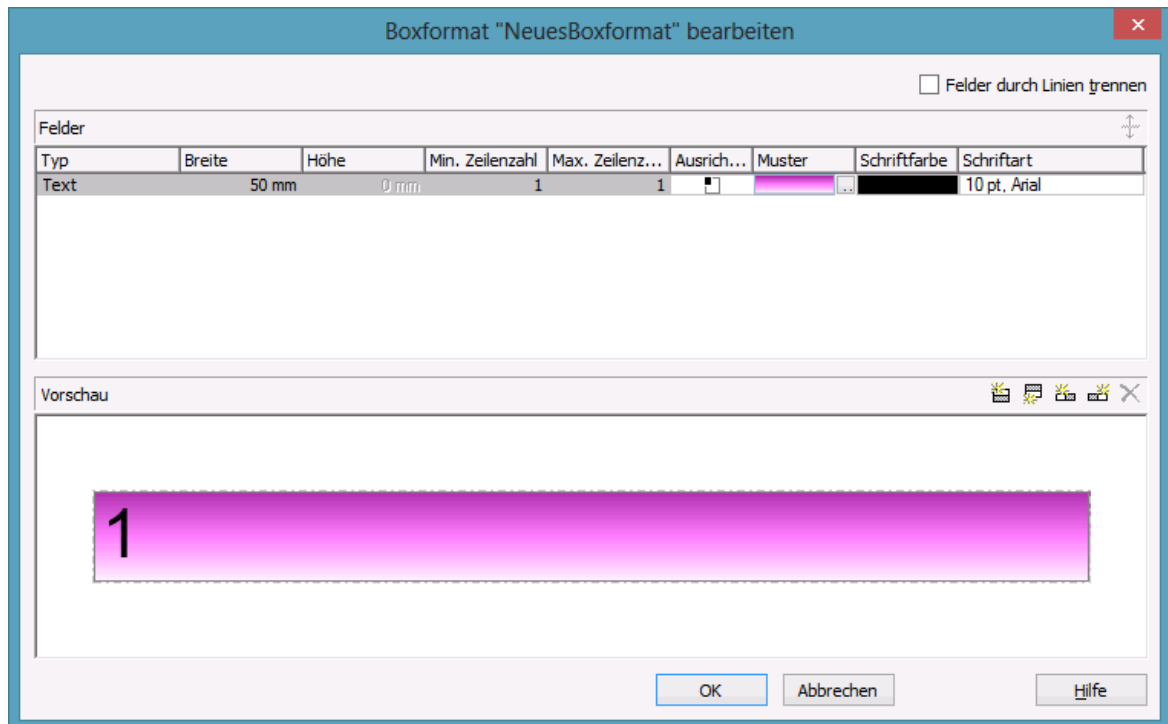
Für jede Box können Sie ein Boxformat wählen. Die Boxformate können Sie selbst festlegen.

Im Dialog **Boxformate verwalten** können Sie Boxformate hinzufügen, kopieren, bearbeiten und löschen. Sie erreichen dieses Dialogfeld über die entsprechende Schaltfläche auf der Eigenschaftenseite **Objekte**.



Im Dialog **Boxformat bearbeiten** können Sie das Boxformat festlegen. Sie erreichen dieses Dialogfeld, indem Sie im Dialogfeld **Boxformate verwalten** auf die Schaltfläche  klicken.

## 78 Wichtige Konzepte: Boxen



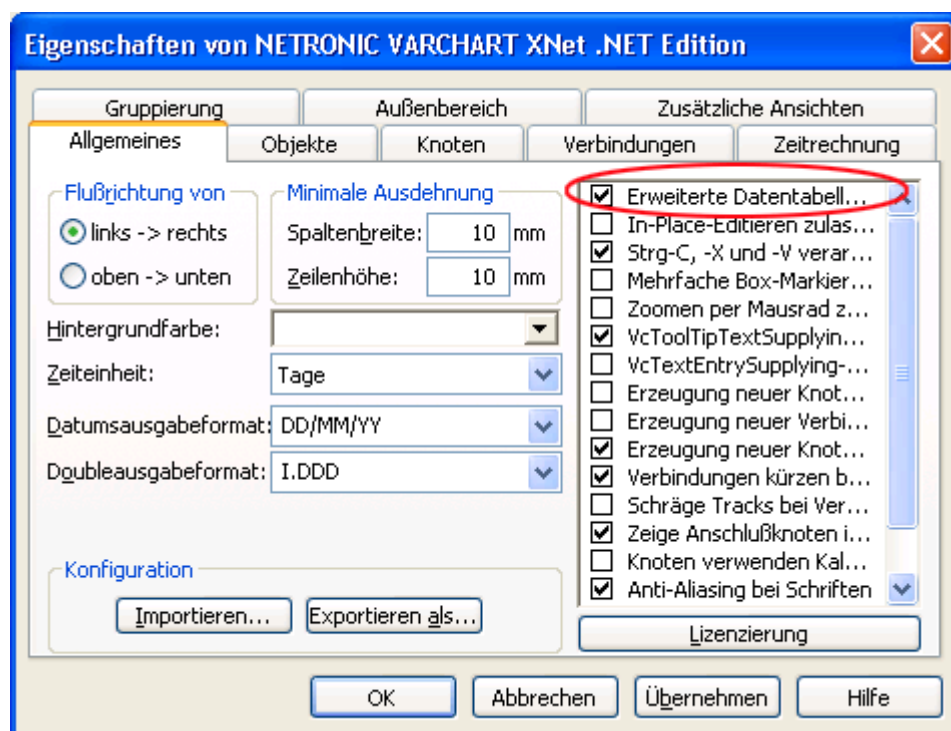
Sie können hier festlegen, ob die Felder der Box durch Linien getrennt werden sollen. Außerdem können Sie für jedes Feld der Box folgendes festlegen:

- Feldtyp (Text oder Grafik)
- Breite und Höhe
- Zeilenzahl
- Ausrichtung
- Füllfarbe und -muster
- Schriftattribute

## 3.2 Datentabellen

Zur Darstellung von Netzdiagrammen verwendet VARCHART XNet zwei Standard-Datentabellen für Knoten und Verbindungen, deren Felder individuell festgelegt werden können. In VARCHART XNet 4.0 wurde dieses Konzept erweitert. Es können jetzt bis zu 90 Datentabellen vereinbart werden und zusätzlich Beziehungen in Form von 1:n Relationen zwischen diesen Tabellen definiert werden. Damit werden in bestimmten Situationen Redundanzen vermieden und der Zugriff auf die Felder des Hauptdatensatzes vom Detaildatensatz aus erleichtert.

Aus Kompatibilitätsgründen mit bestehenden Anwendungen arbeitet VARCHART XNet im bisherigen Modus. Erst durch Aktivieren der entsprechenden Option zur Design- oder Laufzeit können die neuen Möglichkeiten genutzt werden. Auf der Eigenschaftenseite **Allgemeines** finden Sie dazu die Option **Erweiterte Datentabellen zulassen**:



Alternativ können die erweiterten Datentabellen statt zur Designzeit auch zur Laufzeit über die API eingeschaltet werden, indem die Eigenschaft **ExtendendDataTablesEnabled** des Objekts **VcNet** auf **True** gesetzt wird.




### > Datentabellen verwalten

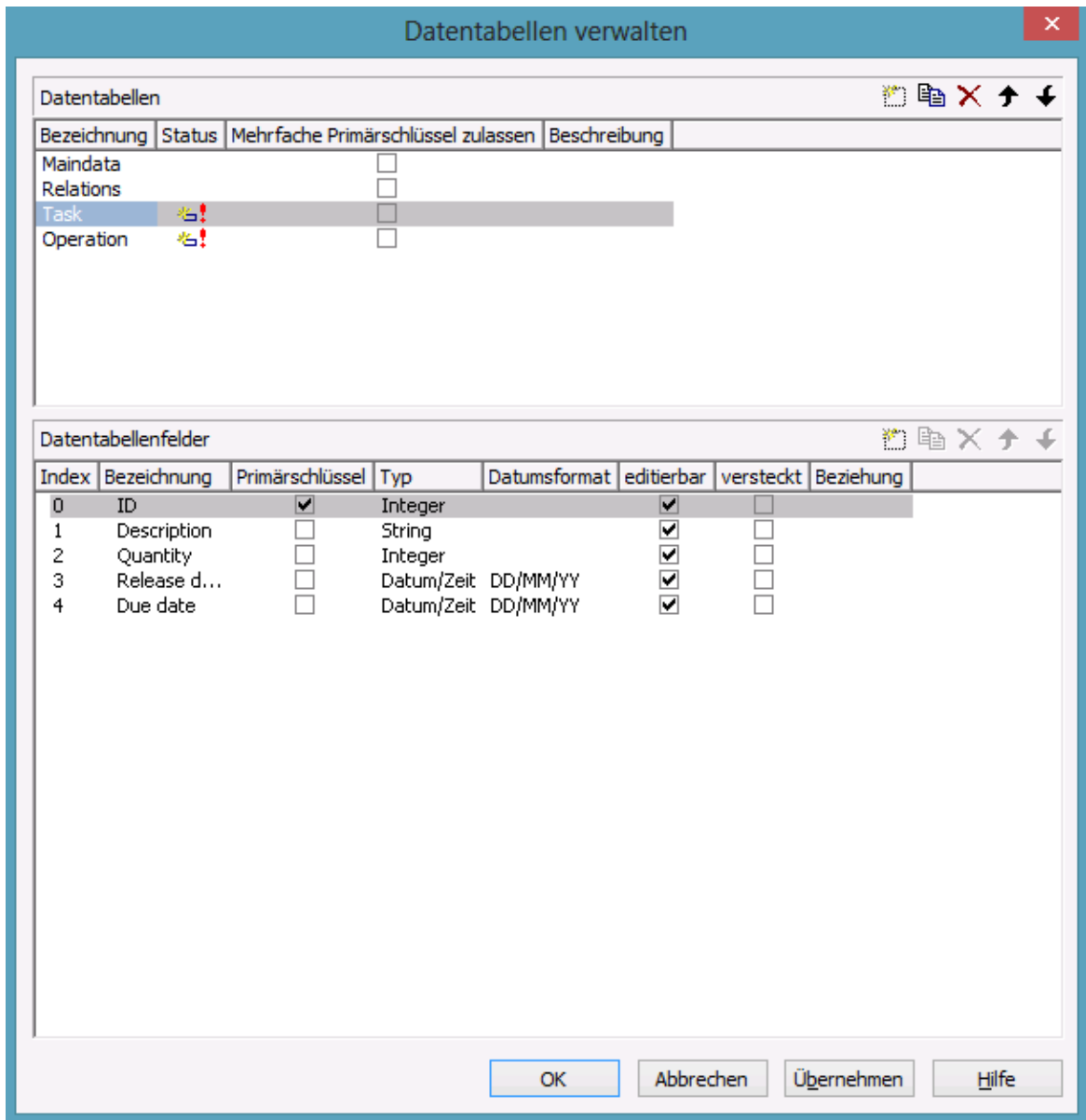
Standardmäßig sind die beiden Datentabellen **Maindata** und **Relations** vorhanden. Auf der Eigenschaftenseite **Objekte** gelangen Sie über die



## 80 Wichtige Konzepte: Datentabellen

Schaltfläche **Datentabellen...** in den Dialog **Datentabellen verwalten**. Nur im Modus **Erweiterte Datentabellen zulassen** ist es möglich, neue Datentabellen anzulegen.

Im Bereich **Datentabellenfelder** können Sie neue Felder anlegen , bestehende Felder löschen  oder Felder kopieren .



Die Spalte **Index** besitzt eine zentrale Bedeutung, denn der Zugriff auf die Inhalte der Datenfelder innerhalb eines Datensatzes geschieht ausschließlich über den Index. Wenn Sie die Reihenfolge der Felder ändern, nachdem bereits Programmcode existiert, dann müssen Sie den Programmcode für den Datenfeldzugriff ebenso anpassen.

Die Änderung des Datentyps eines Feldes kann bewirken, dass bereits definierte Formate und Layer geändert werden müssen, um den Zugriff mit dem richtigen Datentyp sicherzustellen.

Mit der Primärschlüssel-Eigenschaft kennzeichnen Sie das Feld, dessen Werte die Datensätze einer Datentabelle eindeutig unterscheidbar machen. Der Primärschlüssel kann auch aus mehreren Feldern - *jedoch nicht mehr als 3* - bestehen. Eine ausführliche Beschreibung zur Handhabung von zusammengesetzten Primärschlüsseln finden sie im Kapitel **Datentabellen verwalten**.

Für jede Datentabelle, auf die in einer Verknüpfung verwiesen wird, ist die Festlegung eines Primärschlüsselfeldes zwingend erforderlich.

Die Verknüpfung zweier Tabellen ist sinnvoll, wenn eine inhaltliche 1:n Beziehung zwischen den Tabellen besteht und von jedem Detaildatensatz direkt auf ein Datenfeld im Hauptdatensatz Bezug genommen werden soll.

Zwischen zwei Tabellen A und B ist derzeit nur eine einzige 1:n-Beziehung möglich; es kann sich also kein zweites Feld der Tabelle B auf den Primärschlüssel in A beziehen. Allerdings kann sich ein Feld aus einer weiteren Tabellen C auf den Primärschlüssel in A beziehen.

**Hinweis:** Wird eine Datentabelle mit einem zusammengesetzten Primärschlüssel in einer Beziehung verwendet, muss auch die Beziehung entsprechend definiert werden. Andernfalls ist eine eindeutige Anbindung nicht möglich. Ist die Beziehung nicht vollständig definiert - was weder an der API noch im Dialog **Datentabellen verwalten** überprüft wird, findet **keine** Datensatzanbindung statt, was zum Ereignis **VcDataRecordNotFound** führt.

In der VARCHART XNet Version 4.0 sind aufgrund der erweiterten Datentabellen eine Reihe neuer Objekttypen entstanden, die bestehende Objekttypen ablösen werden. Aus Gründen der Kompatibilität sind in dieser Version die bisherigen Objekttypen noch enthalten. Für neue Anwendungen und beim Aktualisieren von bestehenden Anwendungen sollten nur noch die neuen Objekttypen eingesetzt werden.

Bisher	Ab Version 4.0
VcDataDefinition	VcDataTableCollectionVcDataTable
VcDataDefinitionTable	VcDataTableFieldCollection
VcDefinitionField	VcDataTableField
	VcDataRecordCollection
	VcDataRecord

### > Datensätze anlegen und ändern

Nach der Definition der Datentabellenfelder können Sie einer Tabelle über die API Datensätze hinzufügen. Es gibt zwei Wege, auf denen die Datensätze mit Daten gefüllt werden können. Der normale und empfehlenswerte Weg besteht in der Vereinbarung eines Arrays vom Datentyp Object, wobei dessen Elementanzahl auf die Anzahl der Datentabellenfelder gesetzt wird.

#### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection

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

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

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

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

#### Code-Beispiel C#

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

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

VcDataRecord dataRec1;
VcDataRecord dataRec2;

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

Ein neuer Datensatz wird durch die Methode **Add()** des Objekts **DataRecordCollection** erzeugt, der als Parameter das Object-Array mitgegeben wird.

#### Code-Beispiel VB.NET

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

**Code-Beispiel C#**

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

Der zweite Weg besteht in der Verwendung einer Zeichenkette, bei der die Datenwerte durch Semikolon getrennt aneinander gereiht werden.

**Code-Beispiel VB.NET**

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

**Code-Beispiel C#**

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

Wenn ein Semikolon selber im Datenwert enthalten ist, dann muss die Zeichenkette in doppelte Hochkommata eingeschlossen werden.

**Code-Beispiel VB.NET**

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

**Code-Beispiel C#**

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

Der Verweis auf ein Datensatzobjekt kann mit Hilfe des Primärschlüssels über die Methode **DataRecordByID ()** schnell gefunden werden.

**Code-Beispiel VB.NET**

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

**Code-Beispiel C#**

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

Einzelne Datenfeldinhalte eines Datensatzes können Sie leicht über die indizierte Eigenschaft **DataField()** ändern. Mit der Eigenschaft **AllData** lassen sich alle Datenfeldinhalte eines Datensatzes auf einmal ersetzen.

**Code-Beispiel VB.NET**

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

## 84 Wichtige Konzepte: Datentabellen

```
dataRec1.Update()
```

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

### Code-Beispiel C#

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

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

Jede Änderung in einem Datensatz wird erst darstellungswirksam, wenn die Methode **Update()** des Objekts **DataRecord** aufgerufen wird.

Das Auslesen der Werte mit **Alldata** eignet sich für die schnelle Anzeige aller Datenwerte während der Programmentwicklung und zum leichten Übertragen des Datensatzinhalts in den Datensatz einer anderen Tabelle. Ebenso wird dieses Datenformat bei OLE Drag & Drop zum Informationsaustausch benutzt.

### Code-Beispiel VB.NET

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

### Code-Beispiel C#

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

### Tipp:

Um die Lesbarkeit bei den Datenfeldzugriffen zu erhöhen, können Sie globale Konstanten definieren, deren Namen sprechender sind als die Indexpzahlen.

Hier noch einmal das gesamte Programmfragment im Zusammenhang:

### Code-Beispiel VB.NET

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

```

'...

Dim dataRec1 As VcDataRecord
Dim dataRec2 As VcDataRecord

Dim content As String

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

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

ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

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

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

VcNet1.EndLoading()

'...

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

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

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

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

'...

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

```

**Code-Beispiel C#**

```

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

```

## 86 Wichtige Konzepte: Datentabellen

```
//...

VcDataRecord dataRec1;
VcDataRecord dataRec2;

string content;

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

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

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

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

vcNet1.EndLoading();

//...

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

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

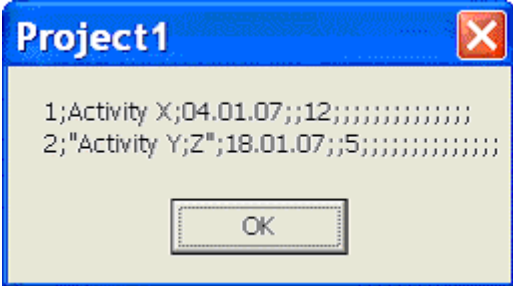
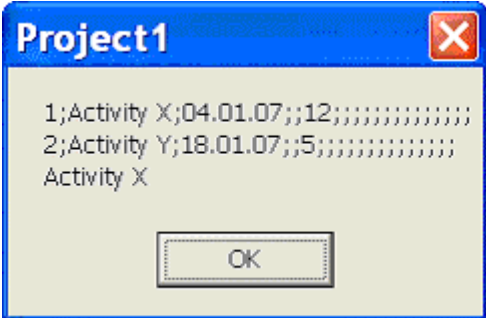
dataRec2.AllData = "2;Activity Y;18.01.13;;5";
dataRec2.Update();

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

//...

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

**Erzeugte Ausgabe:**





### 3.3 Datumsangaben und Zeitumstellung

Datumsangaben in VARCHART Komponenten sind immer bezogen auf die im System eingestellte Zeitzone. Es ist nicht möglich, Datumsangaben aus anderen Zeitzonen anzugeben, d.h. diese müssen vor der Übergabe an die Komponente für die eingestellte Zeitzone umgerechnet werden. Dabei beachtet die VARCHART-Komponente automatisch die im System vorhandenen Informationen zu Beginn und Ende der Sommerzeit.

Damit eine VARCHART-Komponente die Umschaltzeitpunkte vom System mitgeteilt bekommt, ist es wichtig, dass die Option **Uhr automatisch auf Sommer-/Winterzeit umstellen** gesetzt ist, wie im Bild gezeigt. Der Dialog ist im Windows-Betriebssystem unter "Start" > "Einstellungen" > "Systemsteuerung" > "Datum und Uhrzeit" über einen Doppelklick auf die Uhrzeit in der Task-Leiste erreichbar.



Eine VARCHART-Komponente nimmt bei der Umschaltung den Startzeitpunkt, den Endzeitpunkt inkl. Stunde, Monat und Tag, den das System als Regel mitteilt. Das heißt aber auch, dass die Umschaltzeiten für die Jahre vor und nach dem aktuellen Jahr extrapoliert werden, sodass etwaige Abweichungen, die für vorherige Jahre galten bzw. für kommende Jahre gelten werden, nicht berücksichtigt werden können, weil sie dem Betriebssystem ebenfalls nicht bekannt sind. Beispielsweise wurde die Sommerzeit in den USA vor wenigen Jahren um Wochen am Beginn und Ende verlängert. Da dem System hier nur die aktuelle Regelung bekannt ist,

werden Datumsangaben in der betroffenen Vergangenheit hier systembedingt falsch interpretiert.

Augenblicklich können `VARCHART` Komponenten bei der Sommerzeit nur eine Zeitdifferenz zur Winterzeit von exakt einer Stunde berücksichtigen. Außerdem darf die Umschaltung nur zur vollen Stunde erfolgen.

Da die `VARCHART`-Komponenten die Datumsangaben immer in lokaler Zeit entgegennimmt und ausgibt, gibt es am Beginn der Sommerzeit eine nicht erlaubte Stunde und am Ende der Sommerzeit zwei Stunden mit derselben Zahl, die derzeit bei Übergabe, bei Rückgabe und bei Ausgabe nicht unterschieden werden kann.

---

## 3.4 Drag & Drop

Neben dem Verschieben oder Kopieren von Vorgängen innerhalb einer Instanz der VARCHART XNet Komponente kann ein Benutzer Vorgänge auch über die Grenzen einer Instanz (Quellkomponente) hinaus in eine andere Instanz (Zielkomponente) bewegen. In diesem Kapitel werden Themen behandelt, die vom Programmierer für diesen Interaktionstyp von Bedeutung sind.

Während sich bei einer Bewegung von Knoten innerhalb einer Instanz die Daten (Termine) der Knoten ändern, erfolgt beim Verschieben zwischen zwei Instanzen keine Datenänderung (welche aber dann in einem zweiten Bewegungsprozess innerhalb der zweiten Instanz möglich wäre).

Die Bewegung eines Knotens teilt sich in zwei Teilbewegungen: zum einen die Bewegung aus der Quellkomponente hinaus und zum anderen die Bewegung in die Zielkomponente hinein. Beide Teilbewegungen müssen von der jeweiligen Komponente erlaubt werden.

XNet ermöglicht es, einzelne oder mehrere Vorgänge gleichzeitig zu verschieben oder zu kopieren. Wird die linke Maustaste auf einem Vorgang gedrückt, wird intern ein Objekt des Typs **System.Windows.Forms.DataObject** angelegt und mit den Vorgangsdaten im CSV-Format (d.h. Text bzw. Typ **System.String**) gefüllt. Anschließend wird direkt das Ereignis **VcDragStarting** ausgelöst. Damit kann die Anwendung die erlaubten Aktionen (Kopieren und/oder Verschieben) selbst bestimmen. Als Vorgabe ist beides, sowohl Kopieren als auch Verschieben, je nach Status der <Strg>-Taste möglich: wird sie beim Loslassen der Maustaste gedrückt, wird kopiert, ansonsten verschoben.

Abschließend wird direkt das Ereignis **VcDragCompleting** ausgelöst, um der Anwendung zu erlauben, die durchgeführte Aktion (Kopieren, Verschieben oder Abbruch) zu erfahren und evtl. darauf zu reagieren.

Anschließend werden auf der Quellkomponente die Ereignisse **Control.GiveFeedback** und **Control.QueryContinueDrag** ausgelöst. Auf der Zielkomponente werden die Ereignisse **Control.DragEnter**, **Control.DragOver** und **Control.DragLeave** ausgelöst.

Die Beschreibung des .NET-Mechanismus für Drag&Drop entnehmen Sie bitte der Beschreibung des .NET-Frameworks. Zusätzlich gibt es fünf Eigenschaften, die das Verhalten des Drag&Drop beeinflussen:

> **Control.AllowDrop**

Mit dieser Boole'schen Eigenschaft der Basisklasse **Control** bestimmen Sie, ob Objekte, die über das Steuerelement gezogen wurden, fallen gelassen werden dürfen. Vorgänge, die innerhalb eines VARCHART-Steuerelements verschoben oder kopiert werden, sind hiervon ausgenommen (d.h. sie dürfen immer fallen gelassen werden).

> **VcNet.LeavingControlWhileDraggingAllowed**

Mit dieser Boole'schen Eigenschaft des VcNet-Objektes können Sie bestimmen, ob ein Ziehen eines oder mehrerer Vorgänge über die Grenzen der Quellkomponente hinaus erlaubt sein soll. Damit können Sie z.B. einen Knoten von einem zu einem andern VARCHART-Steuerelement oder in andere Steuerelemente der gleichen Anwendung oder sogar in andere Anwendungen verschieben oder kopieren.

> **VcNet.NodeCreationAtDroppingEnabled**

Mit dieser Boole'schen Eigenschaft des VcNet-Objektes können Sie festlegen, ob beim Fallenlassen eines gezogenen Objekts in der Zielkomponente automatisch ein Vorgang im VARCHART-Steuerelement erzeugt werden soll.

> **VcNet.PhantomDrawingWhileDraggingEnabled**

Mit dieser Boole'schen Eigenschaft des VcNet-Objektes können Sie in der Zielkomponente festlegen, ob beim Ziehen das für das VARCHART-Steuerelement übliche Phantom angezeigt werden soll.

> **VcNet.InbuiltMouseCursorWhileDraggingEnabled**

Mit dieser Boole'schen Eigenschaft können Sie in der Zielkomponente festlegen, ob beim Ziehen die für das VARCHART-Steuerelement charakteristischen, eingebauten Mauszeiger angezeigt werden sollen oder der für das Drag&Drop übliche Mauszeiger (Pfeil mit Rechteck bzw. Verbotssymbol), oder gar anwendungseigene.

---

## 3.5 Ereignisse

Über Ereignisse werden die Interaktionen des Anwenders vom VARCHART Steuerelement an die Applikation weitergegeben. Immer wenn der Anwender mit dem VARCHART Steuerelement interagiert, indem er beispielsweise Daten ändert oder auf irgendeine Stelle des Steuerelements klickt, wird ein entsprechendes Ereignis gemeldet. Sie können in Ihrem Programmcode die Ereignisse auffangen und mit Ihrer Anwendung beliebig darauf reagieren.

In jeder Entwicklungsumgebung werden für die Ereignisse entsprechende Funktionen bereitgestellt, in denen die vom VARCHART-Windows-Forms-Steuerelement zur Verfügung gestellten Parameter schon eingetragen sind. Die einzelnen Ereignisse sind in der API-Referenz ausführlich beschrieben. Hier sei nur kurz darauf hingewiesen, dass Sie unter Verwendung des Parameters **returnStatus** in den Ereignissen alle in VARCHART Steuerelement verfügbaren Kontextmenüs abschalten (und ggf. durch eigene ersetzen) und alle Interaktionen kontrollieren und bei Bedarf unterbinden bzw. rückgängig machen können.

### > Rückgabewerte

Die folgende Tabelle enthält die Rückgabewerte für VARCHART-Ereignisse:

Konstante	Wert	Beschreibung
vcRetStatDefault	2	standardmäßige Vorbesetzung
vcRetStatFalse	0	Abbrechen der Aktion
vcRetStatNoPopup	4	Kontextmenü erscheint nicht

## 3.6 Filter

Ein Filter enthält Bedingungen für Knoten oder Verbindungen. Mit Hilfe eines Filters können Sie alle Knoten bzw. Verbindungen, die die Filterbedingungen erfüllen, auswählen, um sie beispielsweise grafisch hervorzuheben.

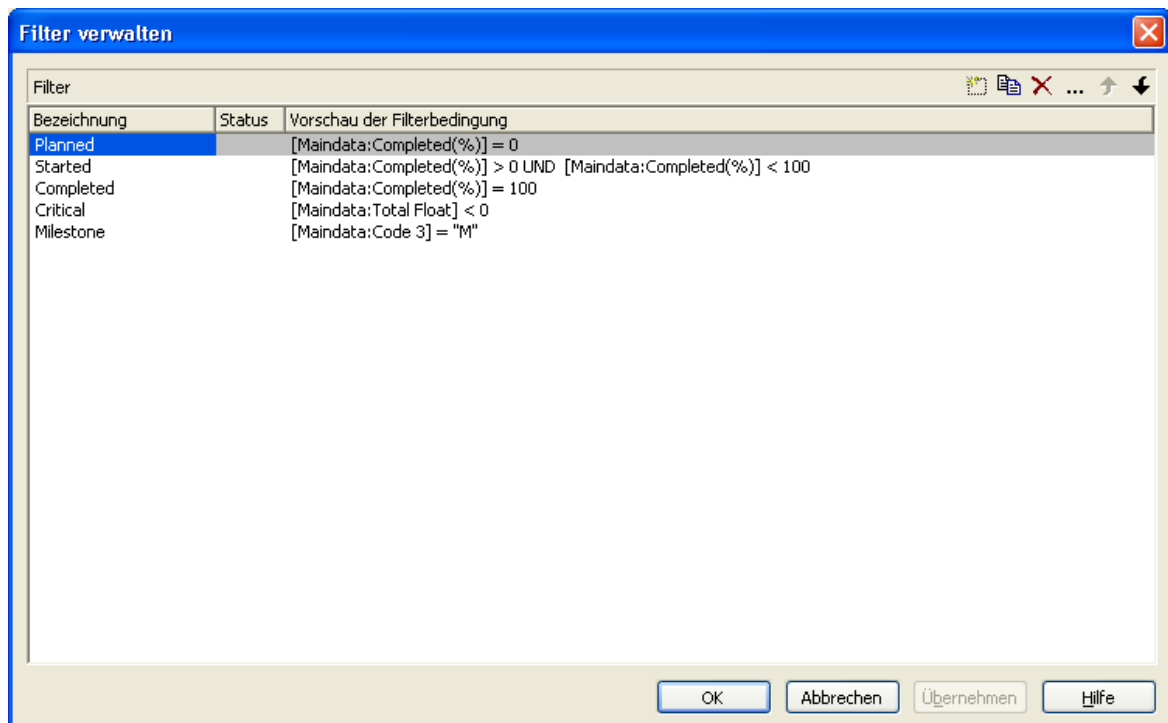
Wenn Sie einen Filter anwenden, werden die Informationen des Datensatzes eines Knotens bzw. einer Verbindung mit den Kriterien aus dem Filter verglichen. Es werden die Knoten bzw. Verbindungen ausgewählt, die die Filterkriterien erfüllen.

Beispielsweise können Sie den Filter "Alle Knoten mit Beginn ab Januar 2001" definieren, um alle Knoten auszuwählen, die ab Januar 2001 beginnen.

Filter können nur im Design-Modus erzeugt, bearbeitet und verwaltet werden.

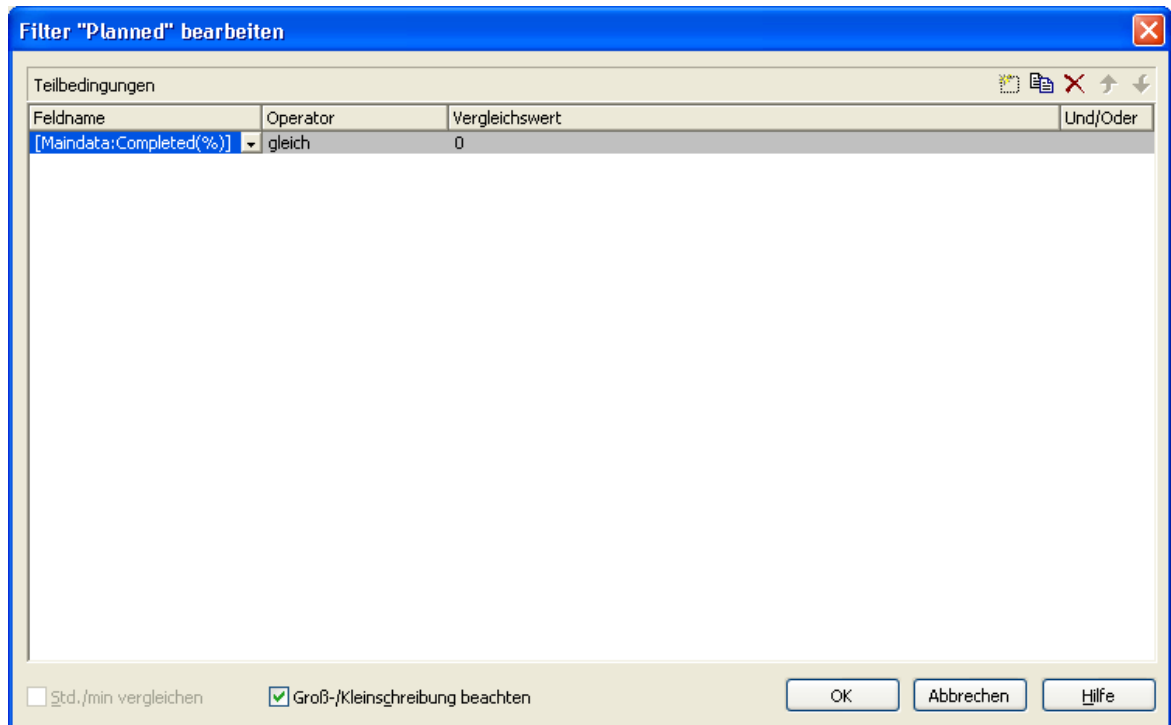
Sie erreichen das Dialogfeld **Filter verwalten** über die Eigenschaftenseite **Objekte**. Das Dialogfeld **Filter verwalten** für Verbindungen erreichen Sie außerdem über die Eigenschaftenseite **Verbindungen**.

Im Dialogfeld **Filter verwalten** können Sie Filter umbenennen, kopieren, löschen, bearbeiten oder neu definieren.



Einen vorhandenen Filter bearbeiten können Sie im Dialogfeld **Filter bearbeiten**, das Sie vom Dialogfeld **Filter verwalten** aus erreichen.

## 94 Wichtige Konzepte: Filter



---

## 3.7 Grafikformate

VARCHART unterstützt die folgenden Grafikformate, was für den Export von Grafiken für Bedeutung ist, vor allem für die Aufrufe **VcNet1.ShowGraphicsExportDialog** und **VcNet1.ExportGraphics**.

Das XNet-Steuerelement unterstützt sowohl den Import von Grafikdateien z.B. für die Darstellung in Knoten oder in Boxen wie auch den Export ganzer Charts in Grafikdateien. Dabei spielen die verschiedenen unterstützten Grafikformate eine nicht ganz unwichtige Rolle in Bezug auf die Qualität der Darstellung der Grafik nach dem Import im Steuerelement bzw. nach dem Export in einem externen Anzeigeprogramm. Im folgenden werden die einzelnen Grafikformate mit ihren Vorzügen und Beschränkungen kurz vorgestellt. Dabei sind grundsätzlich zwei Typen zu unterscheiden:

**Vektorgrafikformate** speichern einzelne geometrische Figuren wie Linien, Ellipsen oder Rechtecke als Beschreibung der Figur mit darauf bezogenen Parametern wie Startkoordinaten, Ausdehnung und Farbe. Sie sind damit auflösungsunabhängig, d.h. bei stärkerem Hineinzoomen werden Linien weiterhin sauber dargestellt. Eine Einschränkung gibt es hier höchstens bei der Größe des zur Verfügung stehenden Koordinatenraums insbesondere beim WMF-Format. Allgemein haben Vektorgrafikformate also einen großen Vorteil durch die Auflösungsunabhängigkeit und oft auch bei der sich ergebenden Dateigröße. Leider hat sich kein plattformunabhängiges, genormtes Format durchgesetzt.

**Bitmapgrafikformate** speichern alle Bildpunkte mit ihrer Farbe in einer vorgegebenen Ausdehnung. Beim stärkeren Hineinzoomen werden die Grafiken dann automatisch "pixelig". Um einer ausufernden Dateigröße entgegenzuwirken, werden Bitmapgrafiken bei vielen Formaten verlustfrei oder gar verlustbehaftet komprimiert. Ein Verlust ist aber nur bei Fotos und nicht bei Diagrammen hinnehmbar. Ein Vorteil haben Bitmapgrafikformate nur dadurch, dass sie sich über Digitalkameras und das Internet durchgesetzt und plattformunabhängig weitverbreitet sind.

### > **WMF (Windows Metafile Format)**

Dieses Vektorgrafik-Format existiert seit Windows 3.0 und besteht intern aus Befehlsdatensätzen, die den GDI-Befehlen der Windows-API entsprechen. Hiermit können GDI-Befehle sozusagen persistent gemacht werden. Dieses Format war aber schon zu Zeiten seiner Entwicklung nicht vollständig: So besaß und besitzt es bis heute nur einen eingeschränkten Koordinatenraum. Außerdem fehlt das Clipping, die Koordinatentransformation und das Füllen komplexer Polygone. Das Problem, nicht die tatsächliche Ausdehnung einer



WMF-Datei in "echten" Koordinaten wie cm oder Zoll festlegen zu können, wurde schon früh durch die Firma Aldus angegangen, die den sogenannten "Aldus Placeable Header" entwickelte, der seit langem in praktisch allen Programmen zur Anzeige von WMF-Dateien erkannt und genutzt wird mit Ausnahme der Windows-API selbst, die bis heute diesen Header nicht erzeugen oder verarbeiten kann, obwohl er in der Microsoft-Dokumentation erwähnt und erklärt wird.

Mit Windows NT und 95 wurde das Format von Microsoft eigentlich "in Rente" geschickt und sein Nachfolger namens EMF eingeführt. Trotzdem erfreut sich WMF bis heute einiger Beliebtheit vornehmlich im Bereich von ClipArt-Grafiken, wo die erweiterten Möglichkeiten des Nachfolgeformats nicht die Rolle spielen. Neuere Entwicklungen in Windows 95 und NT und deren Nachfolgern flossen nicht mehr in das Format ein; es ist seitdem unverändert geblieben.

WMF kennt auch einen Kommentardatensatz, der dazu genutzt werden kann, dort EMF-Befehle unterzubringen. Wenn ein Anzeigeprogramm solche Kommentare entdeckt, also auch EMF-Dateien anzeigen kann, dann verwirft es automatisch die WMF-Befehlsdatensätze und zeigt die EMF-Befehlsdatensätze. So kann eine einzige Datei WMF- wie auch EMF-Grafik enthalten. Dies ist wohl aus Kompatibilitätsgründen eingebaut worden, bläht aber die Dateigröße stark auf.

Formatbeschreibung siehe:

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

Beschränkungen des Formats siehe:

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

### > **EMF (Enhanced Metafile Format)**

Dieses Vektorgrafik-Format wurde mit den 32bit-Betriebssystemen Windows NT und 95 eingeführt und behebt die Einschränkungen des WMF-Formats und besteht intern aus Grafikbefehlen, die den GDI32-Befehlen Windows-API entsprechen. Der Koordinatenraum ist nun also 32bittig, Transformationen und Clipping werden unterstützt. Die später ins GDI32 hinzu genommenen Befehle zum maskierten oder mit AlphaBlending versehenem Blenden von Speicher-Bitmaps werden aber nicht unterstützt.

Das Format ist bis heute trotz der Vorteile, die es gegenüber WMF bietet, recht unbekannt geblieben. Alle Anzeigeprogramme und Office-Pakete können allerdings mit EMF-Dateien umgehen.

Ein Nachteil bei der Verwendung von GDI+ ergibt sich dergestalt, dass die von GDI+ neu eingeführten Möglichkeiten der grafischen Gestaltung wie

Farbverläufen und Transparenzen nicht voll unterstützt werden und außerdem beim Export in eine EMF-Datei unterbrochene Linien (gestrichelt u.ä.) in einzelne kleine nicht unterbrochene Linien gespeichert werden, wodurch zum einen der Speicherbedarf stark ansteigt und zum anderen eine Anzeige einer so geschriebenen Datei sehr lange Zeit benötigt.

EMF kennt auch einen Kommentardatensatz, der dazu genutzt werden kann, dort EMF+-Befehle unterzubringen. Wenn ein Anzeigeprogramm solche Kommentare entdeckt, also auch EMF+-Dateien anzeigen kann, dann verwirft es automatisch die EMF-Befehlsdatensätze und zeigt die EMF+-Befehlsdatensätze. So kann eine einzige Datei EMF- wie auch EMF+-Grafik enthalten. Dies ist wohl aus Kompatibilitätsgründen eingebaut worden, bläht aber die Dateigröße stark auf.

Druckjobs werden Windows-intern übrigens auch als EMF-Datenstrom ggf. zwischengespeichert und an den Druckertreiber übergeben.

Formatbeschreibung siehe:

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

### > **EMF+ (Enhanced Metafile Format Plus)**

Obwohl der Name nahelegt, dass es sich um eine Erweiterung des EMF handelt, ist dies ein eigenes Vektorgrafikformat, das mit der Vorstellung der GDI+-Windows-API eingeführt wurde und intern aus Grafikbefehlsdatensätzen besteht, die den GDI+-Befehlen entspricht (GDI+ ist übrigens auch keine Erweiterung von der GDI-API, sondern eine eigene Grafikbibliothek). Zusätzlich zu EMF werden hier Transparenzen und Farbverläufe voll unterstützt.

Das Format ist bis heute recht unbekannt geblieben und wird auch von üblichen Anzeigeprogrammen oft nicht unterstützt, außer in Microsoft Office ab 2003. Microsoft hat erst 2003 den Aufbau des EMF+-Dateiformats veröffentlicht.

Formatbeschreibung siehe:

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

### > **GIF (Graphics Interchange Format)**

Dieses Bitmap-Format wurde von CompuServe in Zeiten vor dem Entstehen des World Wide Web zur verlustfreien, komprimierten Speicherung von Grafiken entwickelt und kann nur gleichzeitig 256 Farben darstellen. Daher kann es die heutigen Grafiken nicht zufriedenstellend speichern. Es wird nur noch aus Kompatibilitätsgründen angeboten.

Die Unterart "Animated GIF" wird überhaupt nicht unterstützt.

### > **JPEG (Joint Photographic Experts Group)**

Dieses Bitmap-Format wurde von der JPEG zur verlustbehafteten, komprimierten Speicherung von Fotos entwickelt. Daher ist eine Speicherung von Diagrammen, bei denen es z.B. auf die saubere Speicherung von Linien ankommt, nicht sinnvoll. Es wird nur noch aus Kompatibilitätsgründen angeboten.

### > **BMP (Windows Bitmap)**

Dieses Bitmap-Format wurde von Microsoft zur verlustfreien, unkomprimierten Speicherung von Grafiken entwickelt. Das Format wird auch intern im Speicher von der Windows-API GDI direkt verwendet. Eine einzige Einschränkung gibt es dadurch, dass es keinen Alphakanal unterstützt, d.h. es können maximal 24 Bits pro Pixel gespeichert werden. Aufgrund seines hohen Speicherbedarfs sollte auf dieses Format verzichtet werden. Es wird nur noch aus Kompatibilitätsgründen angeboten.

### > **TIFF (Tagged Image File Format)**

Dieses Bitmap-Format wurde von Aldus (in Adobe aufgegangen) zur Speicherung von Grafiken entwickelt. Die Grafik kann darin verlustbehaftet oder verlustfrei gespeichert werden. Das Format wird seit längerem nicht mehr weiterentwickelt. Es wird nur noch aus Kompatibilitätsgründen angeboten.

### > **PNG (Portable Network Graphics)**

Dieses Bitmap-Format wurde vom World Wide Web Consortium (W3C) zur verlustfreien, komprimierten Speicherung von Grafiken entwickelt, um das vormals Copyright-behaftete und beschränkte GIF Format abzulösen. PNG ist hervorragend geeignet zur Speicherung von VARCHART-Grafiken und beim Einlesen werden transparente Anteile auch transparent gezeichnet. Es wird auch durchgängig von praktisch allen Anzeigeprogrammen und Internetbrowsern unterstützt. Das Format selbst ist frei von Copyrights und vollständig dokumentiert.

Seit der Version 4.2 wird für den Export die frei verfügbare Bibliothek **libpng** verwendet, um durch die Vorgabe der Auflösung beliebig große Bitmaps speichern zu können. Hierbei muss aber darauf geachtet werden, dass sehr große PNG-Dateien zu Schwierigkeiten beim Einlesen führen können, denn üblicherweise wird die PNG-Datei im Speicher erst komplett entpackt und dann dargestellt.

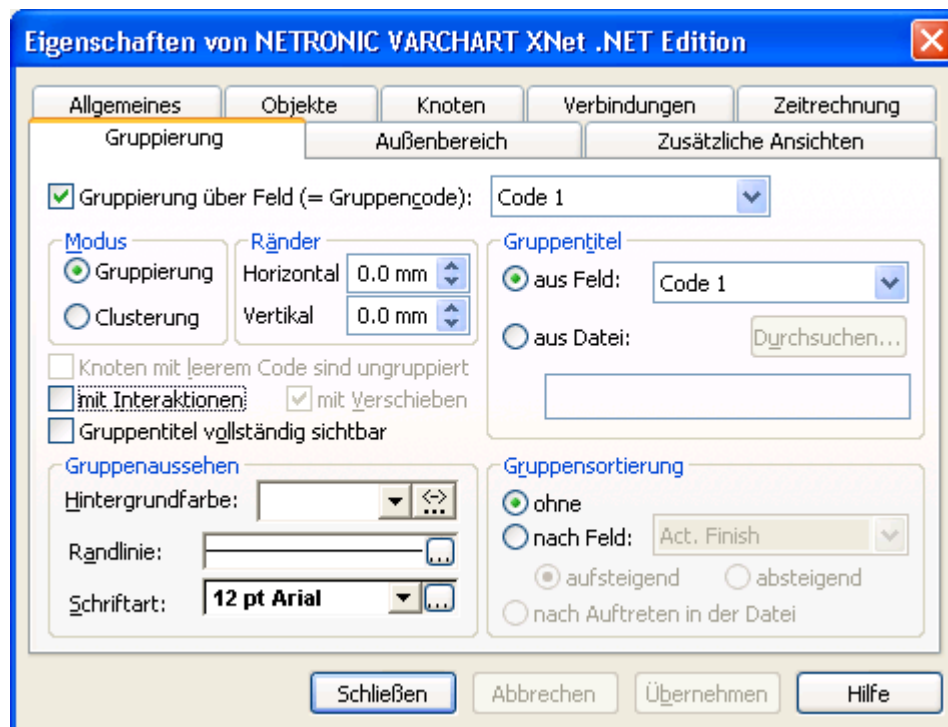
Formatbeschreibung siehe:

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

## 3.8 Gruppierung

In vielen Anwendungsfällen ist es gefordert, Knoten in Gruppen einzuteilen und diese Gruppen in der Darstellung besonders hervorzuheben. Beispielsweise werden Knoten häufig nach bestimmten Projektphasen gruppiert (z. B. Planung, Konstruktion, Fertigung, etc.) oder nach bestimmten Abteilungen (Abteilung Konstruktion, Abteilung Rechnungswesen, etc.).

Die Kriterien für die Gruppierung können Sie auf der Eigenschaftenseite **Gruppierung** vornehmen.



Die Knoten werden nach dem Feld gruppiert, das Sie aus der Kombobox **Gruppierung über Feld (= Gruppencode)** auswählen. Legen Sie hier fest, nach welchem Feld gruppiert werden soll. Das Feld, das Sie hier auswählen, wird **Gruppencode** genannt. Alle Knoten, die in dem hier gewählten Feld denselben Eintrag haben, werden in einer Gruppe zusammengefasst.

Wenn der Anwender einen Knoten von einer Gruppe in eine andere schiebt, wird der Wert in dem Datenfeld, das als Gruppencode vereinbart wurde, automatisch angepasst.

Sie können wählen, ob die Gruppentitel aus einem Datenfeld oder aus einer Datei genommen werden sollen.

- Aktivieren Sie das Kontrollkästchen **aus Feld**, damit die Gruppentitel aus dem Datenfeld genommen werden, das Sie hier auswählen. Dieses Feld muss nicht notwendigerweise mit dem Gruppencode identisch sein. Doch damit die Gruppen sinnvoll beschriftet werden, sollten die Einträge im

**Gruppencode-Feld** und im **Gruppentitel-Feld** miteinander korrespondieren.

- Wenn Sie das Kontrollkästchen **aus Datei** aktivieren, werden die Gruppentitel aus der Datei übernommen, die Sie hier auswählen. Der Aufbau einer Gruppentitel-Datei erfolgt nach folgendem Schema:

A Gruppe A  
 B Gruppe B  
 C Gruppe C

> **Beispiel:**

"A" = *Kurztext*

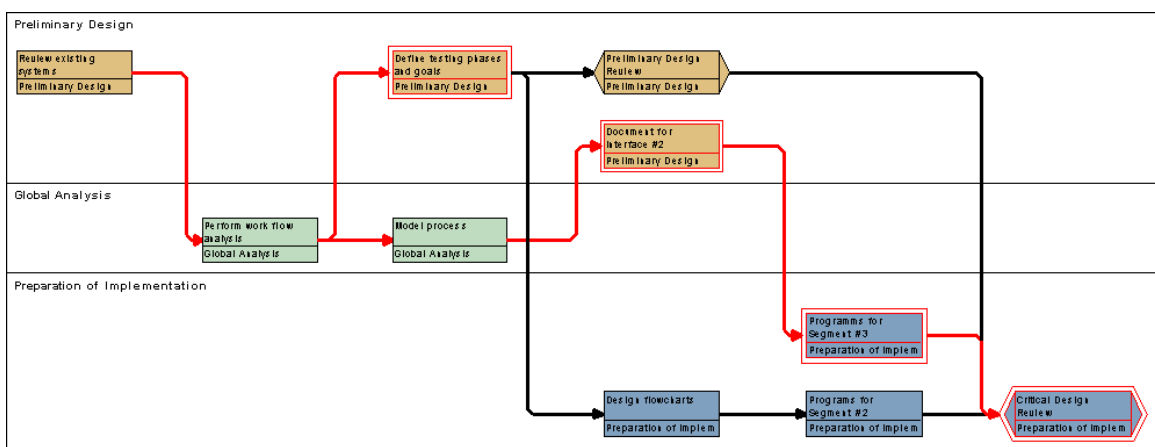
*Leerzeichen = Trennzeichen*

"Gruppe A" = *Langtext*

> **Darstellungsmodi: Gruppierung und Clusterung**

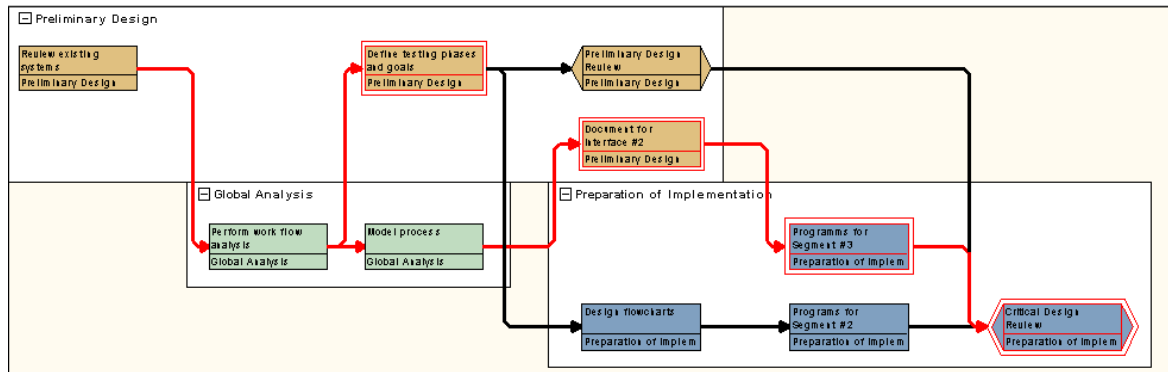
Sie können zwischen zwei Darstellungsmodi wählen:

- **Gruppierung:** normale Gruppendarstellung (die Breite und Höhe jeder Gruppe wird durch die Positionen der Knoten bestimmt; jede Gruppe nimmt jeweils die gesamte Breite bzw. Höhe des Netzdiagramms ein)
- **Clusterung:** Die Gruppen umschließen die darin enthaltenen Knoten möglichst platzsparend, wobei die Gruppen frei im Netzdiagramm verteilt sind. Im Cluster-Modus können die Gruppen durch Klick auf das Plus- bzw. Minus-Symbol kollabiert und expandiert werden (wenn dies über die Eigenschaft `VcNet.GroupInteractionsAllowed` eingeschaltet ist). Kollabierete Gruppen können mit der Maus verschoben werden wie Knoten.



*Beispiel für Gruppierungsmodus*

## 102 Wichtige Konzepte: Gruppierung



### Beispiel für Cluster-Modus

In beiden Modi können Sie interaktiv Knoten verschieben, löschen oder neu erzeugen.

Den Gruppierungsmodus können Sie auf der Eigenschaftenseite **Gruppierung** unter **Modus** auswählen oder über die API mittels **VcNet.GroupingType** setzen.

#### > Sortierung der Gruppen

Sie können die Gruppen sortieren lassen: entweder **aufsteigend** oder **absteigend** nach einem Kriterium Ihrer Wahl (**nach Feld**) oder nach dem Auftreten der Gruppen in der Datei.

#### > Aussehen der Gruppen

Auch das Aussehen der Gruppen können Sie frei gestalten, indem Sie die Trennlinien zwischen den Gruppen (**Randlinie**), den **Hintergrund** der Gruppen und die **Schriftart** der Gruppentitel festlegen.

#### > Ereignisse

Auf folgende Ereignisse können Sie reagieren:

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

### 3.9 In-Flow-Gruppierung

Bei einer In-Flow-Gruppierung werden die Knoten eines Netz-Diagramms nach einer bestimmten Reihenfolge angeordnet. Die Reihenfolge kann zeitlich oder durch einen bestimmten Code (ein bestimmtes Datenfeld) bestimmt sein.

Bei einer zeitorientierten Anordnung (In-Flow-Gruppierung nach einem Terminfeld) werden Knoten nach ihrer zeitlichen Reihenfolge in x-Richtung (bei Orientierung des Diagramms von links nach rechts) bzw. in y-Richtung (bei Orientierung des Diagramms von oben nach unten) angeordnet und in Zeitintervalle eingeordnet. Die Länge dieser Intervalle können Sie festlegen.

13.01.2013	27.01.2013	10.02.2013
13.01.2013	27.01.2013	10.02.2013

*Beispiel für eine zeitliche Gruppierung bei Links-Rechts-Orientierung*

13.01.2013		13.01.2013
27.01.2013		27.01.2013

*Beispiel für eine zeitliche Gruppierung bei Oben-Unten-Orientierung*

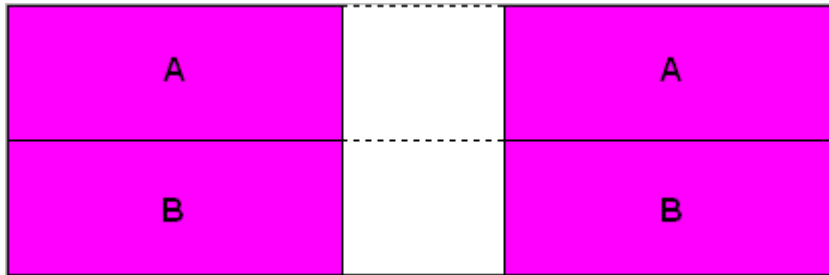
Analog ist eine In-Flow-Gruppierung nach bestimmten Datenfeldern möglich.

A	B	C
A	B	C

*Beispiel für eine Gruppierung nach einem bestimmten Datenfeld bei Links-Rechts-Orientierung*



## 104 Wichtige Konzepte: In-Flow-Gruppierung



*Beispiel für eine Gruppierung nach einem bestimmten Datenfeld bei Oben-Unten-Orientierung*

Im Dialog **In-Flow-Gruppierung bearbeiten** können Sie die Kriterien und das Layout der In-Flow-Gruppierung bearbeiten.

**In-Flow-Gruppierung bearbeiten**

Code aus Feld:   Trennlinien:

Zeitintervall:

**Titelleisten**

oben  unten

Schrift:

Hintergrundfarbe:

Breite:

**Datumsformat**

**Texte**

aus Feld:

aus Datei:

**Vorschau**

04.06.2007	18.06.2007	02.0
04.06.2007	18.06.2007	02.0

Sie erreichen diesen Dialog über die Eigenschaftenseite **Knoten**.

Bei einer Orientierung des Diagramms von links nach rechts können Sie oberhalb und/oder unterhalb des Diagrammbereichs Titelleisten ausgeben. Bei einer Orientierung von oben nach unten können Sie links und/oder rechts des Diagrammbereichs Titelleisten ausgeben. Außerdem können Sie festlegen, ob vertikale bzw. horizontale Trennlinien die Gruppengrenzen darstellen sollen. Die Farbe und die Beschriftung der Titelleisten sowie das Aussehen der Trennlinien lassen sich individuell gestalten.

---

## 3.10 Knoten

Ein Knoten entspricht einem Datensatz aus der Maindata-Tabelle. Knoten können über die API geladen oder interaktiv vom Anwender erzeugt werden.

### > Knoten erzeugen

Wenn auf der Eigenschaftenseite **Allgemeines** die Option **Erzeugung neuer Knoten und Verbindungen zulassen** gewählt ist, kann der Anwender im Erzeugemodus neue Knoten per Mausklick erzeugen.

Ist dort außerdem das Kontrollkästchen **Erzeugung neuer Knoten mit Dialog** aktiviert, öffnet sich das Dialogfeld **Vorgänge bearbeiten**, sobald ein Knoten durch Mausklick erzeugt wurde. Hier werden die Daten des interaktiv erzeugten neuen Knotens angezeigt, und Sie können sie nun bearbeiten.

Sie können Knoten auch über die API mit **InsertNodeRecord** anlegen.

Jedes interaktive Neuanlegen eines Knotens wird der Applikation mit dem Ereignis **VcNodeCreating** mitgeteilt.

### > Markieren von Knoten

Auf der Eigenschaftenseite **Knoten** können Sie unter **Markierungstyp** festlegen, wie und in welcher Farbe markierte Knoten dargestellt werden sollen.

Mögliche Alternativen sind:

- Ohne
- Einrahmen
- Einrahmen innen
- Invertieren
- Pickmarks
- Pickmarks innen

**Hinweis:** Wenn Sie den Markierungstyp "Ohne" ausgewählt haben, können Knoten nicht markiert werden.

Jedes Markieren/Demarkieren von Knoten wird der Applikation mit dem Ereignis **VcNodesMarking** mitgeteilt. Das Ende einer Markier-/ Demarkier-operation wird durch das Ereignis **VcNodesMarked** mitgeteilt.

### > Knoten löschen

Einen oder mehrere Knoten können Sie löschen, indem Sie sie bei gedrückter Umschalt- oder Strg-Taste mit der linken Maustaste markieren und dann die

rechte Maustaste drücken. Wählen Sie dann im Kontextmenü für Knoten den Befehl **Löschen** bzw. **Ausschneiden**.

Markierte Knoten können auch mit der **Entf**-Taste gelöscht werden.

Das interaktive Löschen eines Knotens löst das Ereignis **VcNodeDeleting** aus.

Außerdem können Sie Knoten über die API mit der VARCHART-ActiveX-Methode **DeleteNodeRecord** oder mit der VcNode-Methode **DeleteNode** löschen.

### > **Ereignisse**

Auf folgende Ereignisse können Sie reagieren:

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

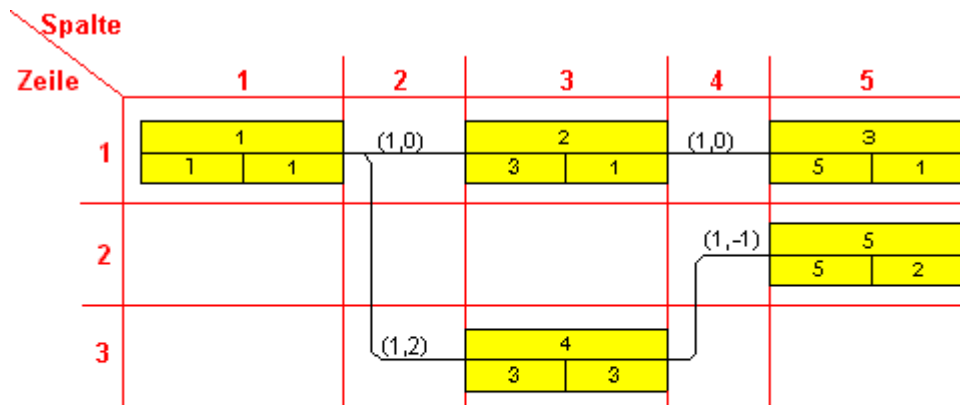
### > **Position von Knoten**

Die Positionen von Knoten und Verbindungsbeschriftungen im Diagramm werden als deren Koordinaten in einem Raster festgelegt.

Die X- und Y-Koordinate eines Knotens geben die absolute Position dieses Knotens im Raster an.

Die X- und Y-Koordinate einer Verbindungsbeschriftung geben die Position dieser Verbindungsbeschriftung relativ zu ihrem Vorgängerknoten an.

Das erste Feld des Rasters (ganz links oben) ist definiert als  $(X,Y) = (1,1)$ . Es ist für Knoten reserviert. Von links nach rechts und von oben nach unten wird jeweils in Schritten von 1 weitergezählt. In allen weiteren Feldern können Knoten oder Verbindungsbeschriftungen stehen. Knoten haben daher immer positive X- und Y-Koordinaten. Bei Verbindungsbeschriftungen sind alle Wertepaare außer (0,0) möglich.



**Legende:**

Nummer		Knotenfeld
X-Position	Y-Position	

(x,y) Verbindungsbeschriftungsposition

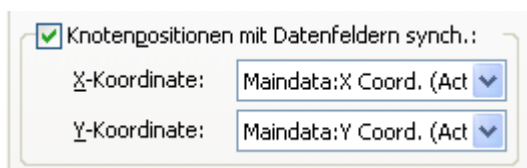
> **Speichern und Laden von Knotenpositionen**

Wenn Sie die Knotenpositionen eines Diagramms wieder laden möchten, müssen Sie diese in bestimmten Datenfeldern speichern.

Um die Positionen der Knoten mit den entsprechenden Datenfeldern zu synchronisieren, aktivieren Sie auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Knotenpositionen mit Datenfeldern synchronisieren** und geben Sie folgende Datenfelder an:

- für die X-Koordinate: "X-Koordinate (Knoten)"
- für die Y-Koordinate: "Y-Koordinate (Knoten)"

(Voraussetzung ist, dass Sie diese Datenfelder beim Einrichten der Schnittstelle entsprechend definiert haben. Siehe hierzu "Tutorium: Schnittstelle einrichten".)



Die Werte dieser Datenfelder können Sie im Dialogfeld **Vorgänge bearbeiten** ausfragen und ggf. bearbeiten.

> **Rang eines Knotens**

Unter dem Rang eines Knotens versteht man die größte Anzahl der Pfeile auf einem Weg zwischen Startknoten und dem betrachteten Knoten.

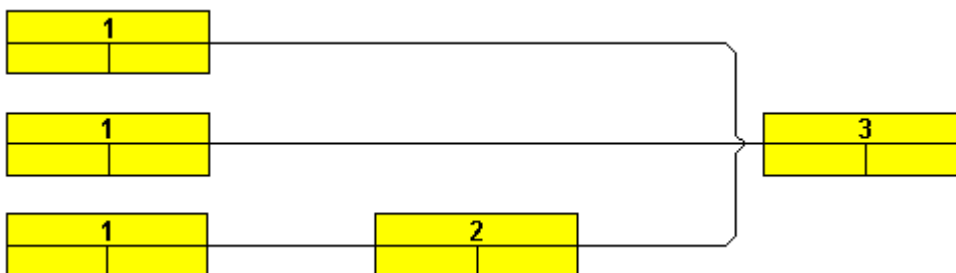
## 108 Wichtige Konzepte: Knoten

Der Rang eines Knotens ohne Vorgänger ist 1. Der Rang eines Knotens mit Vorgängern ist gleich 1 plus Rang desjenigen seiner Vorgänger, der den höchsten Rang besitzt.

Durch diese Definition wird verhindert, dass in Netzdiagrammen Ringbeziehungen (Schleifen) vorkommen.

### Einige Beispiele:

- Der Rang eines Knotens mit einem Vorgänger, der keinen Vorgänger besitzt, ist  $1+1=2$ .
- Der Rang eines Knotens, der drei Vorgänger mit den Rängen 1, 1 bzw. 2 hat, ist  $1+2=3$  (siehe Abbildung).



*Ränge der Knoten bei Flussrichtung von links nach rechts*

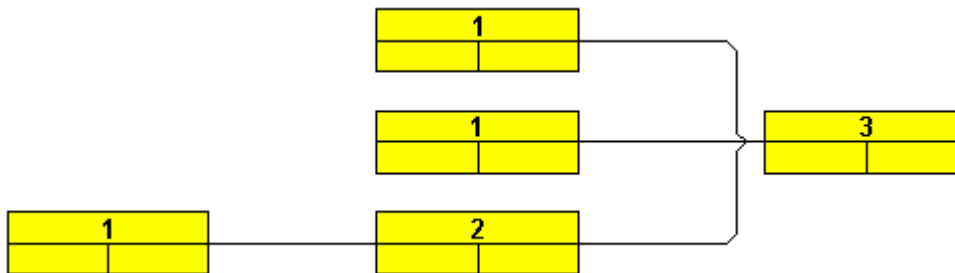
Den Rang von Knoten kann man sich folgendermaßen veranschaulichen:

- Bei Flussrichtung von links nach rechts entspricht der höchste Rang von allen Knoten in einer Diagrammspalte der Nummer dieser Diagrammspalte (gezählt werden hierbei nur die Diagrammspalten für Knoten, nicht die für Verbindungsbeschriftungen).
- Bei Flussrichtung von oben nach unten entspricht der höchste Rang von allen Knoten in einer Diagrammzeile der Nummer dieser Diagrammzeile (gezählt werden hierbei nur die Diagrammzeilen für Knoten, nicht die für Verbindungsbeschriftungen).

Ränge werden nur beim Aufruf von **Anordnen** (Kontextmenü für das Diagramm) berechnet. Sie dienen dem Layout-Algorithmus als Vorgabe für die Positionierung der Knoten in Flussrichtung. Bei Ringbeziehungen zwischen Knoten wählt VARCHART XNet über einen eigenen Algorithmus Verbindungen aus, die bei der Rangberechnung vorübergehend ignoriert werden, so dass das Layout möglichst natürlich aussieht. Die ignorierten Verbindungen sind nach dem Layout als rückführende Verbindungen sichtbar.

Über die Eigenschaft **ShortenedLinks** oder indem Sie das Kontrollkästchen **Verbindungen kürzen beim Anordnen** auf der Eigenschaftenseite **Allgemeines** aktivieren, erreichen Sie, dass die Positionen der Knoten beim

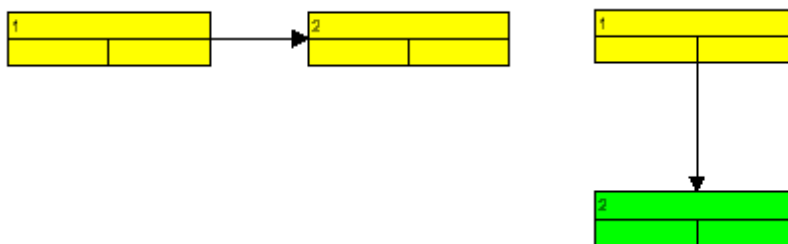
Layout soweit nach rechts bzw. unten verschoben werden, dass die Verbindungen zwischen den Knoten möglichst kurz werden.



### > Hilfsknoten

In vielen Fällen ist es sinnvoll, Knoten nicht in Flussrichtung anzuordnen, sondern entgegen der Flussrichtung unter- bzw. oberhalb ihrer Vorgänger (bei Flussrichtung von links nach rechts) bzw. neben ihren Vorgängern (bei Flussrichtung von oben nach unten). Dies ist in VARCHART XNet möglich, da es erlaubt, den Rang von Knoten zu reduzieren.

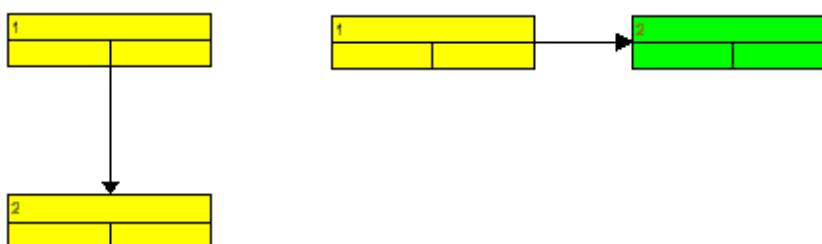
Bei Flussrichtung von links nach rechts steht der Hilfsknoten, dessen Rang um 1 reduziert wurde, unter oder über seinem Vorgänger statt daneben.



*Zwei Knoten mit Rang 1 bzw. 2*

*Der Rang des 2. Knotens wurde um 1 reduziert. Anschließend wurde der Befehl **Anordnen** aufgerufen.*

Bei Flussrichtung von oben nach unten steht der Hilfsknoten, dessen Rang um 1 reduziert wurde, links oder rechts neben seinem Vorgänger statt darunter.



## 110 Wichtige Konzepte: Knoten

*Zwei Knoten mit Rang 1 bzw. 2      Der Rang des 2. Knotens wurde um 1 reduziert.  
Anschließend wurde der Befehl **Anordnen** aufgerufen.*

Um den Rang eines Knotens reduzieren zu können, ist das Datenfeld "Hilfsknoten" vorgesehen. Der Eintrag in diesem Datenfeld legt fest, wie der Hilfsknoten positioniert wird. Mögliche Werte sind 0, 1, 2, 3.

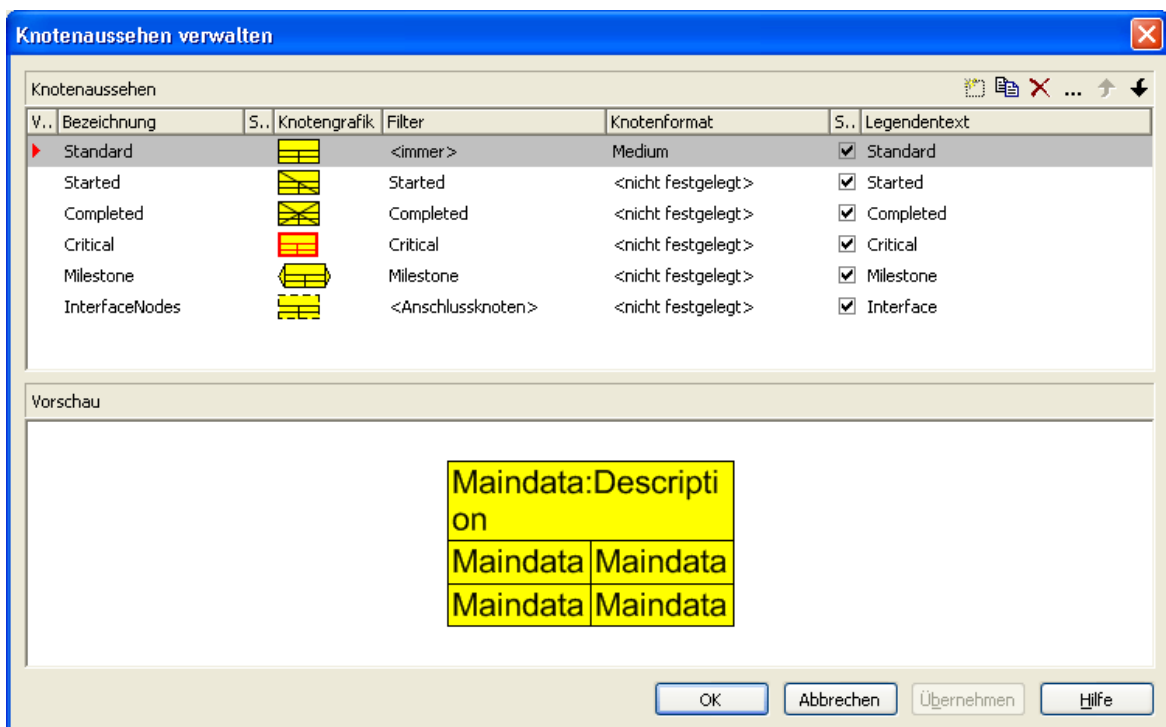
Wert im Feld "Hilfsknoten"	Flussrichtung von oben nach unten	Flussrichtung von links nach rechts
0	Der Rang des Hilfsknotens wird nicht reduziert.	Der Rang des Hilfsknotens wird nicht reduziert.
1	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann links oder rechts neben seinem Vorgänger statt darunter.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann über oder unter seinem Vorgänger statt daneben.
2	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann links von seinem Vorgänger.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann über seinem Vorgänger.
3	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann rechts von seinem Vorgänger.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann unter seinem Vorgänger.

Um Hilfsknoten auf demselben Rang zu positionieren wie ihre Vorgänger, aktivieren Sie auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Knoten auf demselben Rang wie seine Vorgänger anordnen gemäß Datenfeld**. Wählen Sie das Datenfeld "Hilfsknoten" aus, dessen Eintrag bestimmen soll, ob ein Knoten auf demselben Rang positioniert wird wie sein Vorgänger. (Dieses Datenfeld müssen Sie ggf. auf der Eigenschaftenseite **Datendefinition** erst definieren.) Es kann die Werte 0, 1, 2 oder 3 annehmen. Von dem Wert eines Knotens im Datenfeld "Hilfsknoten" hängt dann ab, ob ein Hilfsknoten auf demselben Rang positioniert wird wie sein Vorgänger und wie er positioniert wird.

## 3.11 Knotenaussehen

Das Aussehen von Knoten lässt sich in Abhängigkeit von deren Daten festlegen. Beispielsweise können alle Knoten der Abteilung A durch einen roten Hintergrund und eine doppelte schwarze Rahmenlinie gekennzeichnet werden, Knoten der Abteilung B durch einen blauen Hintergrund und eine einfache gelbe Rahmenlinie etc. Diese grafischen Attribute werden als Knotenaussehen bezeichnet.

Sie können Knotenaussehen festlegen, indem Sie auf der Eigenschaftenseite **Objekte** auf die **Knotenaussehen**-Schaltfläche klicken, um das Dialogfeld **Knotenaussehen verwalten** zu öffnen. Hier können Sie Knotenaussehen kopieren, bearbeiten, löschen und neu definieren oder deren Abarbeitungsreihenfolge verändern.



Jedes Knotenaussehen ist mit einem Filter und einem Knotenformat verbunden. Der Filter gibt die Bedingungen an, unter denen dieses Knotenaussehen auf einen Knoten angewandt wird. Beispielsweise ist das Knotenaussehen "Markiert" mit dem Filter "Markiert" verbunden, der alle markierten Knoten selektiert.

Erfüllt ein Knoten die Filterkriterien mehrerer Knotenaussehen, werden diese Knotenaussehen für den Knoten grafisch überlagert. Begonnen wird dabei jeweils mit dem Knotenaussehen, das in der Tabelle ganz oben steht. Das Knotenaussehen, das ganz unten steht, wird als letztes zugewiesen und überlagert daher alle anderen. Die niedrigste Position hat i. d. R. das

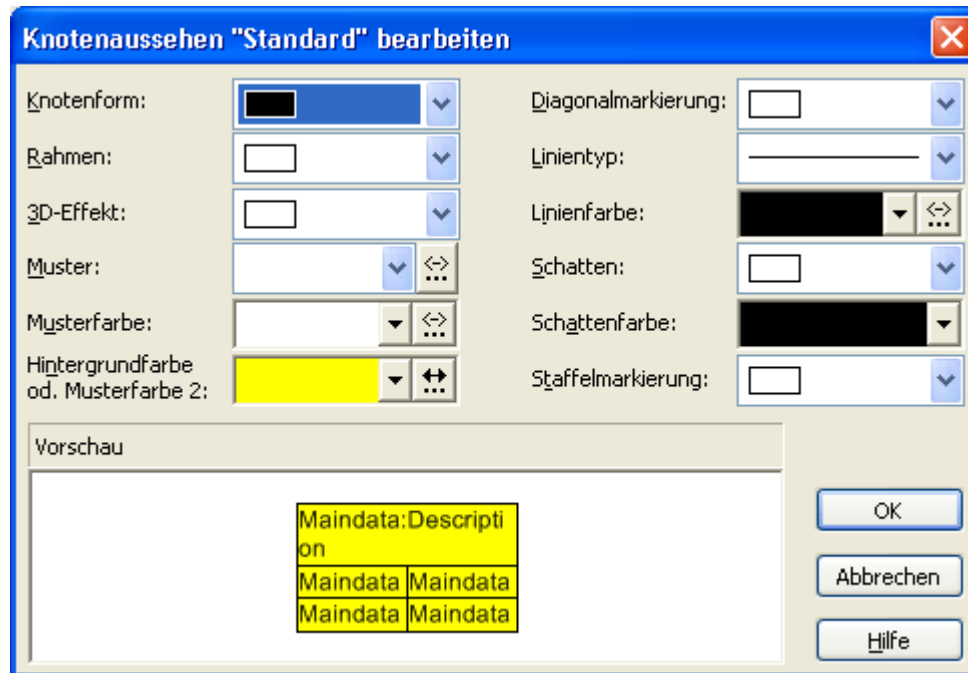


## 112 Wichtige Konzepte: Knotenaussehen

Knotenaussehen "Standard", das normalerweise ganz oben in der Tabelle steht. Es wird auf alle Knoten angewendet und kann nicht gelöscht werden.

↑ ↓ Sie können die Abarbeitungsreihenfolge der Knotenaussehen mit Hilfe der Pfeil-Schaltflächen verändern.

Um ein Knotenaussehen zu bearbeiten, klicken Sie auf die **Knotenaussehen bearbeiten**-Schaltfläche oder doppelklicken Sie auf das **Knotengrafik**-Feld. Sie gelangen dann in das folgende Dialogfeld:

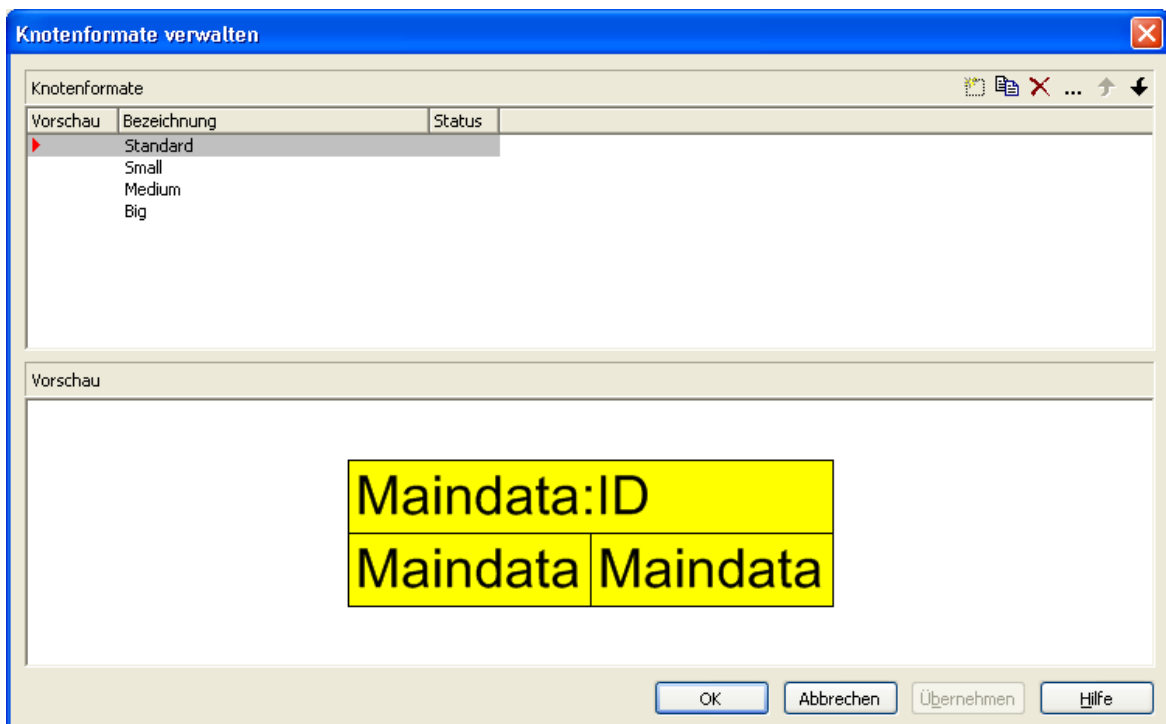


Die Hintergrundfarbe und die Linienfarbe eines Knotenaussehens können Sie mit Hilfe von Zuordnungstabellen in Abhängigkeit von den Knotendaten festlegen. Siehe hierzu das Kapitel "Wichtige Begriffe: Zuordnungstabellen".

## 3.12 Knotenformat

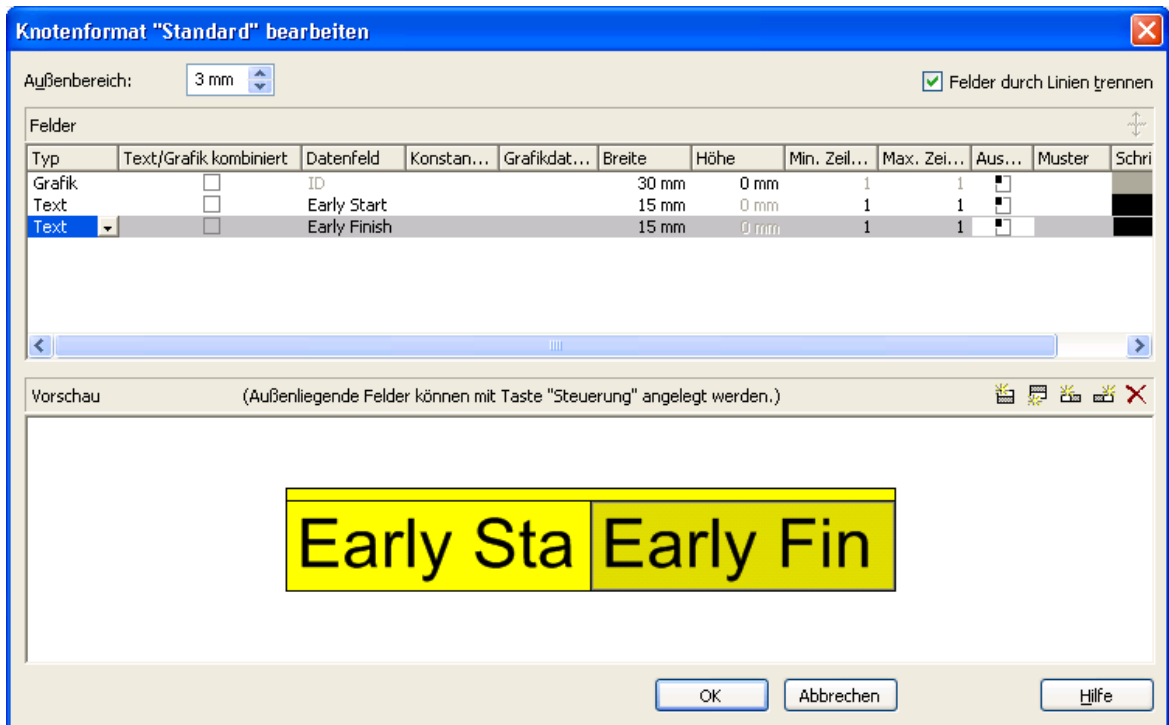
Jedes Knotenaussehen ist mit einem Knotenformat verbunden. Sie können das Knotenformat für ein Knotenaussehen festlegen, indem Sie im Dialogfeld **Knotenaussehen verwalten** aus der Kombobox unter **Knotenformat** das gewünschte Knotenformat auswählen.

Knotenformate werden im Dialogfeld **Knotenformate verwalten** verwaltet. Dieses Dialogfeld erreichen Sie, indem Sie auf der Eigenschaftenseite **Objekte** auf die Schaltfläche **Knotenformate** klicken.



Um das aktuelle Knotenformat zu bearbeiten, klicken Sie auf die **Knotenformat bearbeiten**-Schaltfläche. Sie gelangen dann in das Dialogfeld **Knotenformat bearbeiten**, in dem Sie den Aufbau und das Aussehen des jeweiligen Knotens bearbeiten können.

## 114 Wichtige Konzepte: Knotenformat



In diesem Dialog können Sie für das gewählte Knotenformat Folgendes festlegen:

- ob die Knotenfelder durch Linien voneinander getrennt werden sollen
- den Außenbereich (Abstand in Millimetern, den Knoten mit diesem Knotenformat zu benachbarten Knoten und zum Rand der Darstellung halten)
- den Feldtyp des aktuellen Knotenfeldes (Text oder Grafik)
- für den Typ Text: das Datenfeld, dessen Inhalt in dem aktuellen Knotenfeldes ausgegeben werden soll, oder einen konstanten Text
- für den Typ Grafik: Name und Pfad der Grafikdatei, die in dem gewählten Knotenfeld dargestellt wird
- die Breite und Höhe des markierten Knotenfeldes
- die maximale Anzahl von Textzeilen im aktuellen Knotenfeld
- die Ausrichtung des Textes bzw. der Grafik des markierten Knotenfeldes
- das Füllmuster und die Musterfarben des Knotenfeldes
- die Schriftart und -farbe des Knotenfeldes

### > **Datumsformat der Terminfelder von Knoten**

Das Datumsformat der Terminfelder von Knoten wird auf der Eigenschaftenseite **Allgemeines** festgelegt.

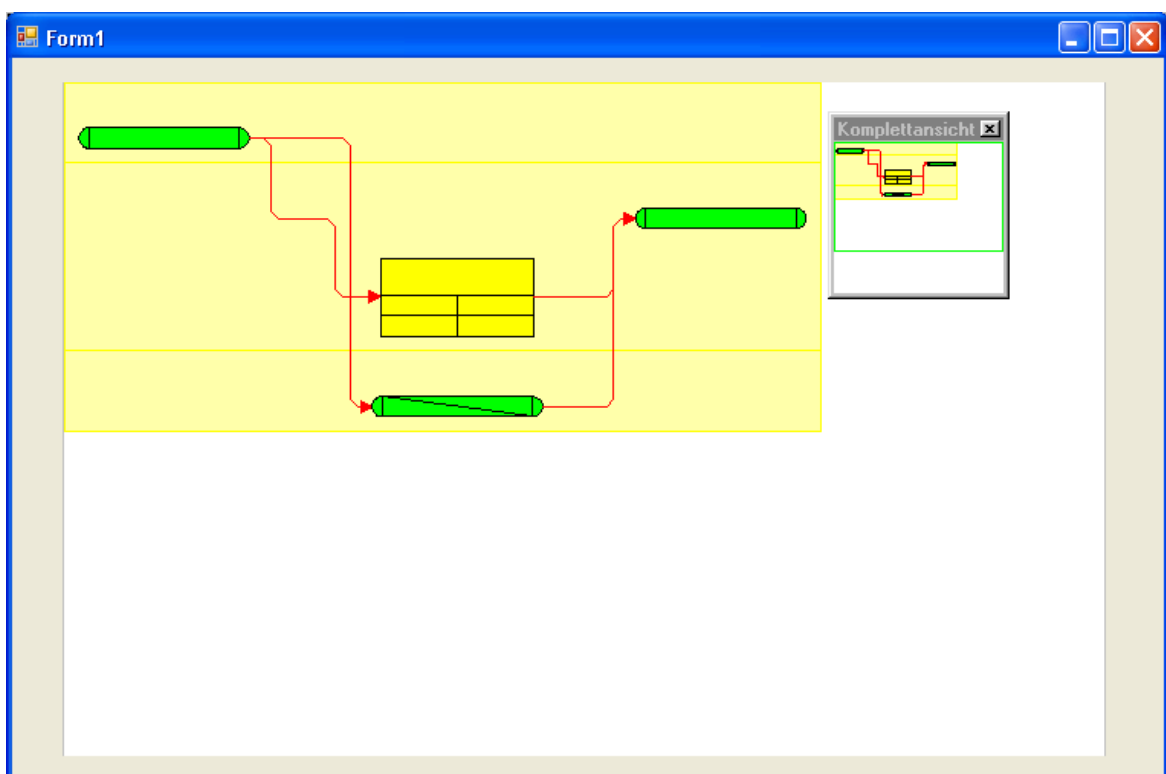
### > **Darstellung von Grafiken in Knotenfeldern**

Sie können für ein Knotenfeld des Typs Grafik eine Grafikdatei wählen, indem Sie das Feld **Grafikdateiname** markieren, auf die dann erscheinende Schaltfläche **Grafikdatei auswählen** (⋮) klicken und anschließend im gleichnamigen Windows-Dialog eine Datei wählen.

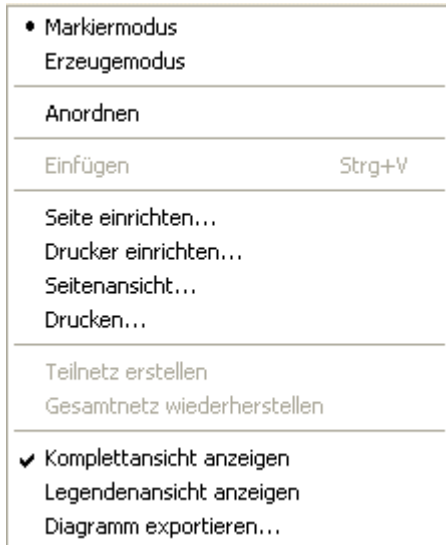
Sie können aber auch eine Zuordnung zwischen den Einträgen eines Datenfeldes und Grafikdateien herstellen. Klicken Sie dazu auf die Schaltfläche **Zuordnungen einstellen** (↔), um den gleichnamigen Dialog zu öffnen. Wenn eine Zuordnung vorgenommen worden ist, wird das durch ein Symbol neben dem Grafikdateinamen dargestellt (↔). Einzelheiten hierzu finden Sie in den Kapiteln "Eigenschaftenseiten und Dialogfelder" und "Wichtige Begriffe: Zuordnungstabellen".

### 3.13 Komplettansicht (World View)

Die Komplettansicht ist ein zusätzliches Fenster, in dem das komplette Diagramm angezeigt wird. Ein Rahmen zeigt an, welchen Diagrammausschnitt das Hauptfenster gerade anzeigt. Wenn Sie mit der Maus diesen Rahmen verschieben, wird der angezeigte Ausschnitt des Hauptfensters beim Loslassen der Maustaste entsprechend verschoben. In ähnlicher Weise können Sie durch Größer- oder Kleinerziehen des Rahmens in der Komplettansicht den realen Bildausschnitt zoomen. Umgekehrt ändern sich Position bzw. Größe des Rechtecks, wenn der Ausschnitt im Hauptfenster gescrollt oder gezoomt wird.



Zur Laufzeit können Sie über den Menüpunkt **Komplettansicht anzeigen** des Standard-Kontextmenüs die Komplettansicht ein- bzw. ausschalten.



Auf der Eigenschaftenseite **Zusätzliche Ansichten** können Sie die Eigenschaften der Komplettansicht festlegen. Einzelheiten hierzu finden Sie im Kapitel "Eigenschaftenseiten und Dialogfelder", Eigenschaftenseite "Zusätzliche Ansichten".

Alternativ können Sie die Eigenschaften der Komplettansicht auch über die API (**VcWorldView**) festlegen.

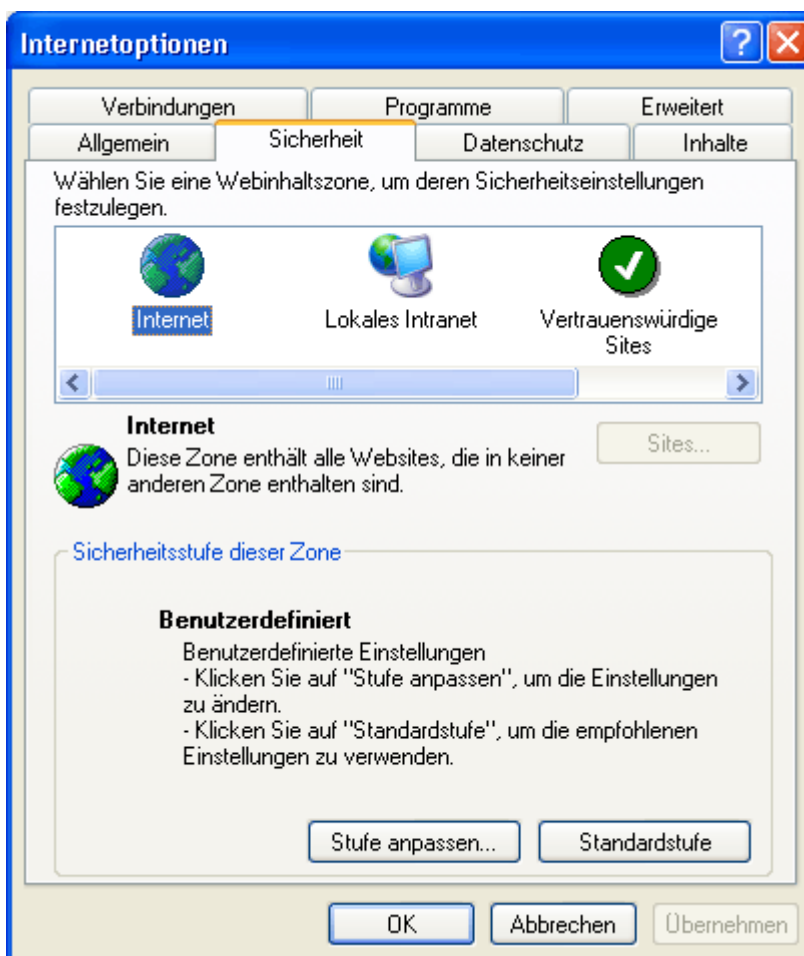
## 3.14 Laufzeitsicherheitsrichtlinien für den Einsatz im Internet Explorer

Die Laufzeitsicherheitsrichtlinien müssen verändert werden, wenn das VARCHART XNet Steuerelement im Internet Explorer innerhalb einer HTML-Seite verwendet wird.

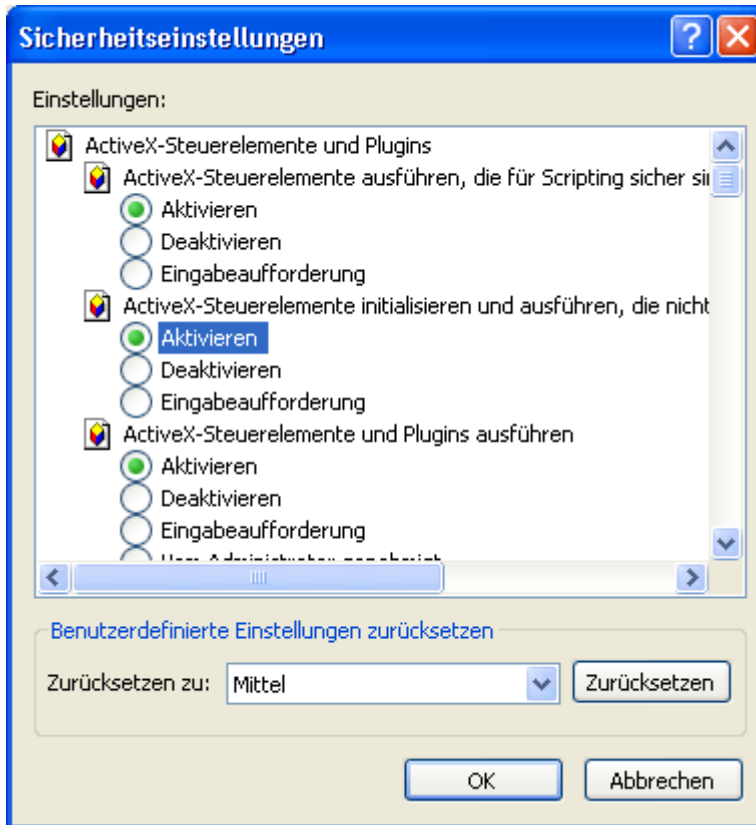
Sobald der Browser das Steuerelement von einem Webserver im Internet lädt, werden die **Laufzeitsicherheitsrichtlinien** der Internet\_Zone wirksam. Die Standardeinstellungen verhindern die Ausführung des Steuerelements. Grundsätzlich muss der Internet Explorer das Ausführen von .NET Komponenten zulassen, damit sie überhaupt sichtbar werden können.

Im Dialog **Sicherheitseinstellungen** können die Richtlinien verändert werden. Sie finden den Dialog im Internet Explorer über

**Extras > Internetoptionen > Sicherheit > Internet** des Internet Explorer. Hier wählen Sie **Internet** oder **Vertrauenswürdige Sites** aus.



Für diese Zone aktivieren Sie bitte unter **Stufe anpassen...** sowohl die Option **ActiveX-Elemente ausführen, die für Scripting sicher sind** als auch **ActiveX-Elemente initialisieren und ausführen, die nicht sicher sind**.



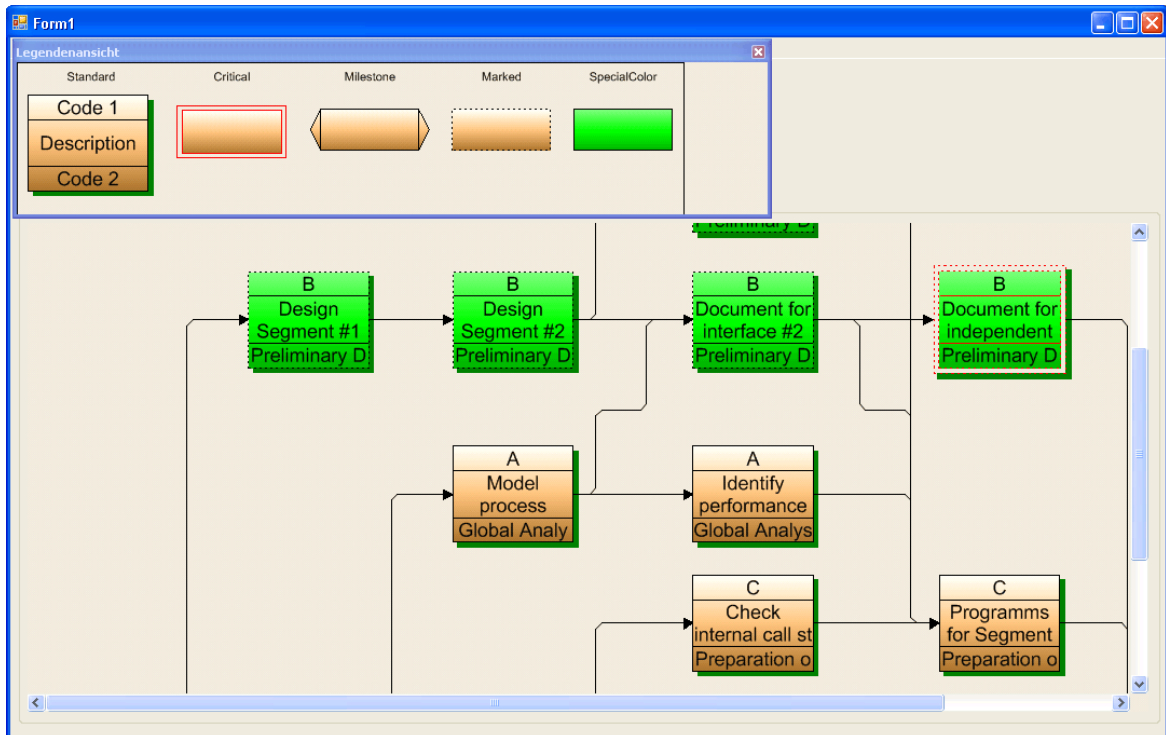
Zusätzlich müssen die Laufzeitsicherheitsrichtlinien auf dem lokalen Rechner eingestellt werden.

Im Unterverzeichnis **CAS** der VARCHART XGantt-Installation finden Sie zwei passende Batch-Dateien, die Sie von der Kommandozeile aus aufrufen können. Die erste ist **AddRights.bat**, mit der Sie einen Berechtigungssatz und eine Code-Gruppe für NETRONIC-Steuererelemente erzeugen können. Wenn Sie später eine Applikation bei Ihrem Endkunden ausliefern wollen, muss die Batch-Datei vorher auf jedem Client-System einmalig ausgeführt werden. Mit der zweiten Datei **RemoveRights.bat** können Sie Berechtigungen rückgängig machen. Auf diese Weise kann das VARCHART XGantt Steuererelement mit der erforderlichen Mindestmenge an Berechtigungssätzen auf einer HTML-Seite im Internet Explorer ausgeführt werden.

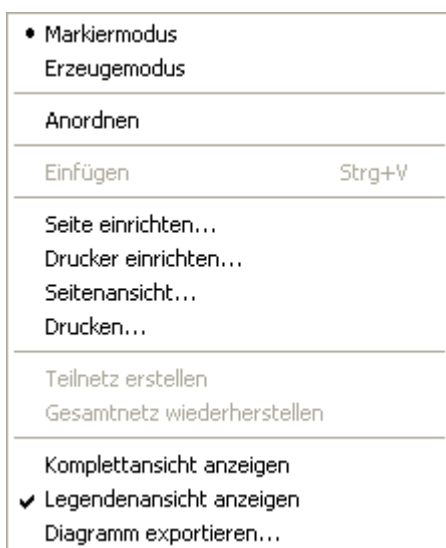


## 3.15 Legendenansicht (Legend View)

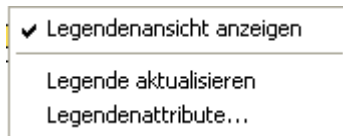
Die Legendenansicht ist ein zusätzliches Fenster zur Darstellung einer Legende auf dem Bildschirm. Das Aussehen der Legende wird festgelegt im Dialog **Legendenattribute**, der über die Eigenschaftenseite **Außenbereich** zu erreichen ist, oder über die Legendenattribute des Objektes **VcBoundingBox**.



Zur Laufzeit können Sie über den Menüpunkt **Legendenansicht anzeigen** des Standard-Kontextmenüs die Legendenansicht ein- und ausschalten.



Die Legende verfügt über ein eigenes Kontextmenü, über das die Legendenansicht ebenfalls ein- und ausgeschaltet werden kann.



Außerdem können Sie über das Kontextmenü auch den Dialog **Legendenattribute** aufrufen sowie die Legende aktualisieren.

Eine Aktualisierung über das Menü kann notwendig sein, da nach Änderungen im Diagramm die Legende nicht automatisch aktualisiert wird. Werden also zum Beispiel Knoten hinzugefügt oder gelöscht, muss eine Aktualisierung entweder über das Kontextmenü oder durch Aus- und Einschalten der Legende durchgeführt werden. Dies gilt auch für das Laden von Knoten. Wenn für die Legendenansicht auf der Eigenschaftenseite **Zusätzliche Ansichten** die Option **Beim Start sichtbar** eingestellt wurde, aber zum Zeitpunkt des Aufbaus noch keine Knoten geladen waren, bleibt die Legende bis zur Aktualisierung leer.

Auf der Eigenschaftenseite **Zusätzliche Ansichten** können Sie die Eigenschaften der Legendenansicht festlegen. Einzelheiten hierzu finden Sie im Kapitel **Eigenschaftenseiten und Dialogfelder, Eigenschaftenseite Zusätzliche Ansichten**.

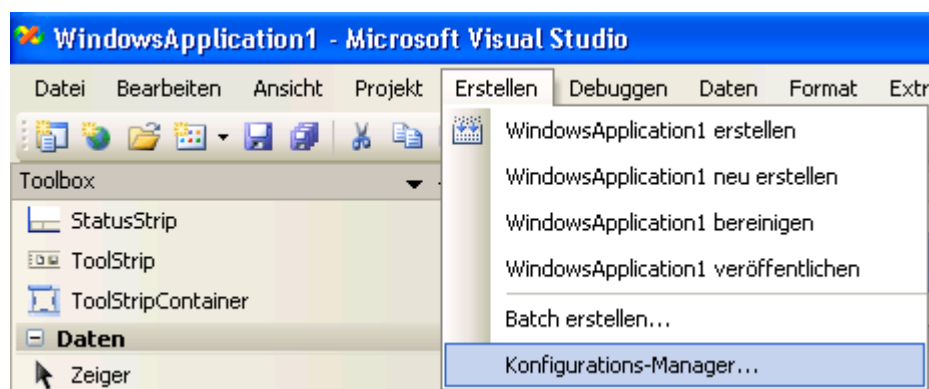
Alternativ können Sie die Optionen der Legendenansicht auch über die API mit der Eigenschaft **VcNet.VcLegendView** festlegen.

## 3.16 Plattformen x86 und x64

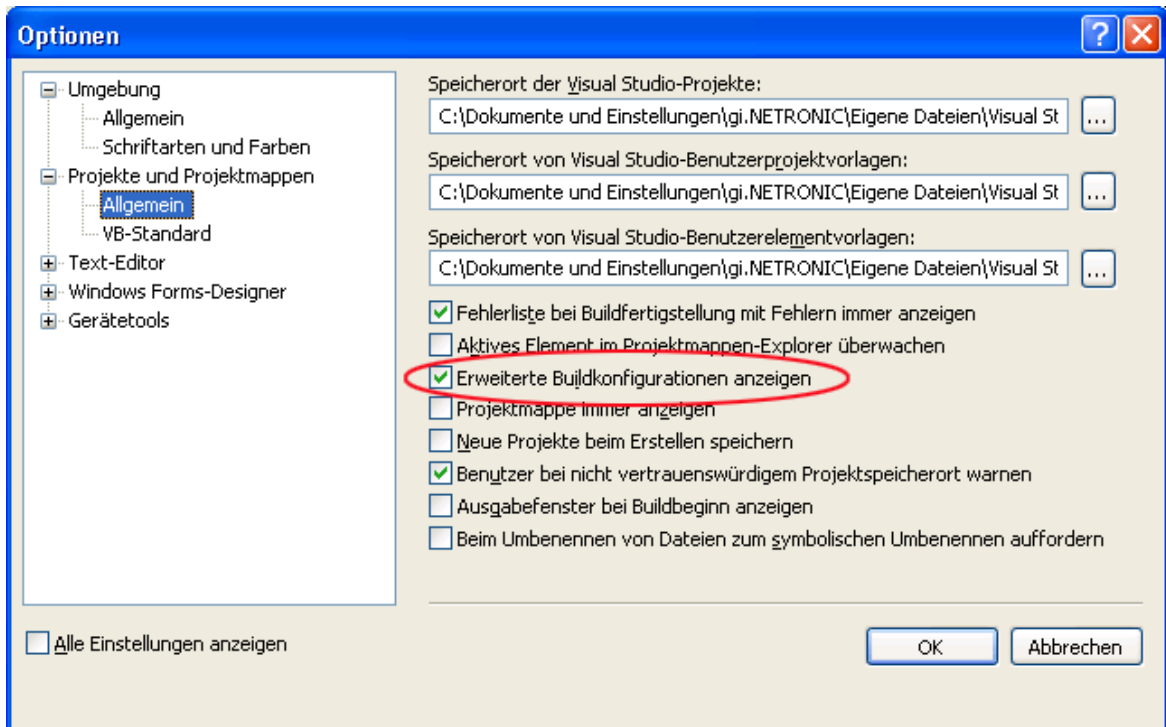
Applikationen, die mit dem .NET Framework geschrieben wurden, werden gewöhnlich über einen Compiler in MSIL, einen prozessorunabhängigen Bytecode, übersetzt. Beim Starten der Applikation wird MSIL in einen Maschinencode übersetzt, der vom jeweilig verwendeten Prozessor verstanden und in dessen voller Geschwindigkeit ausgeführt wird. Applikationen in MSIL sind also unter Windows auf jedem Prozessor ausführbar, wenn sie keine reinen Maschinencode-Komponenten (Assemblies oder DLLs) verwenden. Sie sind sogar auf anderen Betriebssystemen wie z.B. unter Mono mit Linux lauffähig, sofern keine betriebssystemabhängigen Komponenten verwendet werden. Wenn eine Anwendung die Voraussetzungen für die Prozessorunabhängigkeit nicht erfüllt, sollte sie entsprechend markiert werden, damit sie nicht irrtümlich auf einem nicht unterstützten Prozessor gestartet wird und es dann bei erstmaliger Verwendung einer prozessor- oder betriebssystemabhängigen Komponente zu Fehlermeldungen kommt.

VARCHART XNet liegt intern teilweise in reinem Maschinencode vor, was unter .NET als **Mixed Mode** bezeichnet wird. Daher muss XNet für jeden Prozessor, mit dem es verwendet werden soll, neu übersetzt werden. Es sind Varianten für x86-Prozessoren und ab Version 4.3 auch für x64-Prozessoren verfügbar.

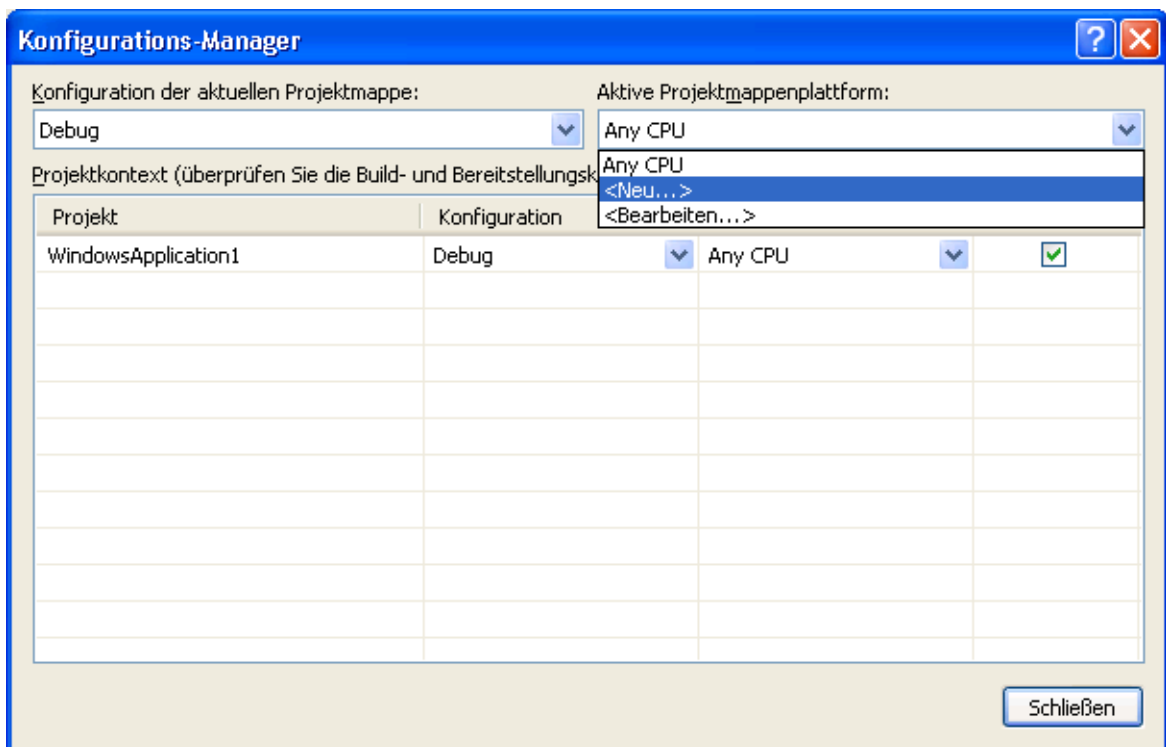
Applikationen, in denen VARCHART XNet verwendet wird, sind also nicht prozessorunabhängig. Da dies von Visual Studio in den Versionen 2005 bis 2010 nicht automatisch erkannt wird, muss man den Prozessor in einem Projekt bzw. einer Projektmappe manuell einstellen. Dies geschieht über den Konfigurations-Manager, den Sie über **Erstellen/Konfigurations- Manager** erreichen.



Sollte dieser Menüpunkt nicht zu sehen sein, muss erst die Einstellung **Erweiterte Buildkonfigurationen** im Dialog **Extras/Optionen/Projekte und Projektmappen/ Allgemein** aktiviert werden.

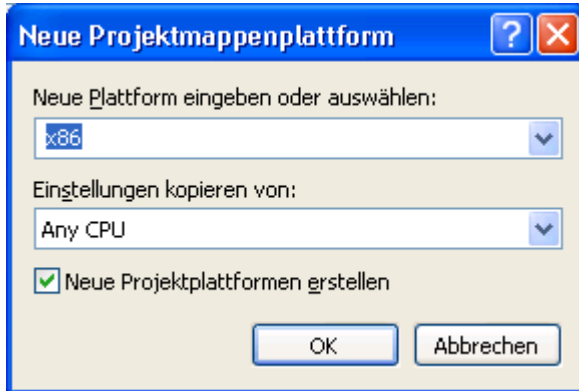


Zum Erstellen einer neuen Plattform wählen Sie bitte im Konfigurationsmanager den Eintrag <Neu> bei **Aktive Projektmappenplattform**.

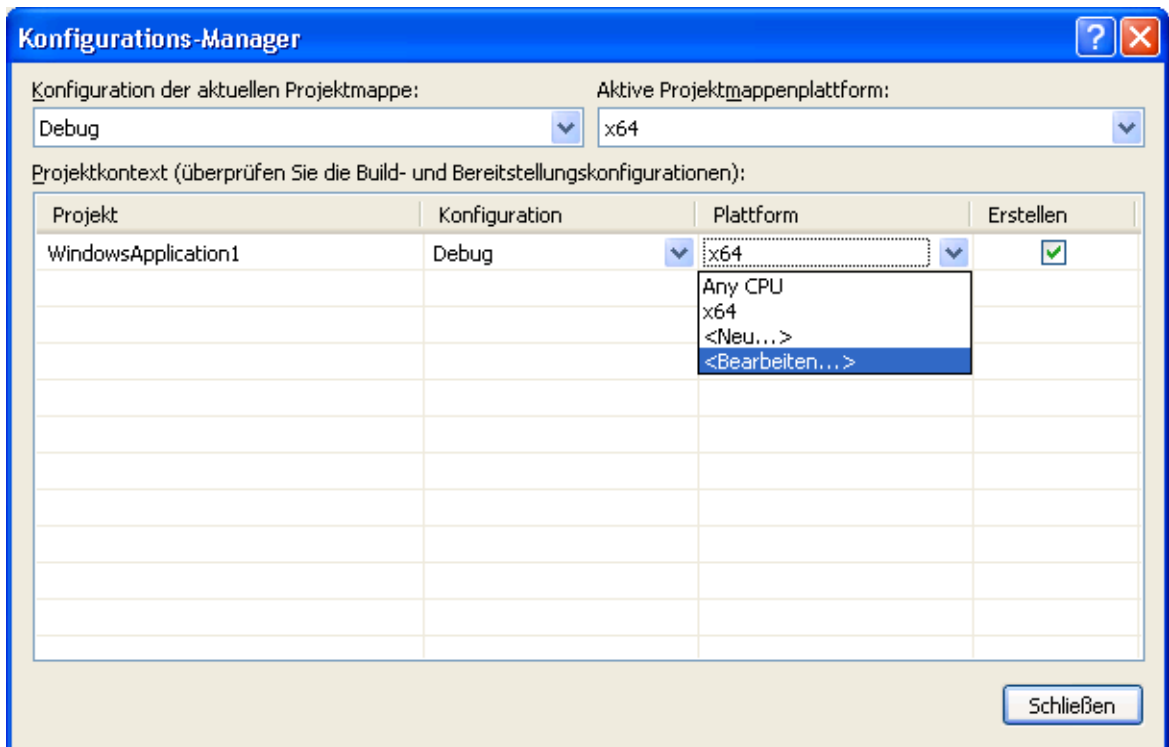


Folgender Dialog erscheint, in dem Sie nun die gewünschten Plattformen x86 oder x64 anlegen können:

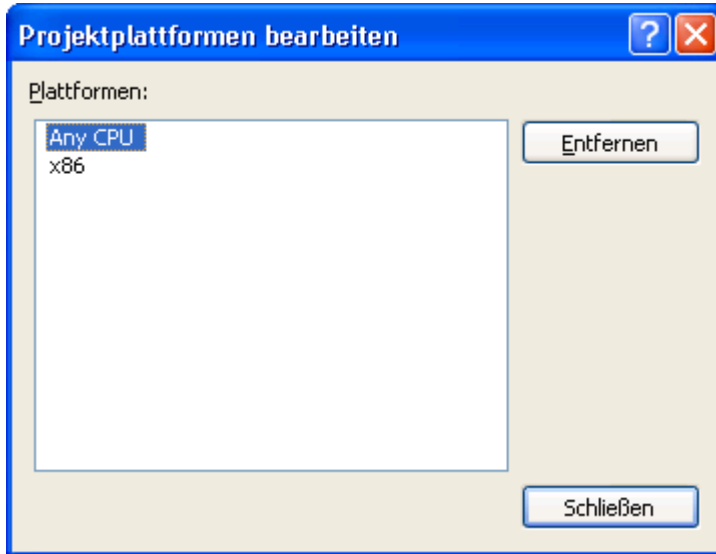
## 124 Wichtige Konzepte: Plattformen x86 und x64



Wenn Sie anschliessend die Plattform **Any CPU** löschen möchten, wählen Sie bitte zunächst im **Projektkontext** des Konfigurations-Managers in der Spalte **Plattform** den Menüpunkt **Bearbeiten** aus der Drop-Down-Liste:



Im nun erscheinenden Dialog können Sie die gewünschte Plattform löschen, indem Sie sie markieren und auf **Entfernen** klicken:



Damit immer die richtige Variante von XNet im Visual Studio benutzt wird, müssen Sie bestimmte Prozeduren in den Pre-Build-Event und den Post-Build-Event Ihres Projektes einbauen. Diese Prozeduren befinden sich im Unterverzeichnis *BuildSteps* innerhalb des XNet-Installationsverzeichnisses (bei Zielframework .NET 2.0 bitte in beiden Build-Events die Zeile "set DOTNET=..." anpassen). Danach kompilieren Sie einmalig Ihr Projekt, was vom Visual Studio nicht unerwartet mit Fehlermeldungen quittiert wird. Im Anschluss fügen Sie eine Referenz auf die *NETRONIC.XNet.dll* im neu entstandenen Verzeichnis *C:\XNetReference* ein (evtl. müssen Sie vorher eine bestehende Referenz auf das XNet-Installationsverzeichnis entfernen) und kompilieren Ihr Projekt noch einmal.

---

## 3.17 Schreiben von PDF-Dateien

Das Schreiben von PDF-Dateien ist nur möglich, wenn ein geeigneter PDF-Druckertreiber installiert ist. Die kostenlosen und kommerziell verfügbaren Treiber unterscheiden sich hinsichtlich der Funktionalität und Qualität der erzeugten PDF-Dateien.

Für die Ansteuerung der Treiber gibt es keinen einheitlichen Standard, sodass jeder Druckertreiber individuell konfiguriert werden muss. So ist beispielsweise bei vielen PDF-Druckertreibern der Zielpfad für die Ausgabedatei fest vorgegeben und kann nur durch Eingriffe in die Windows-Registry, durch Editieren von INI-Dateien oder durch Verwenden treiberspezifischer Funktions-APIs oder COM-Objekte geändert werden.

Ein PDF-Druckertreiber muss die folgenden Anforderungen hinsichtlich der Ansteuerung und Druckqualität erfüllen, damit er sich für den Einsatz eignet:

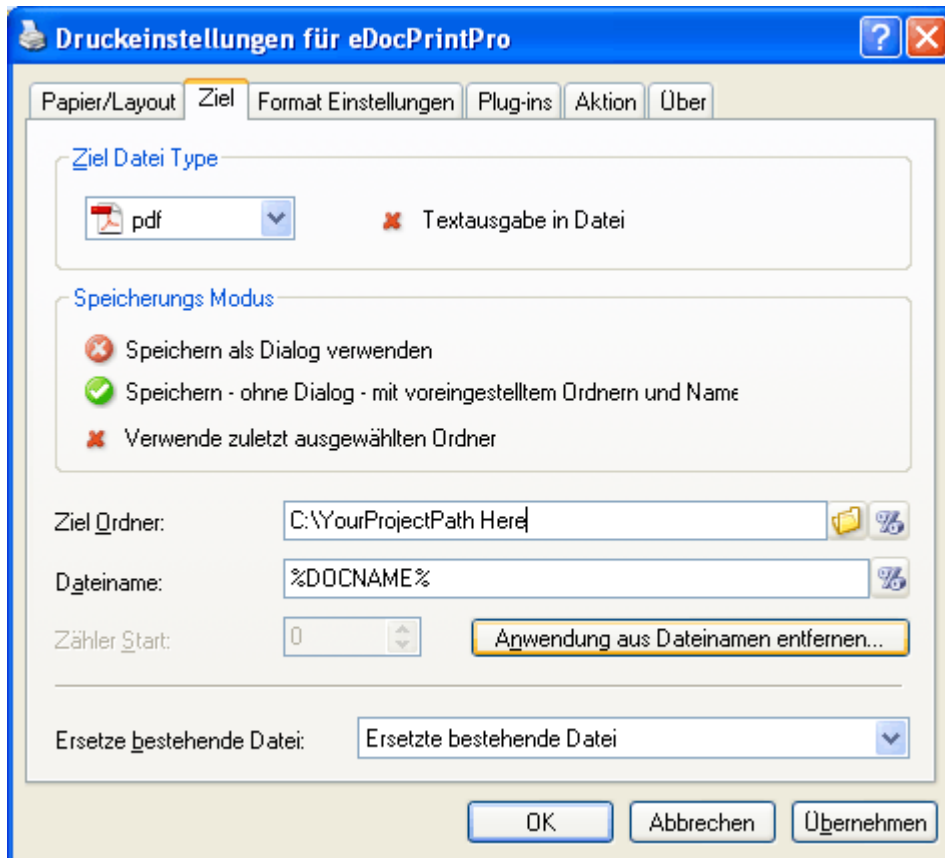
- Je nach Design der Anwendung kann es notwendig sein, den Treiber über die API so einzustellen, dass zur Laufzeit keine Dialoge und Messageboxen erscheinen. Dazu gehören insbesondere Dialoge zur Festlegung von Dateinamen und Pfaden.
- Soll das Setzen von Dateinamen und Pfad erst zur Laufzeit erfolgen, und ist dies nur über das Ändern von Windows-Registry-Einträgen möglich, dann müssen die Rechte des Benutzerkontos dies auch erlauben.
- Zur korrekten Ausgabe von Texten ist Unicode-Unterstützung erforderlich.
- Die Wiedergabe von Füllmustern muss in ausreichender Qualität erfolgen. Dabei ist zu beachten, dass Transparenzen mit Ausnahme bei Bitmaps grundsätzlich nicht dargestellt werden können. Dort können jedoch unerwünschte Artefakte auftreten.
- Der Treiber muss die Ausgabe vertikaler Texte unterstützen, sonst kann die vertikale Beschriftung von Datumslinien in VARCHART XGantt nicht genutzt werden.

Die vorgenannten Anforderungen erfüllen z.B. der in der **Adobe Acrobat Suite** ab Version 6 enthaltene Druckertreiber [[www.adobe.com](http://www.adobe.com)] sowie der kostenlose Treiber **eDocPrintPro** [[www.pdfprinter.at](http://www.pdfprinter.at)].

Im Folgenden werden die Schritte skizziert, die zur Ansteuerung der Druckertreiber notwendig sind. Dies wird exemplarisch am Treiber **eDocPrintPro** verdeutlicht:

- In den **Druckeinstellungen** (erreichbar über die Einstellungen des Treibers in der Systemsteuerung oder über einen eigenen Eintrag des

Treibers unter Start/Programme oder über den normalen Druckdialog in einer Anwendung) kann gegebenenfalls spezifiziert werden, dass die PDF-Erzeugung ohne Dialog abläuft, und dass der Name der Zielfeile z.B. über den Dokumentennamen bestimmt wird. Bei **eDocPrintPro** sehen die notwendigen Einstellungen dann wie folgt aus:



- Im Programm wird dann das VcPrinter-Objekt von VARCHART XGantt folgendermaßen bestückt:

#### Code-Beispiel VB.NET

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

#### Code-Beispiel C#

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

Ganz wenige Druckertreiber erfordern eine andere Code-Sequenz:

#### Code-Beispiel VB.NET

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



## 128 Wichtige Konzepte: Schreiben von PDF-Dateien

### Code-Beispiel C#

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

Für weitere Fragen zur Konfiguration und Benutzung von **eDocPrintPro** bitten wir Sie, sich mit dem Hersteller in Verbindung zu setzen.

## 3.18 Sprachanpassung von Textausgaben

Sie können mit Hilfe des Ereignisses **VcTextEntrySupplying** die Texte aller zur Laufzeit erscheinenden Kontextmenüs, Dialogfelder, Infoboxen und Fehlermeldungen bearbeiten, z. B. um sie in unterschiedliche Sprachen zu übersetzen.

Aktivieren Sie dazu das Kontrollkästchen **VcTextEntrySupplying-Ereignisse** auf der Eigenschaftenseite **Allgemeines**.

Oder setzen Sie dazu die Eigenschaft **TextEntrySupplyingEventEnabled** auf den Wert **True**, um das Ereignis zu aktivieren.

### Code-Beispiel VB.NET

```
VcNet1.TextEntrySupplyingEventEnabled = True
```

### Code-Beispiel C#

```
vcNet1.TextEntrySupplyingEventEnabled = true;
```

Fangen Sie dann das Ereignis **VcTextEntrySupplyingEvent** ab und legen Sie fest, welcher Text erscheinen soll.

### Code-Beispiel VB.NET

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

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

### Code-Beispiel C#

```
private void vcNet1_VcTextEntrySupplying(object sender,
NETRONIC.XNet.VcTextEntrySupplyingEventArgs e)
{
    switch(e.ControlIndex)
    {
        case VcTextEntryIndex.vcTXERibCW:
            e.Text = "KW";
            break;
        case VcTextEntryIndex.vcTXERibDay0:
            e.Text = "Mo";
            break;
    }
}
```

## 130 Wichtige Konzepte: Sprachanpassung von Textausgaben

```
case VcTextEntryIndex.vcTXERibMon8:  
    e.Text = "September";  
    break;  
case VcTextEntryIndex.vcTXERibQuar3:  
    e.Text = "3. Quartal";  
    break;  
}  
}
```

---

## 3.19 Statuszeilentext

Sie können das Ereignis **VcStatusLineTextShowing** verwenden, um Informationen über den mit der Maus berührten Knoten in einer Statusleiste bereitzustellen.

---

## 3.20 Tooltips zur Laufzeit

Sie können Tooltips verwenden, um Informationen über das mit der Maus berührte Objekt bereitzustellen. Mit Hilfe des Ereignisses **VcToolTipTextSupplying** können Sie die Texte aller zur Laufzeit erscheinenden Tooltips (Group, Node, None, LinkCollection) bearbeiten, z. B. um sie in unterschiedliche Sprachen zu übersetzen oder zu unterdrücken.

Setzen Sie dazu die VcNet-Eigenschaft **ToolTipTextSupplyingEventEnabled** auf den Wert True, um das Ereignis zu aktivieren.

Alternativ können Sie auf der Eigenschaftenseite **Allgemeines** das Kontrollkästchen **VcToolTipTextSupplying-Ereignisse** aktivieren.

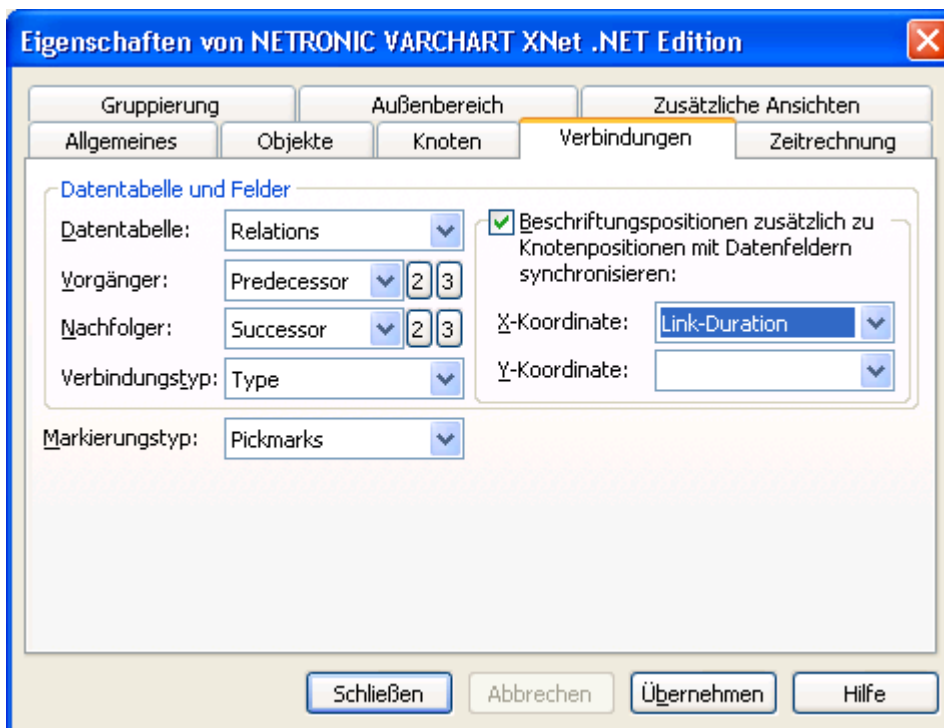
Fangen Sie dann das Ereignis **VcToolTipTextSupplying** ab und legen Sie fest, welcher Text erscheinen soll, oder ob an dieser Stelle kein Tooltip erscheinen soll.

## 3.21 Verbindungen

Eine Verbindung entspricht einem Datensatz aus der Datentabelle, die die Verbindungsdaten enthält. Verbindungsdaten werden zusammen mit den Knotendaten in einer Datei abgelegt. Verbindungen können über die API geladen oder interaktiv vom Anwender erzeugt werden.

### > Verbindungen festlegen

Auf der Eigenschaftenseite **Verbindungen** können Sie festlegen, ob die Verbindungen angezeigt werden sollen, und ggf. die Verbindungen spezifizieren.



Sie können hier die Datenfelder festlegen, in denen die Identifizierung des Vorgänger- und des Nachfolgerknotens sowie des Verbindungstyps festgelegt ist. Wenn die Identifikation eines Vorgänger - oder Nachfolgerknotens mehrteilig ist, muss auch die jeweilige Verbindung entsprechend aufgebaut sein d.h. dass bei **Vorgänger-Feld** bzw. **Nachfolger-Feld** ggf. ein zweites oder drittes Feld entsprechend der ID des jeweiligen Knotens angegeben werden muss. standardmäßig wird jeweils das erste Feld angezeigt. Zur Angabe des zweiten oder dritten Feldes klicken Sie auf die entsprechenden Schaltflächen und wählen dann aus der Drop-Down-Liste das gewünschte Feld aus.

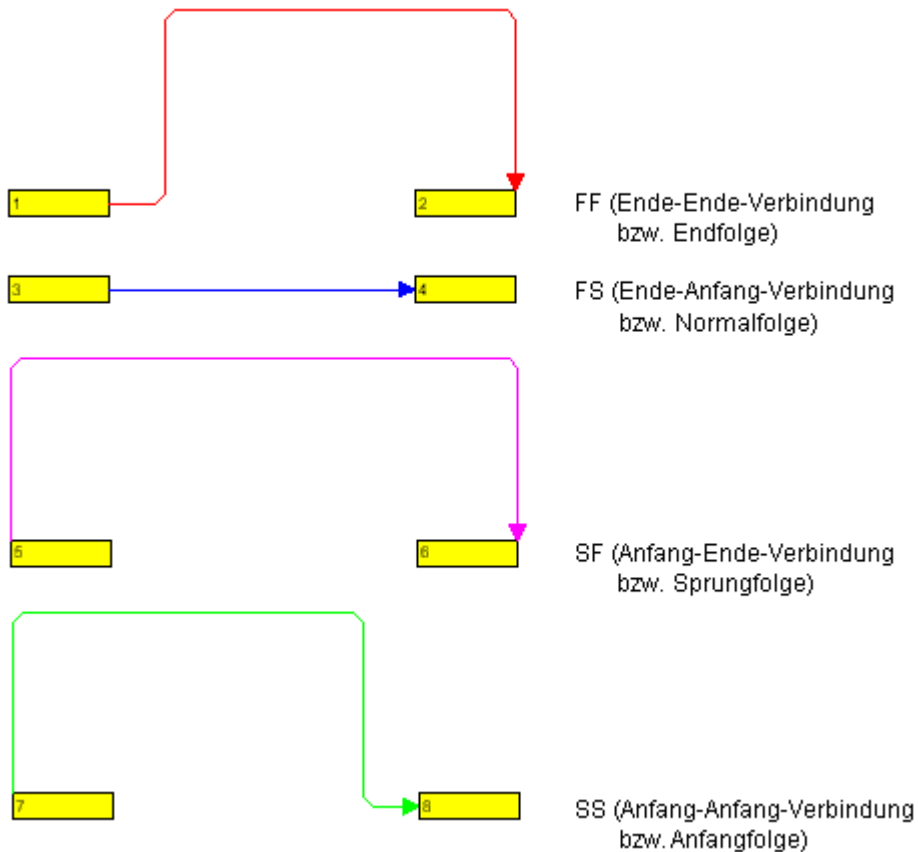
Außerdem können Sie verschiedene Verbindungsausssehen festlegen. Für jedes davon können Sie einen Filter, den Vorgänger- und Nachfolger-Layer, die Linienart sowie die Vorgänger- und Nachfolger-Portsymbole festlegen.

## 134 Wichtige Konzepte: Verbindungen

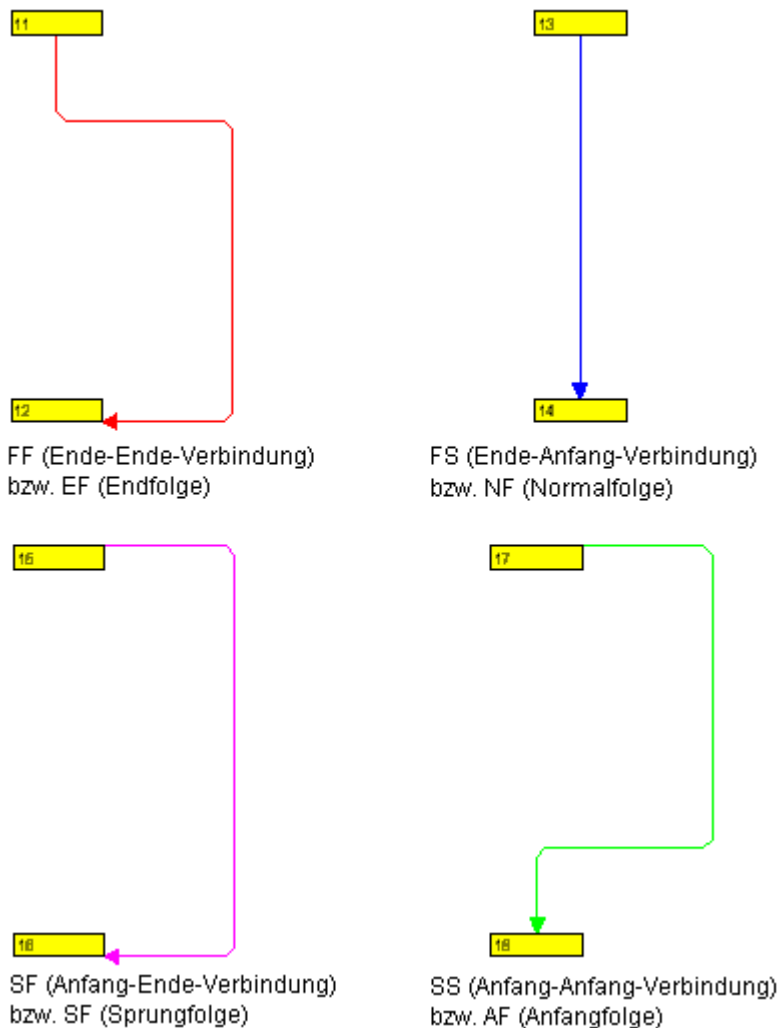
### > Verbindungstypen

Unter **Verbindungstyp** legen Sie fest, aus welchem Datenfeld der Verbindungstyp gelesen werden soll.

Eine Übersicht über das Aussehen der verschiedenen Verbindungstypen bei den beiden Flussrichtungen geben die folgenden Abbildungen:



*Flussrichtung von links nach rechts*



*Flussrichtung von oben nach unten*

### > Position von Verbindungsbeschriftungen

Damit Sie die Positionen der Verbindungsbeschriftungen eines Diagramms wieder laden können, müssen Sie diese mit den entsprechenden Datenfeldern synchronisieren. Aktivieren Sie dazu auf der Eigenschaftenseite **Verbindungen** das Kontrollkästchen **Positionen der Beschriftungen mit Datenfeldern synchronisieren** und geben Sie folgende Datenfelder an:

- für die X-Koordinate: "X-Koordinate (Verbindung)"
- für die Y-Koordinate: "Y-Koordinate (Verbindung)"

(Voraussetzung ist, dass Sie diese Datenfelder beim Einrichten der Schnittstelle entsprechend definiert haben. Siehe hierzu "Tutorium, Schnittstelle einrichten".)



## 136 Wichtige Konzepte: Verbindungen

Beschriftungspositionen zusätzlich zu Knotenpositionen mit Datenfeldern synchronisieren:

X-Koordinate: Relations:X Coord. ( v)

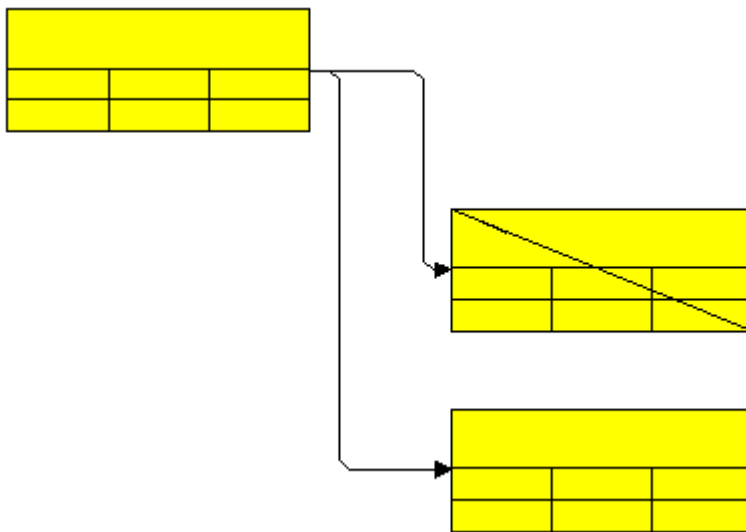
Y-Koordinate: Relations:Y Coord. ( v)

Die Werte dieser Datenfelder können Sie im Dialogfeld **Verbindung bearbeiten** ausfragen und ggf. bearbeiten.

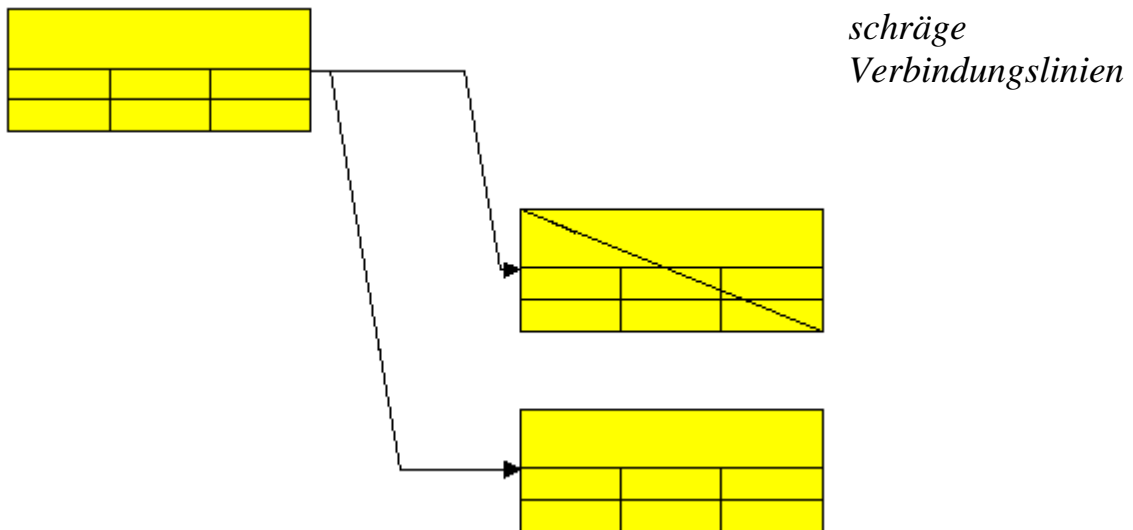
Ein Beispiel zu Positionen von Knoten- und Verbindungsbeschriftungen finden Sie im Abschnitt "Knoten" dieses Kapitels.

### > Orthogonale/Schräge Verbindungslinien

Auf der Eigenschaftenseite **Allgemeines** können Sie unter **Schräge Tracks bei Verbindungen** festlegen, ob die Verbindungslinien orthogonal dargestellt werden sollen oder direkt an den horizontalen Linienstücken ansetzen und schräg verlaufen sollen. Alternativ können Sie dies über die VcNet-Eigenschaft **ObliqueTracksOnLinks** festlegen.



*orthogonale  
Verbindungslinien*



### > Verbindungen erzeugen

Der Anwender kann zur Laufzeit im Erzeugemodus Verbindungen zwischen Knoten ziehen, sofern auf der Eigenschaftenseite **Allgemeines** die Option **Neue Knoten und Verbindungen zulassen** aktiviert ist. Wenn auf der Eigenschaftenseite **Allgemeines** außerdem die Option **Neue Verbindung bearbeiten** ist, erscheint beim Anlegen einer neuen Verbindung der Dialog **Verbindung bearbeiten**, in dem Sie die Daten der Verbindung direkt bearbeiten können.



Sie können Verbindungen auch über die API mit **InsertLinkRecord** anlegen. Jedes Neuanlegen einer Verbindung, gleich auf welche Weise, wird der Applikation mit dem Ereignis **VcLinkCreating** mitgeteilt.

### > Verbindungen markieren

Zur Laufzeit können Sie im Markiermodus eine einzelne Verbindung markieren, indem Sie sie mit der linken Maustaste anklicken. Mehrere Verbindungen lassen sich sammeln und toggeln, indem Sie sie bei gedrückter Strg-Taste mit der linken Maustaste anklicken.

### > Verbindungen bearbeiten

Sie können eine Verbindung bearbeiten, indem Sie sie mit der rechten Maustaste anklicken und dann im Kontextmenü den Befehl **Bearbeiten** wählen. Dann erscheint der Dialog **Verbindung bearbeiten**, in dem Sie alle Daten der Verbindung bearbeiten können.

### > **Verbindungen löschen**

Eine Verbindung können Sie löschen, indem Sie sie mit der rechten Maustaste anklicken und dann im Kontextmenü den Befehl **Löschen** wählen. Außerdem können Sie Verbindungen über die API mit der Methode **VcNet.DeleteLinkRecord** oder mit der Methode **VcLink.DeleteLink** löschen.

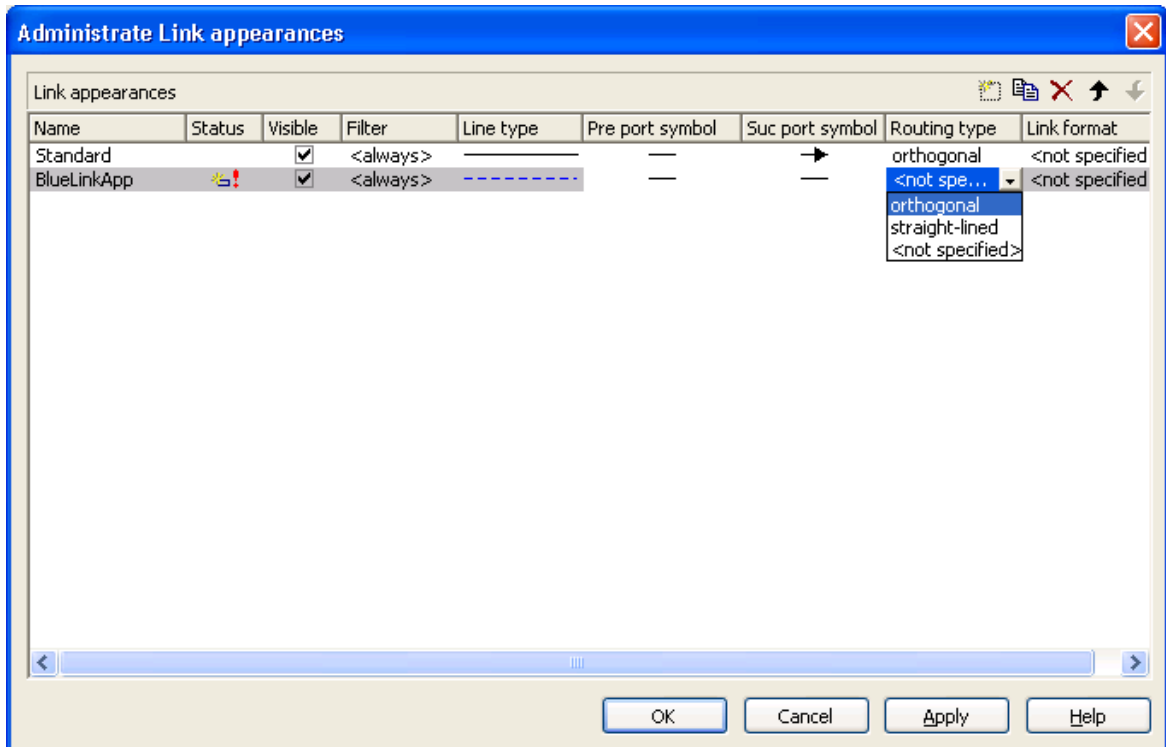
### > **Ereignisse**

Auf folgende Ereignisse können Sie reagieren:

- **VcLinkCreating**
- **VcLinkCreated**
- **VcLinkDeleting**
- **VcLinkDeleted**
- **VcLinksLeftClicking**
- **VcLinksLeftDoubleClicking**
- **VcLinksMarked**
- **VcLinksMarking**
- **VcLinkModified**
- **VcLinkModifying**
- **VcLinksRightClicking**

## 3.22 Verbindungsaussehen

Für die Verbindungen lassen sich im Dialog **Linienaussehen verwalten** unterschiedliche Verbindungsaussehen definieren, die den Verbindungen dynamisch über Filter zugewiesen werden.



Die **Name**-Spalte enthält die Namen aller verfügbaren Verbindungsaussehen.

### > Neues Verbindungsaussehen definieren

Klicken Sie auf die **Neu**-Zeile, um ein neues Verbindungsaussehen zu definieren.

### > Verbindungsaussehen löschen

Mit Hilfe der **Entf**-Taste können Sie ein Verbindungsaussehen aus der **Aussehen**-Tabelle löschen.

### > Filter festlegen

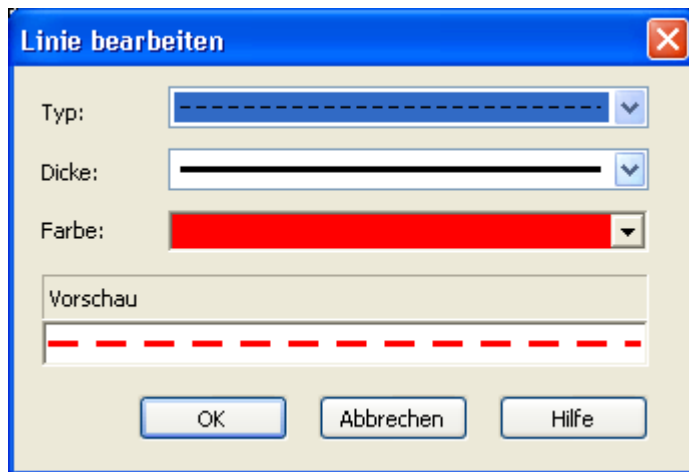
Um den Filter, der für ein bestimmtes Verbindungsaussehen verwendet werden soll, auszuwählen oder neu zu definieren, klicken Sie auf dessen **Filter**-Feld. Klicken Sie auf die Pfeil-Schaltfläche neben dem Filternamen, um die Kombobox mit allen verfügbaren Filtern zu öffnen, und wählen Sie hier einen Filter aus. Oder klicken Sie auf die **Bearbeiten**-Schaltfläche, um das Dialogfeld **Filter verwalten** zu öffnen. Hier können Sie Filter neu

## 140 Wichtige Konzepte: Verbindungsaussehen

definieren, kopieren, bearbeiten und löschen. Änderungen, die Sie an einem Filter durchführen, gelten für jedes Verbindungsaussehen, das mit diesem Filter verbunden ist.

### > Verbindungslinien festlegen

Wenn Sie auf den Eintrag des Feldes **Linienart** klicken, erscheint eine **Bearbeiten**-Schaltfläche, über die Sie das Dialogfeld **Linie bearbeiten** öffnen können. Hier können Sie das Aussehen der Verbindungslinien festlegen.



### > Weitere Festlegungen des Verbindungsaussehens

Weitere Informationen zum Verbindungsaussehen finden Sie im Kapitel 4.28 "Das Dialogfeld Verbindungsaussehen verwalten".

---

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

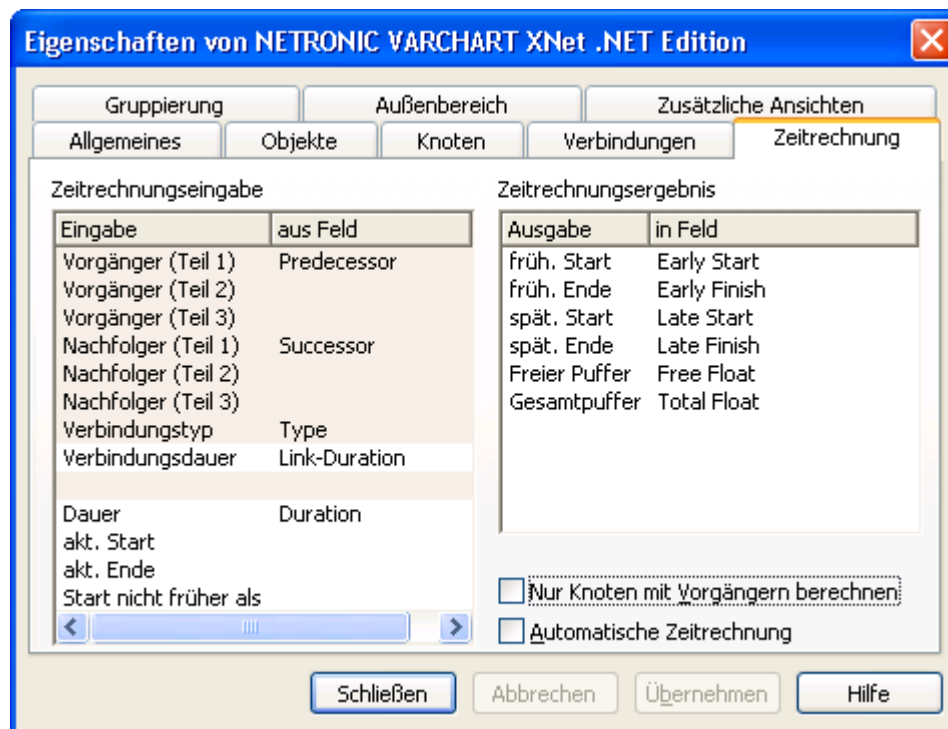
Viewer Metafile (VMF) ist ein Grafikformat, das speziell für den von der NETRONIC Software GmbH entwickelten WebViewer (ein plattform- und browserunabhängiges Java Applet) entwickelt worden ist. Das VMF-Format ermöglicht es, Ihre Diagramme über das Intra- bzw. Internet in einem Browser zu betrachten, zu zoomen oder zu drucken.

Mit der Methode **ExportGraphicsToFile** des Objektes VcNet oder mit dem Standard-Kontextmenü für das Diagramm können Sie ein Diagramm in einer Datei abspeichern.

## 3.24 Zeitrechnung

Mit dem Scheduler von VARCHART XNet können Sie einfache Terminberechnungen durchführen. Die gewünschten Projektstart- und Projektende-Termine werden dabei als Parameter übergeben.

Mit Hilfe der Eigenschaftenseite **Zeitrechnung** können Sie die Zeitrechnung des VARCHART XNet an Ihre Schnittstelle anpassen, indem Sie festlegen, welche Datenfelder für die Eingabe (**Zeitrechnungseingabe**) und die Ausgabe (**Zeitrechnungsergebnis**) des Schedulers verwendet werden sollen. Außerdem können Sie die Zeiteinheit festlegen, die für die Berechnung der Dauer in den entsprechenden Datenfeldern in Knoten und Verbindungen verwendet werden soll.



Unter **Zeitrechnungseingabe** können Sie für jede Eingabe auswählen, aus welchem Feld sie entnommen werden soll. Als Ein- und Ausgaben für die Zeitrechnung verwendet der Scheduler Datenfelder der jeweils eingestellten Knoten- und Verbindungstabellen.

Die Grundlage der Berechnung sind die Dauer der einzelnen Vorgänge, deren logische Abhängigkeiten und der Projektanfang. Daraus werden die frühesten bzw. spätesten Start- und Endtermine sowie der Gesamtpuffer und der freie Puffer berechnet. Die Felder **Vorgänger**, **Nachfolger** und **Verbindungstyp** können in der **Zeitrechnungseingabe**-Tabelle nicht bearbeitet werden. Sie geben nur die auf der Eigenschaftenseite **Verbindungen** vorgenommenen Festlegungen wieder.

Die Ausgaben werden wiederum in Datenfelder der Schnittstelle geschrieben. Als Ausgaben stehen zur Verfügung: **früh. Start**, **früh. Ende**, **spät. Start**, **spät. Ende**, **Freier Puffer** und **Gesamtpuffer**. Jeder dieser Ausgaben können Sie ein Feld aus der in der Datendefinition vereinbarten Liste von Feldern zuweisen.

Es gibt folgende Möglichkeiten, die Zeitrechnung zu beeinflussen:

1. Sie können einen Projektstart vorgeben. Das erreichen Sie über die API mit der VcNet-Methode **ScheduleProject**:

```
VcNet1.ScheduleProject "04.05.2000", 0
```

Mit der Methode **ScheduleProject** können Sie eine Vorwärts- und Rückwärtsberechnung des aktuellen Projekts durchführen. Bei Übergabe des Starttermins wird zunächst eine Vorwärtsberechnung, dann eine Rückwärtsberechnung durchgeführt. Bei Übergabe des Endtermins wird zunächst eine Rückwärtsberechnung, dann eine Vorwärtsberechnung durchgeführt. Es können auch Anfangs- und Enddatum übergeben werden, die Vorgänge erhalten dann entsprechende Pufferzeiten.

#### Mögliche Wahl der Parameter für die Methode **ScheduleProject**:

Anfang	Ende
Termin 1	0
0	Termin 2
Termin 1	Termin 2

2. Sie können aktuelle Start- bzw. Endtermine angeben. Die Knoten sind dann unverrückbar.
3. Sie können für die Bedingungen "Start nicht früher als" und "Ende nicht später als" Referenztermine angeben. Dazu wird auf der Eigenschaftenseite **Zeitrechnung** in der linken Tabelle für die entsprechenden Werte auch jeweils ein Feld aus der Datendefinition festgelegt. Dann liegt der früheste Start eines Knoten nicht vor dem angegebenen Termin bzw. das späteste Ende nicht nach dem angegebenen Termin.

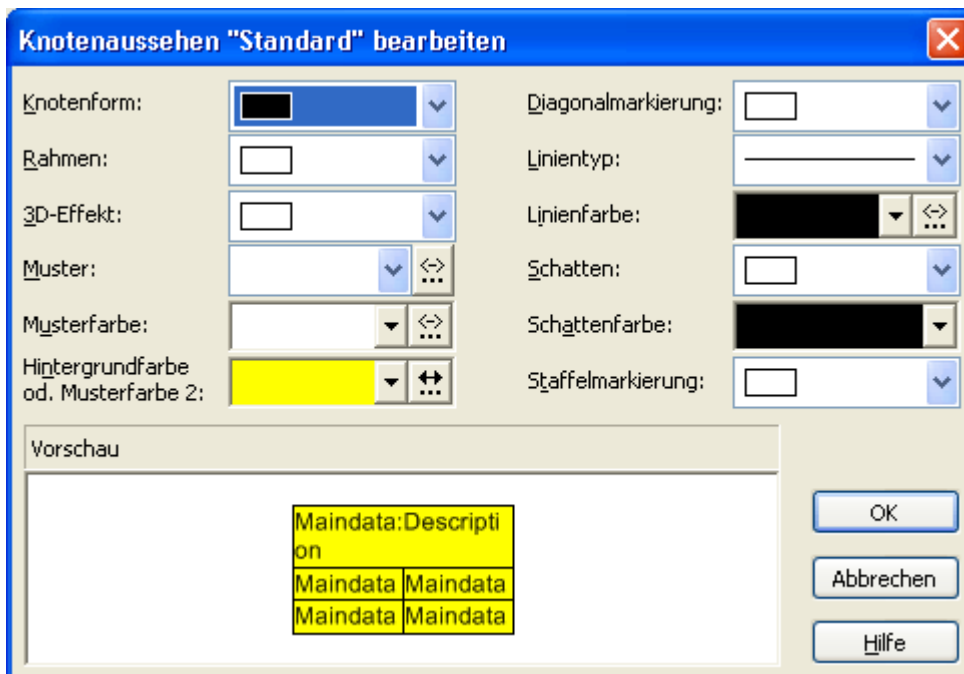


## 3.25 Zuordnungstabellen

Das Knotenaussehen und das Knotenformat können über Zuordnungstabellen datenabhängig gestaltet werden.

### > Knotenaussehen datenabhängig festlegen

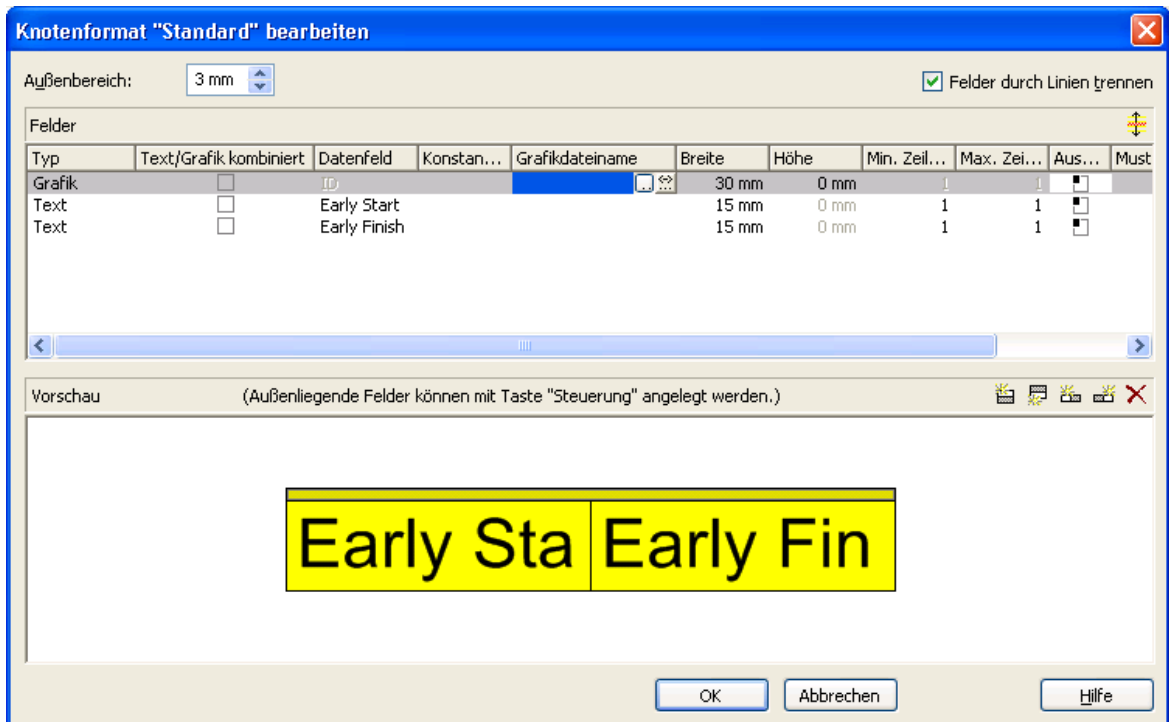
Für jedes Knotenaussehen können Muster, Musterfarbe, Hintergrund- oder 2. Musterfarbe sowie Linienfarbe über Zuordnungstabellen datenabhängig festgelegt werden. Klicken Sie dazu im Dialogfeld **Knotenaussehen bearbeiten** auf die zweite Schaltfläche für die **Hintergrundfarbe** bzw. **Linienfarbe** (☰).



Sie gelangen in den Dialog **Zuordnung einstellen**.

### > Grafik eines Knotenformats datenabhängig festlegen

In den Feldern der Knotenformate können Grafiken über eine Zuordnungstabelle datenabhängig ausgegeben werden.

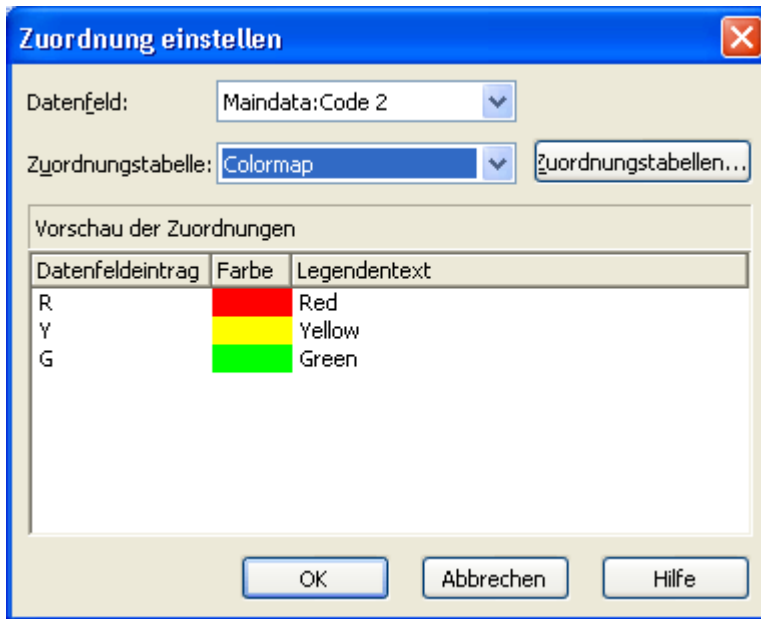


Um eine Zuordnung zwischen den Einträgen eines Datenfeldes vom Typ Grafik und Grafikdateien herzustellen, klicken Sie im Feld **Grafikdateiname** auf die 2. Schaltfläche (**Zuordnungen einstellen**). Der gleichnamige Dialog erscheint dann.

Wenn Sie dort eine Zuordnung vorgenommen haben, erscheint ein Symbol () im Feld **Grafikdateiname**, sobald Sie die entsprechende Zeile verlassen.

### > Zuordnung einstellen

Im Dialogfeld **Zuordnung einstellen** können Sie festlegen, dass in einem bestimmten Knotenformatfeld vom Typ Grafik datenabhängig Grafikdateien dargestellt werden sollen, bzw. dass die Hintergrundfarbe eines bestimmten Knotenaussehens datenabhängig sein soll.



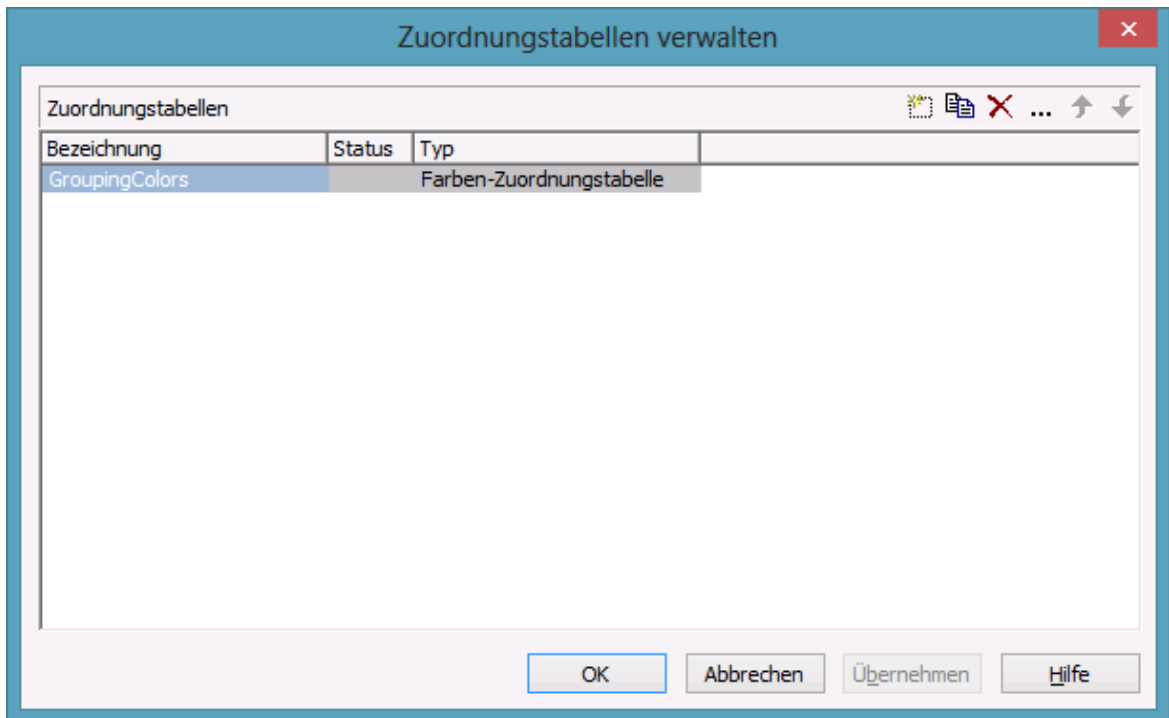
Wählen Sie dazu in der ersten Kombobox das **Datenfeld**, von dessen Einträgen die Grafikdatei des zu bearbeitenden Knotenformatfeldes bzw. die Hintergrundfarbe des zu bearbeitenden Knotenaussehens abhängen soll. Wählen Sie dann in der zweiten Kombobox die **Zuordnungstabelle**, die den einzelnen Datenfeldeinträgen eine Grafikdatei bzw. eine Farbe und einen Legendentext zuordnet.

In der **Vorschau der Zuordnungen** wird dargestellt, wie die gewählte Zuordnungstabelle den einzelnen Datenfeldeinträgen eine Grafikdatei bzw. eine Hintergrundfarbe und einen Legendentext zuordnet.


### > **Zuordnungstabellen verwalten**

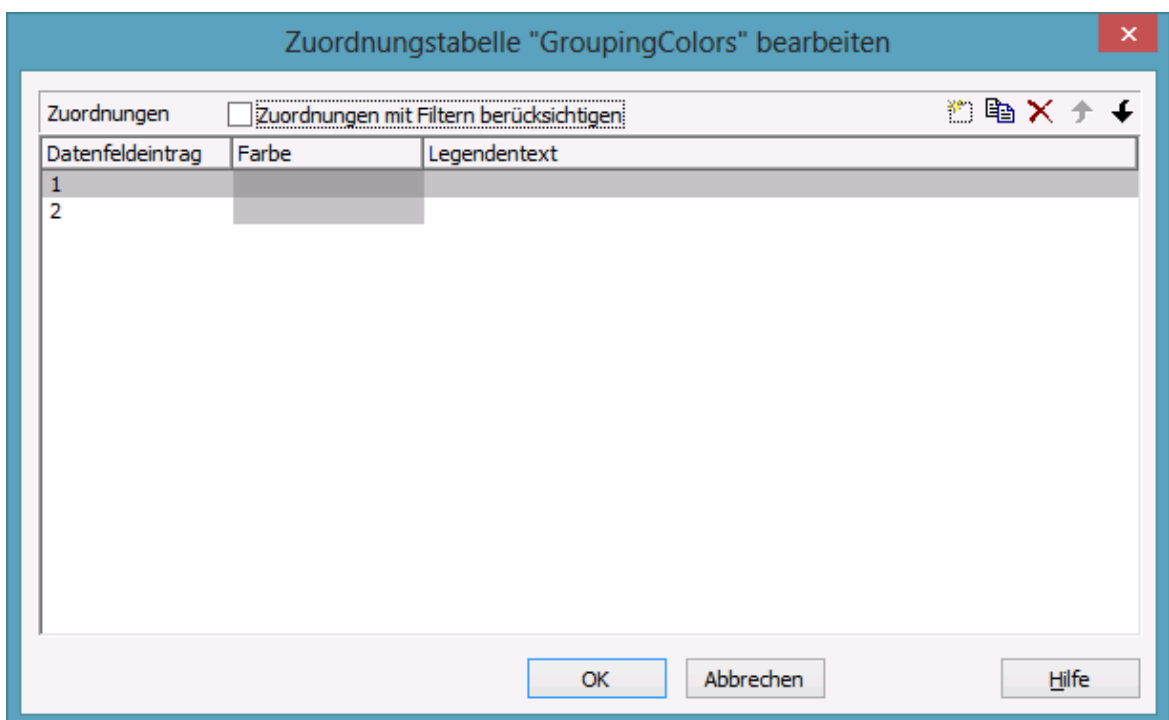
Im Dialogfeld **Zuordnungstabellen verwalten**, das Sie durch Klick auf die Schaltfläche **Zuordnungstabellen** oder über die Schaltfläche **Zuordnungstabellen** auf der Eigenschaftenseite **Objekte** erreichen, können Sie Namen und Typ einer Zuordnungstabelle durch direkte Eingabe verändern sowie über die entsprechenden Schaltflächen oben rechts im Fenster **Zuordnungstabellen** erstellen, kopieren, löschen oder bearbeiten.

Sie können aus verschiedenen Typen von Zuordnungstabellen auswählen, je nachdem, ob den Datenfeldinhalten Farben, Muster, Grafiken, Schrifttypen, Längen oder Nummern zugeordnet werden sollen.



### > Zuordnungstabellen bearbeiten

Um eine Zuordnungstabelle zu bearbeiten, markieren Sie diese in der Tabelle und klicken Sie auf die Schaltfläche  oberhalb der Tabelle. Es erscheint das Dialogfeld **Zuordnungstabelle bearbeiten**.



In der **Zuordnungen**-Tabelle werden für jeden Schlüssel die entsprechenden Werte aufgelistet, in unserem Beispiel sind dies die Hintergrundfarbe und der Legendentext.

## 148 Wichtige Konzepte: Zuordnungstabellen

Über die Schaltflächen oben rechts können Sie Schlüssel (Zuordnungen) hinzufügen, kopieren oder löschen oder deren Reihenfolge verändern.

Wenn die Option **Zuordnungen mit Filtern berücksichtigen** ausgewählt ist, werden nicht nur die in der Liste der Datenfeldeinträge angegebenen festen Werte als Schlüssel berücksichtigt, sondern auch Filter, die aus der Dropdown-Liste ausgewählt werden können. Dadurch hängen die Ausführungswerte nicht mehr nur von einem konkreten Wert, sondern von komplexeren Kriterien ab.

Sie können in einer Zuordnungstabelle maximal 150 Zuordnungen festlegen. Falls Sie weitere Zuordnungen benötigen, erstellen Sie einfach eine neue Zuordnungstabelle, z. B. als Kopie der bereits vorhandenen.

Einzelheiten zu den hier beschriebenen Dialogfeldern finden Sie im Kapitel "Eigenschaftenseiten und Dialogfelder".

### > **Anpassung der Zuordnungstabelle zur Laufzeit**

Sie können die Zuordnungstabellen auch zur Laufzeit noch mit Hilfe der VcMap-Methoden anpassen. Damit geben Sie dem Anwender die Möglichkeit, Ihre Voreinstellungen über einen von Ihnen erstellten Dialog zu verändern.

---

---

## 4 Eigenschaftenseiten und Dialogfelder

---


---

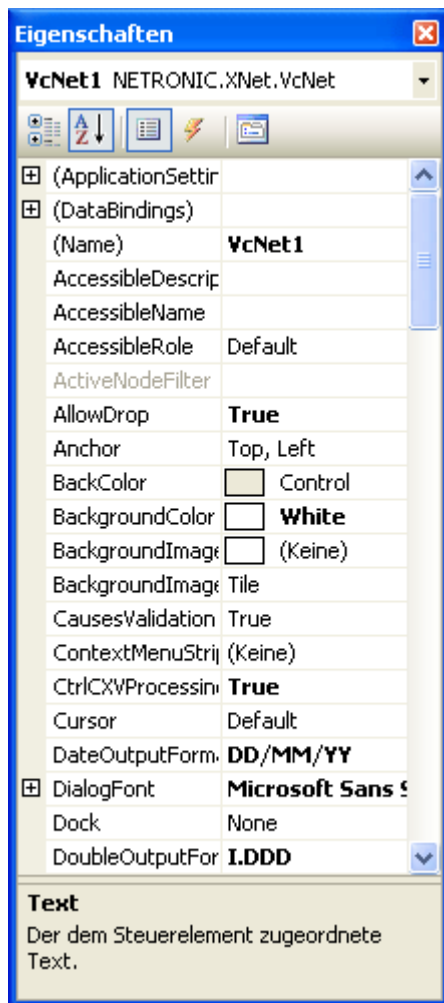
### 4.1 Allgemeines

Über die Eigenschaftenseiten kann VARCHART XNet bereits zur Entwurfszeit konfiguriert werden. Es gibt zwei Möglichkeiten, zu den Eigenschaftenseiten zu gelangen:

- Drücken Sie die rechte Maustaste, wenn der Mauszeiger sich innerhalb des Steuerelements befindet, und wählen Sie im Kontextmenü den Befehl **Eigenschaften** aus.

oder

- Im Eigenschaftfenster (kann mit F4 geöffnet werden) klicken Sie in der Symbolleiste auf das ganz rechts stehende Symbol .



Nähere Informationen zu jeder Eigenschaftenseite bzw. jedem Dialogfeld erhalten Sie, indem Sie auf die **Hilfe**-Schaltfläche klicken oder die F1-Taste drücken. Sie erhalten dann direkt die Online-Hilfe zu der Eigenschaftenseite bzw. dem Dialogfeld.

## 4.2 Eigenschaftenseite "Allgemeines"



Auf dieser Eigenschaftenseite können Sie allgemeine Einstellungen für VARCHART XNet vornehmen.

### Flussrichtung

Legen Sie hier fest, ob die Knoten im Netzdiagramm von links nach rechts oder von oben nach unten angeordnet werden sollen.

### Minimale Spaltenbreite

Legen Sie hier die minimale Spaltenbreite in Millimetern fest. Die eingestellte Breite sollte etwa der mittleren Breite eines Knotens entsprechen. Damit Verbindungen bei der Orientierung von links nach rechts weniger Platz auf dem Bildschirm benötigen, kann die Breite weiter herabgesetzt werden.

### Minimale Zeilenhöhe

Legen Sie hier die minimale Zeilenhöhe in Millimetern fest. Die eingestellte Höhe sollte etwa der mittleren Höhe eines Knotens entsprechen. Damit Verbindungen bei der Orientierung von oben nach unten weniger Platz auf dem Bildschirm benötigen, kann die Höhe weiter herabgesetzt werden.



## Hintergrundfarbe

Wählen Sie hier die Hintergrundfarbe für das Netzdiagramm aus.

## Zeiteinheit

Der hier eingestellte Wert wird für die Berechnung der Dauer (siehe Kapitel "Wichtige Begriffe: Layer") und für das interaktive Verändern und Verschieben Ihrer Knoten in der Darstellung verwendet.

**Beispiel:** Haben Sie hier die Zeiteinheit "Tage" gewählt, lassen sich Knoten nur in Sprüngen von so vielen ganzen Tagen verschieben, wie unter **Kleinstes Zeitintervall** definiert wurde.

Diese Option kann auch über die Eigenschaft **VcNet.TimeUnit** gesetzt werden.

## Datumsausgabeformat

Wählen Sie aus der Kombobox das Datumsformat aus, in dem Ihre Daten ausgegeben werden sollen, oder definieren Sie Ihr eigenes Datumsformat. Dieses Format gilt auch für alle im Steuerelement integrierten Dialogfelder, die zur Laufzeit verfügbar sind.

Diese Option kann auch über die Eigenschaft **VcNet.DateOutputFormat** festgelegt werden.

Für das Datum stehen folgende Kürzel zur Verfügung:

- D: Wochentagsname erster Buchstabe (nicht anpassbar)
- TD: Wochentagsname (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- DD: Tagesnummer zweistellig: 01-31 (nicht anpassbar)
- DDD: die ersten drei Buchstaben des Wochentagsnamens (nicht anpassbar)
- M: erster Buchstabe des Monatsnamens (nicht anpassbar)
- TM: Monatsname (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- MM: zweistellige Monatsnummer: 01-12
- MMM: erste drei Buchstaben des Monatsnamens (nicht anpassbar)
- YY: zweistellige Jahreszahl
- YYYY: vierstellige Jahreszahl

- WW: zweistellige Nummer der Kalenderwoche: 01-53
- TW: Text für "Kalenderwoche" (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- Q: einstellige Quartalsnummer: 1-4
- TQ: Quartalsname (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- hh: Stunde zweistellig im 24-Stunden-Format: 00-23
- HH: Stunde zweistellig im 12-Stunden-Format: 01-12
- Th: Text für "o' clock" (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- TH: "am" oder "pm" (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- mm: Minute zweistellig: 00-59
- ss: Sekunde zweistellig: 00-59
- TS: kurzes Datumsformat, wie in der Windows-Systemsteuerung über die Regions- und Sprachoptionen definiert
- TL: langes Datumsformat, wie in der Windows-Systemsteuerung über die Regions- und Sprachoptionen definiert
- TT: Zeitformat, wie in der Windows-Systemsteuerung über die Regions- und Sprachoptionen definiert

**Hinweis:** Zeichen, die nicht als Teil des Datums interpretiert werden sollen, sollten mit einem vorangehenden rückwärtigen Schrägstrich '\' gekennzeichnet werden. '\\' zum Beispiel ergibt \'. Die Sonderzeichen ':, /, -' und **Leerzeichen** benötigen keinen '\' als Präfix.

## Double-Ausgabeformat

Markieren Sie in der Kombobox das Format, in dem Ihre Daten vom Typ **Double** ausgegeben werden sollen. Sie können wählen zwischen **I** (ganze Zahl), **I,DDD**, **I,DDDDDD** bzw. **I.DDD**, **I.DDDDDD** (3 oder 6 Dezimalstellen) und **\$ I,III.DD** bzw. **I.III,DD €** (Währung mit 2 Dezimalstellen).

Diese Option kann auch über die Eigenschaft **VcNet.DoubleOutputFormat** festgelegt werden.

## Konfiguration

Alle Einstellungen der Eigenschaftenseiten können Sie auch jederzeit in Form einer Konfiguration außerhalb Ihres Projektes speichern und nach Bedarf wieder einlesen. Dies ist sehr praktisch, wenn Sie zu einem früheren Stand der Einstellungen zurückkehren oder die gleichen Einstellungen für andere Projekte verwenden möchten.

Eine gespeicherte Konfiguration besteht aus zwei Dateien mit gleichem Namen aber unterschiedlichen Dateierendungen. Zu einer Konfiguration gehört jeweils eine INI- und eine IFD-Datei, die beide zwingend benötigt werden.

Als Konfigurationsdatei können Sie eine lokale Datei mit Pfad oder eine URL angeben.

Die Angabe einer URL als Konfigurationsdatei ist nur sinnvoll, wenn die Konfiguration über die API zur Laufzeit festgelegt wird, da dann die INI- und IFD-Dateien von der angegebenen URL heruntergeladen werden. (Gibt man schon zur Designzeit eine URL als Konfigurationsdatei an, werden die INI- und IFD-Dateien zwar ebenfalls heruntergeladen, aber als Ressource dem Projekt hinzugefügt und zur Laufzeit verwendet, statt die Dateien direkt herunterzuladen.)

### So speichern Sie Ihre aktuelle Konfiguration:

Klicken Sie auf die Schaltfläche **Exportieren als...** und vergeben im folgenden Dialog den gewünschten Dateinamen für die INI-Datei. Die IFD-Datei wird automatisch erzeugt.

### So lesen Sie eine bereits gespeicherte Konfiguration wieder ein:

Klicken Sie auf die Schaltfläche **Importieren...** und wählen die gewünschte Datei aus.

## Erweiterte Datentabellen zulassen

Wenn Sie dieses Kontrollkästchen aktivieren, können Sie mehr (bis zu 99) Datentabellen anlegen und nutzen, als die standardmäßig vorhandenen **Main data** und **Relations**. Diese Option kann auch über die Eigenschaft **VcNet.ExtendedDataTablesEnabled** gesetzt werden.

## In-Place-Editieren zulassen

Aktivieren Sie dieses Kontrollkästchen, wenn das direkte Editieren in Knotenfeldern und in Boxen möglich sein soll. Diese Option kann auch über die Eigenschaft **VcNet.InPlaceEditingAllowed** gesetzt werden.

Wenn einzelne Datenfelder nicht editierbar sein sollen, so darf in der Datendefinition die Option **editierbar** nicht ausgewählt werden.

## Strg-C, -X und -V verarbeiten

Aktivieren Sie dieses Kontrollkästchen, damit die Tastenkombinationen Strg+C, Strg+X und Strg+V automatisch in die Zwischenablage-Operationen **CopyNodesToClipboard**, **CutNodesToClipboard** bzw. **PasteNodesFromClipboard** übersetzt werden. Dieses Verhalten kann abgeschaltet werden, damit in Visual Basic kein Konflikt mit Bearbeitungsmethoden für Menüpunkte entsteht, die dieselben Tastaturkombinationen benutzen. Diese Option kann auch über die Eigenschaft **VcNet.CtrlCXVProcessingEnabled** gesetzt werden.

## Mehrfache Box-Markierung zulassen

Wenn Sie dieses Kontrollkästchen aktivieren, können zur Laufzeit mehrere Boxen gleichzeitig durch Anklicken markiert werden, ohne gleichzeitig die STRG-Taste gedrückt halten zu müssen. Diese Option ist standardmäßig deaktiviert.

Diese Option kann auch über die Eigenschaft **VcNet.MultipleBoxMarkingAllowed** gesetzt werden

## Zoomen per Mausrad zulassen

Aktivieren Sie dieses Kontrollkästchen, um das Zoomen per Mausrad zuzulassen. Um zu zoomen, muss der Anwender die Strg-Taste festhalten und das Mausrad drehen.

Diese Option kann auch über die Eigenschaft **VcNet.ZoomingPerMouseWheelAllowed** gesetzt werden.

## VcToolTipTextSupplying-Ereignisse

Aktivieren Sie dieses Kontrollkästchen, um das Ereignis **VcToolTipTextSupplying** freizuschalten. Dies kann auch durch die Eigenschaft **ToolTipTextSupplyingEventEnabled** geschehen. Mit Hilfe dieses Ereignisses können Sie die Texte der Tooltips, die bei Objekten angezeigt werden sollen, festlegen.

## VcTextEntrySupplying-Ereignisse

Aktivieren Sie dieses Kontrollkästchen, um das Ereignis **VcTextEntrySupplying** freizuschalten. Mit Hilfe dieses Ereignisses können Sie die Texte aller Kontextmenüs, Dialogfelder, Infoboxen und Fehlermeldungen, die zur Laufzeit erscheinen, verändern, beispielsweise um sie in unterschiedliche Sprachen zu übersetzen.

Diese Option kann auch über die Eigenschaft **VcNet.TextEntrySupplyingEventEnabled** gesetzt werden.

## Erzeugung neuer Knoten mit Dialog

Wenn Sie diese Option wählen, öffnet sich das Dialogfeld **Vorgänge bearbeiten** automatisch, wenn der Anwender interaktiv einen neuen Knoten erzeugt hat. Das Dialogfeld **Vorgänge bearbeiten** kann - auch wenn diese Option deaktiviert ist - nach dem Erzeugen eines Knotens durch einen Doppelklick auf diesen geöffnet werden.

Diese Option kann auch über die Eigenschaft **VcNet.NodeCreationWithDialog** eingestellt werden.

## Erzeugung neuer Verbindungen mit Dialog

Wenn Sie diese Option wählen, öffnet sich das Dialogfeld **Daten bearbeiten** automatisch, wenn der Anwender interaktiv eine neue Verbindung erzeugt hat. Das Dialogfeld **Daten bearbeiten** kann nach dem Erzeugen einer Verbindung durch einen Doppelklick auf die Verbindung geöffnet werden.

## Erzeugung neuer Knoten und Verbindungen zulassen

Nur wenn Sie diese Option ankreuzen, lassen Sie zu, dass der Anwender in einem geöffneten Projekt im Erzeugemodus interaktiv neue Knoten und Verbindungen erzeugt. Ein neuer Knoten wird dann erzeugt, wenn der Anwender mit der linken Maustaste in die Diagrammfläche klickt. Eine neue Verbindung wird erzeugt, indem der Anwender den Cursor auf einem Knoten positioniert und den Cursor dann mit gedrückter Maustaste zu einem anderen Knoten zieht oder einen neuen Knoten erzeugt. Sobald die Maustaste losgelassen wird, wird die Verbindung zwischen den beiden Knoten gezogen.

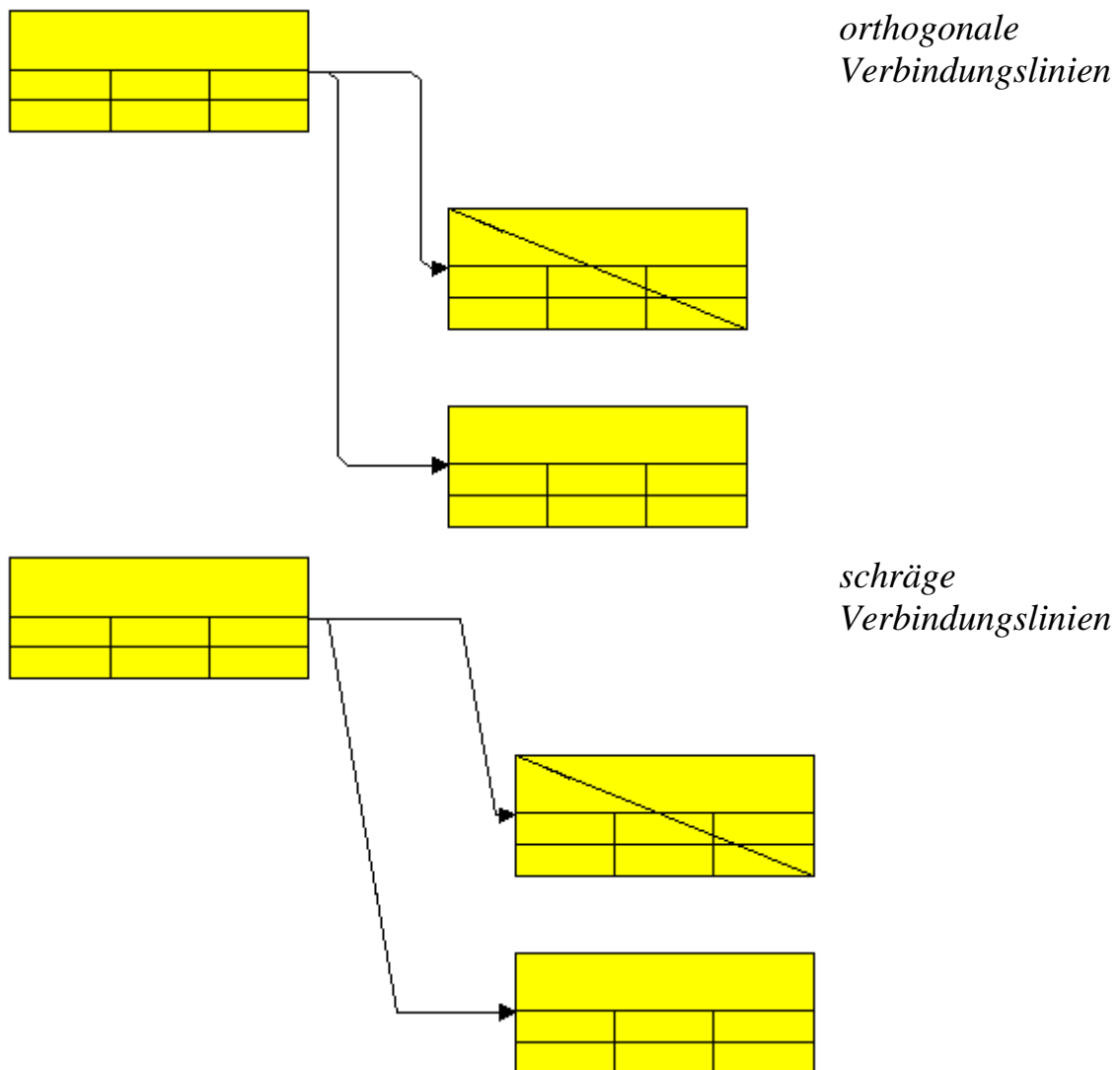
## Verbindungen kürzen beim Anordnen

Diese Eigenschaft wirkt sich auf das Layout eines Netzdiagramms aus und wird von der Methode **Arrange** berücksichtigt. Wird die Eigenschaft auf

True gesetzt, werden alle Knoten so nahe wie möglich an ihre Nachfolgerknoten herangeschoben, um die Verbindungen möglichst kurz zu halten. Wird sie auf False gesetzt, werden Knoten grundsätzlich so weit links bzw. oben wie möglich angeordnet, und so entstehen lange Verbindungslinien.

## Schräge Tracks bei Verbindungen

Aktivieren Sie dieses Kontrollkästchen, damit die Verbindungslinien direkt an den horizontalen Linienstücken ansetzen und schräg verlaufen. Andernfalls werden die Verbindungslinien orthogonal dargestellt. Alternativ können Sie dies über die VcNet-Eigenschaft **ObliqueTracksOnLinks** festlegen.



## Zeige Anschlussknoten im Teilnetz

Aktivieren Sie dieses Kontrollkästchen, wenn bei der Erstellung eines Teildiagramms auch die Anschlussknoten dargestellt werden sollen. Das Aussehen dieser Anschlussknoten können Sie im Dialog **Knotenaussehen bearbeiten** bestimmen, indem Sie als Spezialfilter <InterfaceNodes> angeben.

## Knoten verwenden Kalender

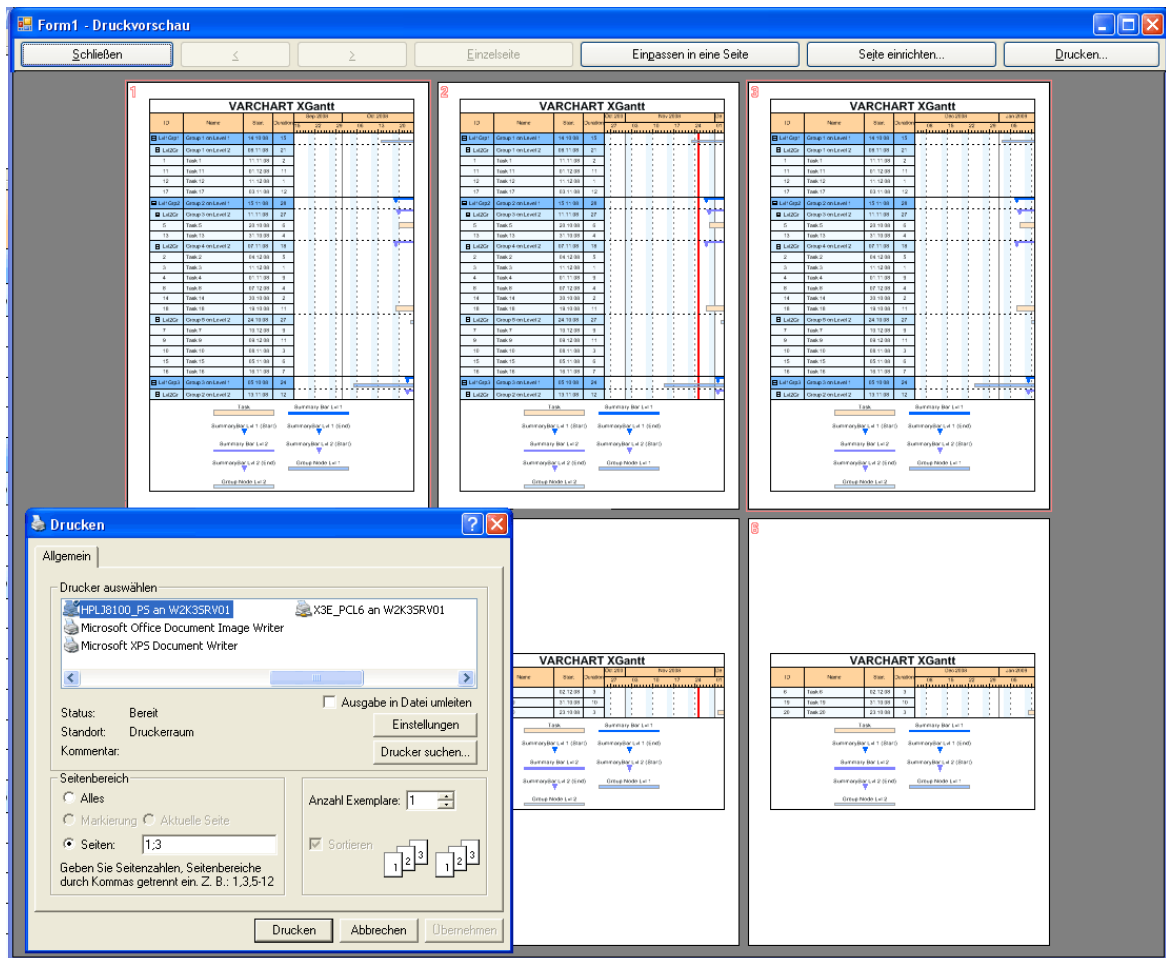
Mit dieser Eigenschaft können Sie festlegen, ob das Zeitrechnungsmodul (Scheduler) für die Zeitrechnung den Kalender benutzen soll. Dabei werden beim Berechnen der Dauer von Knoten die arbeitsfreien Zeiten berücksichtigt. Standardmäßig ist ein Fünf-Tage-Kalender definiert. Sie können aber über die Objekte VcCalendar, VcWorkweek und VcWorkday eigene Kalender definieren und einen davon mittels VcCalendarCollection.Active aktivieren.

## PrintDlgEx Dialog verwenden

Wenn diese Option ausgewählt ist, kann zur Laufzeit der Dialog **Drucker einrichten** weder aus dem Kontextmenü noch aus der Druckvorschau aufgerufen werden, da er sich nun im (erweiterten) **Drucken**-Dialog befindet. Bei neuen Projekten ist die Option standardmäßig aktiviert, bei bestehenden Projekten ist sie aus Gründen der Kompatibilität ausgeschaltet.

In der Druckvorschau können nun einzelne oder mehrere Seiten durch Klick bzw. STRG+Klick ausgewählt werden. Diese Seiten sind dann im Dialog **Drucken** unter **Seitenbereich** bereits voreingestellt.

Wenn man aus der Druckvorschau den **Drucken**-Dialog aufruft, sind alle Blätter mit einer Seitennummer versehen, um so die Seitenauswahl zu erleichtern.



Diese Option kann **nicht** an der API gesetzt werden.

## Abgerundete Schrägen bei Verbindungen

Aktivieren Sie dieses Kontrollkästchen, damit Schrägen bei Verbindungen vom Routing-Typ **vcLRTOrthogonal** als Viertelkreise statt als gerade Linien dargestellt werden. Diese Option kann auch über die VcNet-Eigenschaft **RoundedLinkSlantsEnabled** gesetzt werden.

## Wartecursor bei zeitkritischen Operationen eingeschaltet

Wählen Sie diese Option, wenn bei zeitkritischen Aufrufen (wie **ScheduleProject**) von uns intern ein Wartecursor gesetzt werden soll.

Die Option kann auch über die Eigenschaft **VcNet.WaitCursorEnabled** gesetzt werden.



## **Panning-Modus zulassen**

Diese Option muss aktiviert werden, damit Sie zur Laufzeit einen bestimmten Bildschirmausschnitt verschieben können. Im Kontextmenü des Diagramms erscheint dann der zusätzliche Eintrag **Verschiebe-Modus**.

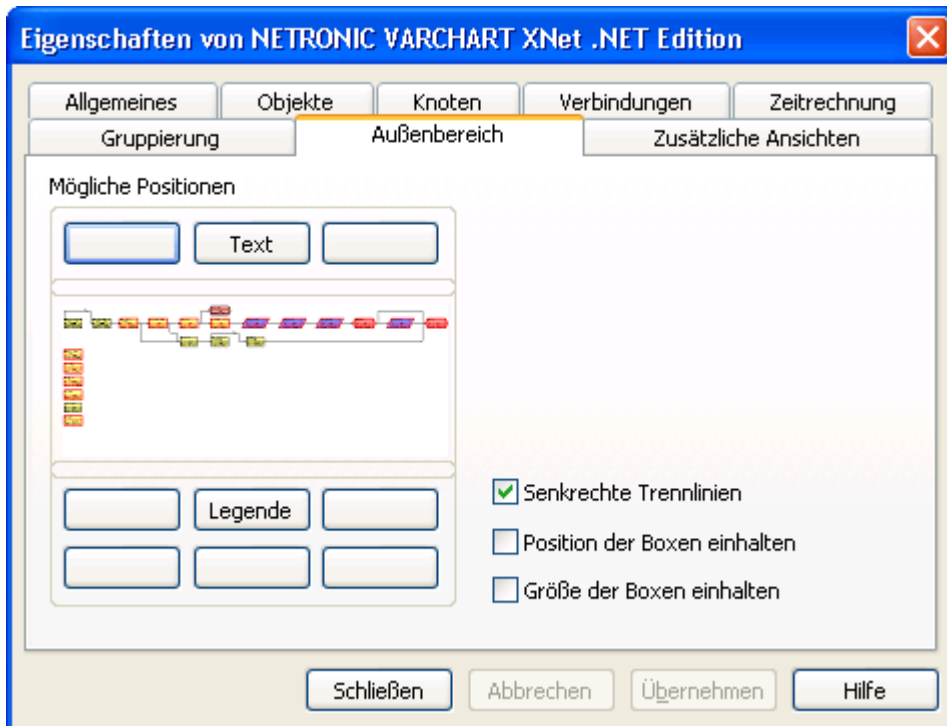
Der Verschiebemodus gilt standardmäßig für **alle** grafischen Grundelemente. Mithilfe der Eigenschaft **VcGantt.VcViewComponent** kann die Verwendung auf einzelne Elemente beschränkt werden.

Die Option kann auch über die Eigenschaft **VcNet.PanningModeAllowed** gesetzt werden.

## **Lizenzierung**

Über diese Schaltfläche gelangen Sie in den Dialog **Lizenzierung**. Für weitere Informationen s. Kap. **Lizenzierung**

## 4.3 Eigenschaftenseite "Außenbereich"



### Mögliche Positionen

Oberhalb der Grafik stehen Ihnen drei und unterhalb der Grafik sechs Bereiche zur Verfügung, in denen Sie Texte, Grafiken oder eine Legende plazieren können. Jeder dieser Bereiche wird in diesem Dialog durch je eine Schaltfläche repräsentiert. Alle diese Bereiche werden nur in der Seitenansicht und im Ausdruck angezeigt. Klicken Sie auf eine der Schaltflächen ober- bzw. unterhalb der Grafik, um den Dialog **Texte**, **Grafiken** und **Legende festlegen** zu öffnen.

### Senkrechte Trennlinien

Aktivieren Sie dieses Kontrollkästchen, wenn die Bereiche für Texte, Grafiken oder Legende durch senkrechte Trennlinien getrennt werden sollen.

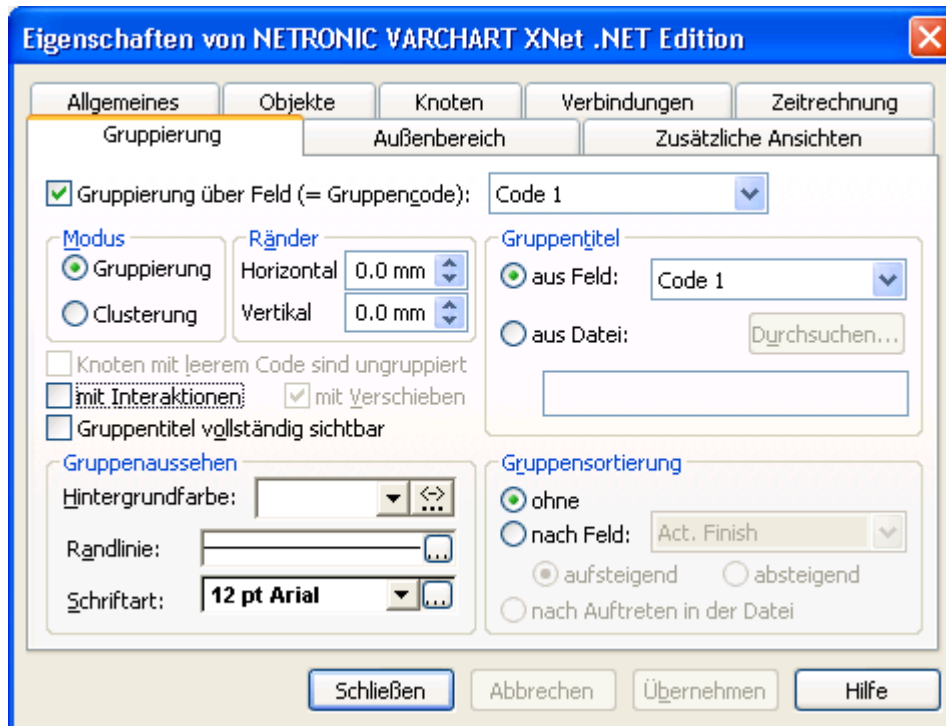
### Position der Boxen einhalten

Aktivieren Sie dieses Kontrollkästchen, wenn die Position der Boxen möglichst genau eingehalten werden soll. Andernfalls wird der vorhandene Platz proportional auf die in der jeweiligen Zeile vorhandenen Elemente verteilt.

## **Größe der Boxen einhalten**

Aktivieren Sie dieses Kontrollkästchen, wenn die Größe der Boxen möglichst genau eingehalten werden soll. Gegebenenfalls wird das Diagramm vergrößert und/oder die Texte in den Boxen abgeschnitten.

## 4.4 Eigenschaftenseite "Gruppierung"



### Gruppierung über Feld (= Gruppencode)

Aktivieren Sie dieses Kontrollkästchen, wenn die Knoten gruppiert dargestellt werden sollen. Nur wenn dieses Kontrollkästchen aktiviert ist, sind die übrigen Optionen dieser Eigenschaftenseite aktiviert.

Legen Sie hier fest, nach welchem Feld gruppiert werden soll. Das Feld, das Sie hier auswählen, wird **Gruppencode** genannt. Alle Vorgänge, die in dem hier gewählten Feld denselben Eintrag haben, werden in einer Gruppe zusammengefasst.

### Modus

Wählen Sie hier den Modus:

- **Gruppierung:** normale Gruppendarstellung (die Breite und Höhe jeder Gruppe wird durch die Positionen der Knoten bestimmt; jede Gruppe nimmt jeweils die gesamte Breite bzw. Höhe des Netzdiagramms ein)
- **Clustering:** Die Gruppen umschließen die darin enthaltenen Knoten möglichst platzsparend, wobei die Gruppen frei im Netzdiagramm verteilt sind.

## Ränder

Stellen Sie hier die Breite der horizontalen bzw. vertikalen Ränder der Gruppen ein. Möglich sind 0 bis 9,9 mm.

## Knoten mit leerem Code sind ungruppiert

*(nur für Modus Clusterung)* Wenn dieses Kontrollkästchen aktiviert ist, werden Knoten, deren Gruppencode der leere String "" ist, nicht innerhalb einer Gruppe dargestellt. Andernfalls werden sie in einer eigenen Gruppe für Vorgänge ohne Gruppencode-Angabe dargestellt.

## mit Interaktionen

Wenn das Kontrollkästchen aktiviert ist, können die Gruppen interaktiv kollabiert bzw. expandiert werden (mit Hilfe des Plus- bzw. Minus-Zeichens neben dem Gruppentitel).

## mit Verschieben

*(nur für Modus Clusterung)* Wenn das Kontrollkästchen aktiviert ist, können die Cluster interaktiv verschoben werden.

## Gruppentitel vollständig sichtbar

Wenn diese Option ausgewählt ist, sind beim horizontalen Rollen die Gruppentitel stets sichtbar.

## Hintergrundfarbe

Wählen Sie hier die Hintergrundfarbe für die Gruppen.

## Randlinie

Hier wird das Aussehen der Randlinien der Gruppen angezeigt. Um es zu bearbeiten, klicken Sie auf die **Bearbeiten**-Schaltfläche. Sie gelangen dann in das Dialogfeld **Linie bearbeiten**, in dem Sie Farbe, Typ und Dicke der Linien, mit denen die einzelnen Gruppen getrennt werden, festlegen können.

## Schriftart

In diesem Feld werden Schriftart und Schriftfarbe der Gruppentitel angezeigt. Um die Schriftfarbe zu bearbeiten, klicken Sie auf die Pfeil-Schaltfläche, um

die Farbauswahltable zu öffnen. Um die Schriftart zu bearbeiten, klicken Sie auf die **Bearbeiten**-Schaltfläche. Sie gelangen dann in das Windows-Dialogfeld **Schriftart**.

## Gruppentitel aus Feld

Aktivieren Sie dieses Kontrollkästchen, damit die Gruppentitel aus dem Datenfeld genommen werden, das Sie hier auswählen. Dieses Feld muss nicht notwendigerweise mit dem Gruppencode identisch sein. Doch damit die Gruppen sinnvoll beschriftet werden, sollten die Einträge im **Gruppencode**-Feld und im **Gruppentitel**-Feld miteinander korrespondieren.

## Gruppentitel aus Datei

Wenn Sie dieses Kontrollkästchen aktivieren, werden die Gruppentitel aus der Datei übernommen, die Sie hier auswählen. Über die **Durchsuchen**-Schaltfläche gelangen Sie in das Windows-Dialogfeld **Gruppentiteldatei auswählen**, in dem Sie die Datei öffnen können, aus der die Gruppentitel entnommen werden sollen.

Standardmäßig werden die Gruppentitel aus einer Datei vom Typ \*.txt gelesen. Sie können aber auch einen anderen Dateityp dafür festlegen.

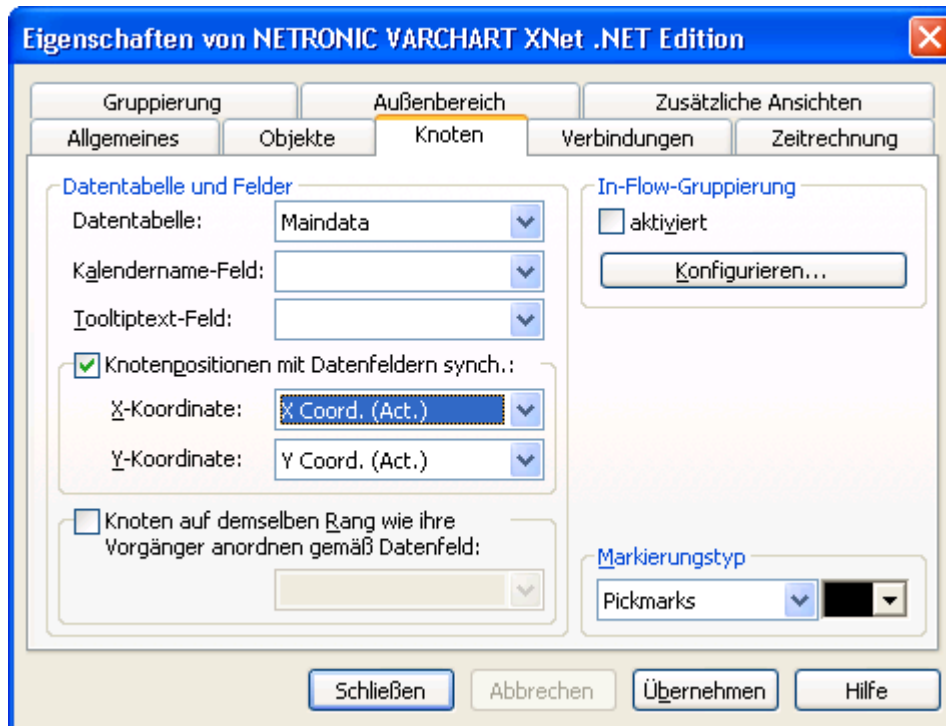
Wird ein relativer Dateiname angegeben, so wird die Datei zur Laufzeit zuerst in dem Verzeichnis gesucht, das in der Eigenschaft **FilePath** des Objektes VcNet gesetzt ist. Wird sie dort nicht gefunden, wird sie zuerst im gerade aktiven Arbeitsverzeichnis der Applikation und dann im Installationsverzeichnis des VARCHART-XNet-Steuerelements gesucht.

## Gruppensortierung

Hier können Sie festlegen, ob und ggf. nach welchem Kriterium die Gruppen sortiert werden. Sie können zwischen den folgenden Alternativen wählen:

- **ohne:** Die Gruppen werden nicht sortiert.
- **nach Feld:** Sie können ein Feld angeben, nach dem die Gruppen sortiert werden sollen, und festlegen, ob **aufsteigend** oder **absteigend** sortiert werden soll.
- **nach Auftreten in der Datei:** Die Gruppen werden in der Reihenfolge sortiert, in der sie in der Gruppentiteldatei auftreten.

## 4.5 Eigenschaftenseite "Knoten"



### Kalendernamen-Feld

Wenn individuelle Kalender für Knoten verwendet werden sollen, können Sie hier das Datenfeld auswählen, das den Namen des für einen Knoten zu verwendenden Kalenders enthalten soll.

Dazu muss auf der Eigenschaftenseite **Allgemeines** die Option **Scheduler benutzt internen Kalender** aktiviert sein. Außerdem müssen die Kalender vor dem Laden der Vorgänge erzeugt worden sein.

Diese Option kann auch über die Eigenschaft **VcNet.NodeCalendarName-DataFieldIndex** festgelegt werden..

### Tooltiptext-Feld

Das Datenfeld, das Sie hier auswählen, ist nur für den VMF-Export von Bedeutung. Wenn Sie eine VMF-Datei mit dem WebViewer ansehen und dort auf einen Knoten rechtsklicken, wird der Inhalt des gewählten Datenfeldes als Tooltip angezeigt. Es sind keine weiteren Einstellungen notwendig.

Um in Ihrer Applikation Tooltips anzuzeigen, müssen Sie auf der Eigenschaftenseite **Allgemeines** das Kontrollkästchen **VcToolTipTextSupplying-Ereignisse** aktivieren bzw. die Eigenschaft **ToolTipTextSupplyingEvent-**

**Enabled** von VcNet auf True setzen und im **VcToolTipTextSupplying**-Ereignis programmieren, welcher Text angezeigt werden soll.

Diese Option kann auch über die Eigenschaft **VcNet.NodeToolTipText-DataFieldIndex** gesetzt werden.

## Knotenpositionen mit Datenfeldern synchronisieren

Aktivieren Sie dieses Kontrollkästchen, um die Knotenpositionen mit Datenfeldern Ihrer Wahl zu synchronisieren. Wählen Sie dann je ein Datenfeld aus, aus dem die X- bzw. Y-Koordinate jeder Knotenposition gelesen und wohin sie zurückgeschrieben werden soll. Die Synchronisierung der Knotenpositionen mit den Datenfeldern ist dann erforderlich, wenn die Knotenpositionen nach dem Schließen Ihres Projekts wiederhergestellt werden müssen.

## Knoten auf demselben Rang wie ihre Vorgänger anordnen gemäß Datenfeld

Wenn bestimmte Knoten auf demselben Rang positioniert werden sollen wie ihre Vorgänger, aktivieren Sie dieses Kontrollkästchen. Wählen Sie dann das Datenfeld aus, dessen Eintrag bestimmt, ob ein bestimmter Knoten auf demselben Rang positioniert wird wie sein Vorgänger (z. B. das Datenfeld "Hilfsknoten"). Dieses Datenfeld müssen Sie ggf. im Dialog **Datentabellen verwalten** erst definieren. Es kann die Werte 0, 1, 2 oder 3 annehmen.

Wert im gewählten Datenfeld	Flussrichtung von oben nach unten	Flussrichtung von links nach rechts
0	Der Rang des Hilfsknotens wird nicht reduziert.	Der Rang des Hilfsknotens wird nicht reduziert.
1	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann links oder rechts neben seinem Vorgänger statt darunter.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann über oder unter seinem Vorgänger statt daneben.
2	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann links von seinem Vorgänger	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann über seinem Vorgänger.
3	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann rechts von seinem Vorgänger.	Der Rang des Hilfsknotens wird um 1 reduziert. Der Hilfsknoten steht dann unter seinem Vorgänger.



*Zur Erläuterung:* Unter dem Rang eines Knotens versteht man eine Zahl, die folgendermaßen definiert ist: Der Rang eines Knotens ohne Vorgänger ist 1. Der Rang eines Knotens mit Vorgängern ist gleich 1 plus Rang desjenigen seiner Vorgänger, der den höchsten Rang besitzt.

(Weitere Informationen finden Sie im Kapitel "Wichtige Begriffe: Knoten".)

## Markierungstyp

Wählen Sie hier aus, ob und welche Art der interaktiven Markierung von Knoten dem Anwender geboten werden soll. Folgende Alternativen stehen zur Verfügung:

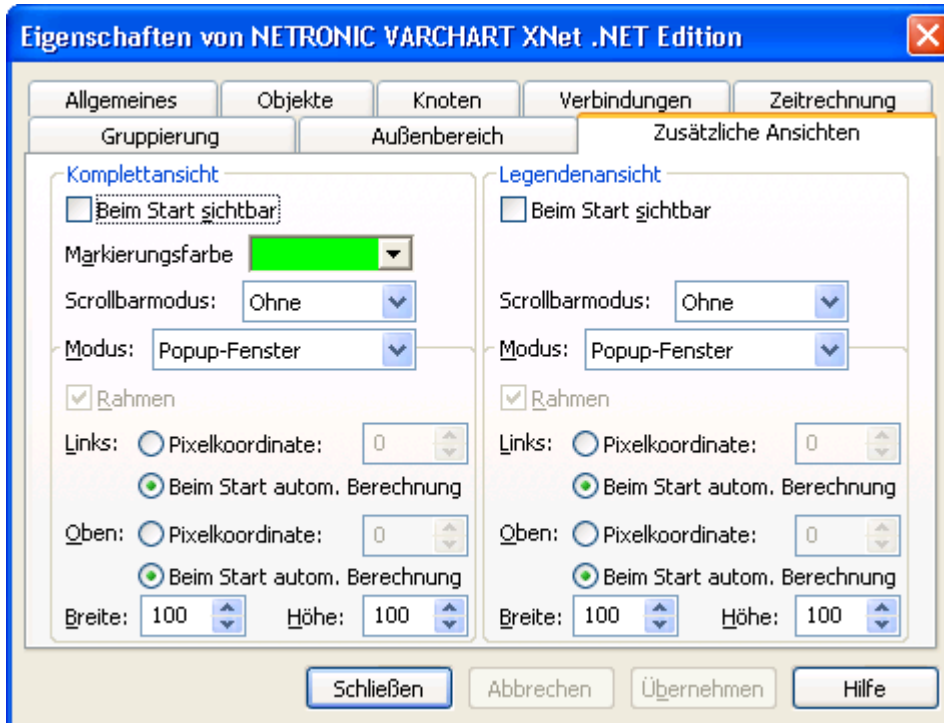
- ohne
- Einrahmen
- Einrahmen innen
- Invertieren
- Pickmarks
- Pickmarks innen

**Hinweis:** Wenn Sie den Markierungstyp "Ohne" ausgewählt haben, werden Knoten nicht grafisch markiert.

## In-Flow-Gruppierung

Über die Schaltfläche **Konfigurieren** lässt sich das Dialogfeld **In-Flow-Gruppierung bearbeiten** öffnen. Wenn das **aktiviert**-Kontrollkästchen aktiviert ist, ist die In-Flow-Gruppierung bei Programmstart aktiv.

## 4.6 Eigenschaftenseite "Zusätzliche Ansichten"



Auf dieser Eigenschaftenseite können Sie die Eigenschaften der Komplettansicht (World View) sowie der Legendenansicht (Legend View) festlegen. Die Komplettansicht ist ein zusätzliches Fenster, in dem das komplette Diagramm angezeigt wird. Ein Rahmen darin zeigt an, welchen Ausschnitt des Diagramms das Hauptfenster gerade anzeigt.

Mithilfe der Legendenansicht lässt sich, ebenfalls in einem zusätzlichen Fenster, eine Legende auf dem Bildschirm darstellen.

Um die Ansichten anzeigen zu lassen, wählen Sie zur Laufzeit im Standard-Kontextmenü für das Diagramm **Komplettansicht anzeigen** bzw. für die Legende **Legendenansicht anzeigen**. Über diese Menüpunkte können die Ansichten auch wieder ausgeschaltet werden (alternativ über die **Schließen**-Schaltfläche in der Titelleiste des jeweiligen Fensters).

im Folgenden sind die möglichen Einstellungen für beide Ansichten gleichzeitig beschrieben. Sollte ein Kriterium nicht für beide Ansichten gelten, so wird gesondert darauf hingewiesen.

### Beim Start sichtbar

Aktivieren Sie dieses Kontrollkästchen, damit die Ansicht beim Start des Programms sichtbar ist.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Visible** bzw. **VcLegendView.Visible** der Programmierschnittstelle gesetzt werden.

## Markierungsfarbe (nur für Komplettansicht)

Wählen Sie hier die Farbe der Linie des Rechtecks aus, das in der Komplettansicht den aktuell gewählten Ausschnitt anzeigt.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.MarkingColor** bzw. **VcLegendView.MarkingColor** der Programmierschnittstelle gesetzt werden.

## Scrollbarmodus

Wählen Sie hier, ob und welche Bildlaufleisten in der Ansicht dargestellt werden sollen. Durch die Verwendung von Bildlaufleisten werden Leerbereiche vermieden und das Diagramm bzw. die Legende ist besser zu erkennen, weil es bzw. sie größer dargestellt wird. Folgende Möglichkeiten stehen zur Verfügung:

- **Ohne:** In der Ansicht wird immer alles vollständig dargestellt. Dadurch können Leerbereiche entstehen, wenn die Ansicht in ihren Proportionen nicht denen des Charts( oder der Legende) entspricht.
- **Horizontal:** Es wird, wenn notwendig, eine horizontale Bildlaufleiste dargestellt.
- **Vertikal:** Es wird, wenn notwendig, eine vertikale Bildlaufleiste dargestellt.
- **Automatisch:** Es wird, wenn notwendig, ein horizontaler oder eine vertikale Bildlaufleiste dargestellt.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.ScrollBarMode** bzw. **VcLegendView.ScrollBarMode** der Programmierschnittstelle gesetzt werden.

## Modus

Wählen Sie hier den Modus für die Gesamtansicht aus. Es gibt folgende Möglichkeiten:

- **fest an linker Seite:** Die Ansicht wird links im Fenster des Steuerelementes angezeigt. Dann kann nur die Breite festgelegt werden, während Position und Höhe vorgegeben sind.

- **fest an rechter Seite:** Die Ansicht wird rechts im Fenster des Steuerelementes angezeigt. Dann kann nur die Breite festgelegt werden, während Position und Höhe vorgegeben sind.
- **fest an oberer Seite:** Die Ansicht wird oben im Fenster des Steuerelementes angezeigt. Dann kann nur die Höhe festgelegt werden, während Position und Breite vorgegeben sind.
- **fest an unterer Seite:** Die Ansicht wird unten im Fenster des Steuerelementes angezeigt.
- **nicht fest positioniert:** Die Ansicht ist ein untergeordnetes Kindfenster des aktuellen Vaterfensters des Steuerelementes und kann an beliebiger Position mit beliebiger Ausdehnung angeordnet werden. Das Vaterfenster kann bei Bedarf über die Eigenschaft **VcWorldView.ParentHWND** geändert werden.
- **Popup-Fenster:** Die Ansicht ist ein Popup-Fenster, das einen eigenen Rahmen besitzt und vom Benutzer in Position und Größe verändert werden kann. Es kann über das Standard-Kontextmenü ein- bzw. ausgeschaltet oder über die **Schließen**-Schaltfläche in der Titelleiste ausgeschaltet werden.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Mode** bzw. **VcLegendView.Mode** der Programmierschnittstelle gesetzt werden.

## Rahmen

*Nicht aktiviert, wenn der Modus **Popup-Fenster** gewählt wurde.* Aktivieren Sie dieses Kontrollkästchen, wenn die Ansicht einen Rahmen erhalten soll. Die Rahmenfarbe kann aus der Drop-Down-Liste gewählt werden.

Diese Optionen können auch über die Aufrufe **VcWorldView.Border** und **VcWorldView.Border.Color** bzw. **VcLegendView.Border** und **VcLegendView.Border.Color** der Programmierschnittstelle gesetzt werden.

## Links

*Nur aktiviert, wenn der Modus **nicht fest positioniert** oder **Popup-Fenster** gewählt wurde.* Legen Sie hier die linke Position der Ansicht fest. Dabei gibt es zwei Möglichkeiten:

1. Geben Sie unter **Pixelkoordinate** einen Wert an. Dabei handelt es sich um Gerätekoordinaten.
2. Wählen Sie die Option **Beim Start automat. Berechnung**, damit die Position der Ansicht beim Programmstart automatisch berechnet wird.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Left** bzw. **VcLegendView.Left** der Programmierschnittstelle gesetzt werden.

## Oben

*Nur aktiviert, wenn der Modus **nicht fest positioniert** oder **Popup-Fenster** gewählt wurde.* Legen Sie hier die obere Position der Ansicht fest. Dabei gibt es zwei Möglichkeiten:

1. Geben Sie unter **Pixelkoordinate** einen Wert an. Dabei handelt es sich um Gerätekoordinaten.
2. Wählen Sie die Option **Beim Start automat. Berechnung**, damit die Position der Ansicht beim Programmstart automatisch berechnet wird.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Left** bzw. **VcLegendView.Left** der Programmierschnittstelle gesetzt werden.

## Breite

*Nicht aktiviert, wenn der Modus **fest an oberer/unterer Seite** gewählt wurde.* Legen Sie hier die horizontale Ausdehnung der Ansicht fest. Der Wert wird in Pixelkoordinaten (Gerätekoordinaten) angegeben.

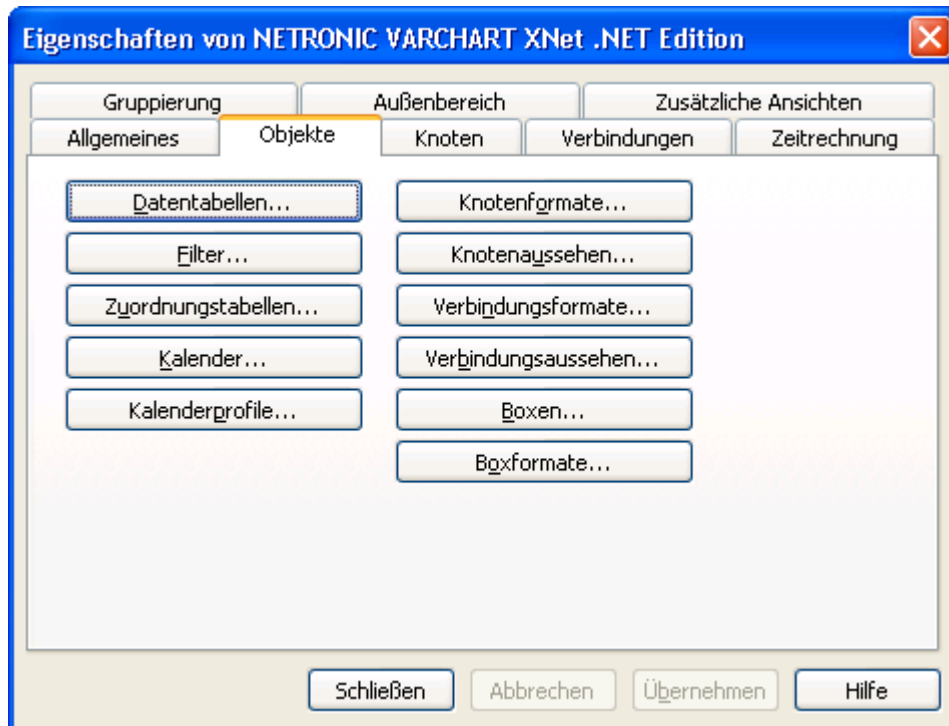
Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Width** bzw. **VcLegendView.Width** der Programmierschnittstelle gesetzt werden.

## Höhe

*Nicht aktiviert, wenn der Modus **fest an linker/rechter Seite** gewählt wurde.* Legen Sie hier die vertikale Ausdehnung der Ansicht fest. Der Wert wird in Pixelkoordinaten (Gerätekoordinaten) angegeben.

Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Height** bzw. **VcLegendView.Height** der Programmierschnittstelle gesetzt werden.

## 4.7 Eigenschaftenseite "Objekte"



### Datentabellen

Über diese Schaltfläche öffnen Sie das Dialogfeld **Datentabellen verwalten**.

### Filter

Über diese Schaltfläche öffnen Sie das Dialogfeld **Filter verwalten**.

### Zuordnungstabellen

Über diese Schaltfläche öffnen Sie das Dialogfeld **Zuordnungstabellen verwalten**.

### Kalender

Über diese Schaltfläche öffnen Sie das Dialogfeld **Kalender festlegen**.

### Knotenformate

Über diese Schaltfläche öffnen Sie das Dialogfeld **Knotenformate verwalten**.

## **Knotenaussehen**

Über diese Schaltfläche öffnen Sie das Dialogfeld **Knotenaussehen verwalten**.

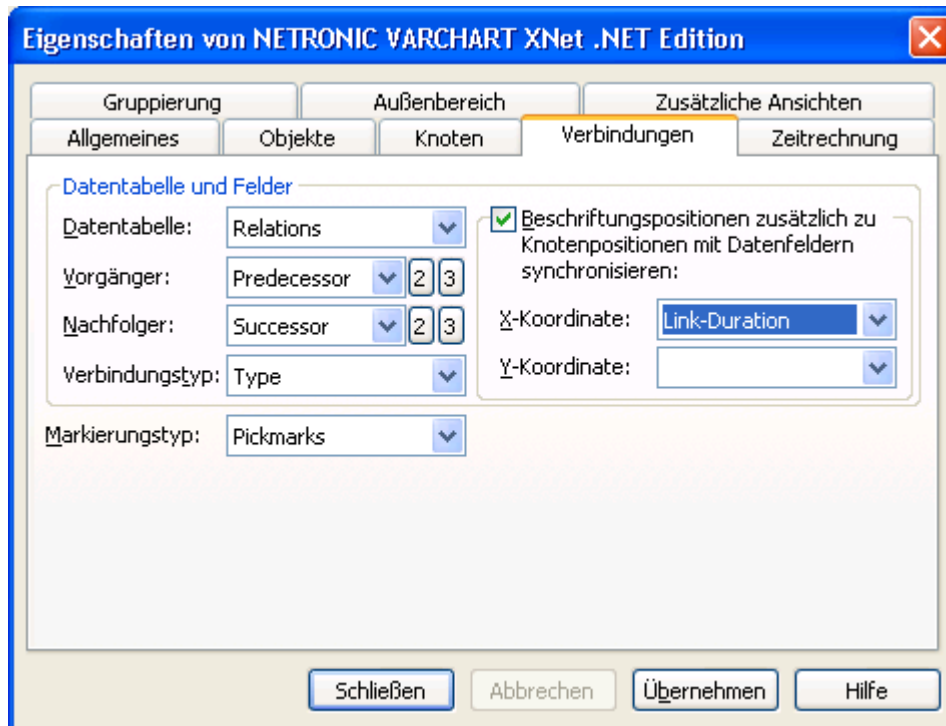
## **Boxen**

Über diese Schaltfläche öffnen Sie das Dialogfeld **Boxen verwalten**.

## **Boxformate**

Über diese Schaltfläche öffnen Sie das Dialogfeld **Boxformate verwalten**.

## 4.8 Eigenschaftenseite "Verbindungen"



Auf dieser Eigenschaftenseite können Sie wählen, ob die Verbindungen zwischen Knoten dargestellt werden, und ihr Aussehen festlegen.

### Datentabelle

Wählen Sie hier eine Datentabelle aus, die die Felder für die Verbindungen enthält. Diese Option kann auch über die Eigenschaft **VcNet.LinkData-TableName** festgelegt werden.

### Vorgänger

Wählen Sie hier das Datenfeld aus, in dem die Identifizierung des Vorgängerknotens der Verbindungen enthalten ist.

### Nachfolger

Wählen Sie hier das Datenfeld aus, in dem die Identifizierung des Nachfolgerknotens der Verbindungen enthalten ist.

### Verbindungstyp

Wählen Sie hier das Datenfeld aus, in dem die Information über den Typ von Verbindungen enthalten sein soll. Es gibt folgende Verbindungstypen:



- Anfang-Anfang (AA oder SS) (Anfangsfolge)
- Anfang-Ende (AE oder SF) (Sprungfolge)
- Ende-Anfang (EA oder FS) (Normalfolge)
- Ende-Ende (EE oder FF) (Endfolge).

In Klammern stehen jeweils die möglichen Feldinhalte, die den jeweiligen Typ repräsentieren.

## **Markierungstyp**

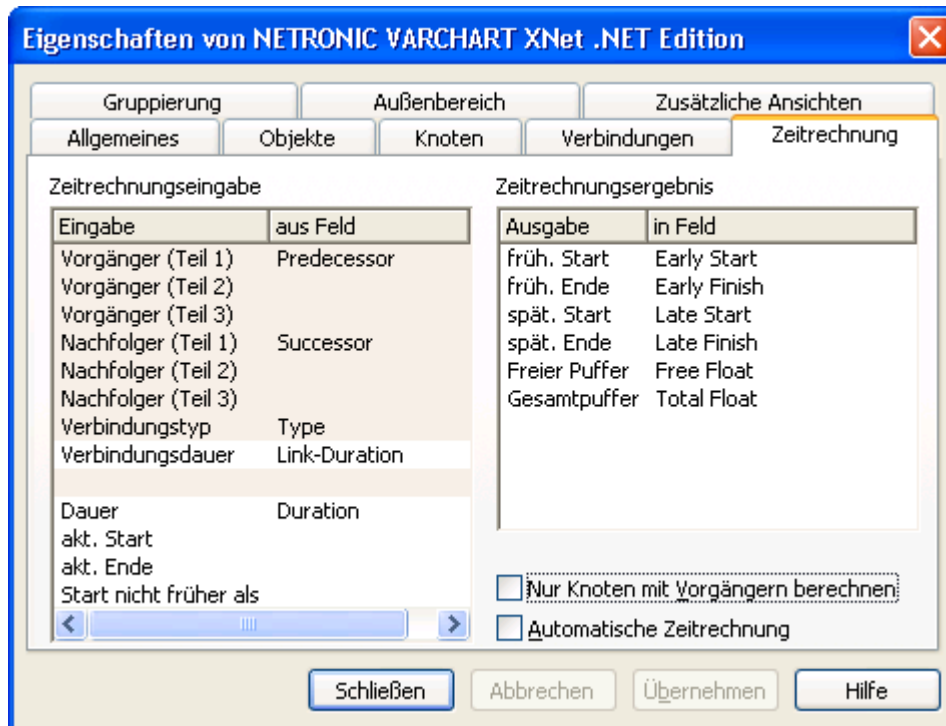
Wählen Sie hier aus, ob und welche Art der interaktiven Markierung von Verbindungen dem Anwender geboten werden soll. Folgende Alternativen stehen zur Verfügung:

- Einrahmen
- Invertieren
- ohne
- Pickmarks

## **Positionen der Beschriftungen mit Datenfeldern synchronisieren**

Wählen Sie je ein Datenfeld aus, aus dem die X- bzw. Y-Koordinate der Position jeder Verbindungsbeschriftung gelesen und wohin sie zurückgeschrieben werden soll. Die Synchronisierung der Positionen der Verbindungsbeschriftungen mit den Datenfeldern ist erforderlich, damit die Positionen der Verbindungsbeschriftungen nach dem Schließen Ihres Projekts wiederhergestellt werden können.

## 4.9 Eigenschaftenseite "Zeitrechnung"



Mit Hilfe dieser Eigenschaftenseite können Sie die Zeitrechnung des VARCHART XNet an Ihre Schnittstelle anpassen, indem Sie festlegen, welche Datenfelder für die Eingabe (**Zeitrechnungseingabe**) und die Ausgabe (**Zeitrechnungsergebnis**) des Schedulers verwendet werden sollen.

### Zeiteinheit für Dauern

Die hier eingestellte Einheit dient als Grundlage für die Berechnung der Termine und Pufferzeiten in den entsprechenden Datenfeldern in Knoten und Verbindungen.

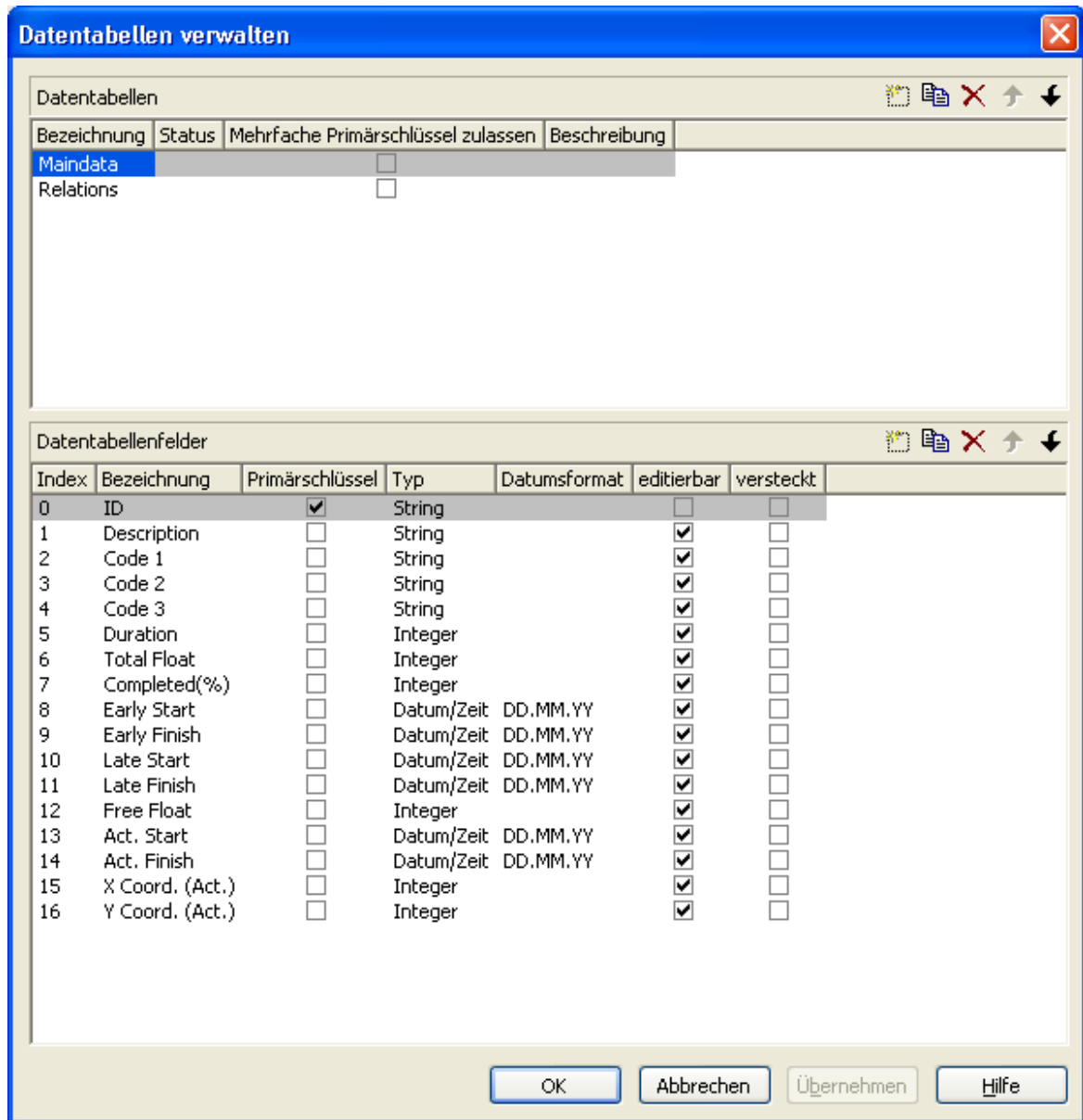
### Zeitrechnungseingabe

Wählen Sie hier für jede Eingabe aus, aus welchem Feld sie entnommen werden soll. Als Eingaben für die Zeitrechnung verwendet der Scheduler Datenfelder der jeweils eingestellten Knoten- und Verbindungstabellen. Die Grundlage der Berechnung sind die Dauer der einzelnen Vorgänge, deren logische Abhängigkeiten und der Projektanfang. Die Felder **Vorgänger** und **Nachfolger** können in der **Zeitrechnungseingabe**-Tabelle nicht bearbeitet werden. Sie geben nur die auf der Eigenschaftenseite **Verbindungen** vorgenommenen Einstellungen wieder.

## **Zeitrechnungsergebnis**



Wählen Sie hier für jedes Ergebnis aus, in welches Feld es geschrieben werden soll. Die Ausgabe des Schedulers erfolgt nur in Datenfelder der **Maindata**-Tabelle. Aus der Dauer der einzelnen Vorgänge, deren logischen Abhängigkeiten und dem Projektanfang werden die frühesten bzw. spätesten Start- und Endtermine sowie der Gesamtpuffer und der freie Puffer berechnet.

## 4.10 Dialogfeld "Datentabellen verwalten"




Sie gelangen in diesen Dialog über die Eigenschaftenseite **Objekte**. Sie können hier Datentabellen sowie die zugehörigen Datenfelder anlegen und bearbeiten.

### Datentabellen

- **Bezeichnung:** In dieser Spalte stehen die Namen aller vorhandenen Datentabellen. Die Namen sind editierbar.
- **Status:** In der Spalte **Status** wird jede Datentabelle gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt () und/oder geändert () worden ist.

- **Mehrfache Primärschlüssel zulassen:** Bestimmen Sie hier, ob der Primärschlüssel der Tabelle aus **einem** oder **mehreren (maximal 3)** Feldern bestehen soll. Sobald Sie **Mehrfache Primärschlüssel zulassen** ausgewählt haben, sind im Bereich **Datentabellenfelder** bis zu 3 Felder für den Primärschlüssel wählbar. Die Einstellung **Mehrfache Primärschlüssel zulassen** kann erst dann wieder deaktiviert werden, wenn im Bereich **Datentabellenfelder** nur noch ein Feld für den Primärschlüssel ausgewählt ist.
- **Beschreibung:** Geben Sie hier eine Beschreibung für die Datentabelle ein.

## Datentabelle hinzufügen / kopieren / löschen / nach oben / unten

 Mit Hilfe dieser Schaltflächen können Sie Datentabellen hinzufügen, kopieren, löschen und in der Liste nach oben oder unten verschieben.

## Datentabellenfelder

Hier können Sie für die im Bereich **Datentabellen** ausgewählte Datentabelle Datentabellenfelder anlegen und bearbeiten.

- **Index:** Der Index der Datentabellenfelder ist nicht veränderbar, da er intern als Referenz dient. In der API werden Datenfelder über den Index angesprochen.
- **Bezeichnung:** Diese Spalte zeigt die Namen der Felder der Datentabelle. Nach Anklicken können Sie diese ändern.
- **Primärschlüssel:** Hier können Sie festlegen, welches Feld in der Spalte der Primärschlüssel der Datensätze dieser Tabelle sein soll.
- **Typ:** Hier können Sie den Datentyp für jedes Datentabellenfeld festlegen. Zur Auswahl stehen:
  - Alphanumerisch
  - Integer
  - Datum/Zeit
  - Double
- **Datumsformat:** Für die Datentabellenfelder vom Typ **Datum/Zeit** können Sie hier jeweils das Datumsformat an das Datumsformat Ihrer Knotendaten anpassen. Einige gebräuchliche Datumsformate stehen zur

Auswahl. Sie können außerdem ein eigenes Datumsformat definieren, z. B. "DD.MM.YY hh:mm".


Das Datumsformat wird aus den Kombinationen **YY** (zweistellige Jahreszahl), **YYYY** (vierstellige Jahreszahl), **MM** (zweistellige Monatszahl), **MMM** (dreistelliges Monatsnamenkürzel), **DD** (zweistellige Tageszahl), **hh** (zweistellige Stundenzahl), **mm** (zweistellige Minutenzahl) und **ss** (zweistellige Sekundenzahl) zusammengesetzt.

Beachten Sie bitte, dass das Datumsformat, das Sie hier festlegen, mit dem Datumsformat Ihrer Knotendaten übereinstimmen muss.

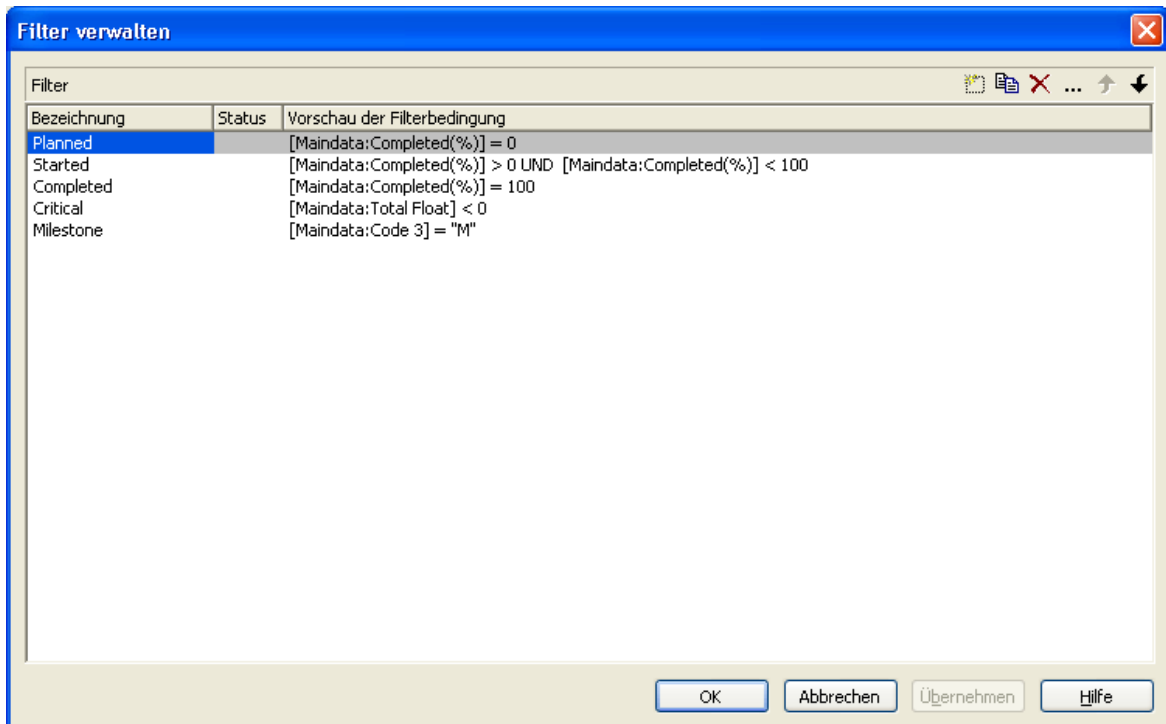
Dieses Datumsformat ist nur für die Dateneingabe, aber nicht für die Darstellung von Daten in Ihrer Grafik relevant.

- **Editierbar:** Aktivieren Sie dieses Kontrollkästchen für alle Datentabellenfelder, die der Anwender im Dialogfeld **Vorgänge bearbeiten** bearbeiten können soll.
- **Versteckt:** Aktivieren Sie dieses Kontrollkästchen für alle Datentabellenfelder, die dem Anwender im Dialogfeld **Vorgänge bearbeiten** verborgen bleiben sollen.
- **Beziehung:** Hier können Sie eine Verknüpfung zu einer anderen Tabelle festlegen, so dass die Datensätze dieser Tabelle über das als Primärschlüssel definierte Feld einer anderen verknüpft sind. Aus diesem Grund werden Ihnen zur Auswahl alle Tabellen angeboten, für die Sie einen Primärschlüssel definiert haben.

## Datentabellenfeld hinzufügen / kopieren / löschen / nach oben / unten

 Mit Hilfe dieser Schaltflächen können Sie Datentabellenfelder hinzufügen, kopieren, löschen und in der Liste nach oben oder unten verschieben.

## 4.11 Dialogfeld "Filter verwalten"





Sie gelangen in dieses Dialogfeld über die Eigenschaftenseite **Objekte** oder **Verbindungen**.

### Bezeichnung

In dieser Spalte stehen die Namen aller vorhandenen Filter. Die Namen sind editierbar.

### Status

In der Spalte **Status** wird jeder Filter gekennzeichnet, der seit dem Aufruf des Dialogs hinzugefügt (  ) und/oder geändert (  ) worden ist.


### Datendefinitionstabelle

Hier wird die Datendefinitionstabelle angezeigt, die dem jeweiligen Filter zu Grunde liegt. Diese Spalte wird nur angezeigt, wenn auf der Eigenschaftenseite **Allgemeines** die Option **Erweiterte Datentabellen zulassen** nicht ausgewählt ist.

## Vorschau der Filterbedingung

In dieser Spalte wird die Bedingung jedes Filters angezeigt. Sie kann hier nicht editiert werden. Um die Filterbedingung zu bearbeiten, klicken Sie auf die **Filter bearbeiten**-Schaltfläche.

## Filter hinzufügen


 Ein neuer Filter mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

Neue Filter werden kontextabhängig angelegt, d. h. es wird immer die jeweils passende Datendefinitionstabelle (**Maindata** bzw. **Relations**) verwendet.


## Filter kopieren

 Der markierte Filter wird kopiert.



## Filter löschen

 Der Filter, den Sie in der Liste markiert haben, wird gelöscht. Es können nur Filter gelöscht werden, die zur Zeit nicht benutzt werden.

## Filter bearbeiten

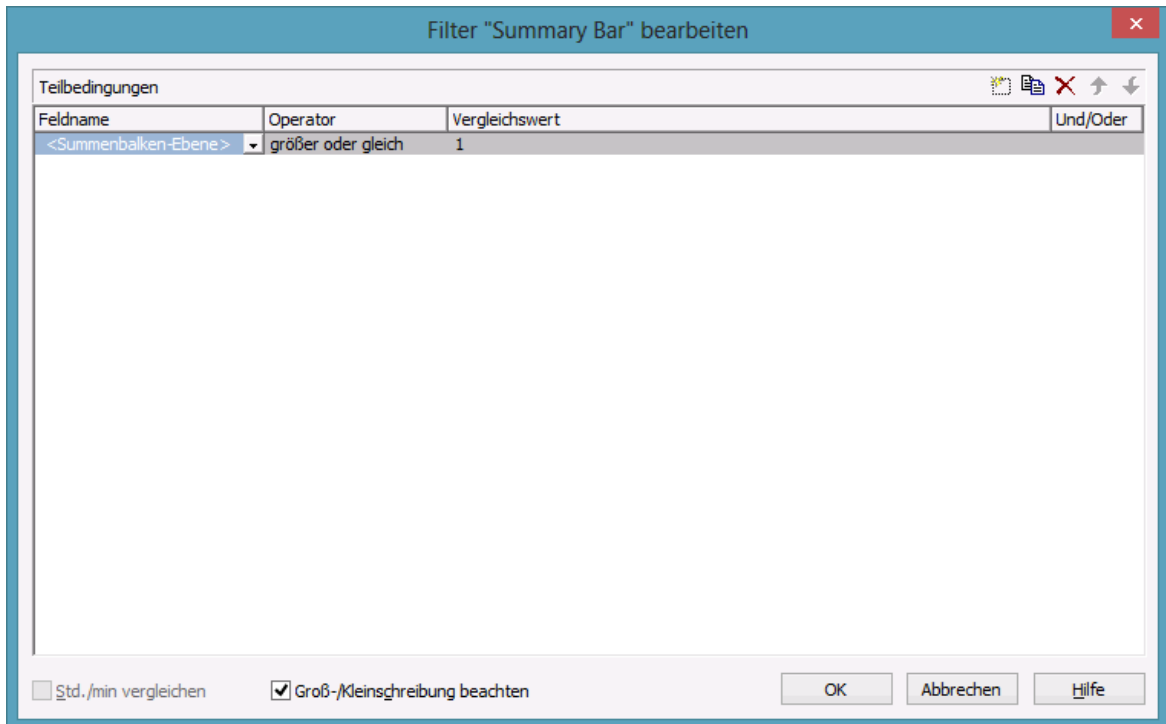
 Um die Bedingungen eines Filters anzusehen oder zu ändern, klicken Sie auf die Schaltfläche **Filter bearbeiten**. Es erscheint das Dialogfeld **Filter bearbeiten**. Nun können Sie die für den Filter formulierten Bedingungen bearbeiten.

## Filter eine Zeile nach oben/unten

  Mit Hilfe dieser Schaltflächen können Sie den markierten Filter eine Zeile nach oben/unten schieben.



## 4.12 Dialogfeld "Filter bearbeiten"




Sie erreichen dieses Dialogfeld in dem Sie entweder

- von der Eigenschaftenseite **Objekte**
- aus dem Dialog **Knotenaussehen verwalten oder**
- aus dem Dialog **Verbindungsaussehen verwalten**

den **Filter verwalten**-Dialog aktivieren und anschließend die Schaltfläche **Filter bearbeiten** anklicken. In der Kopfzeile dieses Dialogfeldes steht der Name des aktuellen Filters.

### Teilbedingung hinzufügen

 Vor der markierten Zeile wird eine neue Zeile für eine Teilbedingung eingefügt.


### Teilbedingung kopieren

 Die markierte Teilbedingung wird kopiert.

### Teilbedingung löschen

 Die markierte Teilbedingung wird gelöscht.

## Teilbedingung früher/später abarbeiten

 Wenn ein Filter mehrere Teilbedingungen enthält, werden die einzelnen Teilbedingungen in der Reihenfolge, in der sie in der **Teilbedingungen**-Tabelle stehen, abgearbeitet.

Klicken Sie die Schaltfläche **Teilbedingung früher/später abarbeiten** an, um eine markierte Teilbedingung in der Tabelle eine Position höher/tiefer zu setzen, damit sie früher/später abgearbeitet wird.

## Feldname

Hier können Sie das Datenfeld auswählen, das mit dem Vergleichswert verglichen werden soll. Die Liste enthält alle verfügbaren Datenfelder.

## Operator

Wählen Sie hier den Vergleichsoperator für den Vergleich des unter **Feldname** gewählten Datenfeldes mit dem unter **Vergleichswert** angegebenen Wertes bzw. Datenfeldes aus.

## Vergleichswert

Hier wird der aktuelle Vergleichswert angezeigt. Im Feld **Vergleichswert** werden in eckigen Klammern alle Felder des passenden Datentyps angeboten, die jeweils als Vergleichswert eingesetzt werden können. Wurde unter **Feldname** beispielsweise das Feld "Frühester Anfang" eingetragen, werden als Vergleichswerte nur Terminfelder (z. B. "Frühestes Ende") und die Optionen <heute> und <Eingabe> angeboten.

Mit Hilfe des Vergleichswertes <Eingabe> können Sie einen variablen Filter definieren. In variablen Filtern sind nur der Feldname und der Operator, aber nicht der Vergleichswert definiert. Diesen können Sie bei Bedarf angeben. Sie können einen variablen Filter verwenden, wenn Sie ein Projekt öffnen und die darzustellenden Vorgänge begrenzen möchten.

Termine werden in dem Format, das auf der Eigenschaftenseite **Allgemeines** unter **Datumsausgabeformat** festgelegt wurde, eingegeben. Wenn Sie unter **Feldname** ein Terminfeld gewählt haben, erscheinen im Feld **Vergleichswert** zwei Pfeil-Schaltflächen, sobald Sie dieses Feld anklicken. Über die erste Pfeil-Schaltfläche öffnen Sie eine Kombobox, die Ihnen alle verfügbaren Termin-Datenfelder anzeigt. Über die zweite Pfeil-Schaltfläche öffnen Sie den Datumsdialog, aus dem Sie das gewünschte Datum per Mausklick auswählen können. Sie können das Datum aber auch direkt editieren.

Zahlenwerte oder Texte müssen direkt eingegeben werden.

Bei den Operatoren "gleich" und "ungleich" können Sie für Textfelder auch Wildcards verwenden:

\*: kein oder mehrere beliebige Zeichen

?: genau ein beliebiges Zeichen

Wenn Sie die Zeichen \* oder ? nicht als Wildcards verwenden, sondern auf diese Zeichen abfragen wollen, müssen Sie dies durch einen vorangestellten Backslash kenntlich machen:

\\*: \*

\?: ?

Wenn dem Backslash kein \* oder ? folgt, wird nach dem Vorhandensein von \ gefragt.

### **Beispiele:**

Vorgang 1 : Name = "Baugenehmigung"

Vorgang 2 : Name = "\*Baugenehmigung"

Mögliche Filter für Vorgang 1:

[Name] = B\*

[Name] = B?ugenehmigung

Mögliche Filter für Vorgang 2:

[Name] = \\*B\*

[Name] = \\*\*

[Name] = ?B\*

## **Und/Oder**

Hier wird die logische Verknüpfung der aktuellen mit der jeweils nächsten Teilbedingung in der Tabelle angezeigt.

Wenn Sie den UND-Operator wählen, werden nur die Objekte ausgewählt, die beide Teilbedingungen erfüllen. Wählen Sie den ODER-Operator, werden die Objekte ausgewählt, die mindestens eine der verknüpften Teilbedingungen erfüllen.

Haben Sie mehrere Teilbedingungen formuliert und diese zum Teil mit UND, zum Teil mit ODER verknüpft, werden erst die UND-Verknüpfungen abgearbeitet. (UND bindet stärker als ODER.)

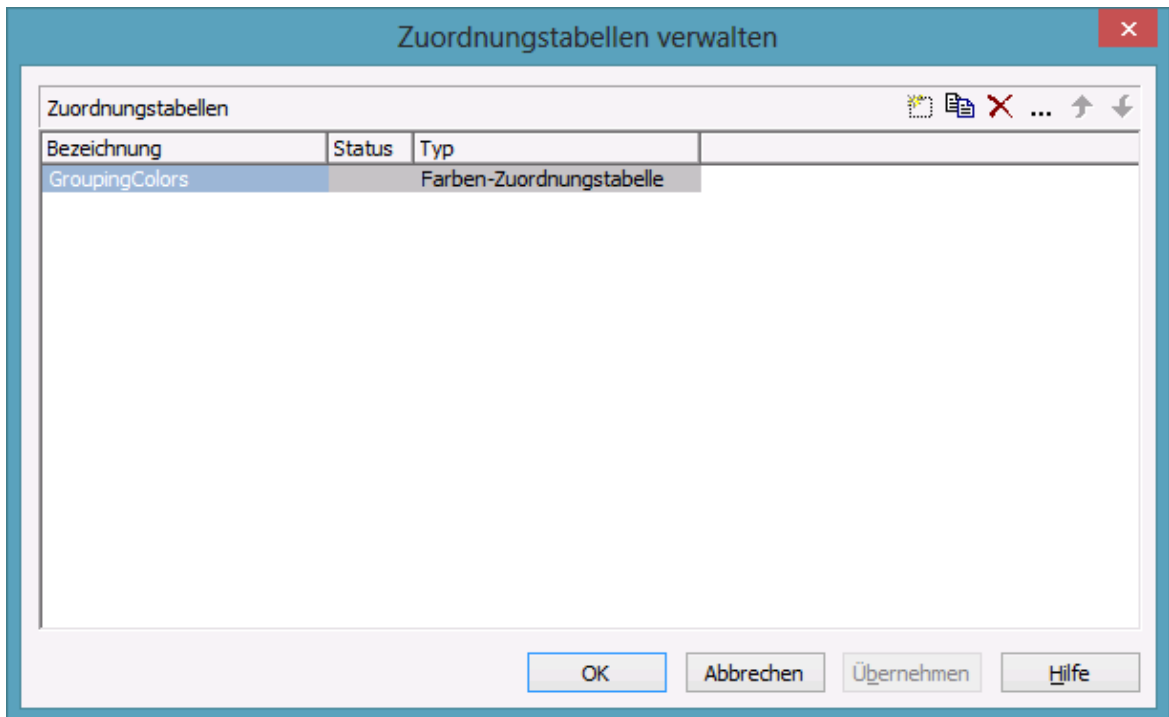
### **Std./min vergleichen**

Aktivieren Sie dieses Kontrollkästchen, wenn beim Datumsvergleich auch Stunde und Minute berücksichtigt werden sollen.

### **Groß-/Kleinschreibung beachten**

Aktivieren Sie dieses Kontrollkästchen, wenn beim Vergleich die Groß-/Kleinschreibung beachtet werden soll.

## 4.13 Dialogfeld "Zuordnungstabellen verwalten"



Sie gelangen über die Eigenschaftenseite **Objekte** in dieses Dialogfeld oder indem Sie im Dialogfeld **Zuordnung einstellen** auf die Schaltfläche **Zuordnungstabellen** klicken.

### Bezeichnung

In dieser Spalte stehen die Namen aller vorhandenen Zuordnungstabellen. Die Namen sind editierbar.

### Status

In der Spalte **Status** wird jede Zuordnungstabelle gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.


### Typ

Wählen Sie hier den Typ der Zuordnungstabelle aus:


- Farben-Zuordnungstabelle
- Schraffuren-Zuordnungstabelle (für spätere Anwendungen)

- Grafikdateien-Zuordnungstabelle


## Zuordnungstabelle hinzufügen

 Eine neue Zuordnungstabelle mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

## Zuordnungstabelle kopieren

 Die markierte Zuordnungstabelle wird kopiert.

## Zuordnungstabelle löschen

 Die Zuordnungstabelle, die Sie in der Liste markiert haben, wird gelöscht. Es können nur die Zuordnungstabellen gelöscht werden, die zur Zeit nicht benutzt werden.

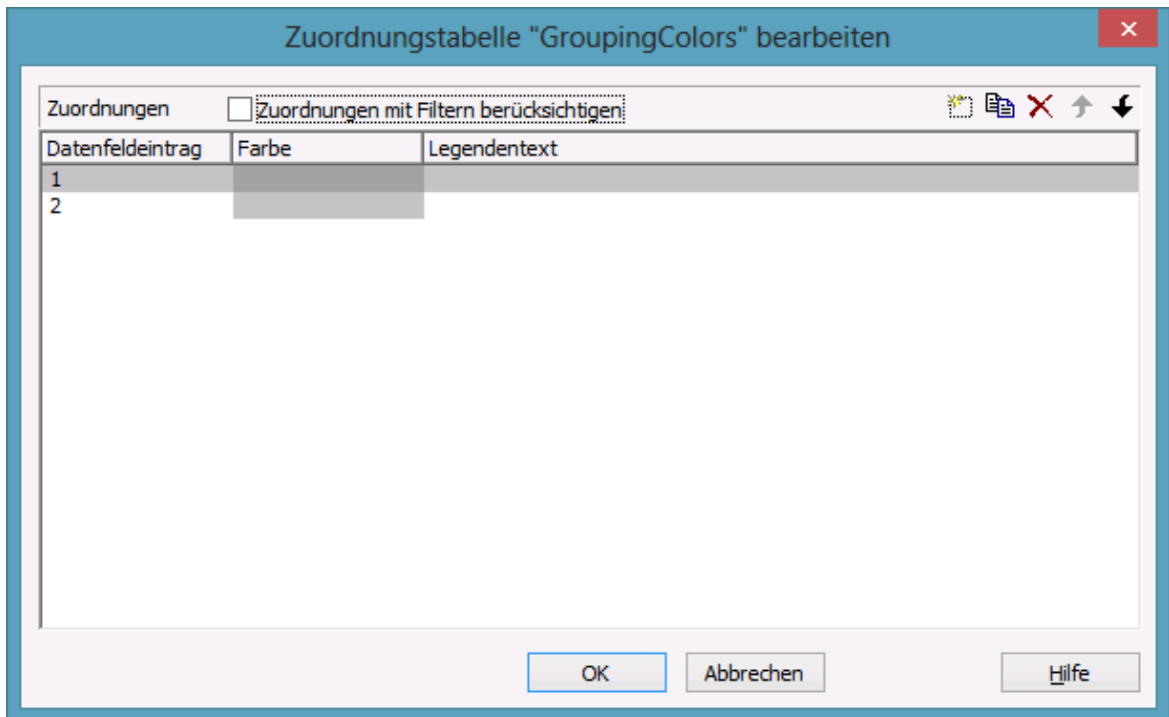
## Zuordnungstabelle bearbeiten

 Sie gelangen in das Dialogfeld **Zuordnungstabelle bearbeiten**.

## Zuordnungstabelle eine Zeile nach oben/unten

 Mit Hilfe dieser Schaltfläche können Sie die markierte Zuordnungstabelle in der Tabelle eine Zeile nach oben/unten schieben.

## 4.14 Dialogfeld "Zuordnungstabelle bearbeiten"



Sie gelangen in dieses Dialogfeld, indem Sie im Dialogfeld **Zuordnungstabellen verwalten** auf die Schaltfläche **Zuordnungstabelle bearbeiten** (...) klicken.

Sie können in einer Zuordnungstabelle maximal 150 Zuordnungen festlegen. Falls Sie noch mehr Zuordnungen benötigen, erstellen Sie einfach eine neue Zuordnungstabelle, z. B. als Kopie der bereits vorhandenen.

### Zuordnungen mit Filtern berücksichtigen

Wenn die Option **Zuordnungen mit Filtern berücksichtigen** ausgewählt ist, werden nicht nur die in der Liste der Datenfeldeinträge angegebenen festen Werte als Schlüsselwerte berücksichtigt, sondern auch Filter, die aus der Dropdown-Liste ausgewählt werden können. Dadurch hängen die Ausführungswerte nicht mehr nur von einem konkreten Wert ab, sondern von einem Wertebereich.

### Datenfeldeintrag

In dieser Spalte werden die Einträge des gewählten Datenfeldes aufgeführt, für die Sie eine Farbe bzw. eine Grafikdatei und den Legendentext vereinbaren können.

## Farbe/Grafikdateiname


Um die Farbe bzw. die Grafikdatei für einen Datenfeldeintrag festzulegen, klicken Sie auf das entsprechende Feld. Dann erscheint eine Schaltfläche, über die Sie in den Dialog zur Auswahl der Farbe bzw. der Grafikdatei gelangen.

Wird ein relativer Dateiname angegeben, so wird die Datei zur Laufzeit zuerst in dem Verzeichnis gesucht, das in der VARCHART-ActiveX-Eigenschaft **FilePath** gesetzt ist. Wird sie dort nicht gefunden, wird sie zuerst im gerade aktiven Arbeitsverzeichnis der Applikation und dann im Installationsverzeichnis des Steuerelements gesucht.

## Legendentext

*(nur für Farb-Zuordnungstabellen)* Hier können Sie für jeden Datenfeldeintrag einen Legendentext festlegen.


## Zuordnung hinzufügen

 Eine neue Zuordnung mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.


## Zuordnung kopieren

 Die markierte Zuordnung wird kopiert.

## Zuordnung löschen

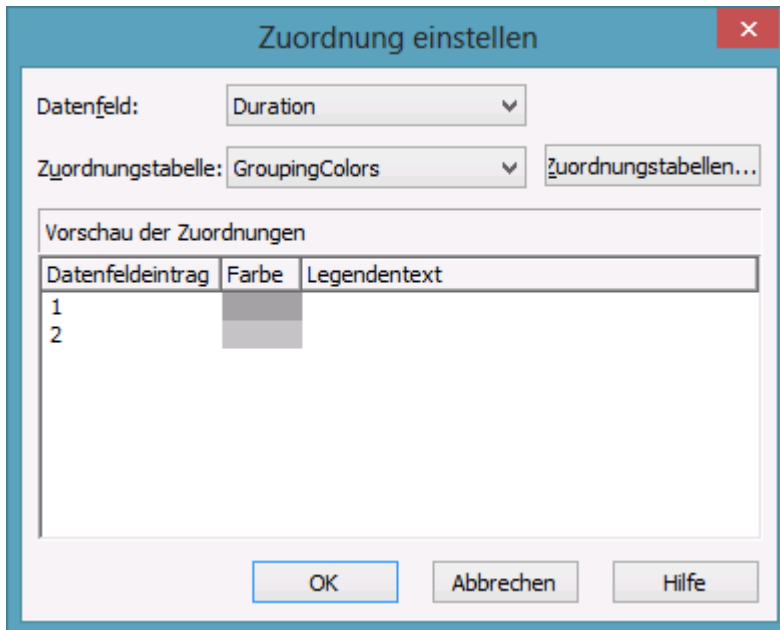
 Die Zuordnung, die Sie in der Liste markiert haben, wird gelöscht. Es können nur die Zuordnungen gelöscht werden, die zur Zeit nicht benutzt werden.


## Zuordnung eine Zeile nach oben/unten

 Die markierte Zuordnung wird in der Tabelle eine Zeile nach oben/unten geschoben.



## 4.15 Dialogfeld "Zuordnung einstellen"



Verbinden Sie in diesem Dialogfeld das Datenfeld eines Knotens mit einer Zuordnungstabelle. Sie erreichen es über verschiedene Dialoge z.B. den Dialog **Layer bearbeiten**. Klicken Sie dort beim gewünschten Attribut auf die Schaltfläche .

### Datenfeld

Wählen Sie hier das Datenfeld aus, von dessen Einträgen die gewünschten Attribute des zu bearbeitenden Objekts abhängen soll.

### Zuordnungstabelle

*(nur aktiv, wenn ein Datenfeld gewählt wurde)* Wählen Sie hier die Zuordnungstabelle aus, die den einzelnen Datenfeldeinträgen eine Farbe bzw. eine Grafikdatei und einen Legendentext zuordnet.

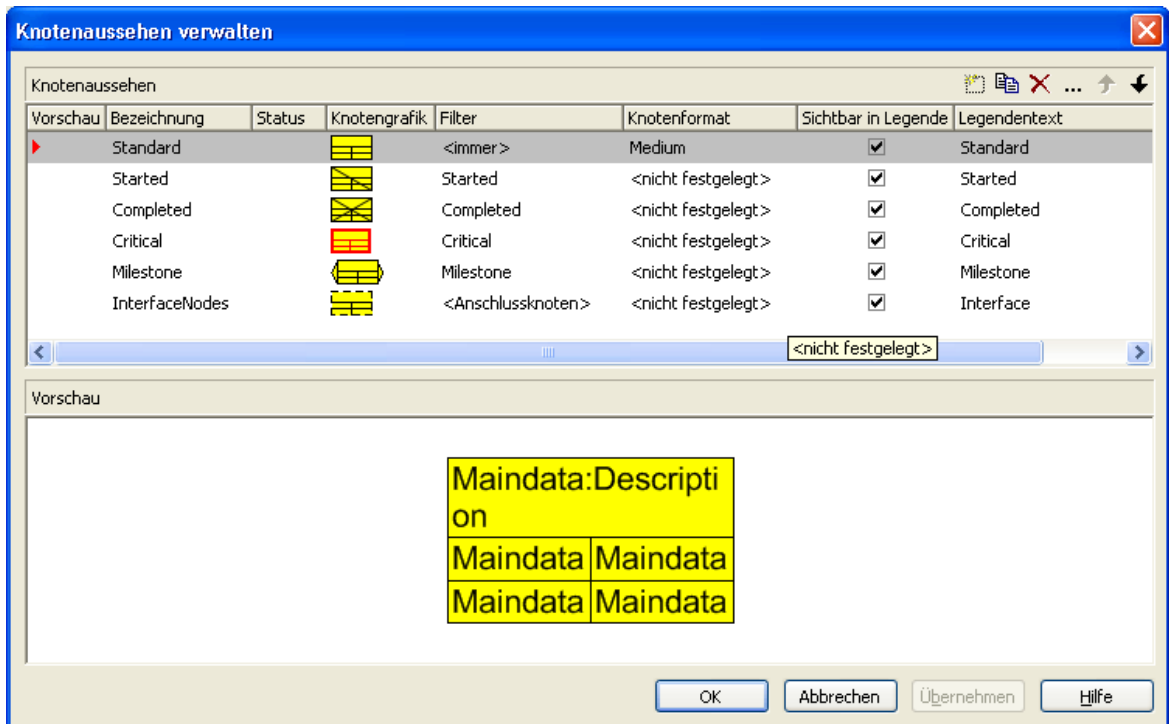
### Zuordnungstabellen

Klicken Sie auf diese Schaltfläche, um das Dialogfeld **Zuordnungstabellen verwalten** aufzurufen. Hier können Sie Zuordnungstabellen erstellen, bearbeiten, kopieren oder löschen.

## **Vorschau der Zuordnungen**

Hier wird die gewählte Zuordnungstabelle dargestellt: alle Datenfelder, die diesen zugeordneten Farben und Legendentexte bzw. die Grafikdateinamen.

## 4.16 Dialogfeld "Knotenaussehen verwalten"



Sie gelangen über die Eigenschaftenseite **Objekte** in dieses Dialogfeld.

Die Darstellung von Knoten wird festgelegt, indem diesen ein oder mehrere Knotenaussehen dynamisch über Filter zugewiesen werden. Aus der Überlagerung der Attribute aller Knotenaussehen, die auf einen Knoten zutreffen, ergibt sich das gesamte Knotenaussehen.

### Vorschau



Alle Knotenaussehen, die in der **Vorschau**-Spalte durch eine kleine rote Pfeilspitze markiert sind, werden im Vorschaufenster überlagert und mit den durch die Abarbeitungsreihenfolge bestimmten Überlagerungen angezeigt.

Das Knotenaussehen, auf dem der Cursor gerade steht, wird durch eine grüne Pfeilspitze gekennzeichnet.

### Bezeichnung

Diese Spalte enthält die Namen aller vorhandenen Knotenaussehen. Die Namen sind editierbar.

## Status

In dieser Spalte wird jedes Knotenaussehen, das seit dem Aufruf des Dialogs hinzugefügt () und/oder geändert () worden ist.

## Knotengrafik

Hier wird jedes Knotenaussehen dargestellt. Um die Knotengrafik, d. h. die grafischen Attribute des Knotenaussehens, zu verändern, klicken Sie auf die **Knotenaussehen bearbeiten**-Schaltfläche oberhalb der Tabelle oder doppelklicken Sie auf den Eintrag **Knotengrafik**. Sie gelangen dann in den Dialog **Knotenaussehen bearbeiten**.

## Filter

Der Filter, der mit einem Knotenaussehen verbunden ist, bestimmt, welche Knoten mit dem betreffenden Knotenaussehen versehen werden.

Sie können für die meisten Knotenaussehen den Filter auswählen. Nur für die Knotenaussehen "Standard" und "Anschlussknoten" ist der Filter festgelegt ("`<immer>`" bzw. "`<Anschlussknoten>`").

Um einem Knotenaussehen einen Filter zuzuweisen, markieren Sie das **Filter**-Feld. Dann erscheinen eine Schaltfläche für eine Kombobox mit allen verfügbaren Filtern sowie eine **Bearbeiten**-Schaltfläche (außer bei den Knotenaussehen mit fest zugewiesenem Filter). Wählen Sie aus der Kombobox einen Filter für das Knotenaussehen aus oder klicken Sie auf die **Bearbeiten**-Schaltfläche des **Filter**-Feldes, um den Dialog **Filter verwalten** aufzurufen. Dort können Sie Filter bearbeiten, kopieren, neu definieren oder löschen.

## Knotenformat

Ein Knotenformat definiert die Anzahl, die Anordnung und das Aussehen der Felder, die zur Beschriftung eines Knotens verwendet werden. Wählen Sie hier das Knotenformat für das Knotenaussehen aus. Markieren Sie dazu das **Knotenformat**-Feld. Dann erscheinen eine Schaltfläche für eine Kombobox mit allen verfügbaren Knotenformaten sowie eine **Bearbeiten**-Schaltfläche. Wählen Sie aus der Kombobox ein Knotenformat für die Knotenaussehen aus. Oder klicken Sie auf die **Bearbeiten**-Schaltfläche des **Knotenformat**-Feldes, um den Dialog **Knotenformate verwalten** aufzurufen. Dort können Sie Knotenformate bearbeiten, kopieren, hinzufügen oder löschen.

## Sichtbar in Legende

Aktivieren Sie dieses Kontrollkästchen für alle Knotenaussehen, die in der Legende dargestellt werden sollen.

## Legendentext

In dieser Spalte können Sie für jedes Knotenaussehen einen Legendentext eintragen.

## Knotenaussehen hinzufügen



Ein neues Knotenaussehen wird am Ende der Liste hinzugefügt.

## Knotenaussehen kopieren



Das markierte Knotenaussehen wird kopiert.

## Knotenaussehen löschen



Knotenaussehen, die Sie nicht mehr verwenden, können Sie über diese Schaltfläche löschen. Es erfolgt noch eine Abfrage, ob Sie das markierte Knotenaussehen wirklich löschen wollen. Das Standard-Knotenaussehen kann nicht gelöscht werden.

## Knotenaussehen bearbeiten




Über diese Schaltfläche gelangen Sie in das Dialogfeld **Knotenaussehen bearbeiten**.


## Knotenaussehen früher/später abarbeiten

Wenn einem Knoten mehrere Knotenaussehen zugewiesen sind, werden die einzelnen Knotenaussehen nacheinander abgearbeitet. In der Tabelle sind die Knotenaussehen nach ihrer Abarbeitungsreihenfolge sortiert. Das Standard-Knotenaussehen steht immer ganz oben in der Tabelle, da es immer zugewiesen ist und immer zuerst abgearbeitet wird. Das Knotenaussehen, das zuletzt abgearbeitet wird, steht ganz unten in der Tabelle.

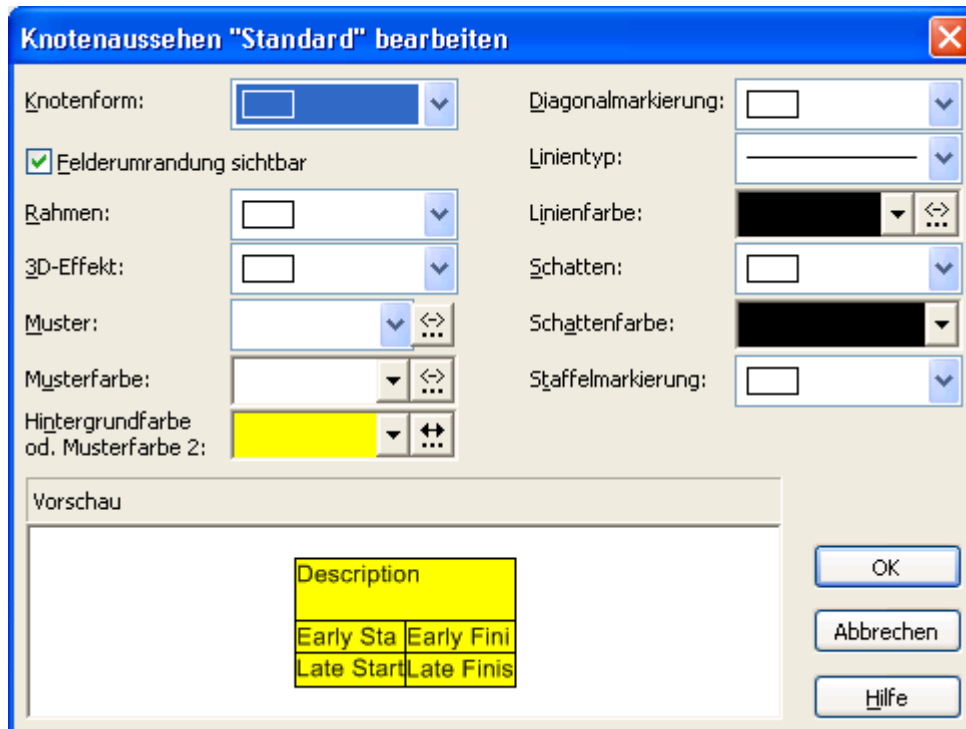
Wenn mehrere Knotenaussehen auf einen Knoten zutreffen, werden die Attribute jedes Knotenaussehens durch die Attribute aller Knotenaussehen, die später abgearbeitet werden, ersetzt. Nur die Attribute, deren Wert "nicht festgelegt" ist, ersetzen nicht die Attribute ihrer Vorgänger.

Mit Hilfe dieser Schaltflächen können Sie die Position eines Knotenaussehens in der Abarbeitungsreihenfolge verändern:

 Wenn Sie die Schaltfläche **Knotenaussehen früher abarbeiten** anklicken, steigt das markierte Knotenaussehen um eine Position in der Tabelle und wird entsprechend früher abgearbeitet.

 Wenn Sie die Schaltfläche **Knotenaussehen später abarbeiten** anklicken, fällt das markierte Knotenaussehen um eine Position in der Tabelle und wird entsprechend später abgearbeitet.

## 4.17 Dialogfeld "Knotenaussehen bearbeiten"



In der Titelzeile wird der Name des Knotenaussehens angezeigt, das Sie in diesem Dialogfeld bearbeiten können.

Wenn einem Knoten mehrere Knotenaussehen zugewiesen sind, werden die Attribute jedes Knotenaussehens durch die Attribute aller Knotenaussehen, die eine höhere Priorität haben, ersetzt. Nur die Attribute, deren Wert <nicht festgelegt> ist, ersetzen nicht die Attribute ihrer Vorgänger.

### Knotenform

Wählen Sie hier die Knotenform für das aktuelle Knotenaussehen aus. Zur Auswahl stehen verschiedene Knotenformen sowie die Einträge <nicht festgelegt> und <ohne Rahmen>.

### Felderumrandung sichtbar

Mit dieser Eigenschaft kann festgelegt werden, ob der Rahmen um die innenliegenden Felder sichtbar ist oder nicht. Die Außenrandlinie der Form ist davon nicht betroffen, daher wirkt sich diese Eigenschaft bei den möglichen Rahmenformen unterschiedlich aus und hat z.B. beim Typ **vcRectangle** keine Auswirkung.

Diese Option kann auch über die Eigenschaft **VcNodeAppearance.FrameAroundFieldsVisible** gesetzt werden.

## Rahmen


Über dieses Feld legen Sie fest, ob die Knoten einen einfachen Rahmen oder einen Doppelrahmen erhalten.


## 3D-Effekt

Über dieses Feld bestimmen Sie, ob die Knoten einen 3D-Effekt erhalten oder nicht.

## Muster

Wählen Sie hier ein Muster für das aktuell ausgewählte Knotenaussehen aus.


 Sie können über die Pfeil-Schaltfläche die Farbzordnungstabelle öffnen, um daraus eine Farbe auszuwählen. Auch transparente Farben stehen zur Verfügung.


 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Hintergrundfarbe vereinbaren.

 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche gefüllt dargestellt.

## Musterfarbe

Wählen Sie hier die Musterfarbe für die Knoten aus.

 Sie können über die Pfeil-Schaltfläche die Farbzordnungstabelle öffnen, um daraus eine Farbe auszuwählen.


 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Linienfarbe vereinbaren.


 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche gefüllt dargestellt.

## Hintergrundfarbe od. Musterfarbe 2

Wählen Sie hier die Hintergrundfarbe für das aktuell ausgewählte Knotenaussehen aus.



 Sie können über die Pfeil-Schaltfläche die Farbzordnungstabelle öffnen, um daraus eine Farbe auszuwählen. Auch transparente Farben stehen zur Verfügung.

 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Hintergrundfarbe vereinbaren.

 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche gefüllt dargestellt.

## Diagonalmarkierung


Über dieses Feld bestimmen Sie, ob die Knoten eine Diagonalmarkierung erhalten und ggf. welche.


## Linientyp

Wählen Sie hier die Linienart für die Umrandung der Knoten.

## Linienfarbe

Wählen Sie hier die Linienfarbe für die Umrandung der Knoten aus.

 Sie können über die Pfeil-Schaltfläche die Farbzordnungstabelle öffnen, um daraus eine Farbe auszuwählen.

 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Linienfarbe vereinbaren.

 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche gefüllt dargestellt.

## Schatten

Über dieses Feld steuern Sie, ob die Knoten einen Schatten erhalten oder nicht.

## Schattenfarbe

Legen Sie hier die Farbe fest, die ggf. der Schatten bzw. die Staffelmartierung erhalten soll.

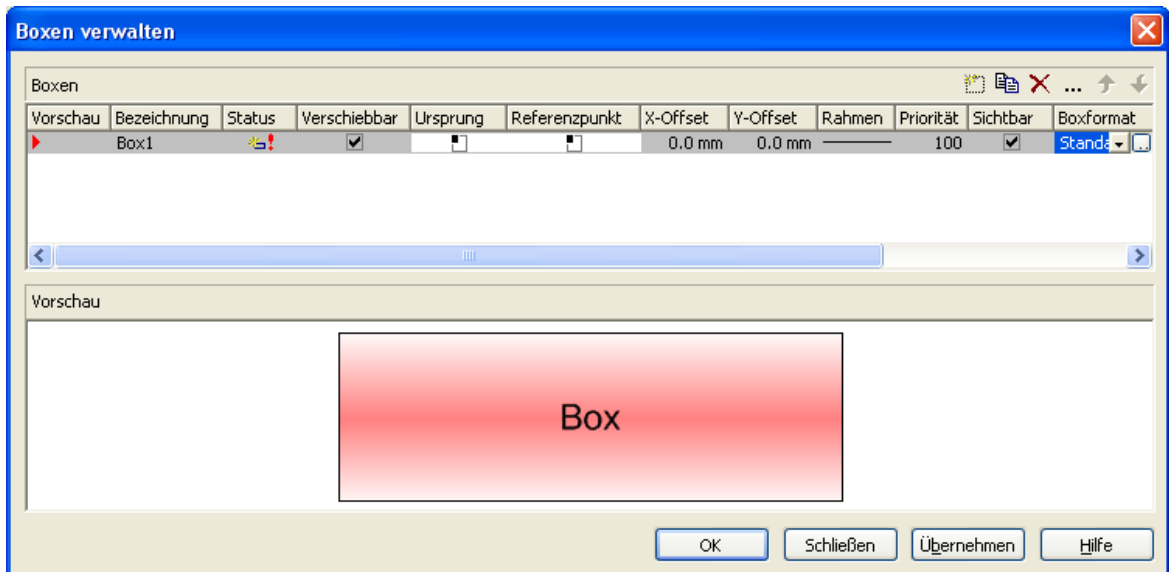
## **Staffelmarkierung**

Über dieses Feld bestimmen Sie, ob die Knoten gestaffelt dargestellt werden sollen (max. 8fach) oder nicht.

## **Vorschau**

In dem Fenster wird das Erscheinungsbild des aktuell festgelegten Knotenaussehens dargestellt.

## 4.18 Dialogfeld "Boxen verwalten"



Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Objekte**. Im Diagrammbereich können beliebig viele Boxen dargestellt werden, die Sie in diesem Dialog verwalten können.



### Vorschau

Die in dieser Spalte markierte Box wird im Vorschaufenster angezeigt.

### Bezeichnung

In dieser Spalte stehen die Namen aller vorhandenen Boxen. Die Namen sind editierbar.

### Status

In der Spalte **Status** wird jede Box gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt (  ) und/oder geändert (  ) worden ist.

### Aktualisierungsverhalten

Wählen Sie hier das gewünschte Aktualisierungsverhalten aus. Bei der Einstellung <nicht festgelegt>, gilt das Aktualisierungsverhalten, das im Dialog **Aktualisierungsverhalten bearbeiten** für Boxen festgelegt wurde.

## Verschiebbar

Durch das Verschieben einer Box wird ihr Offset verändert. Legen Sie hier fest, ob die Box zur Laufzeit frei im Diagrammbereich verschiebbar sein soll. Wenn die Box nicht mehr verschiebbar sein soll, nachdem Sie sie positioniert haben, schalten Sie diese Option danach ab.

## Ursprung

Mithilfe der Eigenschaften **Ursprung**, **Referenzpunkt**, **X-Offset** und **Y-Offset** können Sie jede einzelne Box im Diagrammbereich positionieren, wobei die relative Position der Box zum Diagramm unabhängig von der Diagrammgröße ist.

Legen Sie hier den Ursprung fest, d. h. den Diagrammpunkt, von dem aus der Abstand zum Referenzpunkt der Box in x- bzw. y-Richtung angegeben wird. Mögliche Werte des Ursprungs: links oben, mittig oben, rechts oben, links mittig, mittig mittig, rechts mittig, links unten, mittig unten, rechts unten.

## Referenzpunkt

Legen Sie hier den Referenzpunkt der Box fest, d. h. den Punkt der Box, von dem aus der Abstand zum Ursprung in x- bzw. y-Richtung angegeben wird. Mögliche Werte des Referenzpunkts: oben links, oben mittig, oben rechts, mittig links, mittig mittig, mittig rechts, unten links, unten mittig, unten rechts.

## X-Offset

Legen Sie hier den Abstand zwischen Ursprung und Referenzpunkt in x-Richtung fest.

## Y-Offset

Legen Sie hier den Abstand zwischen Ursprung und Referenzpunkt in y-Richtung fest.

## Rahmen

Wenn Sie auf dieses Feld klicken, erscheint eine **Bearbeiten**-Schaltfläche, über die Sie in den Dialog **Linie bearbeiten** gelangen. Hier können Sie den Typ, die Dicke und die Farbe der Umrandungslinie der Box festlegen.

## Priorität

Legen Sie hier die relative Zeichnungspriorität der Box zu anderen Objekten im Diagramm (Knoten, Liniengitter etc.) fest. Die Priorität von Knoten ist 0. Falls die Boxen eine höhere Priorität als die Knoten haben, überdecken sie die Knoten so, dass darauf interaktiv nicht mehr zugegriffen werden kann.

## Sichtbar

Legen Sie fest, ob die Box zur Laufzeit sichtbar sein soll.

## Boxformat

Hier wird das Boxformat der Box angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:



Aus der Kombobox können Sie ein vorhandenes Boxformat wählen.



Über die **Bearbeiten**-Schaltfläche gelangen Sie in den Dialog **Boxformate verwalten**.

## Box hinzufügen



Eine neue Box mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

## Box kopieren



Die markierte Box wird kopiert.

## Box löschen



Die Box, die Sie in der Liste markiert haben, wird gelöscht.

## Box bearbeiten



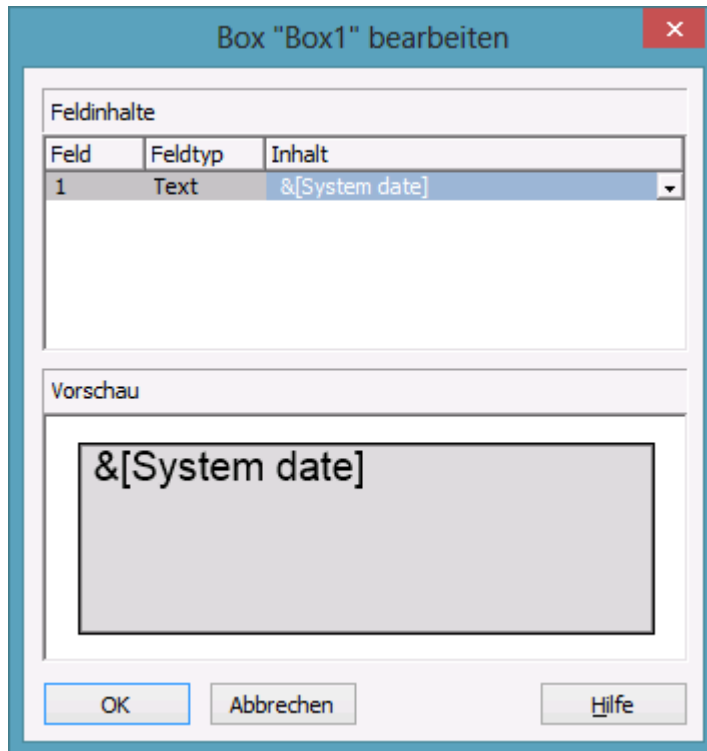
Sie gelangen in das Dialogfeld **Box bearbeiten**.

## Box eine Zeile nach oben/unten



Mit Hilfe dieser Schaltflächen können Sie die markierte Box in der Tabelle eine Zeile nach oben bzw. unten schieben.

## 4.19 Dialogfeld "Box bearbeiten"



Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Objekte** und das Dialogfeld **Boxen verwalten**, indem Sie dort auf die **Box bearbeiten**-Schaltfläche klicken. Dieser Dialog erscheint auch zur Laufzeit, wenn sie auf eine Box doppelklicken.

### Feld

In dieser Spalte werden die Nummern aller Felder der Box aufgeführt. (Die Anzahl der Felder hängt vom gewählten Boxformat ab.)

### Feldtyp

Hier wird der Feldtyp jedes Feldes angezeigt (Text oder Grafik).

### Inhalt

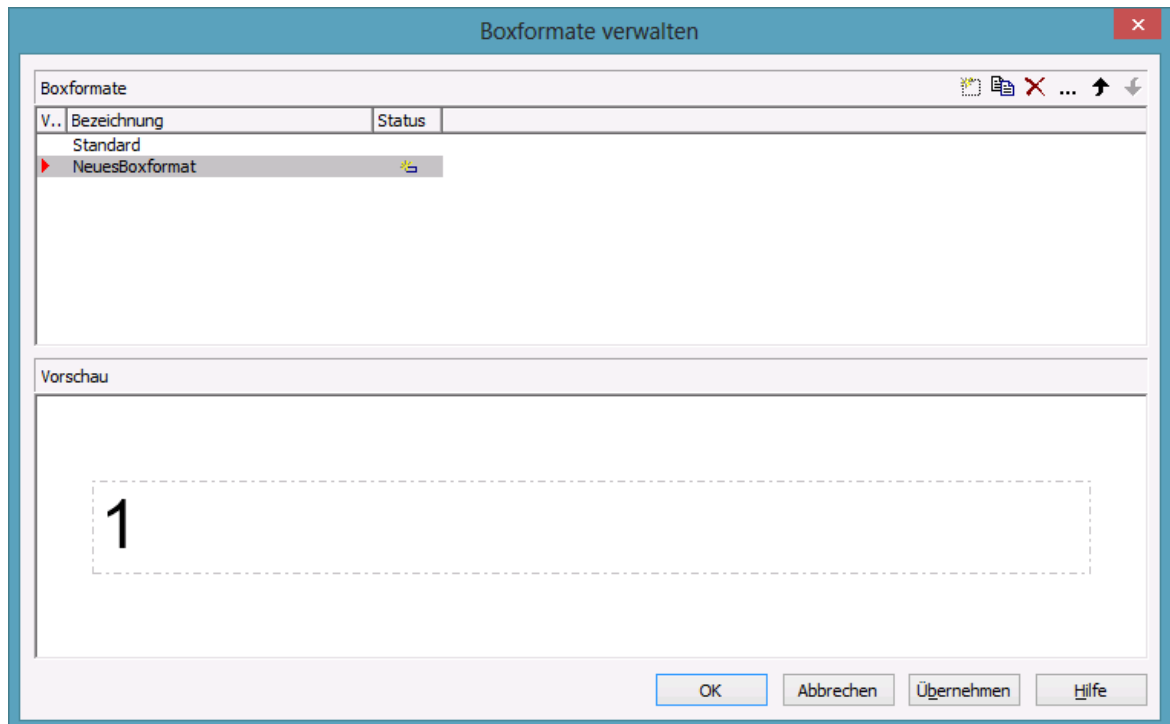
Hier können Sie den Inhalt des Feldes bzw. den Namen einer Grafikdatei eingeben.

Bei mehrzeiligen Textfeldern müssen für einen erzwungenen Umbruch die einzelnen Zeilen des Textfelds mit "\n" im String getrennt sein (Beispiel: "Zeile1\nZeile2"). Ohne erzwungenen Umbruch wird automatisch an Leerzeichen umgebrochen.

**206** Dialogfeld "Box bearbeiten"

Mögliche Grafikformate: WMF, JPG, BMP, GIF, PCX, PNG, TIF.

## 4.20 Dialogfeld "Boxformate/Knotenformate verwalten"



Sie gelangen in dieses Dialogfeld über die Eigenschaftenseite **Objekte**.

### Vorschau

Das in der Vorschau-Spalte markierte Format wird im Vorschau-Fenster angezeigt.

### Bezeichnung


In dieser Spalte stehen die Namen aller vorhandenen Formate. Die Namen sind editierbar.

### Status

In der Spalte **Status** wird jedes Format gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt (🔧) und/oder geändert (⚠️) worden ist.




## Box/Knotenformat hinzufügen

 Ein neues Format mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

## Box/Knotenformat kopieren

 Das markierte Format wird kopiert.



## Box/Knotenformat löschen

 Das Format, das Sie in der Liste markiert haben, wird gelöscht. Es können nur die Formate gelöscht werden, die zur Zeit nicht benutzt werden.

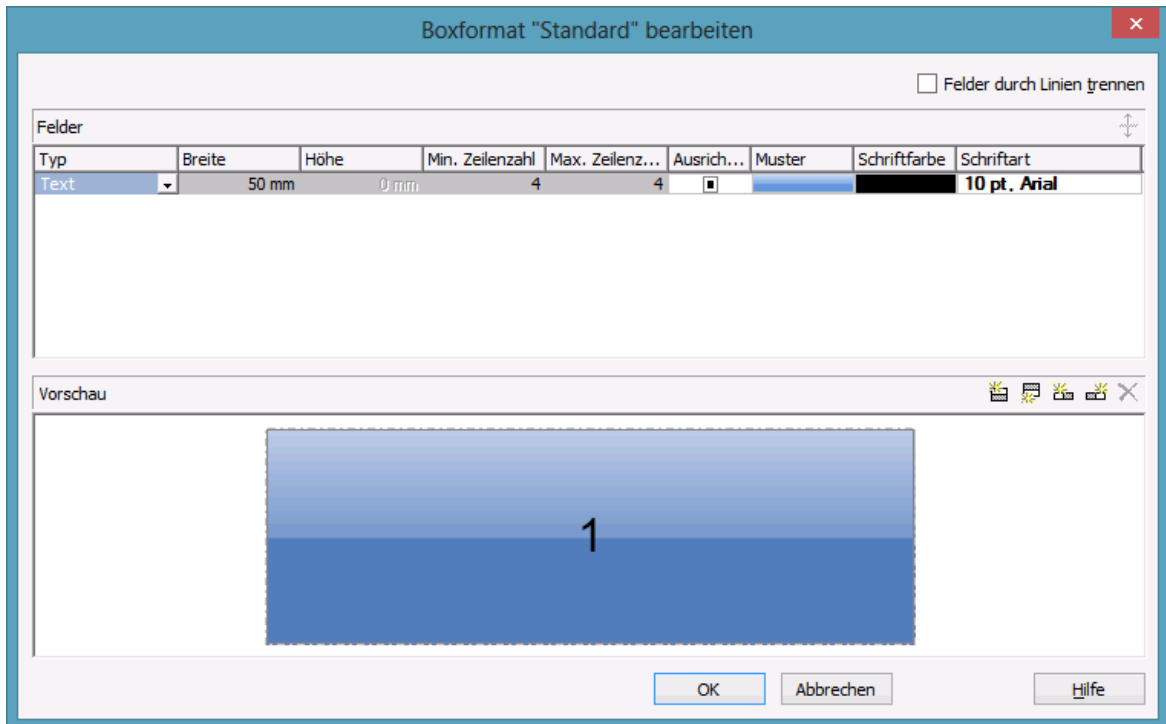
## Box/Knotenformat bearbeiten

 Sie gelangen in das Dialogfeld **Boxformat bearbeiten** bzw. **Knotenformat bearbeiten**.

## Box / Knotenformat eine Zeile nach oben / unten

  Mit Hilfe dieser Schaltflächen können Sie das markierte Format in der Tabelle eine Zeile nach oben bzw. unten schieben.

## 4.21 Dialogfeld "Boxformat bearbeiten"



Sie erreichen dieses Dialogfeld, indem Sie auf der Eigenschaftenseite **Objekte** das Dialogfeld **Boxformate verwalten** aktivieren und dort auf die **Boxformat bearbeiten**-Schaltfläche klicken.

### Felder durch Linien trennen

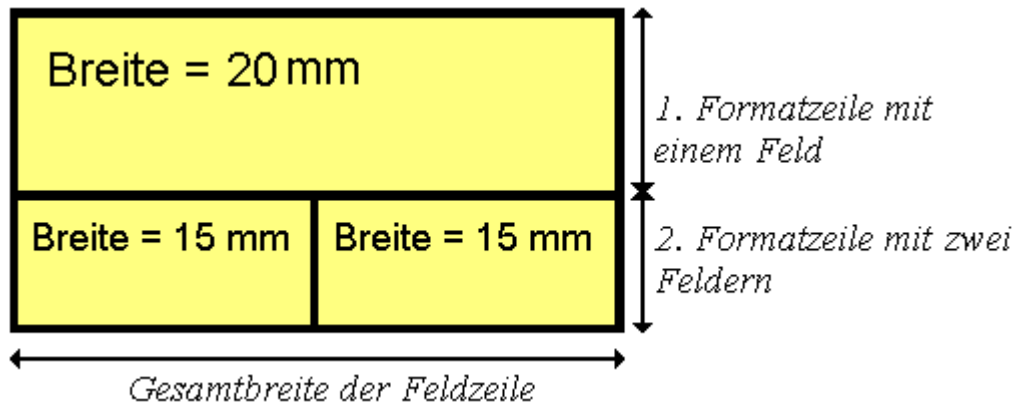
Aktivieren Sie dieses Kontrollkästchen, wenn die Felder der Box durch Linien getrennt werden sollen.

### Typ

Wählen Sie hier den Feldtyp (Text oder Grafik).

### Breite

Legen Sie hier die Breite in Millimetern für das markierte Feld fest. Die maximale Breite eines Feldes beträgt 200 mm. Ist eine Feldzeile in zwei oder mehr Felder unterteilt und die Gesamtbreite der einzelnen Feldzeilen unterschiedlich groß, so richtet sich die Gesamtbreite nach der insgesamt breitesten Feldzeile.



## Höhe

(nur für den Typ *Grafik*) Legen Sie hier die minimale Höhe in Millimetern für das markierte Feld fest. Die maximale Höhe eines Feldes beträgt 200 mm.

## Min./Max. Zeilenzahl

(nur für den Typ *Text*) Bestimmen Sie die minimale bzw. die maximale Anzahl der Textzeilen des aktuellen Feldes. Die maximale Zeilenzahl eines Feldes beträgt neun.

## Ausrichtung

Wählen Sie hier die Ausrichtung des Inhalts in dem markierten Feld (9 Möglichkeiten).

## Muster

Legen Sie hier Hintergrundfarbe und -muster des Feldes fest. Durch Klick auf  gelangen Sie in den Dialog **Musterattribute bearbeiten**, in dem Sie ein Muster, eine Hintergrundfarbe sowie eine 2. Musterfarbe auswählen können. Zusätzlich zu der vorgegebenen Farbpalette können Sie beliebige Farben definieren. Es stehen außerdem transparente Farben zur Verfügung.

## Schriftfarbe

(nur für den Typ *Text*) Die Schriftfarbe des Feldes wird hier angezeigt.


Die Farbzuordnungstabelle öffnet sich, und Sie können daraus eine Schriftfarbe auswählen.

## Schriftart

*(nur für den Typ Text)* Die Schriftart des Feldes wird hier angezeigt.





 Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**.

## selektierte Eigenschaft in alle Felder übernehmen

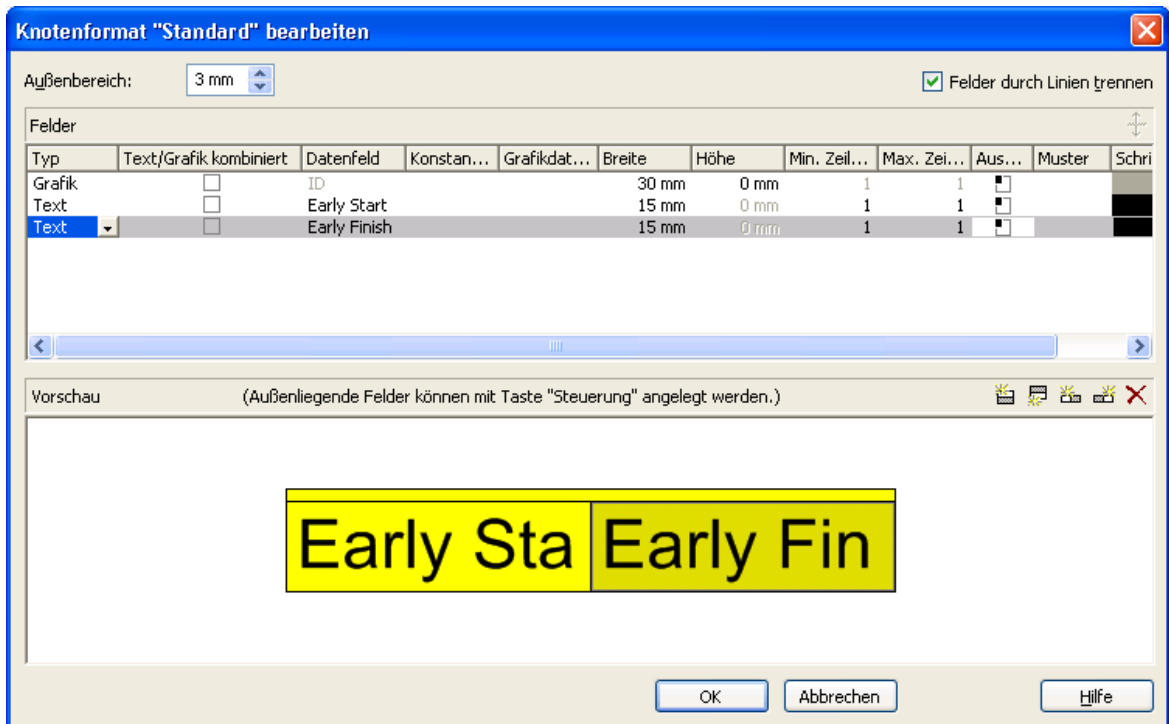
 Klicken Sie auf diese Schaltfläche, damit die selektierte Eigenschaft in alle Felder übernommen wird.

## Vorschau

Hier werden die aktuellen Felder des Boxformats dargestellt. Wenn Sie im Vorschaufenster ein Feld anklicken, können Sie dessen Attribute in der **Felder**-Tabelle bearbeiten.

    Mit Hilfe der Schaltflächen oberhalb der Vorschau können Sie neue Felder anlegen oder das markierte Feld löschen. Zum Löschen von Feldern können Sie auch die Entf-Taste benutzen.

## 4.22 Dialogfeld "Knotenformat bearbeiten"



Sie gelangen in dieses Dialogfeld, indem Sie im Dialogfeld **Knotenformate verwalten** auf die **Knotenformat bearbeiten**-Schaltfläche klicken.

### Außenbereich

Legen Sie hier den Abstand in Millimetern fest, den Knoten mit diesem Knotenformat zu benachbarten Knoten und zum Rand der Darstellung halten sollen. Standardmäßig beträgt der Außenbereich 3 mm. Bei kleineren Werten kann es gelegentlich zu Überlagerungen von grafischen Elementen kommen. Daher sollten Sie den Standardwert nur in begründeten Fällen unterschreiten.

### Felder durch Linien trennen

Aktivieren Sie dieses Kontrollkästchen, wenn die einzelnen Felder durch Linien getrennt werden sollen.

### Typ

Wählen Sie hier den Feldtyp (Text oder Grafik).

## Text/Grafik kombiniert

Wenn dieses Kontrollkästchen aktiviert ist, können in dem Knotenfeld ein Text und eine Grafik kombiniert werden, und zwar folgendermaßen:

- **Typ:** Text, **Text/Grafik kombiniert:** nein: nur Text wird ausgegeben (wie unter **Datenfeld** oder unter **Konstanter Text** angegeben)
- **Typ:** Grafik, **Text/Grafik kombiniert:** nein: nur eine Grafik wird ausgegeben (wie unter **Grafikdateiname** angegeben)
- **Typ:** Text, **Text/Grafik kombiniert:** ja: Text (wie unter **Datenfeld** oder unter **Konstanter Text** angegeben) und Grafik (wie unter **Grafikdateiname** angegeben) werden ausgegeben
- **Typ:** Grafik, **Text/Grafik kombiniert:** ja: nur eine Grafik wird ausgegeben (wie unter **Grafikdateiname** angegeben). Text (unter **Datenfeld** angegeben) ist nur im Tooltip sichtbar; er kann aber, falls geeignet, als Hyperlink genutzt werden.)

## Datenfeld

Wählen Sie hier das Datenfeld, dessen Inhalt in dem aktuellen Feld ausgegeben werden soll. Passt der Inhalt des Datenfeldes nicht in das Feld, wird der Überhang bei der Ausgabe abgeschnitten.


## Konstanter Text

*(nur wenn kein Datenfeld gewählt wurde)* Sie können hier einen konstanten Text eingeben, der in dem Knotenfeld ausgegeben werden soll.

## Grafikdateiname

Hier werden Name und Pfad der Grafikdatei angezeigt, die in dem gewählten Knotenfeld dargestellt werden soll.

Wenn Sie auf ein **Grafikdateiname**-Feld klicken, erscheinen zwei Schaltflächen:

 Der Windows-Dialog **Grafikdatei auswählen** erscheint. Hier können Sie eine Grafikdatei auswählen, die in dem aktuellen Formatfeld erscheinen soll.

Wird ein relativer Dateiname angegeben, so wird die Datei zur Laufzeit zuerst in dem Verzeichnis gesucht, das in der VARCHART-Windows-Forms-Eigenschaft **FilePath** gesetzt ist. Wird sie dort nicht gefunden, wird sie zuerst im gerade aktiven Arbeitsverzeichnis der Applikation und dann im

Installationsverzeichnis des VARCHART Windows-Forms-Steuerelement gesucht.



Klicken Sie auf diese Schaltfläche (**Zuordnungen einstellen**), um eine Zuordnungstabelle zu verwenden, um Grafiken abhängig vom Inhalt eines Datenfelds in einem Knotenfeld erscheinen zu lassen.

Wird im Dialog **Zuordnung einstellen** nur ein Datenfeld, aber keine Zuordnungstabelle ausgewählt, so wird der Inhalt des eingestellten Datenfelds direkt als Name einer Grafikdatei interpretiert. Findet sich in dem Datenfeld bzw. in der Zuordnungstabelle kein gültiger Name einer Grafikdatei, dann wird der direkt in Spalte angegebene Dateiname verwendet.

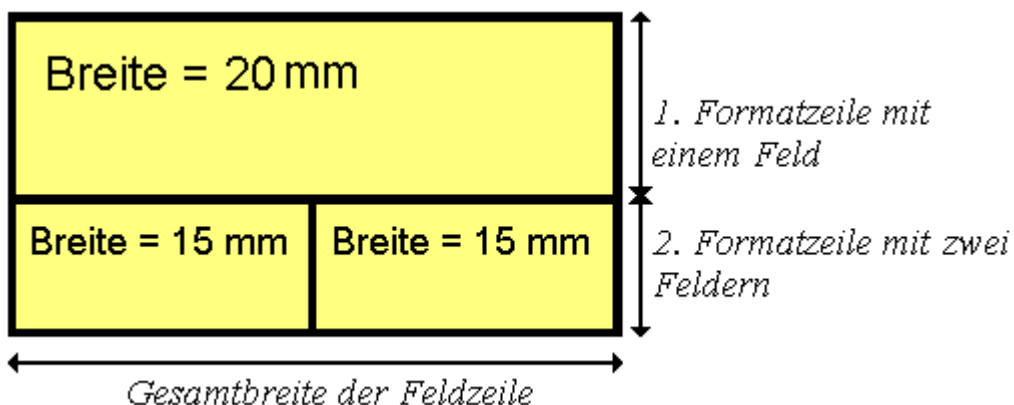
Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der 2. Schaltfläche fett dargestellt (**⇄**).

⇄ Sobald Sie die entsprechende Zeile verlassen, zeigt ein Symbol im Feld **Grafikdateiname** an, dass eine Zuordnung vorgenommen worden ist.

Bei der Darstellung der Grafik wird die Farbe des Pixels von ihrer Ecke links oben durch die Diagramm-Hintergrundfarbe ersetzt, d. h. alle Pixel der Grafik in dieser Farbe werden transparent dargestellt.

## Breite

Legen Sie hier die Breite in Millimetern für das markierte Feld fest. Die maximale Breite eines Feldes beträgt 99 mm. Ist eine Feldzeile in zwei oder mehr Felder unterteilt und die Gesamtbreite der einzelnen Feldzeilen unterschiedlich groß, so richtet sich die Gesamtbreite nach der insgesamt breitesten Feldzeile.



## Höhe

(nur für den Typ Grafik) Legen Sie hier die minimale Höhe in Millimetern für das markierte Feld fest. Die maximale Höhe beträgt 99 mm.


## Min./Max. Zeilenzahl



(*nur für den Typ Text*) Bestimmen Sie die minimale und die maximale Anzahl der Textzeilen des aktuellen Feldes. Die maximale Zeilenzahl eines Feldes beträgt neun.

## Ausrichtung

Hier bestimmen Sie die Ausrichtung des Textes bzw. der Grafik des markierten Feldes.

## Muster


Legen Sie hier Hintergrundfarbe und -muster des Feldes fest. Durch Klick auf  gelangen Sie in den Dialog **Musterattribute bearbeiten**, in dem Sie ein Muster, eine Hintergrundfarbe sowie eine 2. Musterfarbe auswählen können. Zusätzlich zu der vorgegebenen Farbpalette können Sie beliebige Farben definieren. Es stehen außerdem transparente Farben zur Verfügung.



: Über diese Schaltfläche im Dialog **Musterattribute bearbeiten** gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie für das jeweilige Attribut eine datenabhängige Zuordnung vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt (.

Wird hier nichts festgelegt, wird automatisch das Attribut benutzt, das für das Knotenaussehen vereinbart wurde.

## Schriftfarbe

(*nur für den Typ Text*) Die Schriftfarbe des Feldes wird angezeigt, und Sie können sie bearbeiten. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Die Farbzusordnungstabelle öffnet sich, und Sie können daraus eine Schriftfarbe auswählen.


 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Farbe vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt (.



## Schriftart





Die Schriftart des Feldes wird angezeigt, und Sie können sie bearbeiten. Wenn Sie auf dieses Feld klicken, erscheint eine Schaltfläche (⋮), über die Sie den Windows-Dialog **Schriftart** öffnen können.

## selektierte Eigenschaft in alle Felder übernehmen

 Klicken Sie auf diese Schaltfläche, damit die selektierte Eigenschaft in alle Felder übernommen wird.

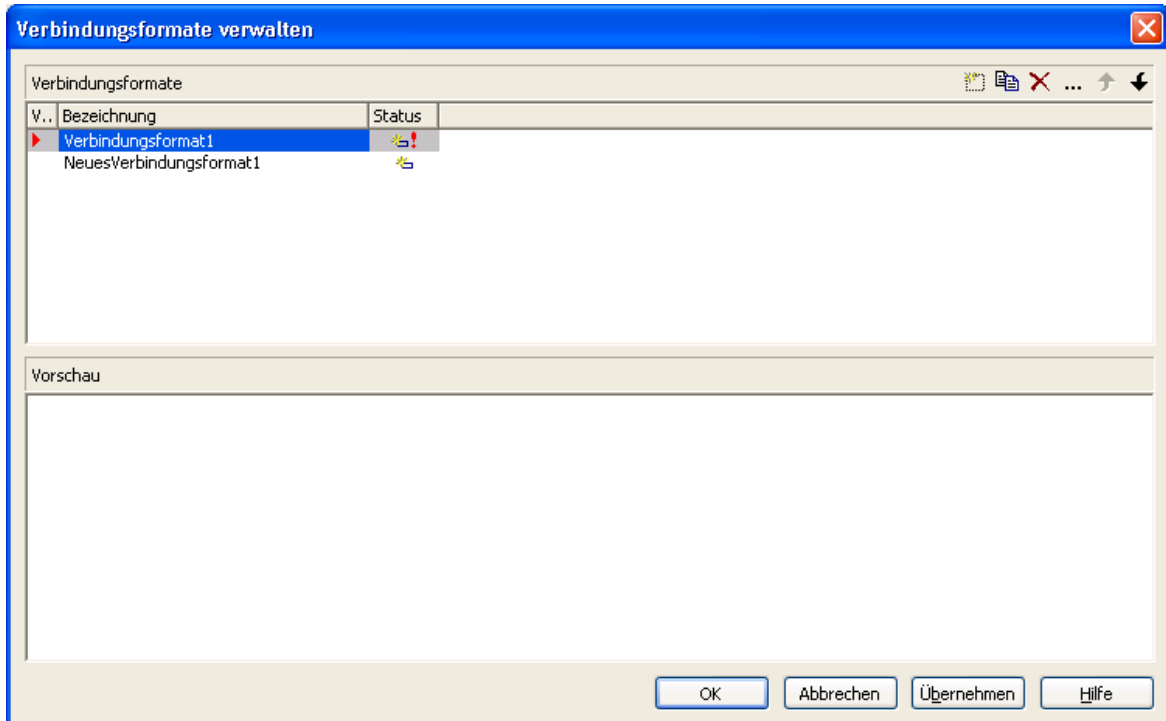
## Vorschau

Hier wird das aktuelle Knotenformat dargestellt. Wenn Sie im Vorschau-fenster ein Feld anklicken, können Sie dessen Attribute in der **Felder**-Tabelle bearbeiten.

    Mit Hilfe der Schaltflächen oberhalb der Vorschau können Sie neue Felder anlegen oder das markierte Feld löschen. Sie können auch die Entf-Tast benutzen, um Felder zu löschen.

Wenn Sie Felder außerhalb des Knotens anlegen möchten, so ist zusätzlich die Strg-Taste zu drücken.

## 4.23 Dialogfeld "Verbindungsformate verwalten"



Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Objekte** oder aus dem Dialog **Verbindungsausssehen verwalten** durch Klick auf **...** im Feld **Verbindungsformate**.

### Vorschau

In dieser Spalte wird das jeweils im Vorschaufenster angezeigte Verbindungsformat gekennzeichnet.


### Bezeichnung

In dieser Spalte werden die Bezeichnungen aller Verbindungsformate aufgeführt. Die Bezeichnungen sind editierbar.

### Status

In dieser Spalte wird jedes Verbindungsformat gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt (⚙️) und/oder geändert (⚙️ !) wurde.


## Verbindungsformat hinzufügen

 Ein neues Linienformat mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.



## Verbindungsformat kopieren

 Das markierte Linienformat wird kopiert.

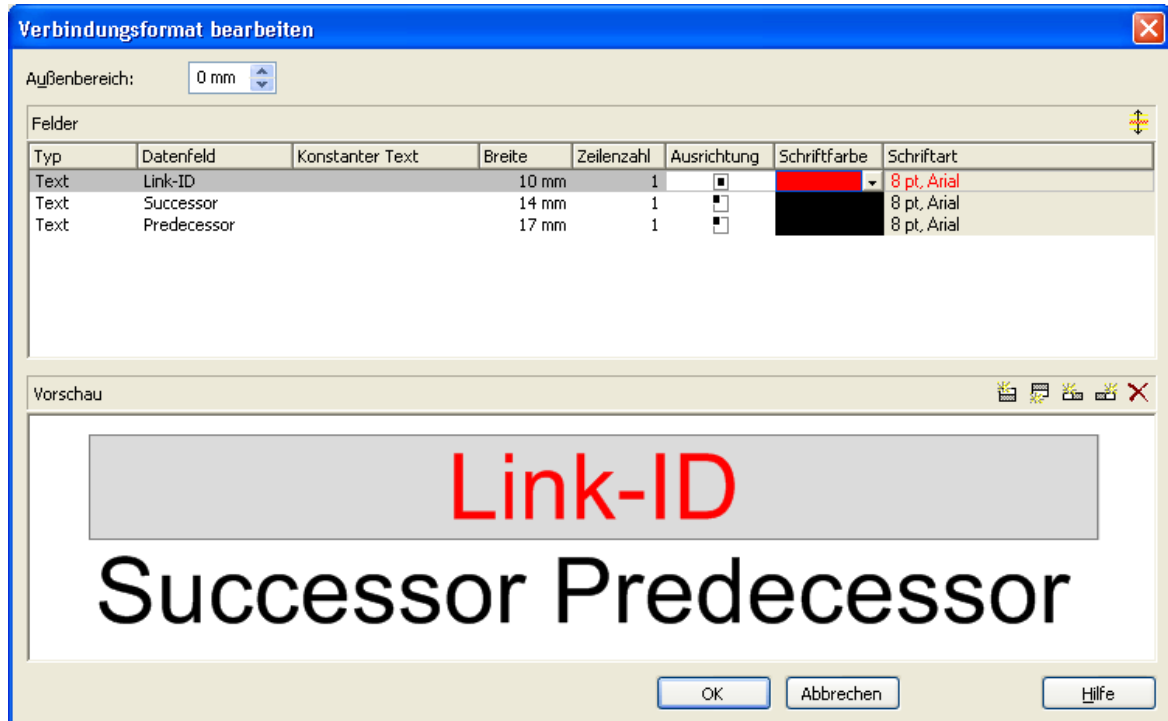
## Verbindungsformat löschen

 Der Filter, den Sie in der Liste markiert haben, wird gelöscht. Es können nur Filter gelöscht werden, die zur Zeit nicht benutzt werden.

## Verbindungsformat eine Zeile nach oben/unten

  Mit Hilfe dieser Schaltflächen können Sie das markierte Linienformat eine Zeile nach oben/unten schieben.

## 4.24 Dialogfeld "Verbindungsformat bearbeiten"



Sie erreichen dieses Dialogfeld wenn Sie auf der Eigenschaftenseite **Verbindungen** auf die **Format bearbeiten**-Schaltfläche klicken.

### Außenbereich

Legen Sie hier den Abstand in Millimetern fest, den Verbindungen mit diesem Verbindungsformat zu benachbarten Knoten halten sollen. Standardmäßig beträgt der Außenbereich 3 mm. Bei kleineren Werten kann es gelegentlich zu Überlagerungen von grafischen Elementen kommen. Daher sollten Sie den Standardwert nur in begründeten Fällen unterschreiten.

### Typ

Der Feldtyp ist Text.

### Datenfeld

Wählen Sie hier das Datenfeld, dessen Inhalt in dem aktuellen Feld ausgegeben werden soll. Passt der Inhalt des Datenfeldes nicht in das Feld, wird der Überhang bei der Ausgabe abgeschnitten.

## Konstanter Text

*(nur wenn kein Datenfeld gewählt wurde)* Sie können hier einen konstanten Text eingeben, der in dem Feld ausgegeben werden soll.

## Breite

Legen Sie hier die Breite in Millimetern für das markierte Feld fest. Die maximale Breite eines Feldes beträgt 99 mm. Ist eine Feldzeile in zwei oder mehr Felder unterteilt und die Gesamtbreite der einzelnen Feldzeilen unterschiedlich groß, so richtet sich die Gesamtbreite nach der insgesamt breitesten Feldzeile.


## Zeilenzahl

Bestimmen Sie die Anzahl der Textzeilen des aktuellen Feldes. Die maximale Zeilenzahl eines Feldes beträgt neun.


## Ausrichtung

Hier bestimmen Sie die Ausrichtung des Textes des markierten Feldes.


## Schriftfarbe

Die Schriftfarbe des Feldes wird hier angezeigt, und Sie können sie hier bearbeiten. Wenn Sie auf dieses Feld klicken, erscheint eine Schaltfläche () , über die Sie die Farbauswahltabelle öffnen können.

## Schriftart

Die Schriftart des Feldes wird hier angezeigt, und Sie können sie hier bearbeiten. Wenn Sie auf dieses Feld klicken, erscheint eine Schaltfläche () , über die Sie den Windows-Dialog **Schriftart** öffnen können.

## selektierte Eigenschaft in alle Felder übernehmen

 Klicken Sie auf diese Schaltfläche, damit die selektierte Eigenschaft in alle Felder übernommen wird.

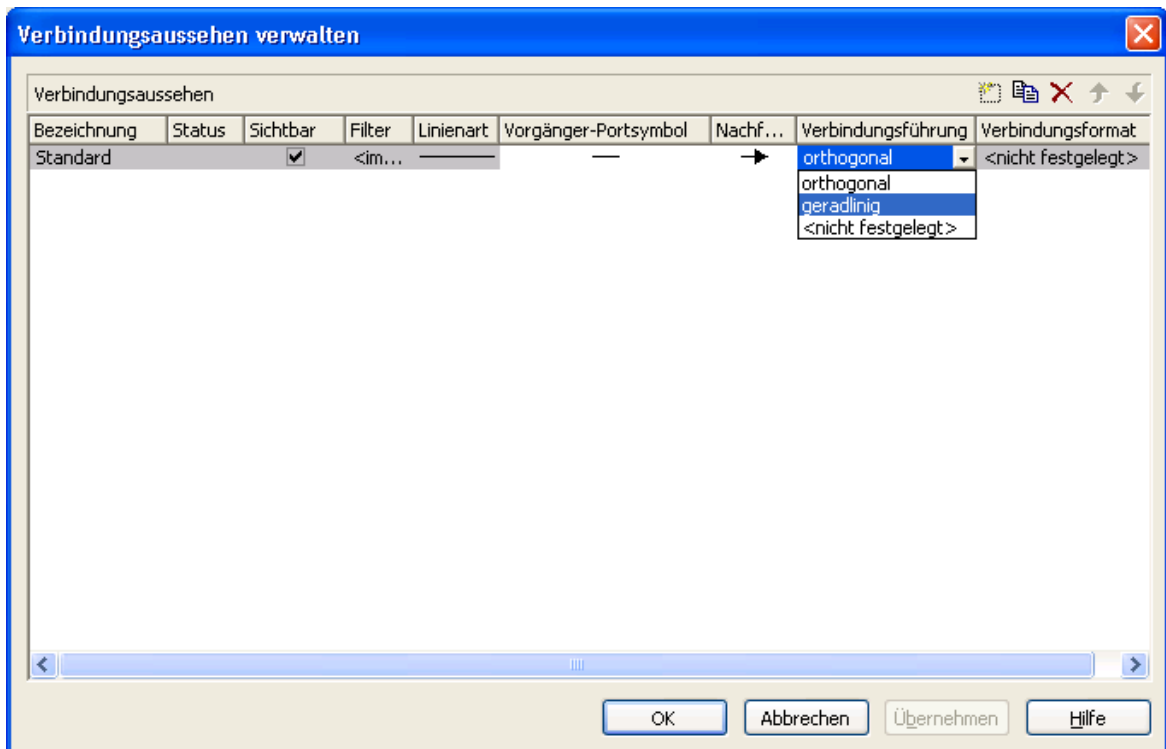
## Vorschau

Hier wird das aktuelle Verbindungsformat dargestellt. Wenn Sie im Vorschauenfenster ein Feld anklicken, können Sie dessen Attribute in der **Felder**-Tabelle bearbeiten.



Mit Hilfe der Schaltflächen oberhalb der Vorschau können Sie neue Felder anlegen oder das markierte Feld löschen. Sie können auch die Entf-Tast benutzen, um Felder zu löschen.

## 4.25 Dialogfeld "Verbindungsausssehen verwalten"





Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Objekte**.

### Bezeichnung

In dieser Spalte werden die Namen der verfügbaren Verbindungsausssehen angezeigt. Die Namen sind editierbar.

Diese Option kann auch über die Eigenschaft **LinkAppearanceName** festgelegt werden.

### Status

In der Spalte **Status** wird jedes Verbindungsausssehen gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

### Sichtbar

Wenn Sie dieses Kontrollkästchen aktivieren, werden die Verbindungen angezeigt.


Diese Option kann auch über die Eigenschaft **VcLinkAppearance.Visible** festgelegt werden.

## Filter

Wählen Sie hier den Filter, der für ein Verbindungsaussehen verwendet werden soll aus.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.Filter-Name** festgelegt werden.

## Linienart

Wenn Sie auf den Eintrag dieses Feldes klicken, erscheint die **Bearbeiten**-Schaltfläche . Durch Klick auf diese Schaltfläche gelangen Sie in das Dialogfeld **Linienattribute bearbeiten**, in dem Sie das Aussehen der Verbindungslinien festlegen können.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.LineType** festgelegt werden.

## Vorgänger-Portsymbol

Wählen Sie hier für jedes Verbindungsaussehen ein Portsymbol, das die Einmündung zum Vorgängerknoten kennzeichnet.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.PredecessorPortSymbol** festgelegt werden.

## Nachfolger-Portsymbol

Wählen Sie hier für jedes Verbindungsaussehen ein Portsymbol, das die Einmündung zum Nachfolgerknoten kennzeichnet.

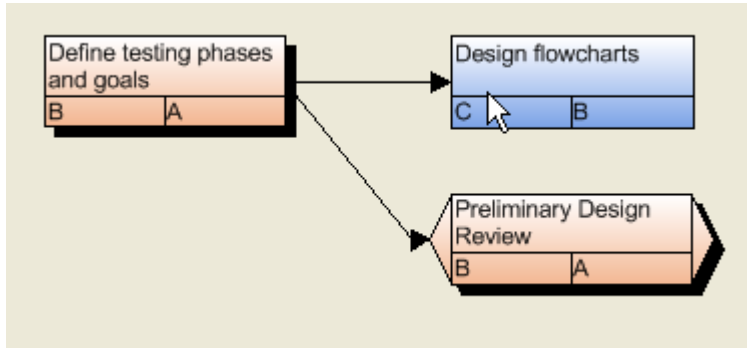
Diese Option kann auch über die Eigenschaft **VcLinkAppearance.SuccessorPortSymbol** festgelegt werden.

## Verbindungsführung

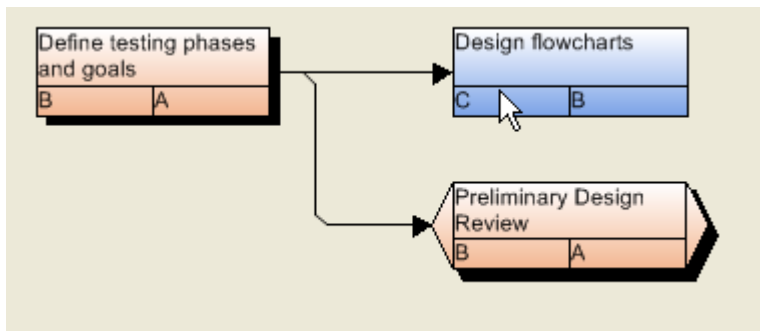
Wählen Sie hier die Art der Verbindungsführung. Der Eintrag <nicht festgelegt> ist erst ab der zweiten Zeile der Tabelle mit den Verbindungsaussehen wählbar, da die erste Zeile immer das Standard-Verbindungsaussehen enthält. Ist <nicht festgelegt> ausgewählt, wird ein in der Liste der LinkAppearance-Objekte weiter vorne stehender Verbindungsführungstyp verwendet.



Sie können die Art der Verbindungsführung auch über die **VcLinkAppearance**-Eigenschaft **RoutingType** setzen.





Geradliniger Verbindungstyp




Orthogonaler Verbindungstyp

## Verbindungsformat


Wählen Sie hier durch Klick auf  ein vorhandenes Verbindungsausssehen aus oder klicken auf  um den Dialog **Verbindungsformate verwalten** zu öffnen, in dem Sie Verbindungsformate erstellen bzw. bearbeiten können.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.Format-Name** festgelegt werden.


## Verbindungsausssehen hinzufügen

 Ein neues Verbindungsausssehen mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.



## Verbindungsausssehen kopieren

 Das markierte Verbindungsausssehen wird kopiert.

## Verbindungsausssehen löschen

 Das Verbindungsausssehen, das Sie in der Liste markiert haben, wird gelöscht. Es können nur Verbindungsausssehen gelöscht werden, die zur Zeit nicht benutzt werden.

## Verbindungsausssehen eine Zeile nach oben/unten

  Mit Hilfe dieser Schaltflächen können Sie das markierte Verbindungsausssehen eine Zeile nach oben/unten schieben.

## 4.26 Dialogfeld "In-Flow-Gruppierung bearbeiten"

Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Knoten**. Sie können hier die Kriterien und das Layout der In-Flow-Gruppierung bearbeiten. Bei einer Orientierung des Diagramms von links nach rechts können Sie oberhalb und/oder unterhalb des Diagrammbereichs eine Titelleiste ausgeben. Bei einer Orientierung von oben nach unten können Sie links und/oder rechts des Diagrammbereichs eine Titelleiste ausgeben.

### Code aus Feld

Wählen Sie das Datenfeld aus, das die In-Flow-Gruppierung bestimmen soll.

### Zeitintervall

*(Nur verfügbar, falls unter **Code aus Feld** ein Termindatenfeld gewählt ist)*  
Legen Sie hier das Zeitintervall fest, in das die Titelleisten unterteilt werden sollen (z. B. 1 Sekunde, 1 Minute, 1 Stunde, 1 Tag, 2 Monate, 1 Jahr).

## Trennlinien

Wenn Sie dieses Kontrollkästchen aktivieren, werden vertikale Trennlinien (bei Anordnung von links nach rechts) bzw. horizontale Trennlinien (bei Anordnung von oben nach unten) im Diagramm ausgegeben. Falls Sie unter **Code aus Feld** ein Termindatenfeld gewählt haben, wird der Abstand dieser Trennlinien durch den unter **Zeitintervall** festgelegten Abstand bestimmt. Andernfalls wird nach jedem Wert des Datenfeldes eine Trennlinie gezogen.

Das Aussehen der Trennlinien können Sie festlegen, indem Sie über die **Bearbeiten**-Schaltfläche den Dialog **Linienattribute öffnen** aufrufen. Hier können Sie Typ, Dicke und Farbe der Trennlinien festlegen.


## Titelleisten oben/unten bzw. links/rechts

Hier können Sie festlegen, ob beschriftete Titelleisten ausgegeben werden sollen:

- Anordnung von links nach rechts: Titelleisten oberhalb und/oder unterhalb des Diagramms
- Anordnung von oben nach unten: Titelleisten links und/oder rechts vom Diagramm.

## Schrift

Die aktuelle Schriftart und Schriftfarbe der Titelleisten werden hier angezeigt.

 Über diese Schaltfläche öffnen Sie die Farbauswahltabelle, in der Sie die Schriftfarbe auswählen können.

 Über die Schaltfläche gelangen Sie in den Windows-Dialog **Schriftart**.

## Hintergrundfarbe

Wählen Sie hier die Hintergrundfarbe der Titelleisten aus.

## Breite

*(Nur für Orientierung von oben nach unten)* Legen Sie hier die Breite der vertikalen Titelleisten in mm fest.

## Datumsformat

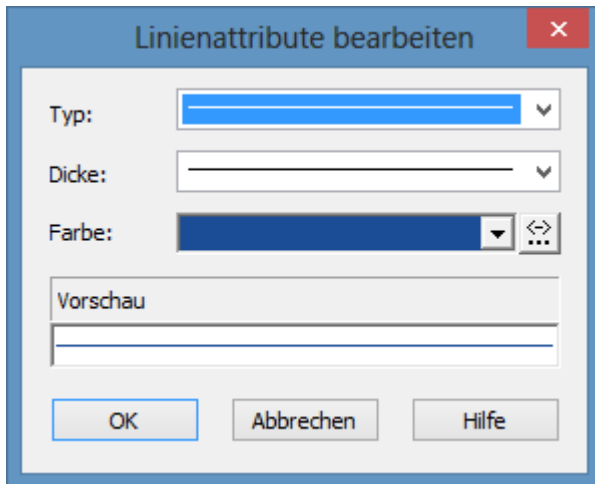
Wählen Sie diese Option, wenn Sie für **Code aus Feld** ein Datumsfeld gewählt haben, und legen Sie dann das Terminformat für die Beschriftung der Titelleisten fest.


## Texte

Legen Sie hier die Beschriftung der Titelleisten fest:

- **aus Feld:** Wählen Sie diese Option, wenn die Beschriftung der Titelleisten durch ein Datenfeld bestimmt werden soll, und wählen Sie dann das entsprechende Datenfeld aus.
- **aus Datei:** Wählen Sie diese Option, wenn die Beschriftung der Titelleisten durch eine Datei bestimmt werden soll, und geben Sie ggf. den Namen dieser Datei an.

## 4.27 Dialogfeld "Linienattribute bearbeiten"



Der Dialog zum Bearbeiten der Linienattribute, der jeweils über die Schaltfläche  aufgerufen werden kann, steht zur Verfügung für das Verbindungsausssehen, Layer, sowie für den Rahmen von Boxen.

### Typ


Wählen Sie hier den Typ der Linie aus (durchgezogen, gestrichelt, etc.).

### Dicke

Wählen Sie hier die Linienstärke aus.

### Farbe

Wählen Sie hier die Farbe der Linien aus.

 Über diese Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**, in dem Sie eine datenabhängige Zuordnung der Linienfarbe vereinbaren können.


 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt.

### Vorschau

Hier wird das Aussehen der Linie mit den gewählten Einstellungen angezeigt.

## 4.28 Dialogfeld "Musterattribute bearbeiten"



Der Muster-Dialog, der jeweils über die Schaltfläche  aufgerufen werden kann, steht zur Verfügung für Box- und Knotenformate.

### **Muster**

Hier können Sie ein Hintergrundmuster auswählen.

### **Musterfarbe**

Wählen Sie hier die Vordergrund-Farbe des Musters aus.

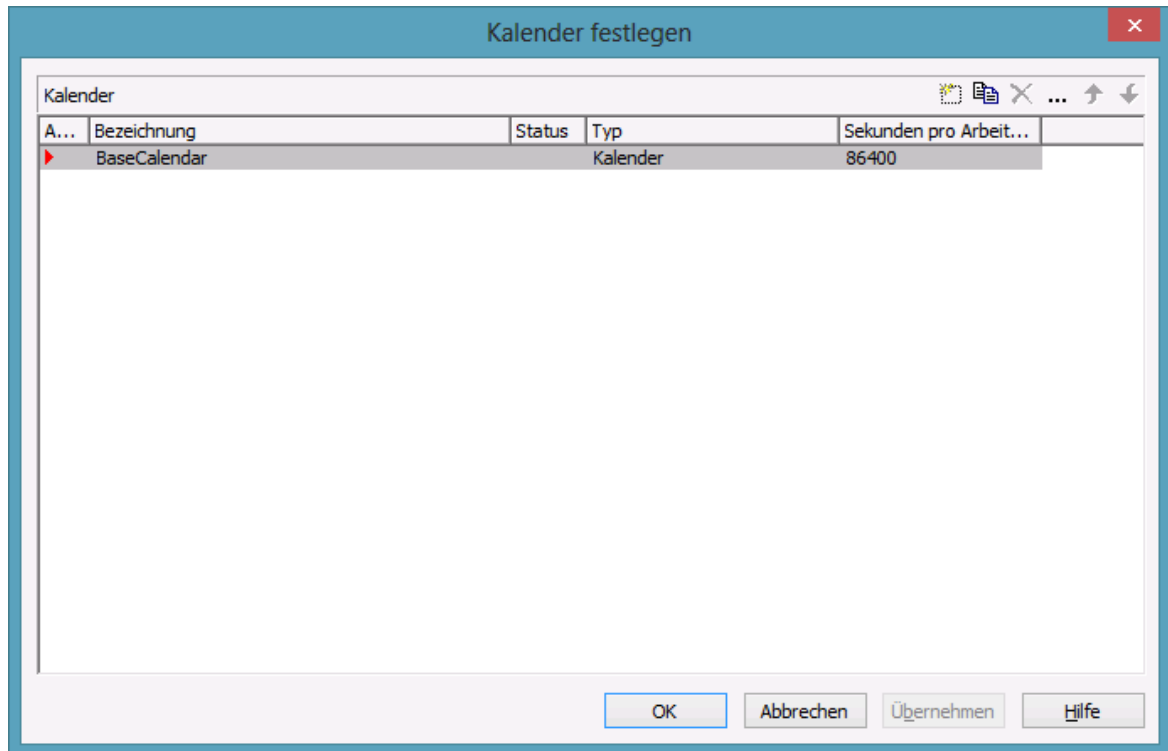
### **Hintergrundfarbe oder Musterfarbe 2**

Wählen Sie hier die Hintergrund-Farbe oder eine zweite Musterfarbe aus.

### **Vorschau**

Hier wird das Aussehen des Musters mit den gewählten Einstellungen angezeigt.

## 4.29 Dialogfeld "Kalender festlegen"



Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Objekte**. In jeder Zeile der Tabelle können Sie einen Kalender definieren.



### Ausgewählt

Der Kalender, den Sie in dieser Spalte durch eine kleine Pfeilspitze markieren, wird für das Kalendergitter verwendet.

### Bezeichnung

In dieser Spalte werden die Namen aller definierten Kalender aufgeführt.

### Status

In der Spalte **Status** wird jeder Kalender gekennzeichnet, der seit dem Aufruf des Dialogs hinzugefügt (  ) und/oder geändert (  ) worden ist.

### Typ


Legen Sie hier den Kalendertyp fest. Außer normalen Kalendern sind auch Schichtkalender möglich.



## Sekunden pro Arbeitstag

Legen Sie hier fest, wie viele Sekunden der Arbeitstag hat.

## Kalender hinzufügen

 Um einen neuen Kalender zu definieren, klicken Sie auf diese Schaltfläche.

## Kalender kopieren

 Der markierte Kalender wird kopiert.

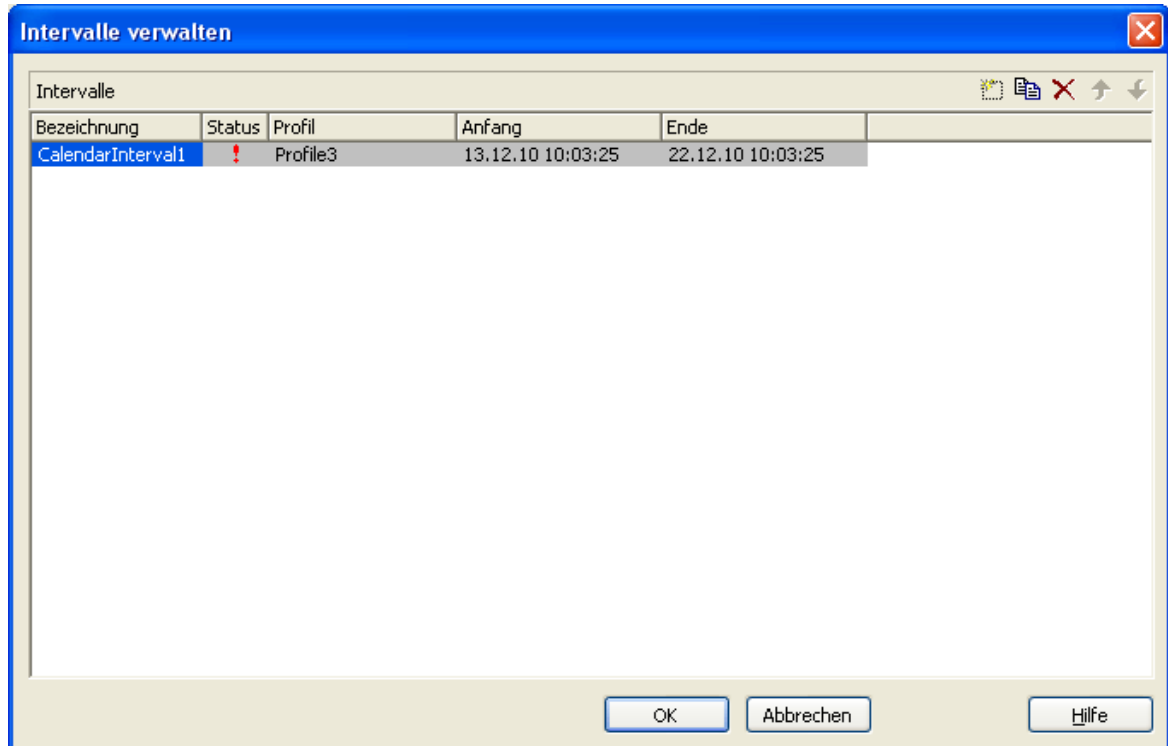
## Kalender löschen

 Der markierte Kalender wird gelöscht.

## Kalender bearbeiten

 Wenn Sie diese Schaltfläche anklicken, öffnet sich der Dialog **Kalender bearbeiten**.

## 4.30 Dialogfeld "Intervalle verwalten" (Kalender)



In diesem Dialogfeld können Sie Intervalle bearbeiten.

### Bezeichnung

In dieser Spalte werden die Bezeichnungen aller Intervalle aufgeführt. Die Bezeichnungen sind editierbar.

### Status

In dieser Spalte wird jedes Intervall gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt (📅) und/oder geändert (!) worden ist.


### Profil

Weisen Sie hier dem Intervall ein Kalenderprofil zu. Über ▾ öffnen Sie eine Liste mit Profilen, aus der Sie das gewünschte Profil auswählen können. Zum Bearbeiten des Profils klicken Sie neben dem Profilnamen auf ... um den Dialog **Kalenderprofile verwalten** zu öffnen.

## Anfang/Ende

Legen Sie hier für das jeweilige Intervall den Anfang bzw. das Ende fest. Das Datum lässt sich komfortabel mithilfe eines Spincontrols eingeben bzw. ändern.

## Intervall hinzufügen

 Ein neues Intervall mit einem Standardnamen wird angelegt. Doppelklicken Sie auf den markierten Namen um ihn zu verändern.

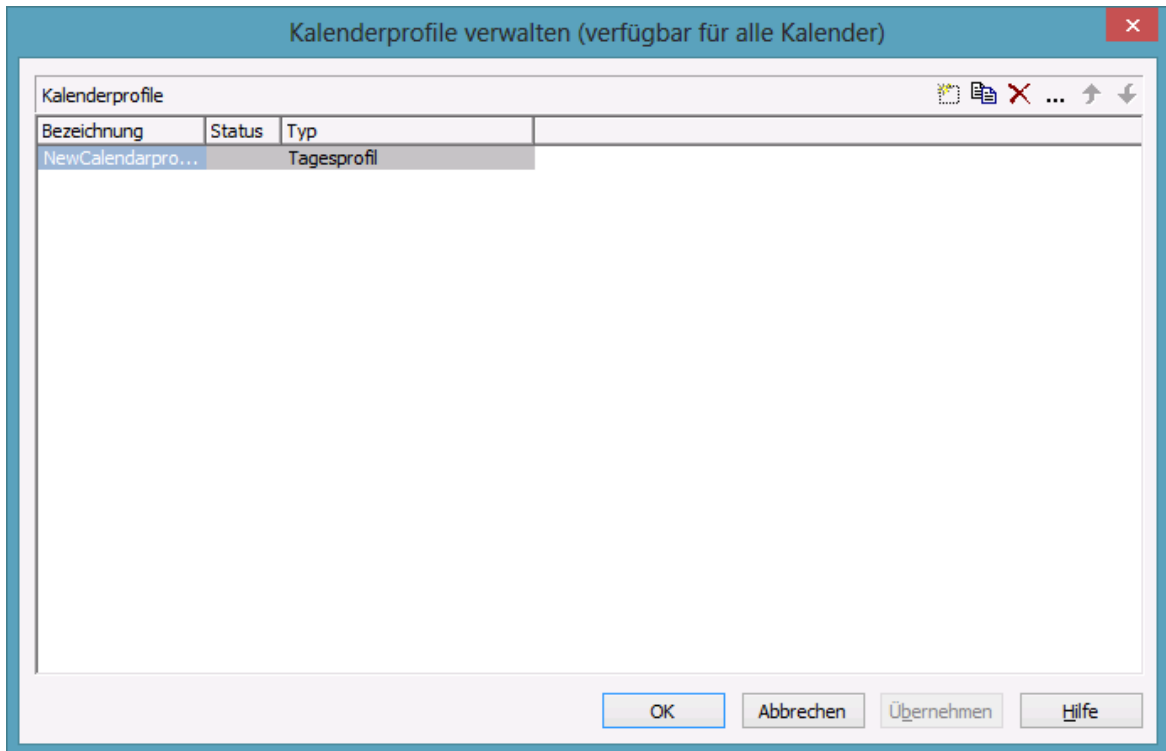
## Intervall kopieren

 Es wird eine Kopie des markierten Intervalls angelegt.

## Intervall löschen

 Das markierte Intervall wird gelöscht.

## 4.31 Dialogfeld "Kalenderprofile verwalten"



In diesem Dialog können Sie Kalenderprofile einrichten und verändern.


### Bezeichnung

In dieser Spalte werden die Bezeichnungen aller Kalenderprofile aufgeführt. Die Bezeichnungen sind editierbar.

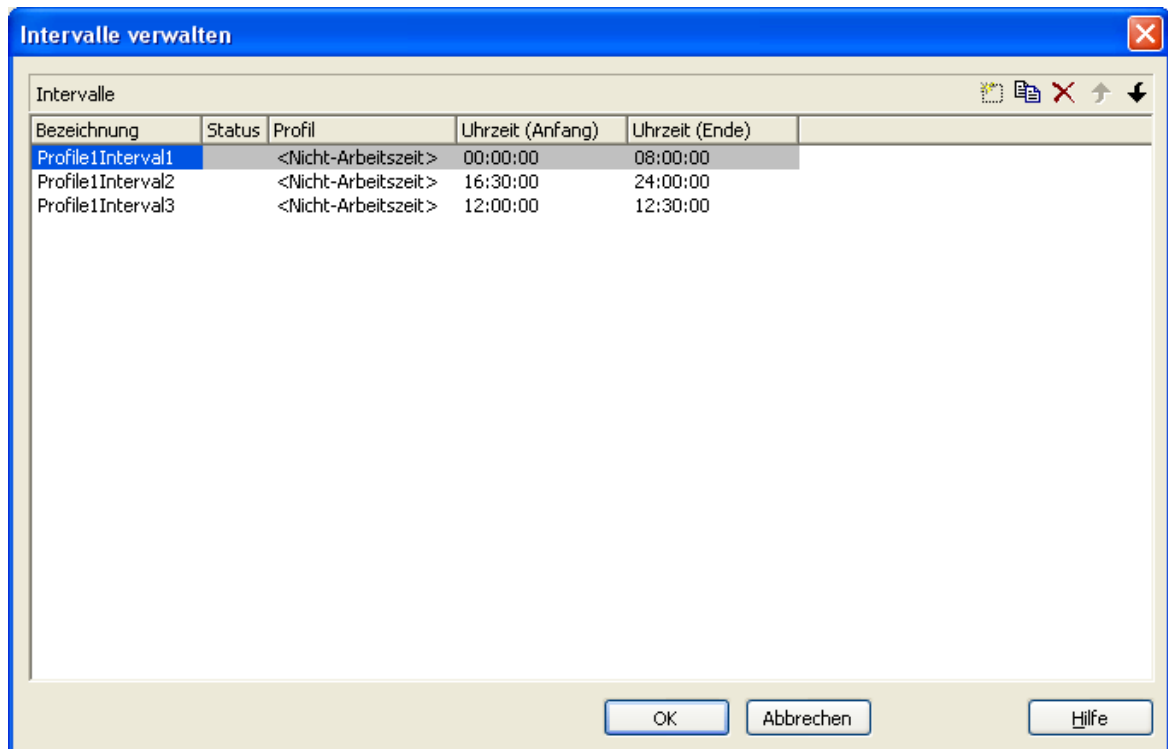
### Status

In dieser Spalte wird jedes Kalenderprofil gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt (🌟) und/oder geändert (⚠️) worden ist.

### Typ

Wählen Sie hier durch Klick auf  den Typ des Kalenderprofils aus. Zur Verfügung stehen die Profiltypen <Tagesprofil>, <Wochenprofil>, <Jahresprofil> und <Variables Profil>.

## 4.32 Dialogfeld "Intervalle verwalten" für Tagesprofil





Zu diesem Dialogfeld gelangen Sie, wenn Sie auf der Eigenschaftenseite "Objekte" das Dialogfeld "Kalenderprofile verwalten" aktivieren und dann die "Bearbeiten"-Schaltfläche eines Kalenderprofils drücken. Die unterschiedlichen Profiltypen bieten entsprechende Optionen; dieses Dialogfeld dient dazu, für ein Tagesprofil Intervalle einzurichten und zu verändern.

### Bezeichnung

In dieser Spalte werden die Bezeichnungen aller Intervalle aufgeführt. Die Bezeichnungen sind editierbar.

### Status

In dieser Spalte wird jedes Intervall gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt (  ) und/oder geändert (  ) worden ist.

### Profil

Weisen Sie hier dem Intervall ein Kalenderprofil zu. Über  öffnen Sie eine Liste mit Profilen, aus der Sie das gewünschte Profil auswählen können.

## Uhrzeit Anfang/Uhrzeit Ende

Legen Sie hier für das jeweilige Intervall mithilfe der Pfeiltasten die Start- bzw. Endezeit fest.

## Intervall hinzufügen



Ein neues Intervall mit einem Standardnamen wird angelegt. Doppelklicken Sie auf den markierten Namen um ihn zu verändern.

## Intervall kopieren



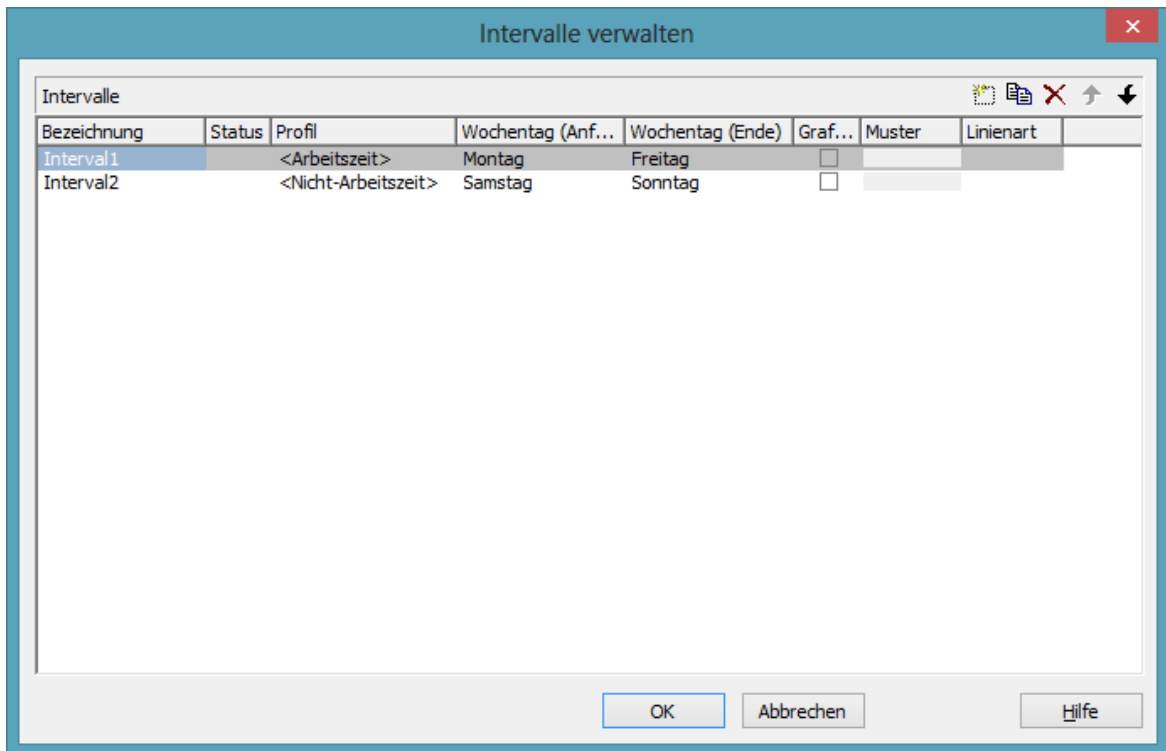
Es wird eine Kopie des markierten Intervalls angelegt.

## Intervall löschen



Das markierte Intervall wird gelöscht.

## 4.33 Dialogfeld "Intervalle verwalten" für Wochenprofil



Zu diesem Dialogfeld gelangen Sie, wenn Sie auf der Eigenschaftenseite "Objekte" das Dialogfeld "Kalenderprofile verwalten" aktivieren und dann die "Bearbeiten"-Schaltfläche eines Kalenderprofils drücken. Die unterschiedlichen Profiltypen bieten entsprechende Optionen; dieses Dialogfeld dient dazu, für ein Wochenprofil Intervalle einzurichten und zu verändern.

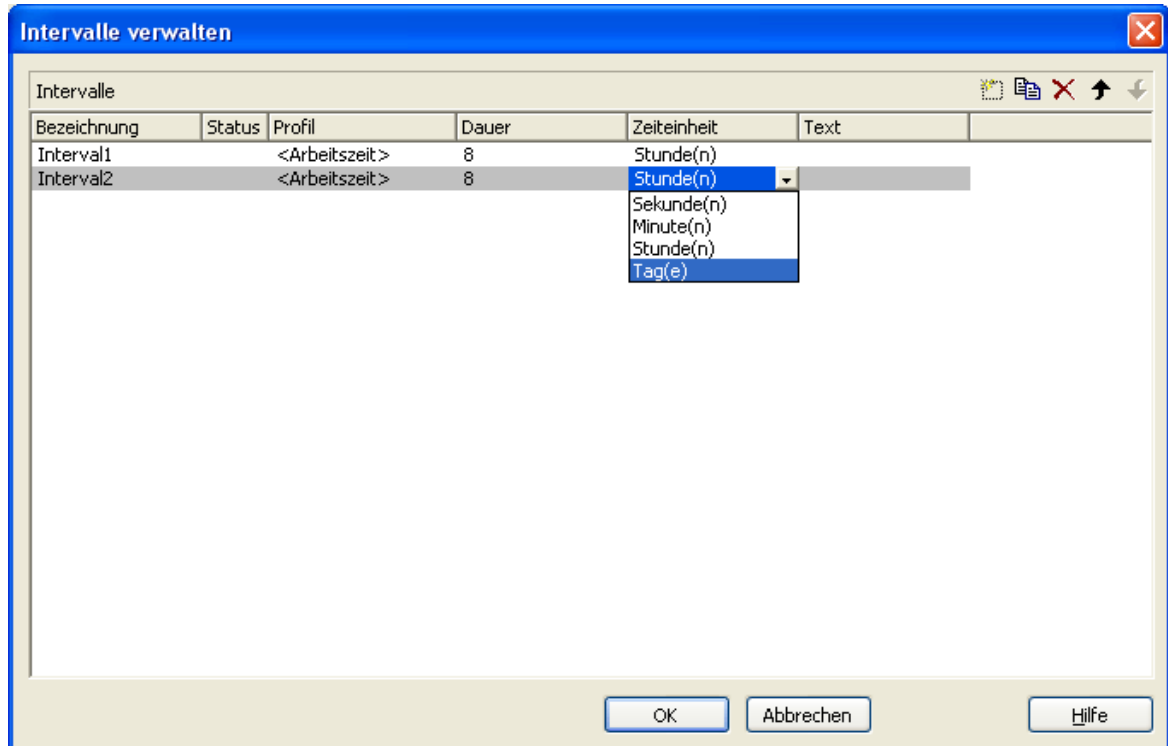
### Wochentag Anfang/Wochentag Ende

Durch Klick auf  können Sie den ersten/letzten Wochentag des Intervalls festlegen.

### Wochentag Anfang/Wochentag Ende

Durch Klick auf  können Sie den ersten/letzten Wochentag des Intervalls festlegen.

## 4.34 Dialogfeld "Intervalle verwalten" für Variables Profil



Zu diesem Dialogfeld gelangen Sie, wenn Sie auf der Eigenschaftenseite "Objekte" das Dialogfeld "Kalenderprofile verwalten" aktivieren und dann die "Bearbeiten"-Schaltfläche eines Kalenderprofils drücken. Die unterschiedlichen Profiltypen bieten entsprechende Optionen; dieses Dialogfeld dient dazu, für ein Variables Profil Intervalle einzurichten und zu verändern.

### Dauer

Legen Sie hier die Dauer des Intervalls fest. Diese Option kann auch über die Eigenschaft **VcInterval.Duration** gesetzt werden.

### Zeiteinheit

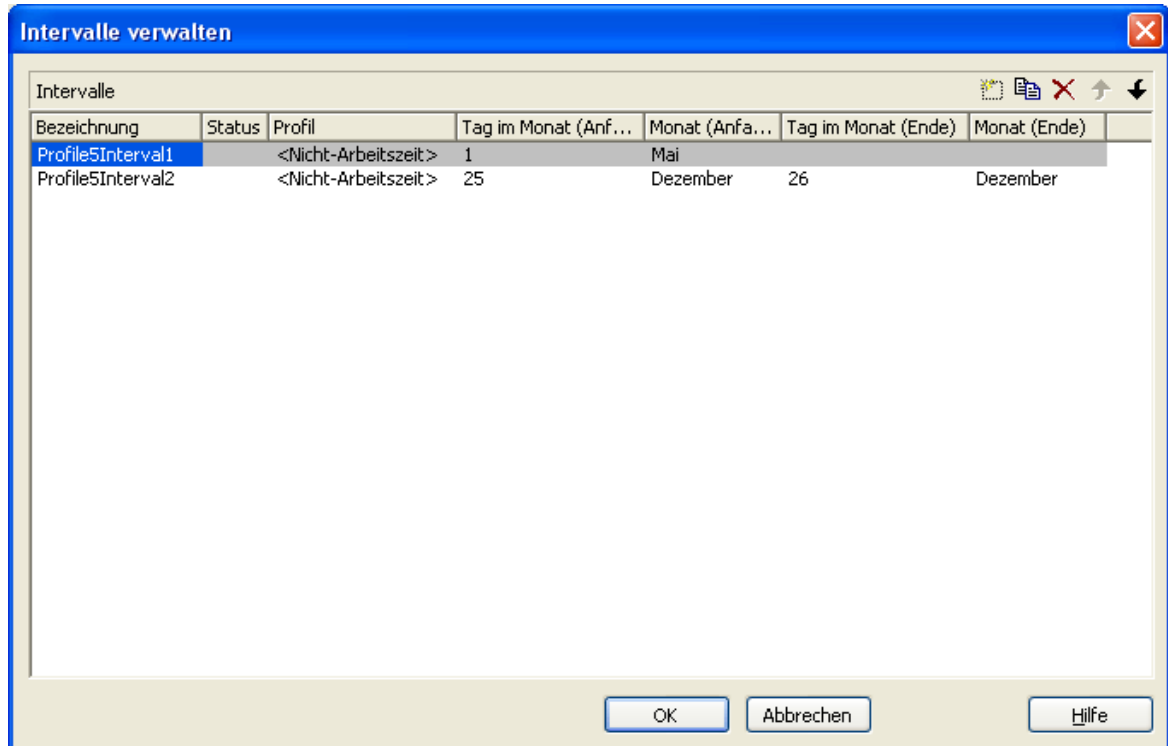
Legen Sie hier die Zeiteinheit des Intervalls fest. Diese Option kann auch über die Eigenschaft **VcInterval.TimeUnit** gesetzt werden.



## **Text**

Legen Sie hier einen Text fest, der im Zeitstreifen des Intervalls erscheinen soll. Diese Option kann auch über die Eigenschaft **VcInterval.Text** gesetzt werden.

## 4.35 Dialogfeld "Intervalle verwalten" für Jahresprofil



Zu diesem Dialogfeld gelangen Sie, wenn Sie auf der Eigenschaftenseite "Objekte" das Dialogfeld "Kalenderprofile verwalten" aktivieren und dann die "Bearbeiten"-Schaltfläche eines Kalenderprofils drücken. Die unterschiedlichen Profiltypen bieten entsprechende Optionen; dieses Dialogfeld dient dazu, für ein Jahresprofil Intervalle einzurichten und zu verändern.


### Tag im Monat (Anfang)/Tag im Monat (Ende)

Durch Klick auf  können Sie den Tag des ersten Monats des Intervalls festlegen. Diese Option kann auch über die Eigenschaft **VcInterval.DayInStart/EndMonth** gesetzt werden.

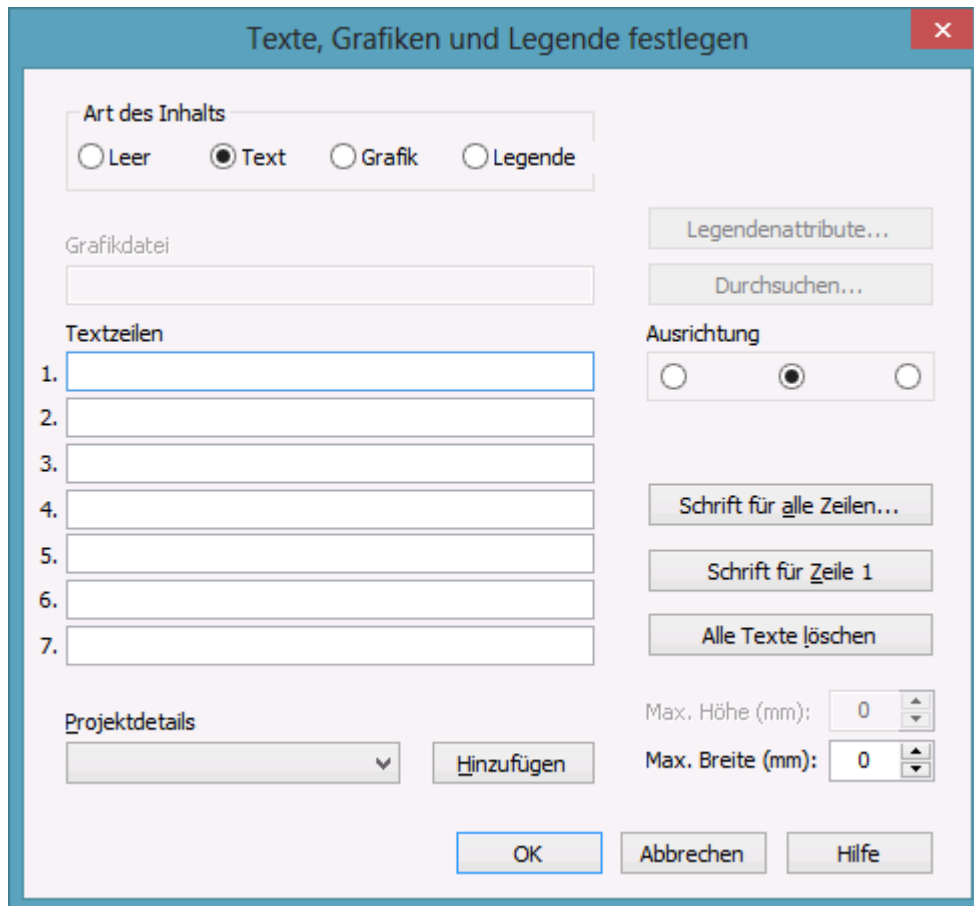
### Monat (Anfang)/Monat (Ende)

Durch Klick auf  können Sie den Start-/ Endmonat des Intervalls festlegen. Diese Option kann auch über die Eigenschaft **VcInterval.Start/EndMonth** gesetzt werden.

### **Monat (Anfang)/Monat (Ende)**

Durch Klick auf  können Sie den Start-/ Endmonat des Intervalls festlegen. Diese Option kann auch über die Eigenschaft **VcInterval.Start/EndMonth** gesetzt werden.

## 4.36 Dialogfeld "Texte, Grafiken und Legende festlegen"



Sie erreichen dieses Dialogfeld, indem Sie auf der Eigenschaftenseite **Außenbereich** auf eine der neun Schaltflächen ober- bzw. unterhalb der Grafik klicken.

### Art des Inhalts

Wählen Sie hier die Art des Inhalts, der in dem zuvor gewählten Bereich der Darstellung ausgegeben werden soll:

- **Leer:** Der gewählte Diagrammbereich bleibt leer.
- **Text:** In dem gewählten Diagrammbereich wird der Text der sechs Textzeilen dargestellt.
- **Grafik:** Sie können in dem gewählten Bereich eine Grafik (z.B. Ihr Firmenlogo) platzieren. Grafiken werden immer mittig ausgerichtet.

- **Legende:** Eine Legende wird in dem gewählten Diagrammbereich ausgegeben. Sie dokumentiert die Layer, die in der aktuellen Darstellung auftreten.

Die jeweils nicht benötigten Bereiche des Dialogs werden abhängig von Ihrer Auswahl deaktiviert. Dabei bleiben alle Angaben erhalten.

## Legendenattribute

*Nur aktiv, wenn das Kontrollkästchen **Legende** angeklickt wurde.* Sie gelangen in den gleichnamigen Dialog, der für die Legende weitere Gestaltungsmöglichkeiten bietet.

## Grafikdatei

*Nur aktiv, wenn das Kontrollkästchen **Grafik** angeklickt wurde.* Tragen Sie hier den Namen der Grafikdatei ein. Falls sich die gewünschte Grafikdatei nicht im Installationsverzeichnis von VARCHART befindet, müssen Sie das Laufwerk und den Pfad auch angeben.

## Durchsuchen

*Nur aktiv, wenn das Kontrollkästchen **Grafik** angeklickt wurde.* Wenn Sie auf diese Schaltfläche klicken, erscheint der Windows-Dialog **Grafikdatei auswählen**, mit dessen Hilfe Sie das Laufwerk, das Verzeichnis und den Dateinamen der Grafik festlegen können.

## Textzeilen

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Vereinbaren Sie den Text (max. 6 Zeilen), mit denen der gewählte Diagrammbereich beschriftet werden soll. Sie können einen beliebigen Text eintragen, aber auch Platzhalter (z.B. &[System-Datum]) für Projektdetails einsetzen. Sind alle sechs Textzeilen leer, wird dieser Bereich nicht dargestellt.

## Projektdetails

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Hier können Sie dem Diagramm verschiedene Informationen (Anzahl der Seiten, Seitennummer, Systemdatum) hinzufügen, indem Sie aus der Kombobox den gewünschten Platzhalter auswählen und auf die Schaltfläche **Hinzufügen** klicken. Die Platzhalter werden in der Druckvorschau oder beim Ausdruck

durch die entsprechenden Daten ersetzt und stets auf dem aktuellen Stand gehalten.

## Hinzufügen

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Nachdem Sie aus der Liste ein Projektdetail ausgewählt haben, bestätigen Sie Ihre Wahl mit der Schaltfläche **Hinzufügen**. Die Projektdetails werden in die Zeile geschrieben, in der sich der Cursor gerade befindet.

## Textausrichtung

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Über die Kontrollkästchen können Sie die Textzeilen linksbündig, zentriert oder rechtsbündig ausrichten.

## Schrift für alle Zeilen

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**. Hier können Sie die Schriftart, Schriftgröße usw. für alle sechs Zeilen festlegen. Beim Ausführen dieser Aktion werden die Einstellungen für die Schrift der einzelnen Zeilen überschrieben.

## Schrift für Zeile 1...6

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**. Hier können Sie die Schriftart, Schriftgröße usw. für die Zeile festlegen, in der sich der Cursor befindet.

## Alle Texte löschen

*Nur aktiv, wenn das Kontrollkästchen **Text** angeklickt wurde.* Wenn Sie diese Schaltfläche anklicken, werden die Texte aller sechs Textzeilen gelöscht.

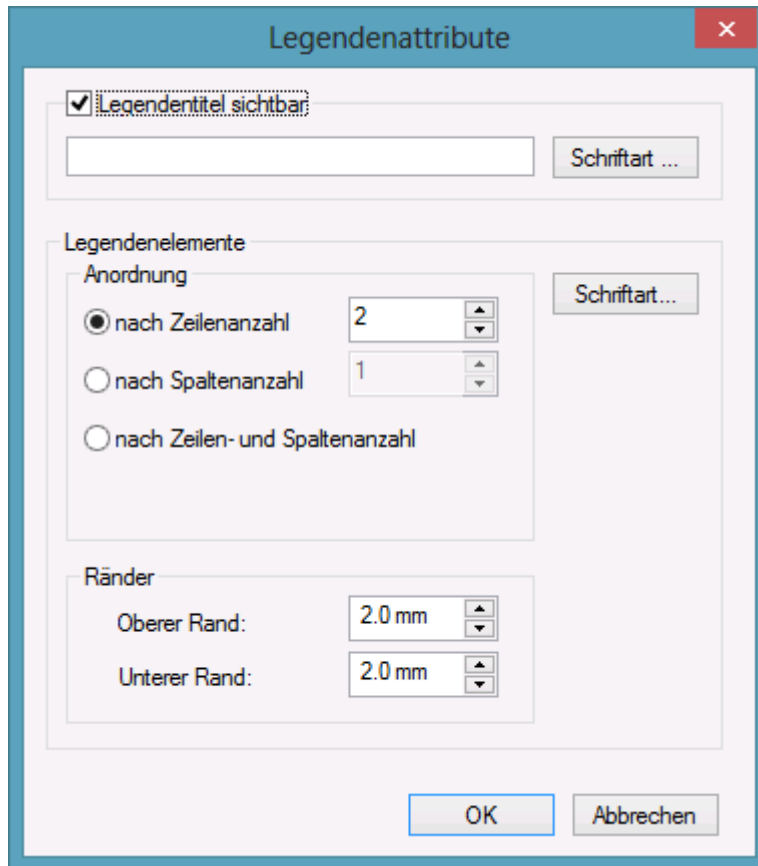
## Max. Höhe (mm)

*Nur aktiv, wenn das Kontrollkästchen **Grafik** angeklickt wurde.* Falls mehrere Felder für Text, Grafik oder Legende vereinbart worden sind, können Sie hier die maximale Höhe des aktuellen Feldes festlegen. So können Sie verhindern, dass Feldinhalte abgeschnitten werden.

### **Max. Breite (mm)**

*Nur aktiv, wenn das Kontrollkästchen **Text** oder **Grafik** angeklickt wurde.*  
Falls mehrere Felder für Text, Grafik oder Legende vereinbart worden sind, können Sie hier die max. Breite des aktuellen Feldes festlegen. So können Sie verhindern, dass Feldinhalte abgeschnitten werden.

## 4.37 Dialogfeld "Legendenattribute"



Sie erreichen dieses Dialogfeld zur Laufzeit über das Kontextmenü der Legende oder zur Designzeit über den Dialog **Texte, Grafiken und Legende festlegen** durch Klick auf die entsprechende Schaltfläche. Diese wird erst wählbar, nachdem Sie bei **Art des Inhalts Legende** ausgewählt haben.

### Legendentitel sichtbar

Legen Sie hier fest, ob ein Legendentitel angezeigt werden soll und geben Sie einen Text ein. Durch Klick auf **Schriftart** öffnen Sie den gleichnamigen Windows-Dialog, in dem Sie die Schriftattribute für den Legendentitel festlegen können.

### Anordnung

- nach Zeilenanzahl: Geben Sie hier an, ob und mit wie vielen Zeilen die Legende dargestellt werden soll.
- nach Spaltenanzahl: Geben Sie hier an, ob und mit wie vielen Spalten die Legende dargestellt werden soll.



- Nach Zeilen- und Spaltenanzahl: Die Legende wird in Spalten und Zeilen dargestellt. Ist die hier eingegebene Zahl niedriger als die vorhandenen Layer, so werden die überzähligen Layer nicht dargestellt.

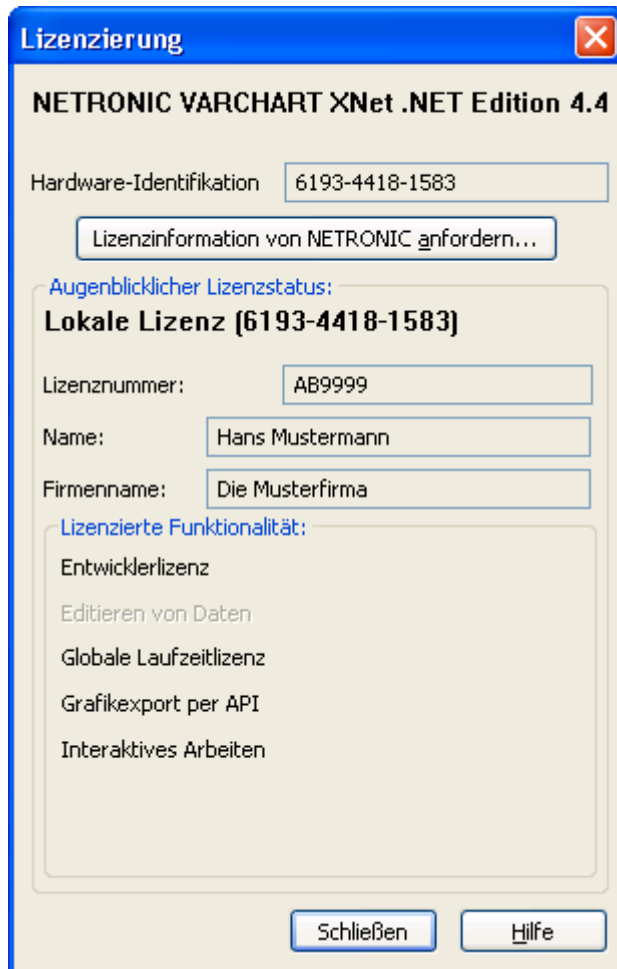
## Ränder

- oberer Rand: Geben Sie ein Maß für den oberen Rand des Legendenelements an.
- unterer Rand: Geben Sie ein Maß für den unteren Rand des Legendenelements an.

## Schrift

Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**, in dem Sie die Schriftattribute für die Legende festlegen können.

## 4.38 Dialogfeld "Lizenzierung"



Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Allgemeines**.

Vor der Lizenzierung ist das Programm nur als Demoversion lizenziert. Folgende Einschränkungen gelten gegenüber der Vollversion: Die Nutzungsdauer zum Testen des Produkts ist auf 30 Tage begrenzt. Nach Ablauf dieses Zeitraums erscheint das Wort "Demo" im Diagramm.

### Hardware-Identifikation

*(nicht editierbar)* Die Nummer, die in diesem Feld angezeigt wird, wird aus der Hardware-Konfiguration Ihres Rechners berechnet. Sie wird von NETRONIC Software GmbH für die Lizenzierung benötigt.

Bei Änderungen an Ihrer Hardware ist eine neue Lizenzierung erforderlich. Der Kundendienst von NETRONIC Software GmbH hilft Ihnen dann gern weiter.

## Anfordern

Um die Lizenzierung vorzunehmen, klicken Sie auf diese Schaltfläche. Dann erscheint der Dialog **Lizenzinformationen anfordern**.

## Lizenznummer/Name/Firmenname

*(nicht editierbar)* Hier werden Ihre Lizenznummer, Ihr Name und der Name Ihrer Firma angezeigt.

## Lizenzierte Funktionalität

Hier wird angezeigt, welche Module für Sie freigegeben wurden. Wenn Sie die Lizenzierung vorgenommen haben, sind die freigegebenen Module aktiviert.

- **Entwicklerlizenz**
- **Globale Laufzeitlizenz** (Das VARCHART ActiveX läuft im Runtime-Modus auf jedem anderen Rechner.)
- **Einzelplatz-Laufzeitlizenzen** (Das VARCHART ActiveX muss auf jedem Rechner, auf dem es im Runtime-Modus laufen soll, einzeln lizenziert werden.)
- **Grafikexport per API**
- **Interaktives Arbeiten**

## Schließen

Sie verlassen den Dialog.

## 4.39 Dialogfeld "Lizenzinformationen anfordern"

**Lizenzinformationen anfordern**

**NETRONIC VARCHART XNet .NET Edition 4.4**

Hardware-Identifikation: 6193-4418-1583

**Erster Schritt: Geben Sie Ihre Benutzerdaten ein:**

Lizenznummer:

Name:

Firmenname:

**Zweiter Schritt: Fordern Sie Ihre Lizenzinformationen an:**

Wenn Sie keine E-Mail direkt von Ihrem Computer versenden können, kontaktieren Sie NETRONIC Software GmbH unter Angabe der obigen vier Einträge:

E-Mail: license@netronic.com  
Telefon: +49/2408/141-0  
Fax: +49/2408/141-33

**Dritter Schritt: Wenn Sie die Lizenzinformationsdatei erhalten haben, kopieren Sie diese in das gleiche Verzeichnis wie die DLL-Datei.**

Geben Sie Ihre Lizenznummer, Ihren Namen und den Namen Ihrer Firma an und klicken Sie auf **E-Mail an NETRONIC senden**. Damit wird automatisch eine E-Mail generiert, die Sie nur noch absenden müssen. Sobald wir Ihre E-Mail erhalten haben, werden wir unverzüglich eine Datei mit Ihren Lizenzinformationen (vcnet.lic) generieren und sie Ihnen zusenden. Bitte kopieren Sie dann diese Datei in das Verzeichnis, in dem die Datei vcnet.ocx steht.

Wenn Sie die neue Lizenzierung vorgenommen haben, müssen Sie diese in jedem Ihrer Projekte aktivieren. Öffnen Sie dazu in jedem Ihrer Projekte eine beliebige Eigenschaftenseite, nehmen Sie dort eine beliebige Änderung vor und speichern Sie diese. Nun ist die neue Lizenzierung aktiviert.



---

---

# 5 Benutzerschnittstelle

---

## 5.1 Übersicht

Die folgende Liste gibt einen Überblick über die Interaktionsmöglichkeiten für Anwender.

- Navigation im Diagramm
- Zoomen
- Knoten und Verbindungen erzeugen
- Knoten und Verbindungen markieren, löschen und verschieben
- Knoten oder Verbindungen bearbeiten
- Legende bearbeiten
- Seite einrichten
- Druckvorschau verwenden

### **Kontextmenüs (rechte Maustaste):**

- für das Diagramm
- für Knoten
- für Verbindungen
- für die Legende

Bei allen Interaktionen wird ein Ereignis ausgelöst, sodass Sie im Programm darüber informiert werden und ggf. darauf reagieren können.

---

## 5.2 Navigation im Diagramm

Sie können mit Hilfe der Pfeil-Tasten mit der Markierung von einem Knoten zum anderen springen.

Sie können bei gedrückter Strg-Taste mit Hilfe der Pfeil-Tasten im Diagramm scrollen.

Weitere Tastenkombinationen für die Navigation im Diagramm:

- **Strg + Pos1:** an den linken oberen Diagrammrand scrollen
- **Strg + Ende:** an den rechten unteren Diagrammrand scrollen
- **Strg + Bild rauf/runter:** an den oberen/ unteren Diagrammrand scrollen
- **Strg + Num +** (Nummernblock): Zoom in
- **Strg + Num -** (Nummernblock): Zoom out
- **Strg + Num \*** (Nummernblock): zum nächsten Knoten scrollen
- **Strg + Num /** (Nummernblock): Komplettansicht

Mit Hilfe von **Strg + C**, **Strg + X** bzw. **Strg + V** können Sie markierte Knoten kopieren, ausschneiden bzw. einfügen. Mit Hilfe der **Entf**-Taste können Sie markierte Knoten löschen.

---

## 5.3 Zoomen

Mithilfe der folgenden Tastenkombinationen lässt sich die Darstellung vergrößern bzw. verkleinern:

- **Strg + -** (Nummernblock): Verkleinern
- **Strg + +** (Nummernblock): Vergrößern

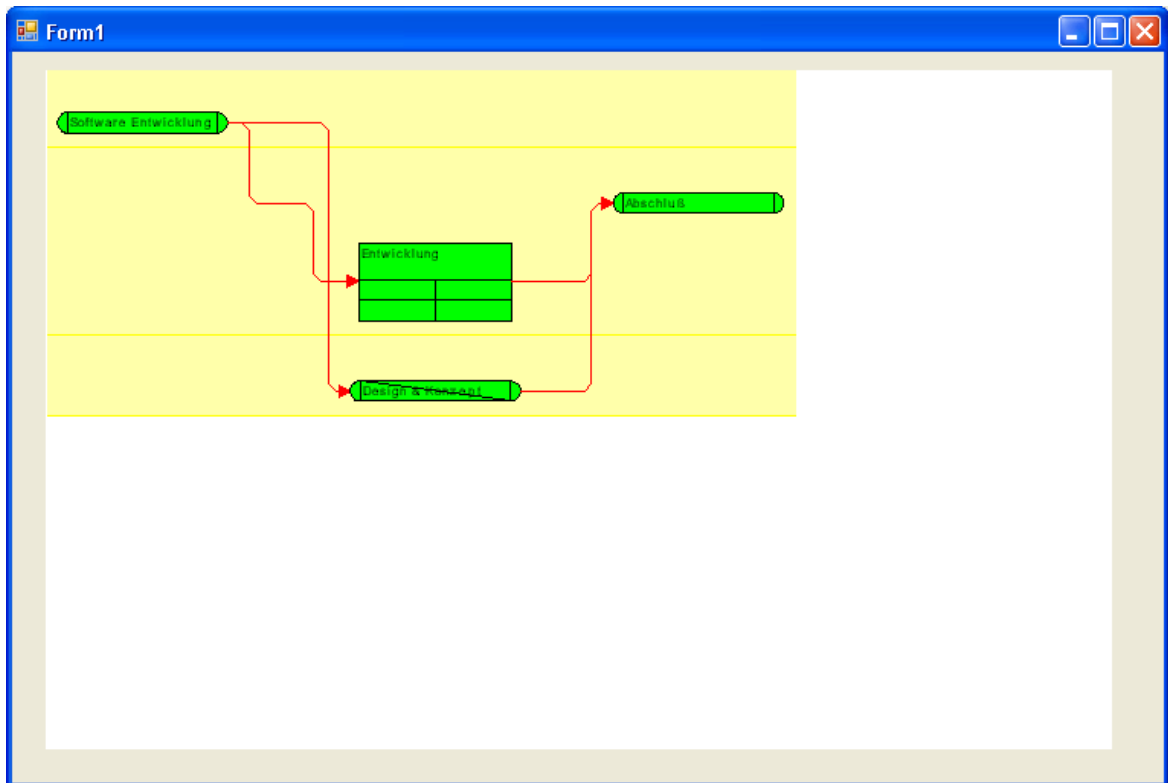
Auch die Maus kann zum Zoomen genutzt werden:

- Drehen Sie das Mousrad während Sie die Strg-Taste gedrückt halten. Dazu muss das Zoomen per Mousrad zugelassen sein. Dies geschieht entweder über die Option **Zoomen per Mousrad zulassen** auf der Eigenschaftenseite **Allgemeines** oder über die API-Eigenschaft **VcNet1.-ZoomingPerMouseWheelAllowed**. (Diese Eigenschaft ist standardmäßig deaktiviert.)
- Sie können einen Ausschnitt Ihres Diagramms bildschirmfüllend darstellen lassen, indem Sie mit gedrückter linker Maustaste ein Rechteck um den zu vergrößernden Ausschnitt aufziehen und dann (bei noch gedrückter linker Maustaste) die rechte Maustaste drücken. Mit Hilfe der Bildlaufleiste können Sie dann das Fenster wie eine Lupe über der Darstellung verschieben und so auch die anderen Bereiche der Darstellung in derselben Vergrößerung betrachten.

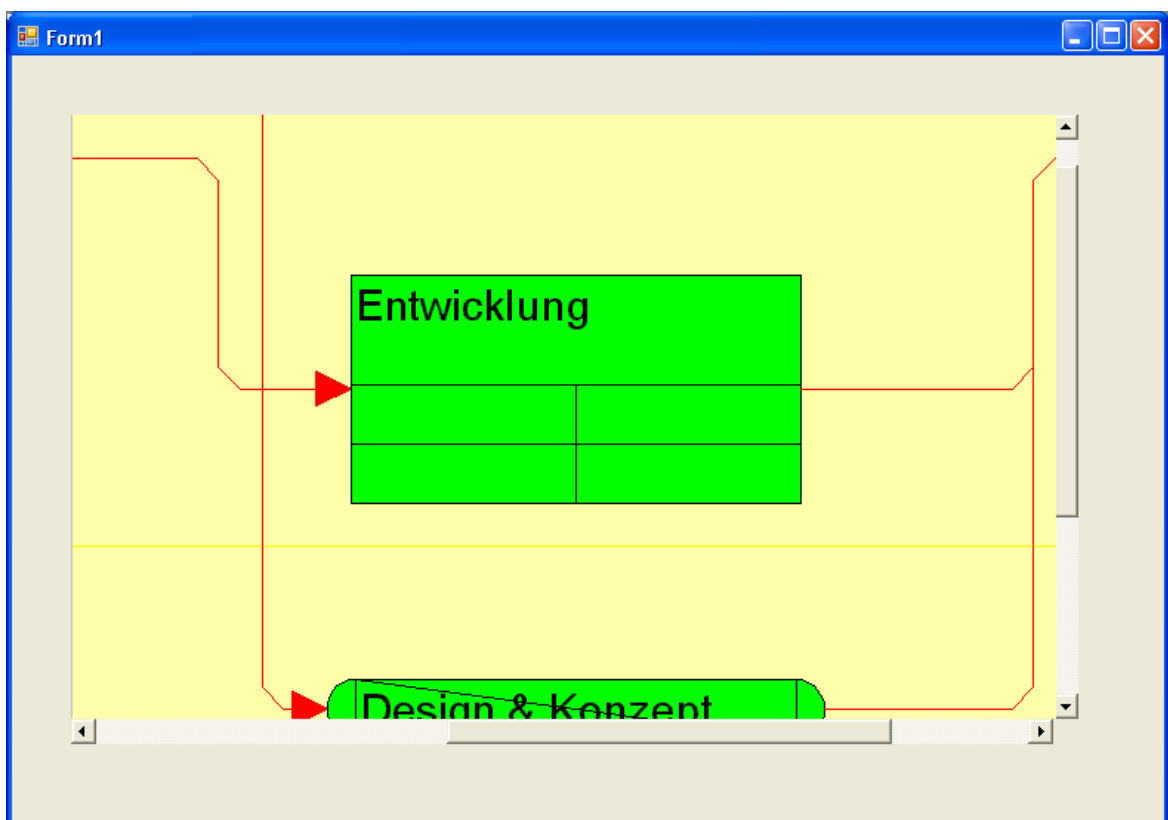
Mit der API-Methode **ShowAlwaysCompleteView** können Sie die Darstellung so einstellen, dass stets das komplette Diagramm angezeigt wird. Der Zoomfaktor passt sich bei jeder Änderung des Diagramms automatisch an. Der maximale Zoomfaktor von 100% wird nicht überschritten, die Knoten werden also höchstens in Originalgröße dargestellt.

Weitere Information zu den Zoommöglichkeiten für den Druck finden Sie in Kapitel 5.21 "Seite einrichten".





*vor dem Zoomen*

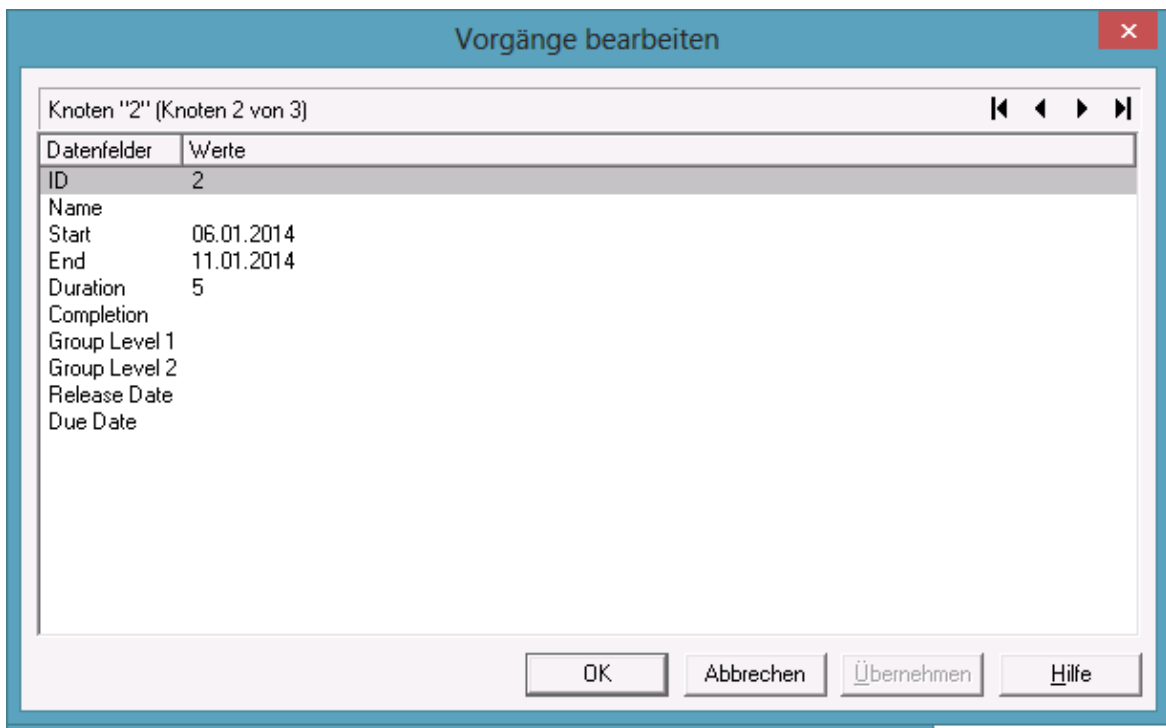


*nach dem Zoomen*

## 5.4 Knotendaten bearbeiten

Alle Daten eines Knotens können Sie im Dialogfeld **Vorgänge bearbeiten** bearbeiten. Sie erreichen das Dialogfeld durch einen Doppelklick auf einen Knoten oder über den Befehl <Bearbeiten> seines Kontextmenüs.

Um die Daten mehrerer Knoten zu bearbeiten, markieren Sie die gewünschten Knoten und wählen Sie aus dem Kontextmenü eines der markierten Knoten ebenfalls den Befehl **Bearbeiten** um das Dialogfeld **Vorgänge bearbeiten** zu öffnen. Jetzt können Sie nacheinander die Daten aller markierten Knoten bearbeiten



Durch einen Doppelklick auf einen Knoten wird das Ereignis **VcNodeLeftDoubleClicking** ausgelöst.

Wenn ein Knoten interaktiv verändert worden ist (hier durch die Veränderung eines Wertes im Dialog **Vorgänge bearbeiten**), wird das Ereignis **VcNodeModifying** ausgelöst. Über den Parameter **modificationType** erhalten Sie nähere Informationen über die Art der Veränderung. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Veränderung unterdrückt.

### Der Dialog "Vorgänge bearbeiten"

Oberhalb der Tabelle wird der Name des aktuellen Knotens angezeigt und ggf. um den wievielten Knoten der markierten Knoten es sich handelt.

Sie können hier alle Werte dieses Knotens bearbeiten und ggf. mit Hilfe der **Übernehmen**-Schaltfläche übernehmen. Mit Hilfe der Pfeil-Schaltflächen können Sie zwischen den Knoten navigieren.

## Datenfelder

In dieser Spalte werden alle Datenfelder angezeigt, durch die der markierte Knoten beschrieben wird und die **nicht** im Dialog **Datentabellen verwalten** als **versteckt** definiert wurden. Welche Datenfelder verfügbar sind, hängt von Ihrer Datendefinition ab.

## Werte

Sie können in dieser Spalte alle Werte des markierten Knotens direkt bearbeiten, sofern sie im Dialog **Datentabellen verwalten** als **editierbar** definiert worden sind.

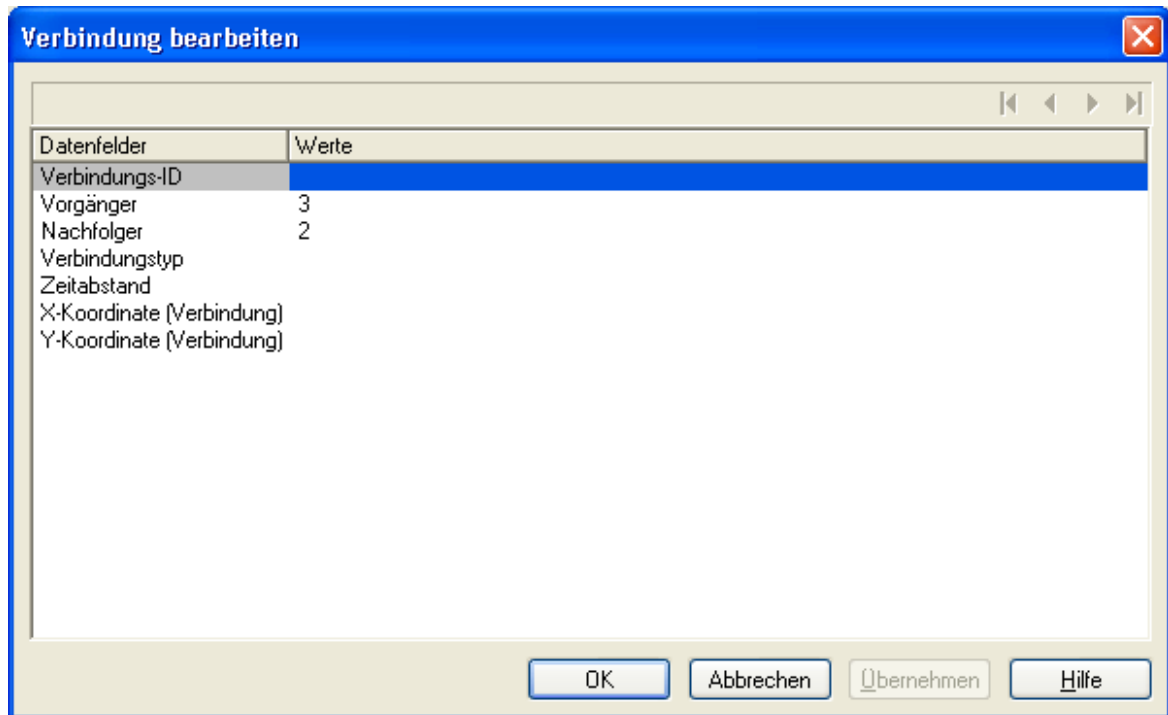
Wenn Sie hier ein Datenfeld vom Typ **Datum/Zeit** bearbeiten, erscheint ein Datumsdialog, in dem Sie das gewünschte Datum anklicken können. Fehler durch die Eingabe eines falschen Datumsformats werden so vermieden.



Das **Datumsausgabeformat** wird auf der Eigenschaftenseite **Allgemeines** festgelegt.

Wenn Sie ein Datenfeld vom Typ **Integer** bearbeiten, erscheint ein Spincontrol, mit dem Sie den gewünschten Wert einstellen können.

## 5.5 Verbindungen bearbeiten



Dieses Dialogfeld erreichen Sie durch einen Doppelklick auf eine Verbindung (Ereignis **VcLinksLeftDoubleClicking**). Sie können hier die Daten einer markierten Verbindung bearbeiten.

Oberhalb der Tabelle wird die Identifikation (ID) der markierten Verbindung angezeigt.

Sie können hier alle Werte der markierten Verbindung bearbeiten und ggf. mit Hilfe der **Übernehmen**-Schaltfläche übernehmen.

### Datenfelder

In dieser Spalte werden alle Datenfelder angezeigt, durch die die markierte Verbindung beschrieben wird und die nicht in der Datendefinition als versteckt definiert sind. Welche Datenfelder verfügbar sind, hängt von Ihrer Datendefinition ab.

### Werte

Sie können in dieser Spalte alle Werte der markierten Verbindung direkt bearbeiten, sofern sie nicht auf der Eigenschaftenseite **Datendefinition** als **nur lesbar** definiert worden sind.

---

## 5.6 Navigation mit Hilfe der Tastatur

Sie können mit Hilfe der Pfeil-Tasten mit der Markierung von einem Knoten zum anderen springen.

Sie können bei gedrückter Strg-Taste mit Hilfe der Pfeil-Tasten im Diagramm scrollen.

Weitere Tastenkombinationen für die Navigation im Diagramm:

- **Strg + Pos1:** an den linken oberen Diagrammrand scrollen
- **Strg + Ende:** an den rechten unteren Diagrammrand scrollen
- **Strg + Bild rauf/runter:** an den oberen/ unteren Diagrammrand scrollen
- **Strg + Num +** (Nummernblock): Zoom in
- **Strg + Num -** (Nummernblock): Zoom out
- **Strg + Num \*** (Nummernblock): zum nächsten Knoten scrollen
- **Strg + Num /** (Nummernblock): Komplettansicht

Mit Hilfe von **Strg + C**, **Strg + X** bzw. **Strg + V** können Sie markierte Knoten kopieren, ausschneiden bzw. einfügen. Mit Hilfe der **Entf**-Taste können Sie markierte Knoten löschen.

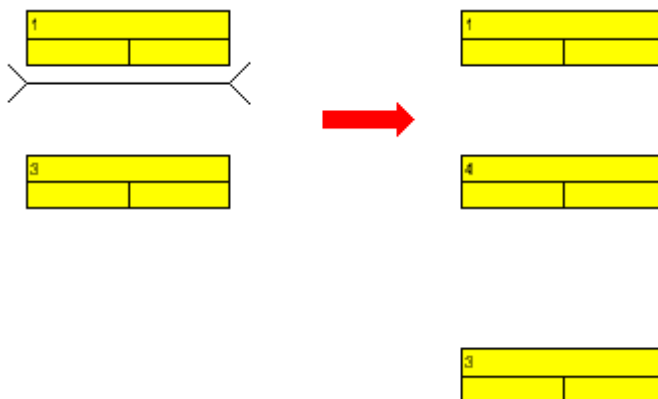
## 5.7 Knoten und Verbindungen erzeugen

VARCHART XNet besitzt zur Laufzeit zwei grundlegende Modi: den Markiermodus und den Erzeugemodus. Knoten und Verbindungen können Sie nur im Erzeugemodus anlegen. Um in den Erzeugemodus zu wechseln, klicken Sie mit der rechten Maustaste in den freien Diagrammbereich und wählen Sie im Kontextmenü den Befehl **Erzeugemodus**.

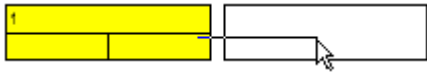


Im Erzeugemodus wird der Mauszeiger im leeren Diagramm zu einem Knotenphantom in der Form eines Rechtecks. Nun können Sie Knoten erzeugen, indem Sie mit der linken Maustaste in den Diagrammbereich klicken.

Wenn Sie die Maus zwischen zwei dicht übereinander oder nebeneinander stehende Vorgänge führen, um dazwischen einen neuen Vorgang anzulegen, verändert das Phantom seine Form und wird zu einer waagerechten Linie mit zwei nach innen gerichteten Pfeilspitzen ("Knochen"). Klicken Sie nun mit der linken Maustaste, so wird der neue Vorgang zwischen den beiden vorhandenen Vorgängen eingefügt.



Verbindungen erzeugen Sie, indem Sie die Maus mit gedrückter linker Maustaste von einem Knoten zu einem anderen Knoten ziehen. Dabei wird der Mauszeiger zu einem Pfeil-Symbol mit einem Knoten-Phantom.



Die Verbindung wird erzeugt, sobald Sie die Maustaste loslassen. Lassen Sie die Maustaste im leeren Diagrammbereich los, wird dort ein neuer Knoten zusammen mit einer Verbindung vom Ausgangsknoten erzeugt. Auf diese Weise können Sie Knoten und Verbindungen gemeinsam erzeugen.

---

## 5.8 Knoten und Verbindungen markieren, löschen oder verschieben

Einen Knoten oder eine Verbindung markieren Sie, indem Sie den Knoten bzw. die Verbindung mit der linken Maustaste anklicken. Mehrere Knoten werden markiert, indem Sie diese bei gedrückter Umschalt- oder Strg-Taste mit der linken Maustaste anklicken. (Bei gedrückter Umschalt-Taste werden dabei zusätzlich die Verbindungen markiert.) Sie können nun beispielsweise alle markierten Knoten auf einmal mit der **Entfernen**-Taste oder mit dem Befehl **Löschen** aus dem Kontextmenü für Knoten löschen.

Sie können auch mehrere Knoten und die zugehörigen Verbindungen auf einmal markieren, indem Sie mit der Maus ein Rechteck darum aufziehen.

Wenn Sie im Markiermodus den Mauszeiger auf einem Knoten positionieren und die linke Maustaste drücken, können Sie ihn beliebig verschieben, solange Sie die linke Maustaste gedrückt halten. Dabei werden die zugehörigen Verbindungen automatisch angepasst.

Wenn Sie im Markiermodus den Mauszeiger auf einer Verbindung positionieren und die linke Maustaste drücken, wird der Mauszeiger zu einem kleinen Quadrat mit vier Pfeilen. Sie können die ausgewählte Verbindung beliebig verschieben, solange Sie die linke Maustaste gedrückt halten.



## 5.9 Seite einrichten

Alle Einstellungen zum Seitenlayout können Sie im Dialog "Seite einrichten" vornehmen. Sie gelangen in diesen Dialog entweder über den entsprechenden Befehl im Diagramm-Kontextmenü über aus der Druckvorschau durch Klick auf die gleichnamige Schaltfläche.

### Modus

Durch Auswahl einer Skalierungsart aus der Drop-Down-Liste und der entsprechenden Werte bei **Zoomfaktor** bzw. **Maximale Breite/Höhe** bestimmen Sie den Maßstab der Darstellung bei der Ausgabe. Nach Klick auf **Übernehmen** werden unter **Aktuell** die Werte angezeigt, die sich aus Ihren Einstellungen ergeben.

## Zoomfaktor

Ein Skalierungsfaktor von 100 % entspricht der Originalgröße, ein kleinerer Wert bewirkt eine entsprechende Verkleinerung, ein größerer Wert eine Vergrößerung.

## Anpassen an Seitenzahl

Durch Auswahl dieser Option können Sie die Anzahl der Seiten in Höhe und Breite vorgeben, auf die das Diagramm bei der Ausgabe maximal aufgeteilt werden soll (**max. Höhe, max. Breite**). Die Diagramme werden so groß wie möglich, aber ohne Verzerrungen dargestellt.

## Knoten nicht schneiden/Titel/Legende wiederholen

Aktivieren Sie dieses Kontrollkästchen, damit bei der Ausgabe des Diagramms auf mehreren Seiten Knoten nicht durchtrennt werden, und damit Titel und Legende - sofern vorhanden - auf jeder Seite ausgegeben werden.

## Seiten mit Leerraum auffüllen

Mithilfe dieser Option können Sie festlegen, ob zwischen dem Diagramm und den Boxen für Titel und Legende so viel Platz gelassen wird, dass die Boxen auf jeder Druckseite immer in voller Breite gedruckt werden können und fest am Blattrand positioniert sind. Wenn diese Option nicht ausgewählt ist, werden die Boxen ohne Zwischenraum am Diagramm gedruckt und können dann je nach Diagramm auf den verschiedenen Druckseiten in der Breite variieren.

## Rahmen außen

Aktivieren Sie dieses Kontrollkästchen, damit ein Rahmen um das Diagramm herum ausgegeben wird. Wenn die Option **Knoten nicht durchtrennen/Titel wiederholen** ausgewählt ist, erhält jede Seite einen Rahmen, andernfalls wird ein Rahmen um das gesamte Diagramm gezogen.

## Ausrichtung

Bestimmen Sie die Ausrichtung des Diagramms auf dem Blatt.

## Zuschnittmarken

Wurde dieses Kontrollfeld aktiviert, wird das Diagramm mit Zuschnittmarken versehen, die das Zusammenkleben der ausgedruckten Einzelseiten zu einer Gesamtgrafik erleichtern.

## Faltmarkierungen (DIN 824)

In der DIN Norm 824 ist für Bauzeichnungen eine ganz bestimmte Art der Faltung vorgeschrieben, mit der man die Zeichnung auf DIN A4-Größe zusammenfalten kann. Die Ausgabe von entsprechenden Faltmarkierungen auf Ihrem Diagramm erleichtert Ihnen die Faltung. Folgende Möglichkeiten stehen zur Verfügung:

- **Form A:** mit Heftrand auf der linken Seite, damit die Zeichnung gelocht und in einem Ordner abgeheftet werden kann.
- **Form B:** insgesamt etwas schmaler, damit ein Heftstreifen angebracht werden kann, der dann gemeinsam mit der Zeichnung die Breite von DIN A4 erreicht.
- **Form C:** die gefaltete Zeichnung wird nicht gelocht, sondern in eine Sichthülle gelegt.

Die vorliegenden Faltmarkierungen können für jedwedes Zielformat ausgegeben werden, während die DIN 824 explizit nur die Formate DIN A0 bis A3 kennt.

## Seitennummerierung

Ist dieses Kontrollkästchen aktiviert, wird auf jeder Seite unten links die Seitennummer ausgegeben. Folgende Möglichkeiten stehen dabei zur Auswahl:

- **Zeile.Spalte:** Sinnvoll, wenn das Diagramm sich auf mehr als eine Seite in der Länge als auch in der Breite erstreckt. Vor dem Punkt wird die Position der Seite in der vertikalen, dann die in der horizontalen Reihenfolge ausgegeben.
- **Spalte.Zeile:** Sinnvoll, wenn das Diagramm sich auf mehr als eine Seite in der Länge als auch in der Breite erstreckt. Vor dem Punkt wird die Position der Seite in der horizontalen, dann die in der vertikalen Reihenfolge ausgegeben.
- **Seite/Anzahl:** Zuerst erscheint die aktuelle Seitenzahl, danach die Anzahl der Gesamtseiten: 1/6, 2/6 etc.

## Text

Aktivieren Sie dieses Kontrollkästchen, um jede Seite unten links mit einem beliebigen Text zu versehen. Dieser Zusatztext wird ggf. rechts von der Seitennummer ausgegeben.

Für die Seitennummerierung können Sie in die Zeile **Zusatztext** folgende Platzhalter eingeben, die dann beim Ausdruck durch die entsprechenden Inhalte ersetzt werden:

{PAGE}	= fortlaufende Seitennummer
{NUMPAGES}	= Gesamtanzahl der Seiten
{ROW}	= Zeilenposition des Ausschnitts im Gesamtdiagramm
{COLUMN}	= Spaltenposition des Ausschnitts im Gesamtchart

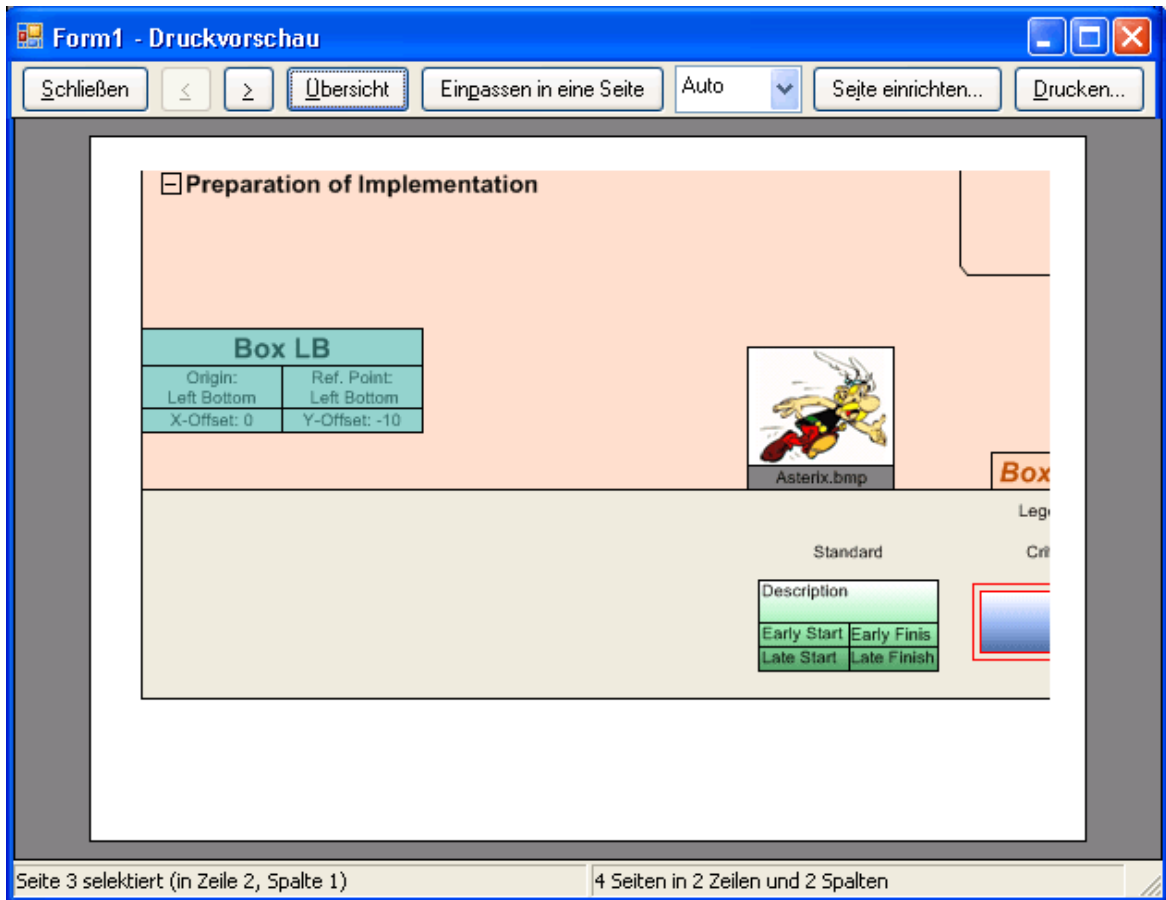
## Druckdatum

Ist dieses Kontrollkästchen aktiviert, wird auf jeder Seite unten links das Druckdatum ausgegeben. Das Druckdatum wird ggf. rechts von der Seitennummer und dem Zusatztext ausgegeben.

## Seitenränder

Über die Felder **Oben**, **Unten**, **Links** und **Rechts** legen Sie den Raum zwischen Papierrand und dem Diagramm in cm fest. Mindestränder, die aus technischen Gründen bei den Druckern entstehen, können nicht unterschritten werden. Bei Druckern mit Mindestrandvorgaben gelten die hier eingetragenen Werte für die Seitenränder zusätzlich zu den Mindestwerten, sodass die tatsächlich ausgegebenen Seitenränder größer sind als hier festgelegt.

## 5.10 Druckvorschau



Vor dem Drucken können Sie das Diagramm in der Druckvorschau prüfen. Es wird so auf dem Bildschirm dargestellt, wie es im Dialogfeld **Seite einrichten** festgelegt ist und wie es gedruckt wird.

Sie können hier einzelne Seiten oder die Gesamtübersicht über alle Seiten Ihrer Darstellung ansehen oder einen Ausschnitt Ihres Diagramms interaktiv vergrößern und anschliessend drucken.

Die Statuszeile informiert über die Gesamtseitenzahl und die horizontale und vertikale Aufteilung der Seiten. In der Ansicht **Einzelseite** wird zusätzlich noch die aktuelle Seitenzahl angezeigt.

### Schließen

Sie verlassen die Druckvorschau und gelangen zurück in die Darstellung.

<



*Nur aktiv, wenn die Schaltfläche **Einzel** gedrückt wurde.* Wenn das Diagramm sich über mehrere Seiten erstreckt, können Sie sich die Seiten


einzel<sup>n</sup> ansehen. Mit Hilfe dieser Schaltfläche gelangen Sie zur vorangehenden Seite. Sie bewegen sich über die Seiten von rechts nach links in aufsteigenden Zeilen.

>

*Nur aktiv, wenn die Schaltfläche **Einzel** gedrückt wurde.* Wenn das Diagramm sich über mehrere Seiten erstreckt, können Sie sich die Seiten einzeln ansehen. Mit Hilfe dieser Schaltfläche gelangen Sie zur nächsten Seite. Sie bewegen sich über die Seiten von links nach rechts in absteigenden Zeilen.

## Einzelseite/Übersicht

Wenn das Diagramm sich über mehrere Seiten erstreckt, können Sie sich die Seiten entweder einzeln oder in der Übersicht ansehen. In der **Übersicht** sehen Sie alle Seiten - je nach Seitenanzahl mehr oder weniger stark verkleinert, im Darstellungsmodus **Einzelseite**, wird zunächst die erste Seite des Diagramms einzeln und vergrößert dargestellt. Mit  oder  können Sie dann durch die Seiten blättern. Mit einem Doppelklick auf eine Seite wechseln Sie bequem zwischen den beiden Darstellungsarten **Einzelseite** und **Übersicht**.

Ein Ausschnitt Ihres Diagramms lässt sich im Darstellungsmodus **Einzelseite** interaktiv vergrößern, indem Sie bei gedrückter linker Maustaste ein Rechteck um den zu vergrößernden Bildausschnitt ziehen. Sobald Sie die linke Maustaste loslassen, wird der eingerahmte Bildausschnitt entsprechend vergrößert und statt der Schaltfläche **Drucken** erscheint die Schaltfläche  über die Sie den Bildausschnitt dann in der aktuellen Vergrößerung drucken können. Beachten Sie bitte, dass der in der Druckvorschau interaktiv gewählte Vergrößerungsfaktor nicht den Skalierungsfaktor im Dialogfeld **Seite einrichten** verändert.

## Einpassen in eine Seite

Mit dieser Schaltfläche lässt sich ein mehrseitiges Diagramm auf eine Seite verkleinern. Auch hier können Teile des Diagramms, wie unter **Einzelseite/Übersicht** beschrieben, interaktiv vergrößert und anschließend gedruckt werden.

## Zoomfaktor

Wählen Sie einen Zoomfaktor aus der Liste oder definieren einen individuellen, um die Darstellungsgröße ihres Diagramms für die Druckvorschau zu verändern. Dies ist nur im Modus "Einzelseite" möglich. Der Zoomfaktor lässt sich auch durch Drehen des Mousrades bei gedrückter <STRG>-Taste verändern. Er hat keinen Einfluss auf den späteren Druck. Je nach gewählter Größe werden vertikale und/oder horizontale Bildlaufleisten angezeigt. Auch das Mousrad kann zum Bewegen des Bildes verwendet werden (ohne Umschalttaste vertikal, mit Umschalttaste horizontal).

Der Zoomfaktor **Auto** ist voreingestellt und verkleinert bzw. vergrößert das Blatt immer so, dass es bildschirmfüllend dargestellt wird.

## Seite einrichten

Sobald Sie auf diese Schaltfläche klicken, gelangen Sie in das Dialogfeld **Seite einrichten** und können dort Änderungen am Seitenlayout vornehmen.

## Drucker einrichten

*Nur sichtbar, wenn auf der Eigenschaftenseite **Allgemeines** die Option **PrintDlgEx Dialog verwenden** nicht gewählt wurde.*

Sobald Sie auf diese Schaltfläche klicken, gelangen Sie in das Windows-Dialogfeld **Drucker einrichten** und können dort Änderungen an den Drucker-einstellungen vornehmen.

## Drucken/Ausschnitt drucken

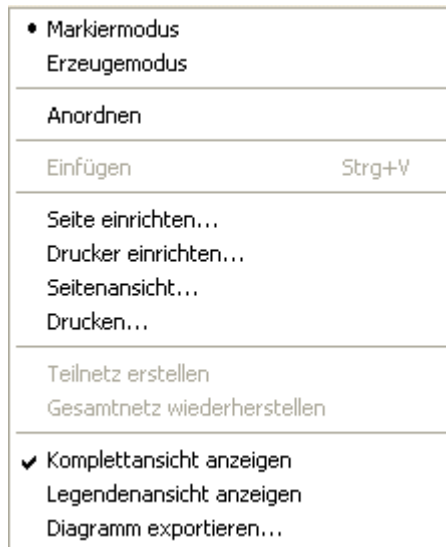
Über diese Schaltfläche gelangen Sie in das Windows-Dialogfeld **Drucken** und können von dort aus den Druckvorgang einleiten.

Wenn Sie in der Druckvorschau einen Ausschnitt interaktiv gezoomt haben, ändert die Schaltfläche ihre Beschriftung zu **Ausschnitt drucken**. Wenn Sie sie anklicken, ist im Windows-Dialogfeld **Drucken** die Option **Markierung** bereits ausgewählt. Durch Klick auf **OK** wird der am Bildschirm dargestellte Ausschnitt gedruckt.

Beachten Sie bitte, dass der in der Druckvorschau interaktiv gewählte Vergrößerungsfaktor nicht den Skalierungsfaktor im Dialogfeld **Seite einrichten** verändert.

## 5.11 Kontextmenü für das Diagramm

Wenn Sie die rechte Maustaste drücken, wenn der Mauszeiger im Diagrammbereich (nicht auf einem Objekt) steht, öffnet sich das folgende Kontextmenü:



### Markiermodus

Der Markiermodus ist der Standardmodus.

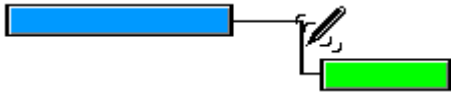
### Erzeugemodus

Dieser Modus kann nur eingeschaltet werden, wenn auf der Eigenschaftenseite **Allgemeines** die Option **Erzeugung neuer Knoten und Verbindungen zulassen** gesetzt ist.

Der Mauszeiger wird im leeren Diagrammbereich zu einem Knotenphantom in der Form eines Rechtecks. Jeder Klick mit der linken Maustaste erzeugt in diesem Modus einen neuen Knoten. Wenn die Option **Erzeugung neuer Knoten mit Dialog** auf der Eigenschaftenseite **Allgemeines** gewählt ist, öffnet sich das Dialogfeld **Vorgänge bearbeiten**, sobald Sie die Maustaste loslassen. Hier können Sie alle Daten des neu angelegten Knotens bearbeiten.

Wenn Sie im Erzeugemodus die Maus von einem Knoten zu einem anderen ziehen, wird der Mauszeiger zu einem kleinen Bleistift, und eine Verbindung zwischen diesen Knoten wird erzeugt. Wenn die Option **Erzeugung neuer Verbindungen mit Dialog** auf der Eigenschaftenseite **Allgemeines** gewählt ist, öffnet sich das Dialogfeld **Vorgänge bearbeiten**, sobald Sie die Maustaste loslassen. Hier können Sie alle Daten der neu angelegten Verbindung bearbeiten.





Der Erzeugemodus lässt sich auf zwei Arten aktivieren:

1. über das standardmäßige Kontextmenü im Diagrammbereich
2. über das Setzen der VcNet-Eigenschaft **InteractionMode** auf den Wert **vcCreateNodesAndLinks**.

## Anordnen

Über diesen Menüpunkt werden die vom Anwender verschobenen Knoten inklusive der zugehörigen Verbindungen optimal angeordnet.

## Einfügen

Dieser Menüpunkt steht nur zur Verfügung, wenn Sie zuvor einen oder mehrere Knoten ausgeschnitten oder kopiert haben. Diese(n) können Sie dann mit diesem Befehl an der Position des Mauszeigers einfügen.

## Seite einrichten

Sie gelangen in das Dialogfeld **Seite einrichten**.

## Drucker einrichten

*Nur aktiv, wenn auf der Eigenschaftenseite **Allgemeines** die Option **PrintDlgEx Dialog verwenden** nicht gewählt wurde.*

Sie gelangen in das Windows-Dialogfeld **Drucker einrichten**.

## Druckvorschau

Sie gelangen in das Dialogfeld **Druckvorschau**.

## Drucken

Wenn Sie diese Option wählen, gelangen Sie in das Windows-Dialogfeld **Drucken**.

## Teilnetz erstellen

*(nur aktiv, wenn Knoten markiert sind)* Wählen Sie diese Option, um ein Teilnetz aus den markierten Knoten darzustellen.

## Gesamtnetz wiederherstellen

*(nur aktiv, wenn zuvor die Option **Teilnetz erstellen** gewählt wurde)* Wählen Sie diese Option, um das Gesamtnetz wiederherzustellen.

## Komplettansicht anzeigen

Über diesen Menüpunkt können Sie die Komplettansicht ein- oder ausschalten. Die Komplettansicht ist ein zusätzliches Fenster, in dem das komplette Diagramm angezeigt wird. Ein farbiger Rahmen markiert den aktuell im Hauptfenster dargestellten Diagrammausschnitt. Wenn Sie diesen Rahmen mit der Maus verschieben, wird auch im Hauptfenster der entsprechende Diagrammausschnitt angezeigt.

Die Komplettansicht lässt sich auch über die Eigenschaft **VcWorldView.Visible** an- oder abschalten.

## Legendenansicht anzeigen

Über diesen Menüpunkt können Sie die Legendenansicht ein- oder ausschalten. Die Legende erscheint in einem zusätzlichen Fenster.

Sie können die Legendenansicht auch über die Eigenschaft **VcLegendView.Visible** an- oder abschalten.

## Diagramm exportieren

Wenn Sie diese Option wählen, gelangen Sie in das Windows-Dialogfeld **Speichern unter**, in dem Sie das dargestellte Diagramm als Grafikdatei speichern können.

Sie können diesen Dialog auch mit der VcNet-Methode **ShowExportGraphicsDialog** aufrufen.

Beim Exportieren wird die Größe des exportierten Diagramms in Pixeln wie folgt berechnet:

- PNG: Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter **SizeX** ein Wert  $\leq -50$  angegeben, so wird die absolute Zahl als DPI-Vorgabe genommen.
- GIF, TIFF, BMP, JPEG: Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter **SizeX** ein Wert  $\leq -50$  angegeben, so wird die absolute Zahl als DPI-Vorgabe genommen. Es gibt aber zusätzlich eine interne Begrenzung auf 50 MB Größe für die im Speicher für das Exportieren benötigte, nicht

komprimierte Ausgangsbitmap, so dass größere Grafiken eine kleinere Auflösung bekommen als gewünscht.

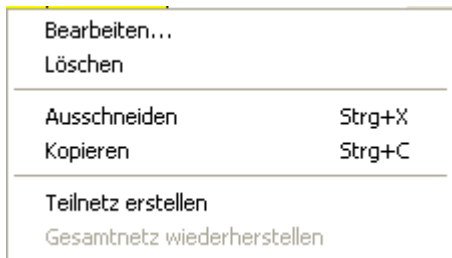
- WMF: Es wird eine feste Auflösung angenommen, bei der die größere Ausdehnung die Koordinaten von 0 bis 10.000 benutzt. Die kleinere Ausdehnung benutzt entsprechend einen kleineren Maximalwert für die Koordinaten für eine verzerrungsfreie Darstellung.
- EMF/EMF+: Es wird die volle Auflösung mit Koordinaten in 1/100 mm Abstand verwendet.

Detaillierte Erläuterungen zu den einzelnen Grafik-Formaten finden Sie im Kapitel: "Wichtige Konzepte: Grafikformate".

---

## 5.12 Kontextmenü für Knoten

Wenn ein oder mehrere Knoten markiert sind und Sie die rechte Maustaste drücken, öffnet sich das folgende Kontextmenü:



### Bearbeiten

Wenn Sie diese Option wählen, öffnet sich das Dialogfeld **Vorgänge bearbeiten**.

### Löschen

Wenn Sie diese Option wählen, werden die markierten Knoten gelöscht.

### Ausschneiden

Mit diesem Befehl können Sie die markierten Knoten ausschneiden.

### Kopieren

Mit diesem Befehl können Sie die markierten Knoten kopieren.

### Teilnetz erstellen

Wählen Sie diese Option, um ein Teilnetz aus den markierten Knoten darzustellen.

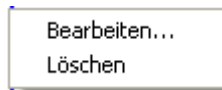
### Gesamtnetz wiederherstellen

*(nur aktiv, wenn zuvor die Option **Teilnetz erstellen** gewählt wurde)* Wählen Sie diese Option, um das Gesamtnetz wiederherzustellen.

---

## 5.13 Kontextmenü für Verbindungen

Wenn Sie eine Verbindung mit der rechten Maustaste anklicken, erscheint folgendes Menü:



### **Bearbeiten**

Mit diesem Befehl öffnen Sie das Dialogfeld **Verbindung bearbeiten**, in dem Sie alle Daten der markierten Verbindung bearbeiten können.

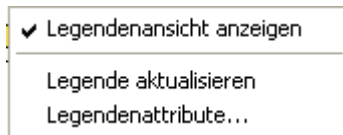
### **Löschen**

Wenn Sie die markierte Verbindung löschen möchten, bestätigen Sie dies durch einen Klick mit der linken Maustaste.

---

## 5.14 Kontextmenü für die Legende

Wenn Sie mit der rechten Maustaste auf die Legende klicken, erscheint das folgende Menü:



### Legendenansicht anzeigen

Über diesen Menüpunkt können Sie die Legendenansicht ein- oder ausschalten.

### Legende aktualisieren

Über diesen Menüpunkt können Sie die Legendenansicht aktualisieren.

Eine Aktualisierung über das Menü kann nötig sein, da nach Änderungen im Diagramm die Legende nicht automatisch aktualisiert wird. Werden also zum Beispiel Knoten hinzugefügt oder gelöscht, muss eine Aktualisierung entweder über das Kontextmenü oder durch Aus- und Einschalten der Legende durchgeführt werden. Dies gilt auch für das Laden von Knoten. Wenn für die Legendenansicht auf der Eigenschaftenseite **Zusätzliche Ansichten** die Option **Beim Start sichtbar** eingestellt wurde, aber zum Zeitpunkt des Aufbaus noch keine Knoten geladen waren, bleibt die Legende bis zur Aktualisierung leer.

### Legendenattribute

Über diesen Menüpunkt öffnen Sie den gleichnamigen Dialog, in dem Sie Einstellungen zum Legendentitel, den Legendenelementen und den Rändern der Legende vornehmen können. Weitere Information zu diesem Dialog finden Sie in Kapitel 4.44 "Dialogfeld Legendenattribute".



---

---

## **6 Häufig gestellte Fragen**

---

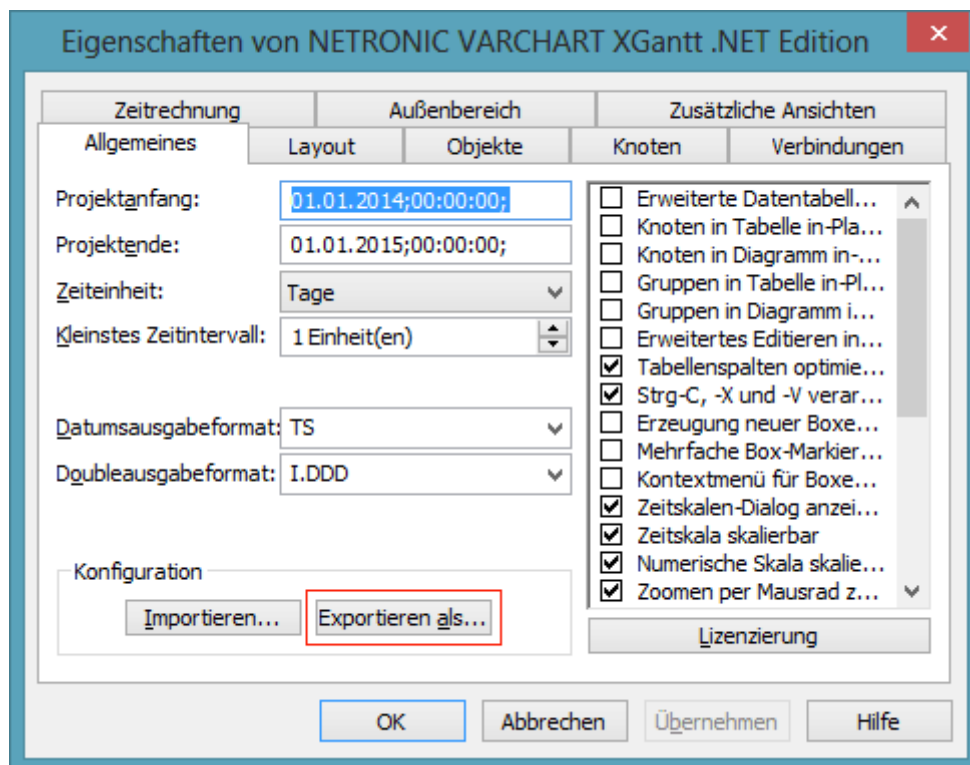
---

### **6.1 Was muss ich tun, wenn ich von VARCHART XGantt 4.4 für .NET nach XGantt 5.0 umsteigen möchte?**

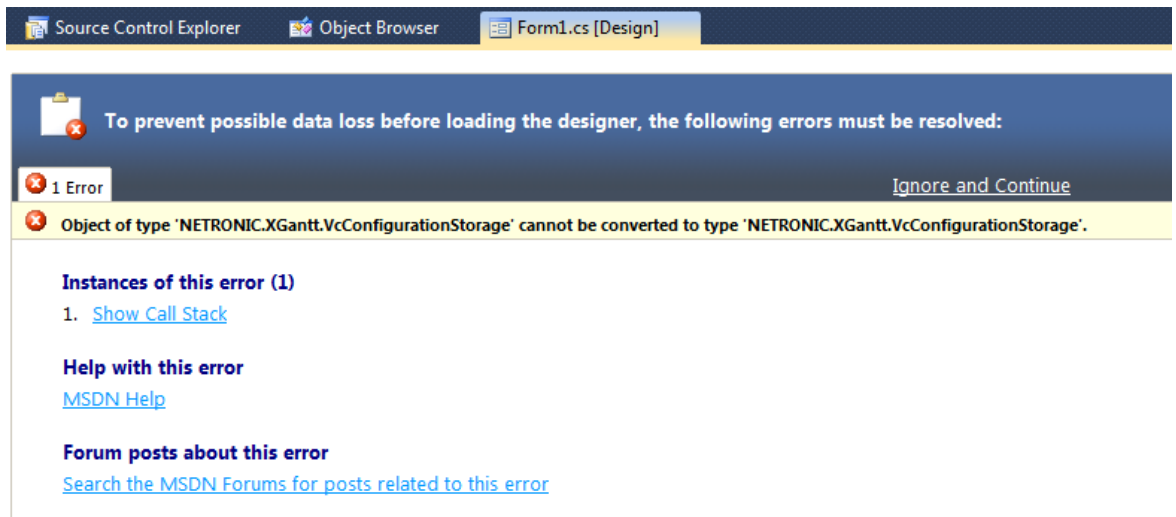


## 6.2 Was muss ich tun, wenn ich im Rahmen eines Service Releases auf einen neuen Build von VARCHART XGantt umsteigen möchte?

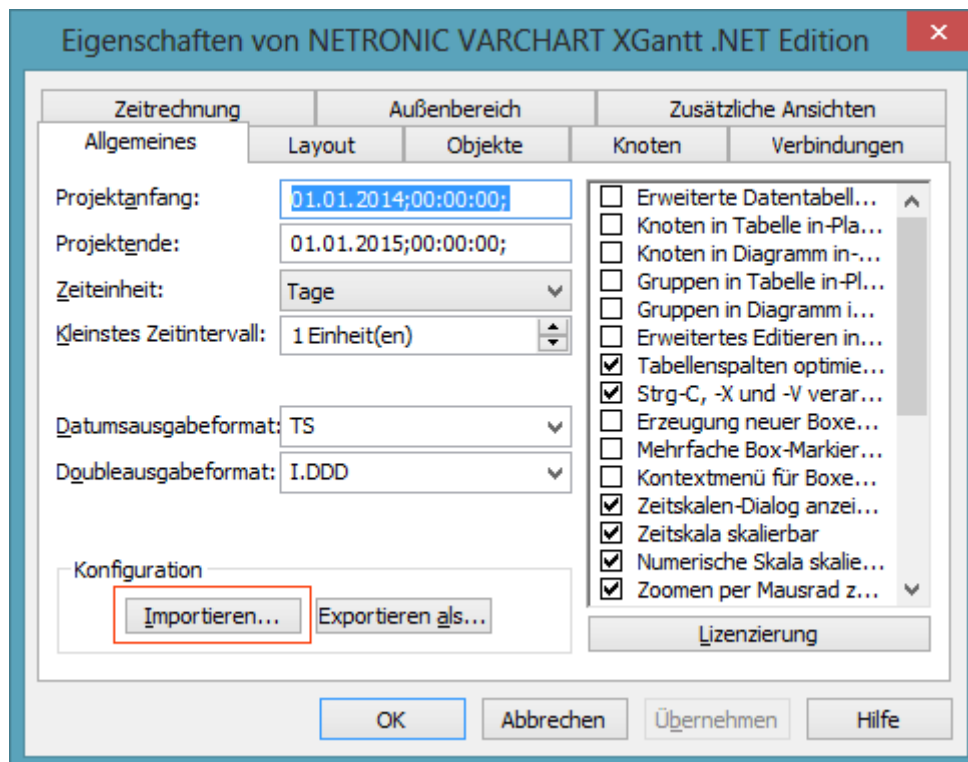
1. Bevor Sie VARCHART XGantt 5.0 für .NET installieren, öffnen Sie im Formdesigner von Visual Studio das Formular, das XGantt 4.4 verwendet, und sichern Sie die aktuelle Konfiguration von XGantt über die Schaltfläche **Exportieren** auf der Eigenschaftenseite **Allgemeines**:



2. Schließen Sie zuerst das Formular und dann Visual Studio.
3. Installieren Sie den neuen Build von VARCHART XGantt für .NET in denselben Ordner, in dem der alte Build vorhanden ist.
4. Öffnen Sie im Formdesigner das Formular, das XGantt enthält. Folgende Fehlermeldung erscheint:

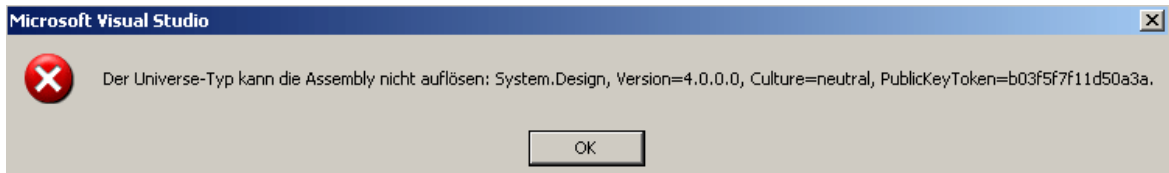


5. Klicken Sie auf **Ignore and Continue**. Damit wird das Formular im Formdesigner wieder korrekt angezeigt, XGantt wurde jedoch auf seine Standardkonfiguration zurückgesetzt.
6. Importieren Sie Ihre zuvor gesicherte Konfiguration über die Schaltfläche **Importieren** auf der Eigenschaftenseite **Allgemeines**:



7. Damit hat VARCHART XGantt dann wieder die von Ihnen eingestellte Konfiguration.

### 6.3 Warum erscheint eine Fehlermeldung, wenn man bei Visual Studio 2010 ein neues Projekt erzeugt und versucht, das Steuerelement auf die Form zu ziehen?



Diese Fehlermeldung erscheint, da bei Visual Studio 2010 das **.NET Framework 4 Client Profile** als Standard voreingestellt ist, das VARCHART-Steuerelement von NETRONIC jedoch das Zielframework **.NET Framework 4** benötigt, da ersteres nicht die von den Eigenschaftenseiten zur Designzeit benötigte System.Design.dll enthält. Sie müssen daher **bevor** Sie das Steuerelement auf die Form ziehen, unter **Projekteigenschaften/Anwendung (C#)** bzw. **Erweiterte Kompilereinstellungen (VB)** das Zielframework von **.NET Framework 4 Client Profile** auf **.NET Framework 4** umstellen.

---

## **6.4 Wie kann das VARCHART Windows Forms neu lizenziert werden?**

1. Entwicklungsumgebung schließen
2. Lizenzdatei NETRONIC.XNet.VcNet.lic in das Installationsverzeichnis von VARCHART XNet.NET kopieren
3. Umgebung wieder starten und Projekt erneut übersetzen

---

## 6.5 Wieso können Knoten u. U. nicht interaktiv erzeugt werden?

Wenn Sie zur Laufzeit keine Knoten mit der Maus anlegen können, stellen Sie sicher, dass auf der Eigenschaftenseite **Allgemeines** das Kontrollkästchen **Erzeugung neuer Knoten und Verbindungen zulassen** aktiviert ist.

Kontrollieren Sie außerdem, ob in Ihrem Programmcode die VARCHART-VcNet-Eigenschaft **NodeAndLinkCreationAllowed** auf **False** gesetzt wurde; dann können ebenfalls keine neuen Knoten angelegt werden.

---

## 6.6 Wieso können Verbindungen u. U. nicht interaktiv erzeugt werden?

Wenn Sie zur Laufzeit keine Verbindungen zwischen Knoten interaktiv anlegen können, sind verschiedene Ursachen dafür möglich:

1. Kontrollieren Sie, ob auf der Eigenschaftenseite **Allgemeines** das Kontrollkästchen **Erzeugung neuer Knoten und Verbindungen zulassen** aktiviert ist. Wenn es aktiviert ist, können Sie Knoten und Verbindungen interaktiv anlegen.
2. Wenn Sie nun immer noch keine Verbindungen sehen können, prüfen Sie Ihre Einstellungen für die Verbindungen. Möglicherweise können Sie Verbindungen zwar anlegen, diese aber nicht sehen. Öffnen Sie dazu die Eigenschaftenseite **Verbindungen** und kontrollieren Sie dort die Linienart jedes Verbindungsaussehens. Wenn Sie feststellen, dass Verbindungslinien dieselbe Farbe haben wie der Diagrammhintergrund, wählen Sie eine Farbe, die sich deutlich davon abhebt.
3. Prüfen Sie die Filterbedingungen. Falsch definierte Filterbedingungen können zur Folge haben, dass Verbindungen nicht dargestellt werden.
4. Wenn das Verbindungsausssehen sinnvoll definiert ist, Sie aber immer noch keine Verbindungen sehen, prüfen Sie, ob die Datenfelder für die Verbindungen (**Vorgänger**, **Nachfolger**, **Verbindungstyp**) richtig definiert sind.

---

## 6.7 Wie verhindert man das interaktive Erzeugen von Knoten?

Sie können auf verschiedene Weisen verhindern, dass Knoten und Verbindungen interaktiv angelegt werden können:

1. Deaktivieren Sie auf der Eigenschaftenseite **Allgemeines** das Kontrollkästchen **Erzeugung neuer Knoten und Verbindungen zulassen**.
2. Setzen Sie Rückgabestatus von **VcNodeCreating** auf **vcRetStatFalse**, damit interaktiv erzeugte Knoten wieder gelöscht werden.
3. Ergänzen Sie die folgenden Codezeilen:

### Code-Beispiel

```
Sub Form_Load
    VcNet1.AllowNewNodesAndLinks = False
End Sub
```

---

## 6.8 Wie lassen sich die Standard-Kontextmenüs abschalten?

Die vordefinierten Kontextmenüs können Sie durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrücken.

### Code-Beispiel VB.NET

```
' Kontextmenü für das Diagramm abschalten
Private Sub VcNet1_VcDiagramRightClicking(ByVal x As Long, ByVal y As
Long, _
                                returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

' Kontextmenü für Verbindungen abschalten
Private Sub VcNet1_VcLinksRightClicking(ByVal linkCltn As _
VcNetLib.VcLinkCollection, _
ByVal x As Long, _
ByVal y As Long, _
returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub

' Kontextmenü für Knoten abschalten
Private Sub VcNet1_VcNodeRightClicking(ByVal node As VcNetLib.VcNode, _
ByVal location As _
VcNetLib.VcLocation, _
ByVal x As Long, ByVal y As Long, _
returnStatus As Variant)
    returnStatus = vcRetStatNoPopup
End Sub
```



---

## 6.9 Wie lässt sich die Performance verbessern?

### > Suspend update

Bei größeren Datenmengen kann es unter Umständen sehr lange dauern, wenn bei einer großen Anzahl von Vorgängen dieselbe Aktion durchgeführt wird. Nicht jeder automatische Aktualisierungsvorgang im Diagramm ist notwendig; in einem solchen Fall können Sie Aktualisierungen für den Einzelfall unterdrücken und nach der Abarbeitung einer Code-Sequenz final einmal durchführen. Unterdrückung und Wiedereinsatz der Aktualisierung erfolgen mit der Methode **SuspendUpdate**, die zu Beginn der Code-Sequenz auf **True** bzw. am Ende auf **False** gesetzt wird. Auf diese Weise können Sie die Performance insgesamt beträchtlich erhöhen.

#### Code-Beispiel

```
VcNet1.SuspendUpdate (True)

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

VcNet1.SuspendUpdate (False)
```

Auch beim Modifizieren von Tabellenformaten lässt sich die Performance steigern.

### > Grafiken

Eine Ursache für eine zu geringe Performance können Grafiken beispielsweise in Tabellen-, Knoten- oder Boxfeldern sein, die zu groß sind oder eine zu große Pixelzahl haben.

---

## 6.10 Fehlermeldungen

> **Fehlermeldungen zur Laufzeit, vom Entwickler verursacht**

Wird noch bearbeitet.

> **Fehlermeldungen zur Laufzeit, verursacht vom Endanwender oder vom Entwickler**

Fehlerursache	Meldung
Zyklen in der Methode <b>ScheduleProject</b> entdeckt	Das Projekt enthält Zyklen!

---

## **6.11 Was tun, wenn das Steuerelement unerwartet nicht in allen Benutzerkonten eines Rechners funktioniert?**

Wenn Sie feststellen, dass das Steuerelement nicht ansprechbar ist, wenn ein anderer Benutzer die gleiche, das Steuerelement nutzende Applikation aufruft, dann liegt es daran, dass das Steuerelement nicht für alle Benutzer installiert wurde. Bei der Erstellung des Setup-Programms, mit dem Sie das Steuerelement auf dem Kundenrechner installieren, sollte die Option "Für alle Benutzer installieren" ausgewählt sein.

Nachträglich oder von Hand lässt sich eine Installation für mehrere Benutzerkonten freischalten, indem Sie die Sicherheitseinstellungen der zum Steuerelement gehörenden Dateien so erweitern, dass andere Benutzerkonten als das, unter dem die Installation stattfand, auf die Dateien zugreifen dürfen. Die Sicherheitseinstellungen können Sie einerseits über den Menüpunkt "Eigenschaften" des Kontextmenüs zur jeweiligen Datei ändern oder per Kommandozeile über den Befehl 'cacls'. Die zum Steuerelement gehörenden Dateien finden Sie im Kapitel "Auslieferung" am Anfang dieses Buches.

---

## 6.12 Sind alle Fonts verwendbar?

Bedingt durch die Umstellung auf GDI+ gibt es Einschränkungen bzgl. der Darstellung von Schriftarten. So ist GDI+ nicht in der Lage, Fonts darzustellen, die als Postscript- oder gar Bitmap-Fonts vorliegen. Zu ersteren gehören auch Fonts, die zwar vom Typ **OpenType** sind, aber als "klassische Fonts" intern eine Art Postscript-Struktur besitzen (wie z.B. "Warnock Pro"). Zu den letzteren gehören z.B. die "alten" Windows-Fonts "Courier" und "Times", aber auch "System" und "MS Sans Serif".

Diese Fonts werden von den Fontauswahldialogen im VARCHART-Steuer-element daher nicht angeboten. Werden sie dennoch über die API gesetzt, so wird ein Ersatzfont angezeigt. Bei den "alten" Windows-Fonts sind von NETRONIC Ersetzungsregeln eingebaut worden, die einen ähnlichen "neuen" Font auswählen, bei externen Fonts wird dann "Arial" als Ersatz genommen, damit überhaupt Text auf dem Bildschirm erscheint.

Es ist zwar unwahrscheinlich, aber nicht auszuschließen, dass in zukünftigen Versionen von GDI+ Unterstützung für die bisher nicht unterstützten Fonts eingebaut wird. Informationen sind bei Microsoft leider nur in Blogs und News-Gruppen zu erhalten, nicht jedoch bei MSDN.



---

---

# 7 API Referenz

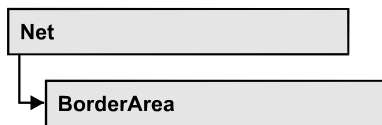
---

## 7.1 Objekttypen

- VcBorderArea
- VcBoundingBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcCalendar
- VcCalendarCollection
- VcCalendarProfile
- VcCalendarProfileCollection
- VcDataRecord
- VcDataRecordCollection
- VcDataTable
- VcDataTableCollection
- VcDataTableField
- VcDataTableFieldCollection
- VcFilter
- VcFilterCollection
- VcFilterSubCondition
- VcGroup
- VcGroupCollection
- VcInterval
- VcIntervalCollection
- VcLegendView
- VcLink
- VcLinkAppearance
- VcLinkAppearanceCollection
- VcLinkCollection
- VcLinkFormat
- VcLinkFormatCollection
- VcLinkFormatField
- VcMap

- VcMapCollection
- VcMapEntry
- VcNet
- VcNode
- VcNodeAppearance
- VcNodeAppearanceCollection
- VcNodeCollection
- VcNodeFormat
- VcNodeFormatCollection
- VcNodeFormatField
- VcPrinter
- VcRect
- VcScheduler
- VcWorldView

## 7.2 VcBorderArea



Ein Objekt vom Typ **VcBorderArea** bezeichnet den Titel- bzw. Legendenbereich der Grafik.

### Methoden

- **BorderBox**

## Methoden

### BorderBox

Methode von **VcBorderArea**

Diese Methode ermöglicht den Zugriff auf ein **BorderBox**-Objekt.

	Datentyp	Beschreibung
<b>Parameter:</b> boxPosition	VcBorderBoxPosition  <b>Mögliche Werte:</b> .vcBBXPBottomBottomCentered 8 .vcBBXPBottomBottomLeft 7 .vcBBXPBottomBottomRight 9 .vcBBXPBottomTopCentered 5 .vcBBXPBottomTopLeft 4 .vcBBXPBottomTopRight 6 .vcBBXPLegend 51 .vcBBXPTopCentered 2 .vcBBXPTopLeft 1 .vcBBXPTopRight 3	Position der Box  zweite Zeile im unteren Bereich, mittig zweite Zeile im unteren Bereich, links zweite Zeile im unteren Bereich, rechts erste Zeile im unteren Bereich, mittig erste Zeile im unteren Bereich, links erste Zeile im unteren Bereich, rechts Legende oben mittig oben links oben rechts
<b>Rückgabewert</b>	VcBorderBox	Box des Titel- und Legendenbereichs

### Code-Beispiel VB.NET

```

Dim boardArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

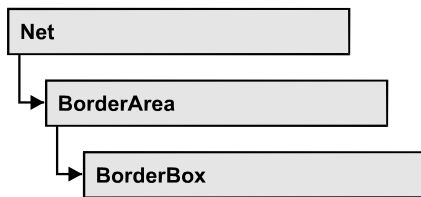
boardArea = VcNet1.BorderArea
bBoxBBL = boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
  
```



### Code-Beispiel C#

```
VcBorderArea boardArea = vcNet1.BorderArea;  
  
VcBorderBox bBoxBBL =  
boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);  
bBoxBBL.LegendTitle = "Explanation";
```

## 7.3 VcBoundingBox



Ein Objekt vom Typ **VcBoundingBox** bezeichnet eine der Boxen des Titel- bzw. Legendenbereichs der Grafik.

### Eigenschaften

- Alignment
- GraphicsFileName
- LegendElementsArrangement
- LegendElementsBottomMargin
- LegendElementsMaximumColumnCount
- LegendElementsMaximumRowCount
- LegendElementsTopMargin
- LegendFont
- LegendTitle
- LegendTitleFont
- LegendTitleVisible
- Text
- TextFont
- Type

---

## Eigenschaften

### Alignment

Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie die Ausrichtung dieses BorderBox-Objektes festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcBoundingBoxAlignment  <b>Mögliche Werte:</b> .vcBBXACentered -1 .vcBBXALeft -3	Ausrichtung der BorderBox  unten mittig links

.vcBBXARight -2	rechts
-----------------	--------

## GraphicsFileName

### Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie den Namen der Grafikdatei angeben oder erfragen, die in dem aktuellen VcBoundingBox-Objekt verwendet wird. Mögliche Formate:

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- \*.GIF (Graphics Interchange Format)
- \*.JPG (Joint Photographic Experts Group)
- \*.PNG (Portable Network Graphics)
- \*.TIF (Tagged Image File Format)
- \*.VMF (Viewer Metafile)
- \*.WMF (Microsoft Windows Metafile, ggf. mit eingebauten EMF)

Nur EMF, EMF+, VMF und WMF sind Vektorformate, in denen das Diagramm auflösungsunabhängig gespeichert werden kann. Die übrigen Formate sind pixelorientiert und bieten damit nicht beliebige Auflösungen.

Das VMF-Format wird in der Zukunft nicht mehr weiterentwickelt, aus Kompatibilitätsgründen für bestehende Anwendungen aber zunächst noch weiter unterstützt.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Grafikdatei

**Code-Beispiel VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBoundingBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomTopRight)
borderBox.Type = VcBoundingBoxType.vcBBXTGraphics
borderBox.GraphicsFileName = "C:\Asterix.jpg"
```

**Code-Beispiel C#**

```
VcBorderArea borderArea = vcNet1.BorderArea;
VcBoundingBox borderBox =
borderArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomTopRight);
borderBox.Type = VcBoundingBoxType.vcBBXTGraphics;
borderBox.GraphicsFileName = @"C:\Asterix.jpg";
```

**LegendElementsArrangement****Eigenschaft von VcBoundingBox**

Mit dieser Eigenschaft können Sie die Anordnung der Elemente der Legende setzen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLegendElementsArrangement	Typ der Anordnung der Legendenelemente
	<b>Mögliche Werte:</b>	
	.vcLEAFixedToColumns 0	Die Legendenelemente werden nur in Spalten ausgerichtet.
	.vcLEAFixedToRows 1	Die Legendenelemente werden nur in Zeilen ausgerichtet.
	.vcLEAFixedToRowsAndColumns 2	Die Legendenelemente werden in Zeilen und Spalten ausgerichtet.

**LegendElementsBottomMargin****Eigenschaft von VcBoundingBox**

Mit dieser Eigenschaft können Sie den Abstand der Elemente zum unteren Rand festlegen oder erfragen (Einheit: mm).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Breite des unteren Randes

## LegendElementsMaximumColumnCount

Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie die Anzahl der Spalten setzen oder erfragen, über die sich die Elemente der Legende verteilen sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Spalten

## LegendElementsMaximumRowCount

Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie die Anzahl der Zeilen setzen oder erfragen, über die sich die Elemente der Legende verteilen sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Zeilen

## LegendElementsTopMargin

Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie den Abstand der Elemente zum oberen Rand festlegen oder erfragen (Einheit: mm).

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite des oberen Randes

## LegendFont

Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie die Schriftattribute der Legende angeben bzw. erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftattribute der Legende

**Code-Beispiel VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendFont.Name)
```

**Code-Beispiel C#**

```
VcBorderArea borderArea = vcNet1.BorderArea;
VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBorderBoxType.vcBBXTLegend;
MessageBox.Show(borderBox.LegendFont.Name);
```

**LegendTitle****Eigenschaft von VcBorderBox**

Mit dieser Eigenschaft können Sie den Legendentitel angeben bzw. erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Legendentitel

**Code-Beispiel VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitle = "Explanation"
```

**Code-Beispiel C#**

```
VcBorderArea borderArea = vcNet1.BorderArea;

VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.LegendTitle = "Explanation";
```

**LegendTitleFont****Eigenschaft von VcBorderBox**

Mit dieser Eigenschaft können Sie die Schriftattribute des Legendentitels angeben bzw. erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.DrawingFont	Schriftattribute des Legendentitels

**Code-Beispiel VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendTitleFont.Name)
```

**Code-Beispiel C#**

```
VcBorderArea borderArea = vcNet1.BorderArea;
VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBorderBoxType.vcBBXTLegend;
MessageBox.Show(borderBox.LegendTitleFont.Name);
```

**LegendTitleVisible****Eigenschaft von VcBorderBox**

Mit dieser Eigenschaft legen Sie fest, ob der Legendentitel sichtbar ist oder nicht.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Legendentitel sichtbar (True)/nicht sichtbar (False)

**Code-Beispiel VB.NET**

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitleVisible = False
```

**Code-Beispiel C#**

```
VcBorderArea borderArea = vcNet1.BorderArea;

VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.LegendTitleVisible = false;
```

**Text****Eigenschaft von VcBorderBox**

Mit dieser Eigenschaft können Sie den Text einer Titelzeile (oberhalb oder unterhalb des Diagramms) angeben oder erfragen. Für die Seitennummerierung oder die Ausgabe des Systemdatums können Sie folgende Platzhalter angeben, die dann beim Ausdruck durch die entsprechenden Inhalte ersetzt werden:

{COLUMN} = Seitennummer in der Breite (einer zweidimensionalen Seitenanordnung)

{NUMPAGES} = Gesamtanzahl der Seiten

{PAGE} = fortlaufende Seitennummer

{ROW} = Seitennummer in der Höhe (einer zweidimensionalen Seitenanordnung)

{SYSTEMDATE} = Systemdatum

Die Eigenschaft Text ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set\_Text (rowIndex, pvn) und get\_Text (rowIndex) angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> rowIndex	System.Int16	Zeilenindex {0..6}
<b>Eigenschaftswert</b>	System.String	Text in Textfeldern

#### Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBoundingBox

borderArea = VcNet1.BorderArea
borderBox = borderArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBoundingBoxType.vcBBXTText
borderBox.Text(index) = "Department A"
```

#### Code-Beispiel C#

```
VcBorderArea borderArea = vcNet1.BorderArea;

VcBoundingBox borderBox =
borderArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBoundingBoxType.vcBBXTText;
borderBox.set_Text(index, "DepartmentA");
```

## TextFont

### Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie die Schriftattribute einer Titelzeile (oberhalb oder unterhalb des Diagramms) angeben bzw. erfragen.



Dies ist eine indizierte Eigenschaft, die in C# über die beiden Methoden **set\_TextFont (rowIndex, pvn)** und **get\_TextFont (row-Index)** angesprochen wird.

Die Eigenschaft **TextFont** ist eine indizierte Eigenschaft, die in C# über die beiden Methoden **set\_TextFont (rowIndex, pvn)** und **get\_FieldText (row-Index)** angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> rowIndex	System.Int16	Zeilenindex {0..6}
<b>Eigenschaftswert</b>	System.DrawingFont	Schriftattribute des Textes

### Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim bBoxTL As VcBoundingBox

Set borderArea = VcNet1.BorderArea
Set bBoxBBL = borderArea.BorderBox(vcBBXPBottomBottomLeft)

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

### Code-Beispiel C#

```
// Text for Title
VcBoundingBox borderBox =
VcNet1.BorderArea.BorderBox(VcBoundingBoxPosition.vcBBXPTopCentered);
borderBox.Type = VcBoundingBoxType.vcBBXTText;

Font titleFont1 = new Font("Arial", 20, FontStyle.Bold);

borderBox.set_Text(1, "Time Scheduler");
borderBox.set_TextFont(1, titleFont1);
```

## Type

### Eigenschaft von VcBoundingBox

Mit dieser Eigenschaft können Sie den Typ des BorderBox-Objekts angeben bzw. erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcBoundingBoxType	Typ der Box
	<b>Mögliche Werte:</b> .vcBBXTGraphics 3 .vcBBXTLegend 4 .vcBBXTNothing 0 .vcBBXTText 1 .vcBBXTTextWithGraphics 2	Grafik Legende Leer Text Text und Grafik

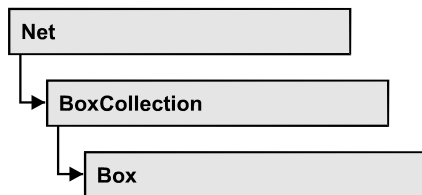
**Code-Beispiel VB.NET**

```
Dim bBoxBBL As VcBoundingBox  
  
bBoxBBL = boardArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomLeft)  
bBoxBBL.Type = VcBoundingBoxType.vcBBXTGraphics
```

**Code-Beispiel C#**

```
VcBorderArea boardArea = vcNet1.BorderArea;  
  
VcBoundingBox bBoxBBL =  
boardArea.BorderBox(VcBoundingBoxPosition.vcBBXPBottomBottomLeft);  
bBoxBBL.Type = VcBoundingBoxType.vcBBXTGraphics;
```

## 7.4 VcBox



Ein Objekt vom Typ **VcBox** beschreibt eine Box, in der Texte und Grafiken ausgegeben werden können.

### Eigenschaften

- FieldText
- FormatName
- LineColor
- LineThickness
- LineType
- Marked
- Moveable
- Name
- Origin
- Priority
- ReferencePoint
- UpdateBehaviorName
- Visible

### Methoden

- GetActualExtent
- GetTopLeftPixel
- GetXYOffset
- IdentifyFormatField
- SetXYOffset
- SetXYOffsetByTopLeftPixel

## Eigenschaften

### FieldText

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie den Inhalt eines Feldes einer Box erfragen oder festlegen. Diese Eigenschaft können Sie auch im Dialog **Box bearbeiten** festlegen.

Bei mehrzeiligen Textfeldern müssen für einen erzwungenen Umbruch die einzelnen Zeilen des Textfelds mit "\n" im String getrennt sein (Beispiel: "Zeile1\nZeile2"). Ohne erzwungenen Umbruch wird automatisch an Leerzeichen umgebrochen.

Die Eigenschaft FieldText ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set\_FieldText (fieldIndex, pvn) und get\_FieldText (fieldIndex) angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fieldIndex	System.Int16	Feldindex
<b>Eigenschaftswert</b>	System.String	Feldinhalt

#### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.FieldText(0) = "User: "
```

#### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.set_FieldText(0, "User: ");
```

### FormatName

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie den Namen des Boxformats erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcBoxFormat	BoxFormat-Objekt oder <b>Nothing</b>

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.FormatName = "Standard"
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.FormatName = "Standard";
```

## LineColor

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie die Farbe der Randlinie der Box festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte {0...255},{0...255},{0...255}

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.LineColor = System.Drawing.Color.Blue
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineColor = System.Drawing.Color.Blue;
```

## LineThickness

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie die Stärke der Randlinie der Box erfragen oder festlegen.

Wenn Sie diese Eigenschaft auf Werte zwischen 1 und 4 setzen, wird damit eine absolute Liniendicke in Pixel definiert, d.h. die Linien haben unabhängig vom Zoomfaktor immer die gleiche feste Linienstärke in Pixeln. Dies wird jedoch aufgrund der besseren Lesbarkeit beim Drucken in eine vom Zoomfaktor abhängige Liniendicke umgewandelt:

Wert	Punkte	mm
1	1/2 Punkt	0,09 mm
2	1 Punkt	0,18 mm
3	3/2 Punkt	0,26 mm
4	2 Punkt	0,35 mm

Ein Punkt ist 1/72 Zoll groß und stellt die Maßeinheit für Schriftgrößen dar.

Wenn Sie diese Eigenschaft auf Werte zwischen 5 und 1.000 setzen, wird damit eine Linienstärke in 1/100 mm definiert, d.h. die Linien bekommen eine tatsächliche Dicke in Pixeln, die abhängig vom Zoomfaktor ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm <b>Standardwert:</b> As defined in the dialog

#### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.LineThickness = 2
```

#### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineThickness = 2;
```

## LineStyle

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie den Typ der Randlinie der Box festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLineType	Linientyp
		<b>Standardwert:</b> vcSolid
	<b>Mögliche Werte:</b>	
	.vcDashed 4	Linientyp <b>gestrichelt</b>
	.vcDashed 4	Linientyp <b>gestrichelt</b>
	.vcDashedDotted 5	Linientyp <b>gestrichelt-gepunktet</b>
	.vcDashedDotted 5	Linientyp <b>gestrichelt-gepunktet</b>
	.vcDotted 3	Linientyp <b>gepunktet</b>
	.vcDotted 3	Linientyp <b>gepunktet</b>
	.vcLineType0 100	Linientyp 0
		<hr/>
	.vcLineType1 101	Linientyp 1
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType10 110	Linientyp 10
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType11 111	Linientyp 11
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType12 112	Linientyp 12
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType13 113	Linientyp 13
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType14 114	Linientyp 14
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType15 115	Linientyp 15
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType16 116	Linientyp 16
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType17 117	Linientyp 17
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType18 118	Linientyp 18
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType2 102	Linientyp 2
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType3 103	Linientyp 3
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType4 104	Linientyp 4
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType5 105	Linientyp 5
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType6 106	Linientyp 6
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType7 107	Linientyp 7
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType8 108	Linientyp 8
		<hr style="border-top: 1px dashed black;"/>
	.vcLineType9 109	Linientyp 9
		<hr style="border-top: 1px dashed black;"/>
	.vcNone 1	Kein Linientyp zugewiesen
	.vcNone 1	Kein Linientyp
	.vcNotSet -1	Kein Linientyp zugewiesen
	.vcSolid 2	Linientyp <b>durchgezogen</b>
	.vcSolid 2	Linientyp <b>durchgezogen</b>

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.LineType = VcLineType.vcDotted
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineType = VcLineType.vcDotted;
```

**Marked****Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie festlegen oder abfragen, ob eine Textbox markiert ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	<b>True:</b> Box markiert; <b>false:</b> Box nicht markiert

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Marked = True
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Marked = true;
```

**Moveable****Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Box interaktiv verschiebbar sein soll.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Verschiebbar (True)/ nicht verschiebbar (False) <b>Standardwert:</b> True



### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Moveable = False
```

### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Moveable = false;
```

## Name

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie den Namen einer Box erfragen oder setzen. Diese Eigenschaft können Sie im Dialog **Boxen verwalten** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name der Box

### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim boxName As String

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Name)
```

### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();

MessageBox.Show(box.Name);
```

## Origin

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie den Ursprungspunkt der Box festlegen oder erfragen, d. h. den Diagrammpunkt, von dem aus der Abstand zum Referenzpunkt der Box in x- bzw. y-Richtung angegeben wird.

Mithilfe der Eigenschaften **Origin** und **ReferencePoint** sowie der Methode **GetXyOffset** können Sie jede einzelne Box im Diagrammbereich positionieren, wobei die relative Position der Box zum Diagramm unabhängig von der Diagrammgröße ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcBoxOrigin	Ursprungspunkt der Box

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Origin = VcBoxOrigin.vcBOTopCenter
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Origin = VcBoxOrigin.vcBOTopCenter;
```

## Priority

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie die Priorität der Box erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Prioritätsstufe

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Priority = 3
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Priority = 3;
```

## ReferencePoint

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie den Referenzpunkt der Box festlegen oder erfragen, d. h. den Punkt der Box, von dem aus der Abstand zum Ursprung in x- bzw. y-Richtung angegeben wird.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcBoxReferencePoint	Referenzpunkt der Box

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight;
```

## UpdateBehaviorName

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Aktualisierungsverhaltens

## Visible

**Eigenschaft von VcBox**

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Box sichtbar sein soll. Diese Eigenschaft können Sie auch im Dialog **Boxen verwalten** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Box sichtbar/unsichtbar <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
box.Visible = False
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Visible = false;
```

## Methoden

### GetActualExtent

**Methode von VcBox**

Mit dieser Methode können Sie die Breite und Höhe der Box erfragen (Einheit: 1/100 mm).

Werden diese Werte beim XY-Offset berücksichtigt, kann man z.B. den Referenzpunkt der Verankerungslinie ändern, ohne die Position der Box zu verändern.

	Datentyp	Beschreibung
<b>Parameter:</b>		
↔ width	System.Int32	Breite der Box
↔ height	System.Int32	Höhe der Box
<b>Rückgabewert</b>	System.Boolean	Ausdehnung der Box wird zurückgegeben/wird nicht zurückgegeben

### GetTopLeftPixel

**Methode von VcBox**

Mit dieser Methode können Sie den gespeicherten XY-Offset für die obere linke Ecke in Pixel umrechnen und ausgeben.

Der x-Wert kann dann z.B. mit der Methode **VcGantt.GetDate** weiter verwendet werden, um ein Datum zu erhalten

	Datentyp	Beschreibung
<b>Parameter:</b>		
↔ x	System.Int32	x-Wert des Offsets
↔ y	System.Int32	y-Wert des Offsets
<b>Rückgabewert</b>	System.Boolean	Offset wird zurückgegeben/nicht zurückgegeben

## GetXYOffset

Methode von VcBox

Mit dieser Methode können Sie den Abstand zwischen Ursprung und Referenzpunkt in x- und y-Richtung erfragen (Einheit: 1/100 mm).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇨ xOffset	System.Int32	x-Wert des Offsets
⇨ yOffset	System.Int32	y-Wert des Offsets
<b>Rückgabewert</b>	System.Boolean	Offset wird zurückgegeben/nicht zurückgegeben

## IdentifyFormatField

Methode von VcBox

Mit dieser Methode können Sie den Index des an der bezeichneten Position befindlichen Formatfeldes erfragen. Falls sich an der bezeichneten Position ein Feld befindet, wird **True** zurückgegeben, ansonsten **False**.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇨ x	System.Int32	X-Koordinate der Position
⇨ y	System.Int32	Y-Koordinate der Position
⇨ format	VcBoxFormat	Identifiziertes Format
⇨ formatFieldIndex	System.Int16	Format-Feldindex
<b>Rückgabewert</b>	System.Boolean	Ein Formatfeld befindet sich/befindet sich nicht an der angegebenen Position

## SetXYOffset

Methode von VcBox

Mit dieser Methode können Sie den Abstand zwischen Ursprung und Referenzpunkt in x- und y-Richtung festlegen (Einheit: 1/100 mm).

Den Offset können Sie auch im Dialog **Boxen verwalten** festlegen.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ xOffset	System.Int32	x-Wert des Offsets
⇒ yOffset	System.Int32	y-Wert des Offsets
<b>Rückgabewert</b>	System.Boolean	Offset wird gesetzt (True)/nicht gesetzt (False)

**Code-Beispiel VB.NET**

```
Dim offSet As Boolean
offSet = VcNet1.BoxCollection.FirstBox.SetXYOffset(100, 100)
```

**Code-Beispiel C#**

```
bool offSet = vcNet1.BoxCollection.FirstBox().SetXYOffset(100, 100);
```

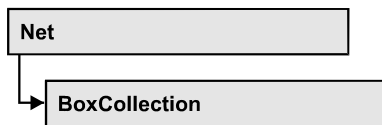
**SetXYOffsetByTopLeftPixel****Methode von VcBox**

Mit dieser Methode können Sie den angegebenen Pixelwert der oberen linken Ecke intern in einen XY-Offset umrechnen und speichern.

Damit kann man z.B. an einer XY-Koordinate aus einem Ereignis eine Box positionieren.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ x	System.Int32	x-Wert des Offsets
⇒ y	System.Int32	y-Wert des Offsets
<b>Rückgabewert</b>	System.Boolean	Offset wird gesetzt (True)/nicht gesetzt (False)

## 7.5 VcBoxCollection



Im VcBoxCollection-Objekt sind alle verfügbaren Boxen zusammengefasst. Über **For Each box In BoxCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Boxen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **BoxByName** und **BoxByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Boxen kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Boxen.

### Eigenschaften

- Count

### Methoden

- Add
- AddBySpecification
- BoxByIndex
- BoxByName
- Copy
- FirstBox
- GetEnumerator
- NextBox
- Remove
- Update

---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcBoxCollection**

Mit dieser Eigenschaft können Sie die Anzahl der Boxobjekte in der Box-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Boxen

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim numberOfBoxes As Integer

boxCltn = VcNet1.BoxCollection
numberOfBoxes = boxCltn.Count
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
int numberOfBoxes = boxCltn.Count;
```

---

## Methoden

### Add

**Methode von VcBoxCollection**

Mit dieser Methode können Sie eine neue Box in der BoxCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Boxobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Um die neu angelegte Box im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ boxName	System.String	Name der Box
<b>Rückgabewert</b>	VcBox	Neues Boxobjekt

**Code-Beispiel VB.NET**

```
newBox = VcNet1.BoxCollection.Add("box1")
```

**Code-Beispiel C#**

```
newBox = vcNet1.BoxCollection.Add("box1");
```

### AddBySpecification

**Methode von VcBoxCollection**

Mit dieser Methode können Sie eine Box über eine Box-Spezifikation erzeugen. Dies dient der Persistenz von Boxobjekten. Die Spezifikation einer



Box kann erfragt (siehe VcBox-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann die gleiche Box mit Hilfe der eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden. Um die neu angelegte Box im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ specification	System.String	Boxspezifikation
<b>Rückgabewert</b>	VcBox	Neues Boxobjekt

#### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
boxCltn.AddBySpecification(textSpecification)
boxCltn.Update()
```

#### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
boxCltn.AddBySpecification(textSpecification);
boxCltn.Update();
```

## BoxByIndex

#### Methode von VcBoxCollection

Mit dieser Methode können Sie auf eine einzelne Box über ihren Index zugreifen. Existiert kein Box-Objekt an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index der Box
<b>Rückgabewert</b>	VcBox	Ermitteltes Box-Objekt

#### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByIndex(0)
box.LineThickness = 2
```

#### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
box.LineThickness = 2;
```

## BoxByName

### Methode von VcBoxCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf eine bestimmte Box zugreifen. Existiert keine Box unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ boxName	System.String	Name der Box
<b>Rückgabewert</b>	VcBox	Box

### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByName("BoxOne")
box.LineThickness = 3
```

### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.BoxByName("BoxOne");
box.LineThickness = 3;
```

## Copy

### Methode von VcBoxCollection

Mit dieser Methode können Sie eine Box kopieren. Wenn die Box mit dem angegebenen Namen existiert und der Name der neuen Box noch nicht verwendet wird, wird das neue Boxobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Um die kopierte Box im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ boxName	System.String	Name der zu kopierenden Box
⇒ newBoxName	System.String	Name der neuen Box
<b>Rückgabewert</b>	VcBox	Boxobjekt

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
boxCltn.Copy("BoxOne", "NewBox")
boxCltn.Update()
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
boxCltn.Copy("BoxOne", "NewBox");
boxCltn.Update();
```

**FirstBox****Methode von VcBoxCollection**

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Box der Box-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextBox** über die nachfolgenden Boxen zu iterieren. Existiert keine Box in der Box-Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcBox	Erste Box

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
```

**GetEnumerator****Methode von VcBoxCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Box-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

**Code-Beispiel VB.NET**

```
Dim box As VcBox

For Each box In VcNet1.BoxCollection
    ListBox1.Items.Add(box.FormatName)
Next
```

**Code-Beispiel C#**

```
foreach (VcBox box in vcNet1.BoxCollection)
    listBox1.Items.Add(box.FormatName);
```

**NextBox****Methode von VcBoxCollection**

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Boxen des BoxCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstBox** den Initialwert erfasst haben. Sind alle Boxen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcBox	Nachfolgende Box

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox

While Not box Is Nothing
    ListBox1.Items.Add(box.Name)
    box = boxCltn.NextBox
End While
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();

while (box != null)
{
    ListBox.Items.Add(box.Name);
    box = boxCltn.NextBox();
}
```

**Remove****Methode von VcBoxCollection**

Mit dieser Methode können Sie eine Box löschen. Um das Löschen der Box im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ boxName	System.String	Name der Box
<b>Rückgabewert</b>	System.Boolean	Box gelöscht (True)/nicht gelöscht (False)

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
boxCltn.Remove(box.Name);
boxCltn.Update();
```

## Update

**Methode von VcBoxCollection**

Mit dieser Methode können Sie eine BoxCollection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

**Code-Beispiel VB.NET**

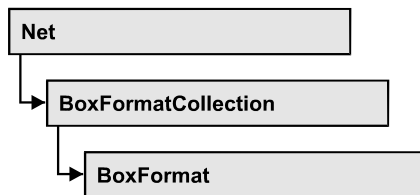
```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.BoxByIndex(0);
boxCltn.Remove(box.Name);
boxCltn.Update();
```

## 7.6 VcBoxFormat



Ein Objekt vom Typ **VcBoxFormat** beschreibt die Formate von Boxen. Über **For Each formatField In BoxFormat** können Sie in einer Schleife auf alle Boxen zugreifen.

### Eigenschaften

- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification

### Methoden

- CopyFormatField
- GetEnumerator
- RemoveFormatField

---

## Eigenschaften

### FieldsSeparatedByLines

Eigenschaft von VcBoxFormat

Mit dieser Eigenschaft können Sie festlegen, ob die Felder durch sichtbare Linien getrennt werden (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Boxfelder werden durch Linien getrennt (True)/ nicht getrennt (False).

**Code-Beispiel VB.NET**

```
Dim boxFormat As VcBoxFormat

boxFormat = VcNet1.BoxFormatCollection.FormatByIndex(0)
boxFormat.FieldsSeparatedByLines = True
```

**Code-Beispiel C#**

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FormatByIndex(0);
boxFormat.FieldsSeparatedByLines = true;
```

**FormatField****Nur-Lese-Eigenschaft von VcBoxFormat**

Mit dieser Eigenschaft können Sie ein VcBoxFormatField-Objekt per Index holen. Der Index muss im Bereich von 0 bis FormatFieldCount-1 liegen.

Die Eigenschaft FormatField ist eine indizierte Eigenschaft, die in C# über die Methode get\_FormatField (index) angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> index	System.Int16 0 ... .FormatFieldCount-1	Index des Boxformatfeldes
<b>Eigenschaftswert</b>	VcBoxFormatField	Boxformatfeld

**Code-Beispiel VB.NET**

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FirstFormat
formatField = boxFormat.FormatField(0)
MsgBox(formatField.FormatName)
```

**Code-Beispiel C#**

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FirstFormat();
VcBoxFormatField formatField = boxFormat.get_FormatField(0);
MessageBox.Show(formatField.FormatName);
```

**FormatFieldCount****Nur-Lese-Eigenschaft von VcBoxFormat**

Mit dieser Eigenschaft können Sie die Anzahl der Felder eines Boxformats ermitteln.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Felder im Boxformat

**Code-Beispiel VB.NET**

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FirstFormat
MsgBox(boxFormat.FormatFieldCount)
```

**Code-Beispiel C#**

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FirstFormat();
MessageBox.Show(boxFormat.FormatFieldCount.ToString());
```

## Name

**Eigenschaft von VcBoxFormat**

Mit dieser Eigenschaft können Sie den Namen eines Boxformats erfragen oder setzen. Diese Eigenschaft können Sie auch im Dialog **Boxformate verwalten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Boxformats

**Code-Beispiel VB.NET**

```
Dim boxFormat As VcBoxFormat

For Each boxFormat In VcNet1.BoxFormatCollection
    ListBox1.Items.Add(boxFormat.Name)
Next
```

**Code-Beispiel C#**

```
foreach (VcBoxFormat boxFormat in vcNet1.BoxFormatCollection)
    listBox1.Items.Add(boxFormat.Name);
```

## Specification

**Nur-Lese-Eigenschaft von VcBoxFormat**

Mit dieser Eigenschaft können Sie die Spezifikation dieses Boxformats auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Boxformats mit der Methode **VcBoxFormatCollection.AddBySpecification** benutzt werden.



	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Spezifikation des Boxformats

**Code-Beispiel VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormat = boxFormatCltn.FirstBoxFormat
MsgBox(boxFormat.Specification)
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat boxFormat = boxFormatCltn.FirstBoxFormat();
MessageBox.Show(boxFormat.Specification);
```

---

## Methoden

### CopyFormatField

**Methode von VcBoxFormat**

Mit dieser Methode können Sie ein Boxformatfeld kopieren. Das neue VcBoxFormatField-Objekt wird zurückgegeben. Es erhält den nächsten, noch nicht vergebenen Index.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ position	VcFormatFieldInnerPosition	Position des neuen Boxformatfeldes
	<b>Mögliche Werte:</b> .vcInnerAbove 1 .vcInnerBelow 3 .vcInnerLeftOf 0 .vcInnerRightOf 4	oberhalb unterhalb links von rechts von
⇒ refIndex	System.Int16	Index des Referenz-Boxformatfeldes
<b>Rückgabewert</b>	VcBoxFormatField	Boxformatfeld-Objekt

**Code-Beispiel VB.NET**

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FormatByIndex(2)
boxFormat.CopyFormatField(VcFormatFieldInnerPosition.vcInnerRightOf, 0)
```

**Code-Beispiel C#**

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FormatByIndex(0);
VcBoxFormatField formatField =
boxFormat.CopyFormatField(VcFormatFieldInnerPosition.vcInnerRightOf, 0);
```

**GetEnumerator****Methode von VcBoxFormat**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Boxformatfelder iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

**Code-Beispiel VB.NET**

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcNet1.BoxFormatCollection.FirstFormat
For Each formatField In boxFormat
    ListBox1.Items.Add(formatField.FormatName)
Next
```

**Code-Beispiel C#**

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FirstFormat();
foreach(VcBoxFormatField formatField in boxFormat)
    listBox1.Items.Add(formatField.FormatName);
```

**RemoveFormatField****Methode von VcBoxFormat**

Mit dieser Methode können Sie ein Boxformatfeld über den angegebenen Index löschen. Anschließend wird ggf. der Index aller Boxformatfelder neu festgesetzt, so dass sie wieder fortlaufend nummeriert sind.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des zu löschenden Boxformatfeldes

## 330 API Referenz: VcBoxFormat

### Code-Beispiel VB.NET

```
Dim boxFormat As VcBoxFormat
Dim i As Integer

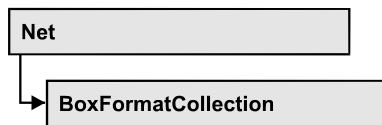
boxFormat = VcNet1.BoxFormatCollection.FirstFormat

For i = 0 To boxFormat.FormatFieldCount - 1
    boxFormat.RemoveFormatField(i)
Next
```

### Code-Beispiel C#

```
VcBoxFormat boxFormat = vcNet1.BoxFormatCollection.FirstFormat();
for (short i=0; i<boxFormat.FormatFieldCount-1; i++)
    boxFormat.RemoveFormatField(i);
```

## 7.7 VcBoxFormatCollection



Im `VcBoxFormatCollection`-Objekt sind alle verfügbaren Boxformate zusammengefasst. Über **For Each boxFormat In BoxFormatCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Formate zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **BoxFormatByName** und **BoxFormatByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Boxformate kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add...**, **Copy...** und **Remove...** ermöglichen das Hinzufügen, Kopieren und Löschen von Boxformaten.

### Eigenschaften

- Count

### Methoden

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

---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcBoxFormatCollection**

Mit dieser Eigenschaft können Sie die Anzahl der Boxformatobjekte in der BoxFormat-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Boxformate

**Code-Beispiel VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim numberOfBoxformats As Integer
```

```
boxFormatCltn = VcNet1.BoxFormatCollection
numberOfBoxformats = boxFormatCltn.Count
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
int numberOfBoxformats = boxFormatCltn.Count;
```

---

## Methoden

### Add

**Methode von VcBoxFormatCollection**

Mit dieser Methode können Sie ein neues Boxformat in der BoxFormatCollection anlegen. Wenn der Name noch nicht verwendet wird, wird das neue Boxformatobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ formatName	System.String	Name des Boxformats
<b>Rückgabewert</b>	VcBoxFormat	Neues Boxformatobjekt

**Code-Beispiel VB.NET**

```
Dim newBoxFormat = VcNet1.BoxFormatCollection.Add("boxFormat1")
```

**Code-Beispiel C#**

```
newBoxFormat = vcNet1.BoxFormatCollection.Add("boxFormat1");
```

### AddBySpecification

**Methode von VcBoxFormatCollection**

Mit dieser Methode können Sie ein Boxformat über eine Boxformat-Spezifikation erzeugen. Dies dient der Persistenz von Boxformat-Objekten. Die Spezifikation eines Boxformats kann erfragt (siehe VcBoxFormat-

Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Boxformat mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ formatSpecification	System.String	Boxformatspezifikation
<b>Rückgabewert</b>	VcBoxFormat	Neues Boxformatobjekt

## Copy

### Methode von VcBoxFormatCollection

Mit dieser Methode können Sie ein Boxformat kopieren. Wenn das Boxformat mit dem angegebenen Namen existiert und der Name des neuen Boxformats noch nicht verwendet wird, wird das neue Boxformatobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ FormatName	System.String	Name des zu kopierenden Boxformats
⇒ newFormatName	System.String	Name des neuen Boxformats
<b>Rückgabewert</b>	VcBoxFormat	Boxformatobjekt

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat")
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat");
```

## FirstFormat

### Methode von VcBoxFormatCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Boxformat des BoxFormatCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextFormat** über die nachfolgenden Boxformate

zu iterieren. Existiert kein Boxformat im BoxFormatCollection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcBoxFormat	Erstes Boxformat

#### Code-Beispiel VB.NET

```
Dim format As VcBoxFormat

format = VcNet1.BoxFormatCollection.FirstFormat
```

#### Code-Beispiel C#

```
VcBoxFormat format = vcNet1.BoxFormatCollection.FirstFormat();
```

## FormatByIndex

### Methode von VcBoxFormatCollection

Mit dieser Methode können Sie auf ein einzelnes Boxformat über ihren Index zugreifen. Existiert kein BoxFormat-Objekt an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Boxformats
<b>Rückgabewert</b>	VcBoxFormat	Ermitteltes Boxformat-Objekt

#### Code-Beispiel VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcNet1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByIndex(2)
```

#### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat format = boxFormatCltn.FormatByIndex(2);
```

## FormatByName

### Methode von VcBoxFormatCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Boxformat zugreifen. Existiert kein Boxformat unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ formatName	System.String	Name des Boxformats
<b>Rückgabewert</b>	VcBoxFormat	Boxformat

**Code-Beispiel VB.NET**

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcNet1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByName("Standard")
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat format = boxFormatCltn.FormatByName("Standard");
```

**GetEnumerator****Methode von VcBoxFormatCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Boxformat-Objekte iterieren.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcObject	Referenzobjekt

**Code-Beispiel VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
    ListBox1.Items.Add(boxFormat.Name)
Next
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
foreach (VcBoxFormat boxFormat in boxFormatCltn)
    listBox1.Items.Add(boxFormat.Name);
```

**NextFormat****Methode von VcBoxFormatCollection**

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Boxformate des BoxFormatCollection-Objekts zugreifen, nachdem Sie mit



der Methode **FirstFormat** den Initialwert erfasst haben. Sind alle Formate durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcBoxFormat	Folgendes Boxformat

### Code-Beispiel VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcNet1.BoxFormatCollection
formatBox = formatBoxCltn.FirstFormat

While Not formatBox Is Nothing
    ListBox1.Items.Add(formatBox.Name)
    formatBox = formatBoxCltn.NextFormat
End While
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormat boxFormat = boxFormatCltn.FirstFormat();

while (boxFormat != null)
{
    ListBox.Items.Add(boxFormat.Name);
    boxFormat = boxFormatCltn.NextFormat();
}
```

## Remove

### Methode von VcBoxFormatCollection

Mit dieser Methode können Sie ein Boxformat löschen. Wenn das Boxformat noch in irgendeinem anderen Objekt benutzt wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ FormatName	System.String	Name des Boxformats
<b>Rückgabewert</b>	System.Boolean	Boxformat gelöscht (True)/nicht gelöscht (False)

### Code-Beispiel VB.NET

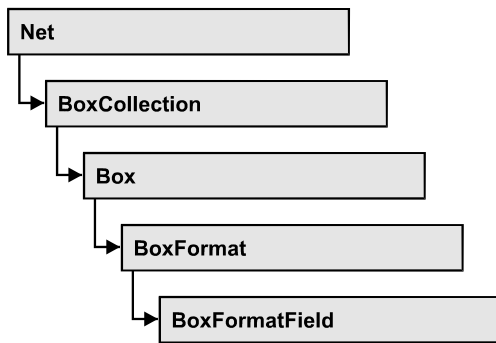
```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormat = boxFormatCltn.FormatByIndex(1)
boxFormatCltn.Remove(boxFormat.Name)
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;  
VcBoxFormat boxFormat = boxFormatCltn.FormatByIndex(1);  
boxFormatCltn.Remove(boxFormat.Name);
```

## 7.8 VcBoxFormatField



Ein Objekt vom Typ **VcBoxFormatField** stellt ein Boxformatfeld, also ein Feld eines VcBoxFormat-Objekts dar. Ein Boxformatfeld besitzt im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, unter dem es im Boxformat untergebracht ist.

### Eigenschaften

- Alignment
- FormatName
- GraphicsHeight
- Index
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColor
- PatternColorAsARGB
- PatternEx
- TextFont
- TextFontColor
- Type

---

## Eigenschaften

### Alignment

**Eigenschaft von VcBoxFormatField**

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Boxformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFormatFieldAlignment	Ausrichtung des Feldinhalts
	<b>Mögliche Werte:</b> .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	 unten unten links unten rechts unten mittig links rechts oben oben links oben rechts

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter;
```

## FormatName

### Nur-Lese-Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie den Namen des Boxformats erfragen, zu dem dieses Boxformatfeld gehört.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Boxformats

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.FormatName)
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
MessageBox.Show(boxFormatField.FormatName);
```

## GraphicsHeight

Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** die Höhe der Grafik in dem Boxformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16 0 ... 99	Höhe der Grafik in mm 0...200

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics;
boxFormatField.GraphicsHeight = 150;
```

## Index

Nur-Lese-Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie den Index des Boxformatfelds im zugehörigen Boxformat erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Index des Boxformatfeldes

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.Index)
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
MessageBox.Show(boxFormatField.Index.ToString());
```

## MaximumTextLineCount

Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie die maximale Anzahl der Zeilen in dem Boxformatfeld setzen oder erfragen, falls das Boxformatfeld vom Typ **vcFFText** ist. Bitte sehen Sie auch die Eigenschaft **MinimumTextLineCount**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0 ... 9	Maximale Zeilenzahl

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFText
boxFormatField.MaximumTextLineCount = 5
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFText;
boxFormatField.MaximumTextLineCount = 5;
```

## MinimumTextLineCount

Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie die minimale Anzahl der Zeilen in dem Boxformatfeld setzen oder erfragen, falls der Typ des Boxformatfeldes auf **vcFFText** gesetzt wurde. Ist in einem Knoten mehr Text vorhanden, als in die minimale Anzahl der Zeilen hineinpasst, wird dieses Feld für diesen Knoten dynamisch bis zur maximalen angegebenen Anzahl der Zeilen ausgedehnt. Wenn Sie dieser Eigenschaft einen Wert zuweisen, sollten Sie anschließend auch erneut der Eigenschaft **MaximumTextLineCount** den gewünschten Wert setzen, sonst könnte es vorkommen, dass das Maximum durch das Minimum überschrieben wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0 ... 9	Minimale Zeilenzahl 0...20

**Code-Beispiel VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTTText
boxFormatField.MinimumTextLineCount = 3
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTTText;
boxFormatField.MinimumTextLineCount = 3;
```

## MinimumWidth

**Eigenschaft von VcBoxFormatField**

Mit dieser Eigenschaft können Sie die minimale Breite des Boxformatfeldes in mm festlegen oder erfragen. Die Breite des Feldes kann sich vergrößern, wenn unter oder über dem Feld andere Felder größere minimale Breiten besitzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16 0 ... 9	Minimale Breite des Boxformatfeldes 0...200

**Code-Beispiel VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.MinimumWidth = 100
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.MinimumWidth = 100;
```

## PatternBackgroundColor

**Eigenschaft von VcBoxFormatField**

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Boxformatfeldes festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255

(ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt.

Wählen Sie den Wert **-1**, wenn das Feld die Hintergrundfarbe des Boxformats besitzen soll.

	Datentyp	Beschreibung

#### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.BackgroundColor = Color.Red
```

#### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.BackgroundColor = Color.Red;
```

## PatternColorAsARGB

### Nur-Lese-Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie die Musterfarbe des Boxformatfeldes festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt. Bei der Umwandlung eines RGB-Wertes in einen ARGB-Wert muss ein Alpha-Wert von 255 hinzugegeben werden.

Wählen Sie den Wert **-1**, wenn das Feld die Hintergrundfarbe des Boxformats besitzen soll.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte {0...255},{0...255},{0...255}) <b>Standardwert:</b> -1








**Code-Beispiel VB.NET**

```
boxFormatField.PatternColor = RGB(0, 255, 0)
```

**PatternEx****Eigenschaft von VcBoxFormatField**

Mit dieser Eigenschaft können Sie für den Hintergrund des Boxformatfeldes ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFieldFillPattern	Mustertyp
	<b>Mögliche Werte:</b> .vcAeroGlassPattern 44	Vertikaler Farbverlauf in der Füllmusterfarbe 
	.vcFieldNoPattern 1276	Kein Füllmuster
	.vcFieldVerticalBottomLightedConvexPattern 43	Vertikaler Farbverlauf von dunkel nach hell 
	.vcFieldVerticalConcavePattern 40	Vertikaler Farbverlauf von dunkel über hell nach dunkel 
	.vcFieldVerticalConvexPattern 41	Vertikaler Farbverlauf von hell über dunkel nach hell 
	.vcFieldVerticalTopLightedConvexPattern 42	Vertikaler Farbverlauf von hell nach dunkel 

**Code-Beispiel VB.NET**

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Pattern = VcFillPatternSingleColored.vcSingleColoredNoPatter
```

**Code-Beispiel C#**

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Pattern = VcFillPatternSingleColored.vcSingleColoredNoPattern;
```

## TextFont

### Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie die Schriftart des Boxformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFTText** gesetzt wurde.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Font	Schriftart des Boxformatfelds

#### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.TextFont.FontFamily.ToString())
```

#### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
MessageBox.Show(boxFormatField.TextFont.Name.ToString());
```

## TextFontColor

### Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie die Schriftfarbe des Boxformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFTText** gesetzt wurde.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	Schriftfarbe des Boxformatfelds <b>Standardwert:</b> Color.Black

#### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFontColor = Color.Red
```

#### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.TextFontColor = Color.Red;
```

## Type

### Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie den Typ des Boxformatfelds erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFormatFieldType  <b>Mögliche Werte:</b> .vcFFTGraphics 64 .vcFFTText 36	Typ des Boxformatfeldes  Grafik Text

### Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcNet1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

### Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcNet1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics;
boxFormatField.GraphicsHeight = 150;
```

## 7.9 VcCalendar



Ein Kalender dient der Definition von Arbeits- und Nichtarbeitszeiten. Er wird durch eine lückenlose Aneinanderreihung von Arbeitszeiten und Nichtarbeitszeiten zusammengesetzt. Diese werden in der Regel durch Arbeitstage und Arbeitswochen (Workday- bzw. Workweek-Objekte) dargestellt, aber auch Intervalle sind möglich.

Ein neu angelegter Kalender enthält standardmäßig ein Arbeitszeitintervall, das den Projektzeitraum umfasst.

Sie können einen Kalender für Zeitberechnungen verwenden, z. B. um den Zeitunterschied zwischen zwei verschiedenen Terminen in Arbeitstagen zu ermitteln.

### Eigenschaften

- CalendarProfileCollection
- IntervalCollection
- Name
- SecondsPerWorkday
- Specification

### Methoden

- AddDuration
- CalcDuration
- Clear
- GetEndOfPreviousWorktime
- GetNextIntervalBorder
- GetPreviousIntervalBorder
- GetStartOfInterval
- GetStartOfNextWorktime
- IsWorktime
- Update

## Eigenschaften

### CalendarProfileCollection

Nur-Lese-Eigenschaft von VcCalendar

Mit dieser Eigenschaft haben Sie Zugriff auf die Kalenderprofilauflistung, in der alle zur Verfügung stehenden Kalenderprofile dieses Kalenderobjektes enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendarProfileCollection	CalendarProfileCollection-Objekt

### IntervalCollection

Nur-Lese-Eigenschaft von VcCalendar

Mit dieser Eigenschaft haben Sie Zugriff auf die Intervallaufistung, in der alle zur Verfügung stehenden Intervalle enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcIntervalCollection	IntervalCollection-Objekt

### Name

Nur-Lese-Eigenschaft von VcCalendar

Mit dieser Eigenschaft können Sie den Namen eines Kalenders erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalenders

#### Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim calendarName As String

calendar = VcNet1.CalendarCollection.FirstCalendar
calendarName = calendar.Name
```

#### Code-Beispiel C#

```
VcCalendar calendar = vcNet1.CalendarCollection.FirstCalendar();
string calendarName = calendar.Name;
```

## SecondsPerWorkday

### Nur-Lese-Eigenschaft von VcCalendar

Mit dieser Eigenschaft lässt sich die Anzahl der Sekunden eines Arbeitstag setzen oder erfragen. Die Option kann auch im Dialog **Kalender festlegen** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Sekunden eines Arbeitstages

## Specification

### Nur-Lese-Eigenschaft von VcCalendar

Mit dieser Eigenschaft können Sie die Spezifikation dieses Kalenders erfragen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Kalenders mit der Methode **VcCalendarCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Kalenders

### Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.FirstCalendar
MsgBox(calendar.Specification)
```

### Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();
MessageBox.Show(calendar.Specification);
```

## Methoden

### AddDuration

Methode von VcCalendar

Mit dieser Methode können Sie zu einem Termin eine Dauer addieren, um so unter Berücksichtigung der Kalenderdefinition einen neuen Termin zu berechnen. Wenn Sie beispielsweise zu einem Freitag die Dauer von drei Arbeitstagen hinzufügen, würde der neue Termin - bei arbeitsfreien Wochenenden - den darauffolgenden Mittwoch ergeben.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ date	System.DateTime	Datum, an dem die Dauer eingefügt werden soll
⇒ duration	System.Int32	Anzahl der Zeiteinheiten (z.B. Tage) der Dauer
<b>Rückgabewert</b>	System.DateTime	Datum, an dem die Dauer eingefügt wurde

#### Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim newDate As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
newDate = calendar.AddDuration("16.06.2017", 3)
```

#### Code-Beispiel C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime newDate = calendar.AddDuration(Convert.ToDateTime("16.06.2017"), 3);
```

### CalcDuration

Methode von VcCalendar

Mit dieser Methode können Sie die Anzahl der Arbeitszeitelemente (z. B. Arbeitstage) berechnen, die zwischen zwei Terminen liegen. Die Einheit (z. B. Tage) des berechneten Wertes entspricht der auf der Eigenschaftenseite **Allgemeines** festgelegten Zeiteinheit im gleichnamigen Feld.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ fromDate	System.DateTime	Anfangsdatum der Zeitdauer, aus der die Anzahl der Arbeitszeitelemente ermittelt werden soll
⇒ toDate	System.DateTime	Enddatum der Zeitdauer, aus der die Anzahl der Arbeitszeitelemente ermittelt werden soll

<b>Rückgabewert</b>	System.Int32	Anzahl der Zeiteinheiten (z.B. Tage) der Dauer
---------------------	--------------	--

**Code-Beispiel VB.NET**

```
Dim calendar As VcCalendar
Dim duration As Integer

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
duration = calendar.CalcDuration("01.01.2014", "31.12.2014")
```

**Code-Beispiel C#**

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
int duration = calendar.CalcDuration(Convert.ToDateTime("01.01.2014"),
Convert.ToDateTime("31.12.2014"));
```

**Clear****Methode von VcCalendar**

Diese Methode löscht alle Profile und Intervalle aus dem Kalender, wodurch er komplett geleert wird (=>100% Arbeitszeit). Damit sich die Änderungen auch optisch zeigen (also z.B. bei Kalendergittern), muss noch ein Update aufgerufen werden.

	Datentyp	Beschreibung

**GetEndOfPreviousWorktime****Methode von VcCalendar**

Mit dieser Methode können Sie das Ende der dem Referenzdatum vorangehenden Arbeitsperiode erfragen. Dabei muss das Referenzdatum in einer Nichtarbeitszeit liegen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ date	System.DateTime	Referenzdatum, zu dem die vorangehende Arbeitszeit ermittelt werden soll
<b>Rückgabewert</b>	System.DateTime	Enddatum der vorangegangenen Arbeitszeit



**Code-Beispiel VB.NET**

```
Dim calendar As VcCalendar
Dim endOfWork As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
endOfWork = calendar.GetEndOfPreviousWorktime("18.06.2014")
```

**Code-Beispiel C#**

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime endOfWork =
calendar.GetEndOfPreviousWorktime(Convert.ToDateTime("18.06.2014"));
```

**GetNextIntervalBorder****Methode von VcCalendar**

Mit dieser Methode können Sie den Anfangstermin des dem Referenzdatum nachfolgenden Intervalls erfragen. Wenn das Intervall, in dem das Referenzdatum liegt, eine Nichtarbeitszeit war, ist das zurückgegebene Datum der Anfang der folgenden Arbeitszeit, und umgekehrt.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ date	System.DateTime	Referenzdatum, zu dem das Anfangsdatum der folgenden Intervallgrenze ermittelt werden soll
<b>Rückgabewert</b>	System.DateTime	Anfangsdatum der folgenden Intervallgrenze

**Code-Beispiel VB.NET**

```
Dim calendar As VcCalendar
Dim nextIntervalBorder As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
nextIntervalBorder = calendar.GetNextIntervalBorder("18.06.2014")
```

**Code-Beispiel C#**

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime nextIntervalBorder =
calendar.GetNextIntervalBorder(Convert.ToDateTime("18.06.2014"));
```

**GetPreviousIntervalBorder****Methode von VcCalendar**

Mit dieser Methode können Sie den Endtermin des dem Referenzdatum vorangehenden Intervalls erfragen. Wenn das Intervall, in dem das Referenzdatum liegt, eine Nichtarbeitszeit war, ist das zurückgegebene Datum das Ende der vorausgehenden Arbeitszeit, und umgekehrt.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ date	System.DateTime	Referenzdatum, zu dem das Enddatum der vorausgehenden Intervallgrenze ermittelt werden soll
<b>Rückgabewert</b>	System.DateTime	Enddatum der vorausgehenden Intervallgrenze

**Code-Beispiel VB.NET**

```
Dim calendar As VcCalendar
Dim previousIntervalBorder As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
previousIntervalBorder = calendar.GetPreviousIntervalBorder("18.06.2014")
```

**Code-Beispiel C#**

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime previousIntervalBorder =
calendar.GetPreviousIntervalBorder(Convert.ToDateTime("18.06.2014"));
```

**GetStartOfInterval****Methode von VcCalendar**

Mit dieser Methode können Sie den Anfang des Intervalls erfragen, in dem das Referenzdatum liegt.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ date	System.DateTime	Referenzdatum des Intervalls, zu dem das Anfangsdatum ermittelt werden soll
<b>Rückgabewert</b>	System.DateTime	Anfangsdatum des Intervalls

**Code-Beispiel VB.NET**

```
Dim calendar As VcCalendar
Dim startOfInterval As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
startOfInterval = calendar.GetStartOfInterval("18.06.2014")
```

**Code-Beispiel C#**

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime startOfInterval =
calendar.GetStartOfInterval(Convert.ToDateTime("18.06.2014"));
```

## GetStartOfNextWorktime

Methode von VcCalendar

Mit dieser Methode können Sie den Anfang der dem Referenzdatum nachfolgenden Arbeitszeit erfragen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ date	System.DateTime	Referenzdatum, zu dem das Anfangsdatum der folgenden Arbeitszeit ermittelt werden soll
<b>Rückgabewert</b>	System.DateTime	Anfangsdatum der folgenden Arbeitszeit

### Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim startOfNextWorktime As Date

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
startOfNextWorktime = calendar.GetStartOfNextWorktime("18.06.2017")
```

### Code-Beispiel C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
DateTime startOfNextWorktime =
calendar.GetStartOfNextWorktime(Convert.ToDateTime("18.06.2017"));
```

## IsWorktime

Methode von VcCalendar

Mit dieser Methode können Sie erfragen, ob sich das übergebene Datum in einer Arbeitszeit befindet.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ date	System.DateTime	Datum, das darauf geprüft werden soll, ob es in einer Arbeitszeit liegt
<b>Rückgabewert</b>	System.Boolean	Übergebenes Datum fällt/fällt nicht in eine Arbeitszeit

### Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim isWorktime As Boolean

calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")
isWorktime = calendar.IsWorktime("18.06.2014")
```

### Code-Beispiel C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");
bool isWorktime = calendar.IsWorktime(Convert.ToDateTime("18.06.2014"));
```

## Update

### Methode von VcCalendar

Mit dieser Methode können Sie einen Kalender aktualisieren, nachdem Sie ihn verändert haben. Diese Methode stellt sicher, dass alle Objekte, die den Kalender verwenden (z. B. das Raster), ebenfalls aktualisiert werden.

	Datentyp	Beschreibung
Rückgabewert	Void	

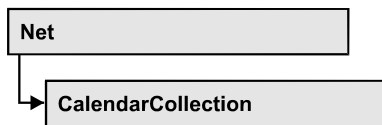
### Code-Beispiel VB.NET

```
Dim calendar As VcCalendar  
  
calendar = VcNet1.CalendarCollection.CalendarByName("WeekCalendar")  
calendar.Update()
```

### Code-Beispiel C#

```
VcCalendar calendar = vcNet1.CalendarCollection.CalendarByName("WeekCalendar");  
calendar.Update();
```

## 7.10 VcCalendarCollection



Im CalendarCollection-Objekt sind alle mit der Methode **CreateCalendar** angelegten Kalender zusammengefasst. Über **For Each calendar In CalendarCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Kalender zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **CalendarByName** und **CalendarByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Kalender kann über die Eigenschaft **Count** erfragt werden. Mit der Eigenschaft **Active** können Sie den dem Kalendergitter zu Grunde liegenden Kalender erfragen.

### Eigenschaften

- Active
- Count

### Methoden

- Add
- AddBySpecification
- CalendarByIndex
- CalendarByName
- Copy
- FirstCalendar
- GetEnumerator
- NextCalendar
- Remove
- Update

---

## Eigenschaften

### Active

**Eigenschaft von VcCalendarCollection**

Mit dieser Eigenschaft kann der aktuelle Standardkalender erfragt oder gesetzt werden, der für Vorgänge verwendet wird, wenn kein anderer Kalender zugewiesen wurde.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendar	Aktuell verwendeter Kalender

### Code-Beispiel VB.NET

```

Dim workday As VcWorkday
Dim freeday As VcWorkday
Dim workweek As VcWorkweek
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

workday = VcNet1.WorkdayCollection.CreateWorkday("Work day")
workday.AddNonWorkInterval("00:00:00", "00:00:00")
workday.AddWorkInterval("08:00:00", "16:30:00")

freeday = VcNet1.WorkdayCollection.CreateWorkday("Workfree day")
freeday.AddNonWorkInterval("00:00:00", "00:00:00")

calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.CreateCalendar("New calendar")

workweek = VcNet1.WorkweekCollection.CreateWorkweek("Work week")
workweek.AddWorkday(workday, VcWeekday.vcMonday, VcWeekday.vcFriday)
workweek.AddWorkday(freeday, VcWeekday.vcSaturday, VcWeekday.vcSunday)

calendar.AddWorkweek(workweek, "01.01.13", "31.12.14")

calendar.Update()
calendarCltn.Active = calendar

```

### Code-Beispiel C#

```

VcWorkday workday = vcNet1.WorkdayCollection.CreateWorkday("Work day");
workday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));
workday.AddWorkInterval(Convert.ToDateTime("08:00:00"),
Convert.ToDateTime("16:30:00"));

VcWorkday freeday = vcNet1.WorkdayCollection.CreateWorkday("Workfree day");
freeday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));

VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
VcCalendar calendar = calendarCltn.CreateCalendar("New calendar");

VcWorkweek workweek = vcNet1.WorkweekCollection.CreateWorkweek("Work week");
workweek.AddWorkday(workday, VcWeekday.vcMonday, VcWeekday.vcFriday);
workweek.AddWorkday(freeday, VcWeekday.vcSaturday, VcWeekday.vcSunday);

calendar.AddWorkweek(workweek, Convert.ToDateTime("01.01.13"),
Convert.ToDateTime("31.12.14"));
calendar.Update();
calendarCltn.Active = calendar;

```

## Count

### Nur-Lese-Eigenschaft von VcCalendarCollection

Mit dieser Eigenschaft kann die Anzahl der Kalender in der Kalender-Auflistung abgefragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Kalender

**Code-Beispiel VB.NET**

```
Dim calendarCltn As VcCalendarCollection
Dim numberOfCalendar As Integer

calendarCltn = VcNet1.CalendarCollection
numberOfCalendar = calendarCltn.Count
```

**Code-Beispiel C#**

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
int numberOfCalendar = calendarCltn.Count;
```

---

## Methoden

### Add

**Methode von VcCalendarCollection**

Mit dieser Methode können Sie einen neuen Kalender in der FilterCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Kalenderobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ calendarName	System.String	Name des Kalenders
<b>Rückgabewert</b>	VcCalendar	Neues Kalenderobjekt

### AddBySpecification

**Methode von VcCalendarCollection**

Mit dieser Methode können Sie einen Kalender über eine Kalender-Spezifikation erzeugen. Dies dient der Persistenz von Kalenderobjekten. Die Spezifikation eines Kalenderobjektes kann erfragt (siehe VcCalendar-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann der gleiche Kalender mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Specification	System.String	Kalenderspezifikation
<b>Rückgabewert</b>	VcCalendar	Neues Kalenderobjekt

## CalendarByIndex

### Methode von VcCalendarCollection

Mit dieser Methode können Sie auf einen einzelnen Kalender über seinen Index zugreifen. Existiert kein Kalender an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Kalenders
<b>Rückgabewert</b>	VcCalendar	Ermitteltes Kalenderobjekt

## CalendarByName

### Methode von VcCalendarCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf einen bestimmten Kalender zugreifen. Existiert kein Kalender unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ calendarName	System.String	Name des Kalenders
<b>Rückgabewert</b>	VcCalendar	Kalender

### Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
calendarCltn = VcNet1.CalendarCollection
calendarCltn.Active = calendarCltn.CalendarByName("Calendar_1")
```

### Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
calendarCltn.Active = calendarCltn.CalendarByName("Calendar_1");
```



## Copy

### Methode von VcCalendarCollection

Mit dieser Methode können Sie einen Kalender kopieren. Wenn der Kalender mit dem angegebenen Namen existiert und der Name des neuen Kalenders noch nicht verwendet wird, wird das neue Kalenderobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ calendarName	System.String	Name des zu kopierenden Kalenders
⇒ newCalendarName	System.String	Name des neuen Kalenders
<b>Rückgabewert</b>	VcCalendar	Kalenderobjekt

## FirstCalendar

### Methode von VcCalendarCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Kalender der Kalenderauflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextCalendar** über die nachfolgenden Kalender zu iterieren. Existiert kein Kalender in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcCalendar	Erster Kalender

### Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar
```

```
calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.FirstCalendar
```

### Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();
```

## GetEnumerator

### Methode von VcCalendarCollection

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C#

wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Kalender-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

#### Code-Beispiel VB.NET

```
Dim calendar As VcCalendar

For Each calendar In VcNet1.CalendarCollection
    MsgBox(calendar.Name)
Next
```

#### Code-Beispiel C#

```
foreach (VcCalendar calendar in vcNet1.CalendarCollection)
    MessageBox.Show(calendar.Name);
```

## NextCalendar

### Methode von VcCalendarCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Kalender des CalendarCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstCalendar** den Initialwert erfasst haben. Sind alle Kalender durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcCalendar	Nachfolgender Kalender

#### Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar
calendarCltn = VcNet1.CalendarCollection
calendar = calendarCltn.FirstCalendar

While Not calendar Is Nothing
    ListBox1.Items.Add(calendar.Name)
    calendar = calendarCltn.NextCalendar
End While
```

#### Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();

while (calendar != null)
{
    ListBox.Items.Add(calendar.Name);
    calendar = calendarCltn.NextCalendar();
}
```

## Remove

### Methode von VcCalendarCollection

Mit dieser Methode können Sie einen Kalender löschen. Wenn der Kalender noch in irgendeinem anderen Objekt verwendet wird, kann er nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Kalender gelöscht (True)/nicht gelöscht (False)

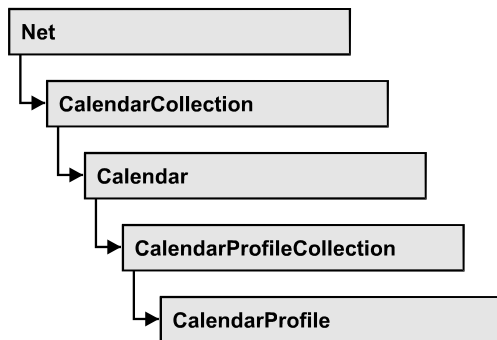
## Update

### Methode von VcCalendarCollection

Mit dieser Methode können Sie eine Kalender-Collection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

## 7.11 VcCalendarProfile



In einem Objekt vom Typ **VcCalendarProfile** legen Sie ein Kalenderprofil fest.

### Eigenschaften

- IntervalCollection
- Name
- Specification
- Type

### Methoden

- PutInOrderAfter

---

## Eigenschaften

### IntervalCollection

**Nur-Lese-Eigenschaft von VcCalendarProfile**

Mit dieser Eigenschaft haben Sie Zugriff auf die Intervallauflistung, in der alle zur Verfügung stehenden Intervalle enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcIntervalCollection	IntervalCollection-Objekt

## Name

**Nur-Lese-Eigenschaft von VcCalendarProfile**

Mit dieser Eigenschaft können Sie den Namen eines Kalenderprofils setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalenderprofils

## Specification

**Nur-Lese-Eigenschaft von VcCalendarProfile**

Mit dieser Eigenschaft können Sie die Spezifikation dieses Kalenderprofils erfragen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Kalenderprofils mit der Methode **VcCalendarProfileCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Kalenderprofils

### Code-Beispiel VB.NET

```
Dim calendarProfileCltn As VcCalendarProfileCollection
Dim calendarProfile As VcCalendarProfile

calendarProfileCltn = VcNet1.CalendarProfileCollection
calendarProfile = calendarProfileCltn.FirstCalendarProfile
MsgBox(calendarProfile.Specification)
```

### Code-Beispiel C#

```
VcCalendarProfileCollection calendarProfileCltn =
vcNet1.CalendarProfileCollection;
VcCalendarProfile calendar = calendarProfileCltn.FirstCalendarProfile();
MessageBox.Show(calendarProfile.Specification);
```

## Type

**Nur-Lese-Eigenschaft von VcCalendarProfile**

Mit dieser Eigenschaft können Sie den Typ des Kalenderprofils setzen oder erfragen. Wenn Sie den Typ ändern, gehen alle für das Kalenderprofil gesetzten Eigenschaften verloren.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendarProfileType	Typ des Kalenderprofils

## Methoden

### PutInOrderAfter

Methode von VcCalendarProfile

Mit dieser Methode können Sie dieses Kalenderprofil in der Auflistung aller Kalenderprofile hinter das durch den Namen angegebene setzen. Wenn als Name "" angegeben wird, wird das Kalenderprofil an die erste Stelle gesetzt. Die Reihenfolge der Kalenderprofile in der Auflistung entscheidet darüber, in welcher Reihenfolge sie in Kalendern angewendet werden.

	Datentyp	Beschreibung
<b>Parameter:</b> refNameParam	String	Name des Kalenderprofils, hinter das das aktuelle Kalenderprofil gesetzt werden soll

#### Code-Beispiel VB.NET

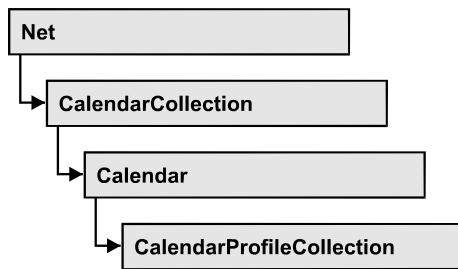
```
Dim calProfCltn As VcCalendarProfileCollection
Dim calProf1 As VcCalendarProfile
Dim calProf2 As VcCalendarProfile

calProfCltn = VcGantt1.CalendarProfileCollection()
calProf1 = calProfCltn.Add("calProf1")
calProf2 = calProfCltn.Add("calProf2")
calProf1.PutInOrderAfter("calProf2")
calProfCltn.Update()
```

#### Code-Beispiel C#

```
VcCalendar ProfileCollection calProfCltn = vcGantt1.Calendar ProfileCollection;
VcCalendar Profile calProf1 = calProfCltn.Add("calProf1");
VcCalendar Profile calProf2 = calProfCltn.Add("calProf2");
calProf1.PutInOrderAfter("calProf2");
calProfCltn.Update();
```

## 7.12 VcCalendarProfileCollection



In einem Objekt vom Typ `VcCalendarProfileCollection` sind automatisch alle verfügbaren Kalenderprofile zusammengefasst. Über **For Each calendar-Profile In CalendarProfileCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Kalenderprofile zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **CalendarProfileByName** und **CalendarProfileByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Kalenderprofile kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Kalenderprofilen.

### Eigenschaften

- Count

### Methoden

- Add
- AddBySpecification
- CalendarProfileByIndex
- CalendarProfileByName
- Copy
- FirstCalendarProfile
- NextCalendarProfile
- Remove
- SelectCalendarProfiles
- Update
- Update

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von VcCalendarProfileCollection

Mit dieser Eigenschaft können Sie die Anzahl der Kalenderprofilobjekte in der Kalenderprofil-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der CalendarProfile-Objekte

## Methoden

### Add

Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie ein neues Kalenderprofil in der FilterCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Kalenderprofilobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ profileName	System.String	Name des Kalenderprofils
<b>Rückgabewert</b>	VcCalendarProfile	Neues Kalenderprofilobjekt

### AddBySpecification

Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie ein Kalenderprofil über eine Kalenderprofil-Spezifikation erzeugen. Dies dient der Persistenz von Kalenderprofilobjekten. Die Spezifikation eines Kalenderprofilobjektes kann erfragt (siehe VcCalendarProfile-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Kalenderprofil mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.



	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Specification	System.String	Kalenderprofilspezifikation
<b>Rückgabewert</b>	VcCalendarProfile	Neues Kalenderprofilobjekt

## CalendarProfileByIndex

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie auf ein einzelnes Kalenderprofil über seinen Index zugreifen. Existiert kein Kalenderprofil unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Kalenderprofils
<b>Rückgabewert</b>	VcCalendarProfile	Ermitteltes CalendarProfile-Objekt

## CalendarProfileByName

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Kalenderprofil zugreifen. Existiert kein Kalenderprofil unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ profileName	System.String	Name des CalendarProfile-Objekts
<b>Rückgabewert</b>	VcCalendarProfile	Zurückgegebenes CalendarProfile-Objekt

## Copy

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie ein Kalenderprofil kopieren. Wenn das Kalenderprofil mit dem angegebenen Namen existiert und der Name des neuen Kalenderprofils noch nicht verwendet wird, wird das neue

Kalenderprofilobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ profileName	System.String	Name des zu kopierenden Kalenderprofils
⇒ newProfileName	System.String	Name des neuen Kalenderprofils
<b>Rückgabewert</b>	VcCalendarProfile	CalendarProfile-Objekt

## FirstCalendarProfile

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie auf den Initialwert, d. h. das erste Kalenderprofil der Kalenderprofil-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextCalendarProfile** über die nachfolgenden Kalenderprofile zu iterieren. Existiert kein Kalenderprofil in der Auflistung, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcCalendarProfile	Erstes CalendarProfile-Objekt

## NextCalendarProfile

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Kalenderprofile der Kalenderprofil-Auflistung zugreifen, nachdem Sie mit der Methode **FirstCalendarProfile** den Initialwert erfasst haben. Sind alle Kalenderprofile durchlaufen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcCalendarProfile	Nachfolgendes CalendarProfile-Objekt

## Remove

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie ein Kalenderprofil löschen. Wenn das Kalenderprofil noch in irgendeinem anderen Objekt verwendet wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ profileName	System.String	Name des Kalenderprofils
<b>Rückgabewert</b>	System.Boolean	Kalenderprofil gelöscht (True)/nicht gelöscht (False)

## SelectCalendarProfiles

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie festlegen, welche Kalenderprofile in der Auflistung der Kalenderprofile verfügbar sein sollen.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ selectionType	CalendarProfileTypeEnum	Auszuwählender Kalenderprofiltyp
<b>Rückgabewert</b>	System.Int32	Anzahl der ausgewählten Kalenderprofile

### Code-Beispiel VB.NET

```
Dim calendarProfileCltn As VcCalendarProfileCollection

Set calendarProfileCltn = VcNet1.CalendarProfileCollection
calendarProfileCltn.SelectCalendarProfile (vcSelected)
```

## Update

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie eine Kalenderprofil-Collection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

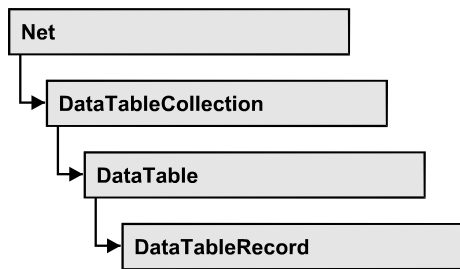
## Update

### Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie eine CalendarProfileCollection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

## 7.13 VcDataRecord



Ein Datensatz ist das logische Grundelement eines Objektes in einem Net-Diagramm, z. B. eines Knotens, eines Gruppenknotens, einer Verbindung, einer Aufgabe, Operation etc. Die Objekte besitzen spezifische Eigenschaften, die in den Feldern des Datensatzes beschrieben werden. Zu den Datenfeldern des Datensatzes existieren entsprechende Beschreibungen, die Datentabellenfelder. Datensätze und Datentabellenfelder werden jeweils zu Collection-Objekten zusammengefasst und bilden eine Datentabelle.

### Eigenschaften

- AllData
- DataField
- DataTableName
- ID

### Methoden

- Delete
- IdentifyObject
- RelatedDataRecord

---

## Eigenschaften

### AllData

**Eigenschaft von VcDataRecord**

Mit dieser Eigenschaft können alle Daten eines Datensatzes gesetzt oder erfragt werden. Beim Setzen ist ein CSV-String (Semikolon als Trennzeichen) oder der Datentyp "Object" erlaubt, der in einem Array alle Datenfelder des Knotens erhält. Beim Erfragen wird eine Zeichenkette (String) zurückgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.Object	Alle Daten des Datensatzes

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRecValue() As Object
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata1")
dataRecCltn = dataTable.DataRecordCollection
ReDim dataRecValue(dataTable.DataTableFieldCollection.Count)
dataRecValue(0) = 1
dataRecValue(1) = "Node One"

'Object
dataRecord = dataRecCltn.Add(dataRecValue)
'CSV
dataRecord.AllData = "1;Node One;"

dataRecord.Update()
```

### Code-Beispiel C#

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

dataRecVal[0] = 1;
dataRecVal[1] = "Node One";

//Object
VcDataRecord dataRecord = dataRecordCltn.Add(dataRecVal);
//CSV
dataRecord.AllData = "1;Node One;";

dataRecord.Update();
```

## DataField

### Eigenschaft von VcDataRecord

Mit dieser Eigenschaft können Sie einem Datenfeld des Datensatzes einen Wert zuweisen oder einen gesetzten Wert erfragen. Wenn ein Datensatz durch diese Methode einen neuen Wert erhalten hat, muss anschließend die grafische Darstellung mit der Methode **UpdateDataRecord** aktualisiert werden.

Die Eigenschaft DataField ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set\_DataField (index, pvn) und get\_DataField (index) angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Datenfeldes
<b>Eigenschaftswert</b>	System.Object	Inhalt des Datenfeldes

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.Update()
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);
dataRecord.set_DataField(1, "Node Two");
dataRecord.Update();
```

**DataTableName****Nur-Lese-Eigenschaft von VcDataRecord**

Mit dieser Eigenschaft können Sie den Namen der Datentabelle erfragen, zu der dieser Datensatz gehört.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name der zugehörigen Tabelle

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

MsgBox(dataRecord.DataTableName)
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

MessageBox.Show(dataRecord.DataTableName);
```

## ID

### Nur-Lese-Eigenschaft von VcDataRecord

Mit dieser Eigenschaft können Sie die ID eines Datensatzes erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Datensatz-ID

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord
dataTable = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)
MsgBox(dataRecord.ID)
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);
MessageBox.Show(dataRecord.ID);
```

## Methoden

### Delete

#### Methode von VcDataRecord

Mit dieser Methode können Sie einen Datensatz löschen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Datensatz erfolgreich (true) / nicht erfolgreich (false) gelöscht

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.Delete()
```



**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

dataRecord.Delete();
```

**IdentifyObject****Methode von VcDataRecord**

Mit dieser Methode kann man erfragen, ob und welches datenbasierte Objekt aus dem Datensatz erzeugt wurde.

Die Methode liefert als Rückgabewert **true**, wenn ein datenbasiertes Objekt ermittelt werden konnte, d.h. wenn aus dem Datensatz für die Grafik ein datenbasiertes Objekt hergestellt wurde.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ establishedObject Param	System.Object	Erkanntes Objekt
establishedObjectTypeParam	VcObjectType	Typ des erkannten Objekts
	<b>Mögliche Werte:</b> .vcObjTypeBox 15 .vcObjTypeGroup 7 .vcObjTypeLinkCollection 3 .vcObjTypeNode 2 .vcObjTypeNone 0	Objekttyp <b>Box</b> Objekttyp <b>Gruppe</b> Objekttyp <b>LinkCollection</b> Objekttyp <b>Knoten</b> kein Objekt
<b>Rückgabewert</b>	System.Boolean	datenbasiertes Objekt wurde/ wurde nicht erzeugt

**RelatedDataRecord****Methode von VcDataRecord**

Mit dieser Eigenschaft können Sie einem Datensatz einen weiteren zuordnen oder einen zugeordneten Datensatz erfragen. Bei der Verwendung von erweiterten Tabellen (extended data tables) können Datensätze einer Tabelle über einen Primärschlüssel den Datensätzen einer anderen Tabelle zugeordnet werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ index	System.Int16	Index des Datenfeldes

---

<b>Rückgabewert</b>	VcDataRecord	Zugeordneter Datensatz
---------------------	--------------	------------------------

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
    Dim dataTable As VcDataTable
    Dim dataRecordCltn As VcDataRecordCollection
    Dim firstDataRecord As VcDataRecord
    Dim secondDataRecord As VcDataRecord

    dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
    dataRecordCltn = dataTable.DataRecordCollection

    firstDataRecord = dataRecordCltn.DataRecordByID(e.Node.DataField(0))
    secondDataRecord = firstDataRecord.RelatedDataRecord(2)

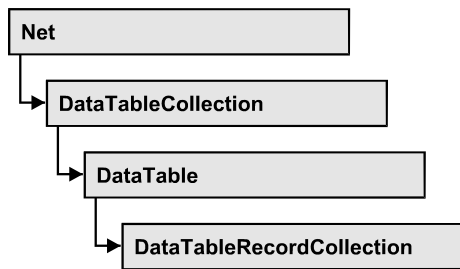
    MsgBox(secondDataRecord.AllData)
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByIndex(0);
    VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
    VcDataRecord firstDataRecord =
dataRecordCltn.DataRecordByID(e.Node.get_DataField(0));
    VcDataRecord secondDataRecord = firstDataRecord.RelatedDataRecord(2);

    MessageBox.Show(secondDataRecord.AllData.ToString());
}
```

## 7.14 VcDataRecordCollection



In einem Objekt vom Typ **VcDataRecordCollection** sind die Datensätze einer Datentabelle zusammengefasst. Mit der Eigenschaft **Count** kann die Anzahl der Datensätze im Collection-Objekt erfragt werden; mit dem Enumerator-Objekt und den Methoden **FirstDataRecord** und **NextDataRecord** können Sie iterativ auf die Datensätze zugreifen sowie mit **DataRecordByID** auf einzelne Datensätze; die Methoden **Add** und **Remove** ermöglichen das Hinzufügen und Entfernen von Datensätzen und mit **Update** können Sie die grafische Darstellung der Datensätze mit neu eingegebenen Daten aktualisieren.

### Eigenschaften

- Count

### Methoden

- Add
- DataRecordByID
- FirstDataRecord
- GetEnumerator
- NextDataRecord
- Remove
- Update

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von VcDataRecordCollection

Mit dieser Eigenschaft können Sie die Anzahl der Datensätze in der DataRecord-Auflistung erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Anzahl der Datensätze im Collection-Objekt

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
MsgBox("Number of DataRecords: " & dataRecordCltn.Count)
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
MessageBox.Show("Number of DataRecords: " + dataRecordCltn.Count);
```

---

## Methoden

### Add

**Methode von VcDataRecordCollection**

Mit dieser Methode können Sie einen neuen Datensatz in der DataRecordCollection anlegen. Wenn die ID noch nicht verwendet wurde, wird der neue Datensatz zurückgegeben, andernfalls wird eine **VcPrimary-KeyNotUniqueException** erzeugt. Nach dem Anlegen des Datensatzes muss die Methode **VcNet.EndLoading** aufgerufen werden, damit die Änderung wirksam wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ dataRecordContent	VcObject	Inhalt des Datensatzes (als Array oder String)
<b>Rückgabewert</b>	VcDataRecord	Neue angelegter Datensatz

### Code-Beispiel VB.NET

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

Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim dataRecVal() As Object

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

Dim dataRec1 As VcDataRecord
ReDim dataRecVal(dataTable.DataTableFieldCollection.Count)

dataRecVal(Main_ID) = 1
dataRecVal(Main_Name) = "Node 1"
dataRecVal(Main_Start) = DateSerial(2014, 1, 8)
dataRecVal(Main_Duration) = 8
dataRec1 = dataRecCltn.Add(dataRecVal)
VcNet1.EndLoading()

' equivalent
' dataRec1 = dataRecCltn.Add("1;Node 1;01.08.14;;8")
```

### Code-Beispiel C#

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

//...

VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
Object [] dataRecVal = new object[dataTable.DataTableFieldCollection.Count];
VcDataRecord dataRec1;

dataRecVal[Main_ID] = "1";
dataRecVal[Main_Name] = "Node 1";
dataRecVal[Main_Start] = "08.01.2014";
dataRecVal[Main_Duration] = 8;
dataRec1 = dataRecCltn.Add(dataRecVal);
VcNet1.EndLoading();

// equivalent
// dataRec2 = dataRecCltn.Add("1;Node 1;01.08.14;;8")
```

## DataRecordByID

### Methode von VcDataRecordCollection

Mit dieser Methode können Sie auf einen einzelnen Datensatz über seine Identifikation zugreifen. Existiert kein Datensatz unter der angegebenen ID, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

Wenn die Identifikation aus mehreren Feldern besteht (zusammengesetzter Primärschlüssel), muss diese mehrteilige ID folgendermaßen angegeben werden:

### ID=ID1|ID2|ID3

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataRecordID	System.String	ID des Datensatzes
<b>Rückgabewert</b>	VcDataRecord	Datensatzobjekt

#### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(0)
```

#### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(0);
```

## FirstDataRecord

### Methode von VcDataRecordCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Datensatz der DataRecord-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextDataRecord** über die nachfolgenden Datensätze zu iterieren. Existiert kein Datensatz in der Datensatzaufliistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcDataRecord	Erster Datensatz

#### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.FirstDataRecord
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.FirstDataRecord();
```

**GetEnumerator****Methode von VcDataRecordCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Datensatz-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Enumerator-Objekt

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

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

VcNet1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
    dataRecord.DataField(4) = "10"
    dataRecord.Update()
    dataRecord = dataRecordCltn.NextDataRecord
End While

VcNet1.SuspendUpdate(False)
```

**Code-Beispiel C#**

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

vcNet1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
    dataRecord.set_DataField(4, "10");
    dataRecord.Update();
    dataRecordCltn.NextDataRecord();
}

vcNet1.SuspendUpdate(false);
```

## NextDataRecord

### Methode von VcDataRecordCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Datensätze des DataRecordCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstDataRecord** den Initialwert erfasst haben. Sind alle Datensätze durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataRecord	Nachfolgende Datensatz

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

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

VcNet1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
    dataRecord.DataField(4) = "10"
    dataRecord.Update()
    dataRecord = dataRecordCltn.NextDataRecord
End While

VcNet1.SuspendUpdate(False)
```

### Code-Beispiel C#

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

vcNet1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
    dataRecord.set_DataField(4, "10");
    dataRecord.Update();
    dataRecordCltn.NextDataRecord();
}

vcNet1.SuspendUpdate(false);
```

## Remove

### Methode von VcDataRecordCollection

Mit dieser Methode können Sie einen Datensatz löschen. Die Methode liefert **true** wenn gelöscht wurde und **false**, wenn nicht gelöscht wurde. Der Datensatzinhalt im Übergabeparameter wird dazu verwendet, um anhand der Identifizierung das Objekt zu finden.



	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ dataRecordContent	VcObject	Inhalt des Datensatzes (als Array oder String)
<b>Rückgabewert</b>	System.Boolean	true

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Remove("1;1Activity; Y;Z;18.01.14;;5")
VcNet1.EndLoading()

' equivalent
' dataRecord = dataRecordCltn.DataRecordByID(1)
' dataRecord.Delete()
' dataRecord.Update()
```

**Code-Beispiel C#**

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

dataRecCltn .Remove("1;1Activity Y;Z;18.01.14;;5");
VcNet1.EndLoading();

// equivalent
// VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);
// dataRecord.Delete();
// dataRecord.Update();
```

## Update

**Methode von VcDataRecordCollection**

Mit dieser Methode können Sie einen Datensatz in der Datensatzliste aktualisieren, nachdem mittels der Methode **Add()** der Datensatz vorher hinzugefügt wurde. Falls der zu aktualisierende Datensatz nicht existiert, d.h. also nicht vorher angelegt wurde, wird er mittels Update() nun neu angelegt. Siehe auch **VcDataRecordCollection.Add()**. Nach dem Aktualisieren des Datensatzes muss die Methode **VcNet.EndLoading** aufgerufen werden, damit die Änderung wirksam wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ dataRecordContent	VcObject	Inhalt des Datensatzes (als Array oder String)
<b>Rückgabewert</b>	System.Boolean	Aktualisierung erfolgt (true) / nicht erfolgt (false)

**Code-Beispiel VB.NET**

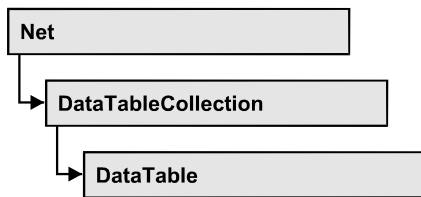
```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Update("1;1.8.2017;;8")
VcNet1.EndLoading()
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Update("1;1.8.2017;;8");
VcNet1.EndLoading();
```

## 7.15 VcDataTable



Eine Datentabelle umfasst **Datensätze** (data records) mit ihren Datenfeldern und ihren Inhalten sowie die **Beschreibungen** der Datenfelder, die **Tabellendatenfelder** (data table fields) genannt werden. Datensätze und Datentabellenfelder können in der Form von eigenen Auflistungen (Collection-Objekten) verwaltet werden.

Datentabellen ihrerseits können ebenfalls in eigenen Auflistungen verwaltet werden.

### Eigenschaften

- DataRecordCollection
- DataTableFieldCollection
- Description
- MultiplePrimaryKeysAllowed
- Name

---

## Eigenschaften

### DataRecordCollection

Nur-Lese-Eigenschaft von VcDataTable

Diese Eigenschaft gibt die in der Datentabelle enthaltene Datensatz-Auflistung zurück. Die Datensatz-Auflistung enthält alle existierenden Datensätze einer Tabelle. Zu Programmbeginn ist sie leer.

	Datentyp	Beschreibung
Eigenschaftswert	VcDataRecordCollection	DataRecordCollection-Objekt

### Code-Beispiel VB.NET

```

Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataRecordCollection.Count)
  
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
MessageBox.Show(dataTable.DataRecordCollection.Count.ToString());
```

**DataTableFieldCollection****Nur-Lese-Eigenschaft von VcDataTable**

Diese Eigenschaft gibt die in der Datentabelle enthaltene Datentabellenfeld-Auflistung zurück. Die Datentabellenfeld-Auflistung enthält die Definition der Datenfelder eines Datensatzes der Tabelle. Zu Programmbeginn enthält sie die bereits zur Designzeit vereinbarten Datenfelder. Weitere Datenfelder können zur Laufzeit über die Methode **Add** des Objekts **DataTableFieldCollection** hinzugefügt werden. Die Definition der Datentabellenfelder muss abgeschlossen sein, bevor die Tabelle mit Datensätzen gefüllt wird.

	Datentyp	Beschreibung
Eigenschaftswert	VcDataTableFieldCollection	DataTableFieldCollection-Objekt

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)
MsgBox(dataTable.DataTableFieldCollection.Count)
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByIndex(0);
MessageBox.Show(dataTable.DataTableFieldCollection.Count.ToString());
```

**Description****Eigenschaft von VcDataTable**

Mit dieser Eigenschaft können sie eine Beschreibung der Datentabelle setzen oder erfragen. Sprechende Namen, z.B. der Name der Tabelle, sind häufig sehr lang und werden daher bei Previews nicht vollständig angezeigt, so dass ihr Nutzen nicht zum Tragen kommen kann. Damit Sie für die vollständige Anzeige kurze Namen verwenden können und trotzdem nicht auf die gewünschte Information verzichten müssen, können Sie in diesem Feld zusätzliche Informationen zum Tabellennamen speichern. Sein Inhalt wird im Dialog zur Datentabelle angezeigt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Beschreibung der Datentabelle <b>Standardwert:</b> Empty string

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.DataTableByName("Maindata")
dataTable.Description = "This table contains data for nodes"
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Maindata");
dataTable.Description = "This table contains data for nodes";
```

**MultiplePrimaryKeysAllowed****Eigenschaft von VcDataTable**

Mit dieser Eigenschaft können Sie angeben oder erfragen, ob die Verwendung zusammengesetzter Primärschlüssel möglich ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Verwendung von zusammengesetzten Primärschlüsseln erlaubt (true)/nicht erlaubt (false) <b>Standardwert:</b> False

**Name****Eigenschaft von VcDataTable**

Mit dieser Eigenschaft können sie den Namen der Datentabelle setzen oder erfragen. Ein Name für die Datentabelle ist obligat und muss eindeutig sein; zudem ist eine leere Zeichenkette nicht erlaubt. Unterscheidungen in der Groß- und Kleinschreibung führen zu unterschiedlichen Namen. Mit der Methode **DataTableByName** des Objekts **DataTableCollection** können Sie über den Tabellennamen eine Referenz auf das Datentabellenobjekt erhalten.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name der Datentabelle <b>Standardwert:</b> Empty string

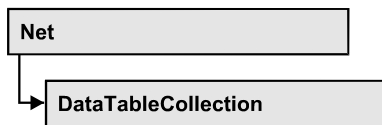
**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable  
  
dataTable = VcNet1.DataTableCollection.DataTableByIndex(0)  
MsgBox(dataTable.Name)
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByIndex(0);  
MessageBox.Show(dataTable.Name);
```

## 7.16 VcDataTableCollection



In einem Objekt vom Typ `VcDataTableCollection` sind die vorhandenen Datentabellen zusammengefasst. Mit der Eigenschaft **Count** kann die Anzahl der Tabellen im Collection-Objekt erfragt werden; mit dem Enumerator-Objekt und den Methoden **FirstDataTable** und **NextDataTable** können Sie iterativ auf die Tabellen zugreifen sowie mit **DataTableByName** und **DataTableByIndex** auf einzelne Tabellen; die Methoden **Add** und **Copy** ermöglichen das Hinzufügen und Kopieren von Tabellen und mit **Update** können Sie dem XNet-Objekt die neuen Änderungen der Datenstrukturen bekanntgeben.

### Eigenschaften

- Count

### Methoden

- Add
- Copy
- DataTableByIndex
- DataTableByName
- FirstDataTable
- GetEnumerator
- NextDataTable
- Update

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von `VcDataTableCollection`

Mit dieser Eigenschaft können Sie die Anzahl der Datentabellen in der DataTable-Auflistung erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Anzahl der Datentabellen im Collection-Objekt

**Code-Beispiel VB.NET**

```
Dim dataTableCltn As VcDataTableCollection

dataTableCltn = VcNet1.DataTableCollection
MsgBox(dataTableCltn.Count.ToString())
```

**Code-Beispiel C#**

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
MessageBox.Show(dataTableCltn.Count.ToString());
```

---

## Methoden

### Add

**Methode von VcDataTableCollection**

Mit dieser Methode können Sie eine neue Datentabelle in der DataTable-Auflistung anlegen. Wenn der Tabellename noch nicht verwendet wurde, wird ein Objekt vom Typ **VcDataTable** zurückgegeben, andernfalls "Nothing" (in Visual Basic) oder "0" (in anderen Sprachen). Nur wenn die Eigenschaft **ExtendedDataTables** auf **True** gesetzt ist, können Tabellen angelegt werden. Maximal 90 Datentabellen können angelegt werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name der neuen Tabelle
<b>Rückgabewert</b>	VcDataTable	Neu angelegte Datentabelle

**Code-Beispiel VB.NET**

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTableCltn.Update()
```

**Code-Beispiel C#**

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.Add("Resources");
dataTableCltn.Update();
```



## Copy

### Methode von VcDataTableCollection

Mit dieser Methode können Sie eine Datentabelle kopieren. Es wird nur die Tabellendefinition kopiert, jedoch nicht die eventuell bereits existierende Datensätze. Nur wenn die Eigenschaft **ExtendedDataTables** auf **true** gesetzt ist, können Tabellen angelegt werden. Wenn die Tabelle kopiert werden konnte, wird ein neues Objekt vom Typ **VcDataTable** zurückgegeben, andernfalls **Nothing** (in Visual Basic) oder **0** (in anderen Sprachen). Es wird bei den Tabellennamen generell zwischen Groß- und Kleinschreibung unterschieden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name der zu kopierenden Datentabelle (Quelltabelle)
⇒ newDataTableName	System.String	Name der neu zu erstellenden Datentabelle (Zieltabelle)
<b>Rückgabewert</b>	VcDataTable	Neu erstelltes Datentabellen-Objekt

### Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Copy("Resources", "NewResources")
dataTableCltn.Update()
```

### Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.Copy("Resources", "NewResources");
dataTableCltn.Update();
```

## DataTableByIndex

### Methode von VcDataTableCollection

Mit dieser Methode können Sie auf eine einzelne Datentabelle über ihren Index zugreifen. Der Index der ersten Tabelle ist 0. Existiert keine Tabelle unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (**Nothing** in Visual Basic oder **0** in anderen Sprachen).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ index	System.Int16	Index der Datentabelle

<b>Rückgabewert</b>	VcDataTable	Ermitteltes Datentabellenobjekt
---------------------	-------------	---------------------------------

**Code-Beispiel VB.NET**

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.DataTableByIndex(2)
MsgBox(dataTable.Name)
```

**Code-Beispiel C#**

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.DataTableByIndex(2);
MessageBox.Show(dataTable.Name);
```

**DataTableByName****Methode von VcDataTableCollection**

Mit dieser Methode können Sie auf eine einzelne Datentabelle über ihren Namen zugreifen. Existiert keine Tabelle unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (**Nothing** in Visual Basic oder **0** in anderen Sprachen).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name der Datentabelle
<b>Rückgabewert</b>	VcDataTable	Ermitteltes Datentabellenobjekt

**Code-Beispiel VB.NET**

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.DataTableByName("Resources")
MsgBox(dataTable.Description)
```

**Code-Beispiel C#**

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.DataTableByName("Resources");
MessageBox.Show(dataTable.Description);
```

**FirstDataTable****Methode von VcDataTableCollection**

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Datentabelle der DataTable-Auflistung zugreifen, um anschließend in einer Schleife mit

der Methode **NextDataTable** über die nachfolgenden Tabellen zu iterieren. Existiert keine Datentabelle in der DataTable-Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataTable	Erste Datentabelle

#### Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
```

#### Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable= dataTableCltn.FirstDataTable();
```

## GetEnumerator

### Methode von VcDataTableCollection

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Datentabellen iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Enumerator-Objekt

#### Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
For Each dataTable In dataTableCltn
    ListBox1.Items.Add(dataTable.Name)
Next
```

#### Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
foreach (VcDataTable dataTable in dataTableCltn)
    listBox1.Items.Add(dataTable.Name);
```

## NextDataTable

### Methode von VcDataTableCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Datentabellen des DataTableCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstDataTable** den Initialwert erfasst haben. Sind alle Tabellen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataTable	Nachfolgende Datentabelle

### Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable
Dim i As Integer

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
For i = 1 To dataTableCltn.Count
    ListBox1.Items.Add(dataTable.Name)
    dataTable = dataTableCltn.NextDataTable
Next
```

### Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.FirstDataTable();
for (int i=0; i<dataTableCltn.Count; i++)
{
    listBox1.Items.Add(dataTable.Name);
    dataTable = dataTableCltn.NextDataTable();
}
```

## Update

### Methode von VcDataTableCollection

Mit dieser Methode können Sie neue Änderungen an Datenstrukturen aktualisieren. Der Aufruf ist notwendig, damit durchgeführte Änderungen an der Datentabellendefinition und an den Datentabellenfeldern in der VARCHART Komponente wirksam werden. Auf diese Weise werden bei mehreren Änderungen unnötige Aktualisierungen vermieden.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (true) / nicht erfolgt (false)

### Code-Beispiel VB.NET

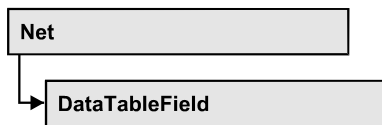
```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcNet1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTable.DataTableFieldCollection.Add("Id")
dataTableCltn.Update()
```

### Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.Add("Resources");
dataTable.DataTableFieldCollection.Add("Id");
dataTableCltn.Update();
```

## 7.17 VcDataTableField



Ein Objekt vom Typ **VcDataTableField** legt die Eigenschaften eines Feldes der Datentabelle fest. Zur Definition eines Datentabellenfeldes gehört der Name, der Datentyp und die Festlegung, ob das Datenfeld als Primärschlüssel dient, der zur eindeutigen Identifizierung eines Datensatzes herangezogen werden kann. Unter Verwendung des Primärschlüssels z.B. kann von anderen Datentabellen auf diese Datentabelle Bezug genommen werden. Um eine Beziehung zu einer Datentabelle mit Primärschlüssel aufzubauen, muss der Primärschlüssel im Feld **RelationshipFieldIndex** benannt werden.

Die DataTableField-Objekte einer Datentabelle werden über die Auflistung **DataTableFieldCollection** verwaltet.

### Eigenschaften

- DataTableName
- DateFormat
- Editable
- Hidden
- Index
- Name
- PrimaryKey
- RelationshipFieldIndex
- Type

---

## Eigenschaften

### DataTableName

Nur-Lese-Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie den Namen der zugehörigen Datentabelle erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Datentabelle

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.FirstDataTable
MsgBox(dataTable.DataTableFieldCollection.FirstDataTableField.DataTableName)
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
MessageBox.Show(dataTable.DataTableFieldCollection.FirstDataTableField().DataTab
leName);
```

**DateFormat****Nur-Lese-Eigenschaft von VcDataTableField**

Mit dieser Eigenschaft können Sie das Datumsformat des Datentabellenfeldes festlegen oder erfragen. Das Datumsformat wird beim Lesen und Schreiben von CSV-Dateien verwendet und wenn der Formattyp **String** beim Hinzufügen eines Datensatzes über die Methode **Add** zur DataRecord-Auflistung verwendet wird. Diese Eigenschaft ist nur wirksam, wenn der Datentyp des Feldes auf **vcDataTableFieldDateTime** eingestellt ist.

**Hinweis:** Es sollte zuerst die Eigenschaft **Type** gesetzt werden, bevor die Eigenschaft **DateFormat** vereinbart wird.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Datumsformat {DMYhms:;./}

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
'DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY"
VcNet1.DataTableCollection.Update()
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
//DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY";
vcNet1.DataTableCollection.Update();
```

## Editable

### Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das spezifizierte Datenfeld zur Laufzeit in der Tabelle (des Diagramms) und des Dialogs **Knoten bearbeiten** editierbar sein soll.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Feld editierbar (True) / nicht editierbar (False) <b>Standardwert:</b> True

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Editable = False
VcNet1.DataTableCollection.Update()
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Editable = false;
VcNet1.DataTableCollection.Update();
```

## Hidden

### Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das Datenfeld zur Laufzeit in den Dialogen **EditNode** oder **EditLink** angezeigt wird.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Feld wird nicht angezeigt (True) / angezeigt (False) <b>Standardwert:</b> False

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Hidden = True
VcNet1.DataTableCollection.Update()
```



### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Hidden = true;
vcNet1.DataTableCollection.Update();
```

## Index

### Nur-Lese-Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie den Index des Datentabellenfelds in der zugehörigen Datentabelle erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Index des Datentabellenfeldes.

## Name

### Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie den Namen des Datenfelds setzen oder erfragen. Der Name des Datenfelds erscheint innerhalb von Laufzeitdialogen wie beispielsweise dem **EditNode**-Dialog. In der API erfolgt der Zugriff auf die Datenfeldinhalte eines Datensatzes jedoch immer über den Index, den dieses Feld im **DataTableFieldCollection**-Objekt besitzt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Feldes <b>Standardwert:</b> Empty string

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.Add("Start")
VcNet1.DataTableCollection.Update()
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Start");
vcNet1.DataTableCollection.Update();
```

## PrimaryKey

### Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob dieses Datenfeld den Primärschlüssel enthält, der zur eindeutigen Identifikation eines Datensatz herangezogen werden kann. In einer Datentabelle kann immer nur ein Datenfeld die Kennung Primärschlüssel tragen. Die aktuelle Setzung hebt eine eventuell vorhandene Setzung bei einem anderen Datenfeld der gleichen Tabelle auf. Die Festlegung eines Primärschlüssels ist unerlässlich, wenn eine Beziehung von einer anderen Tabelle zu dieser Tabelle hergestellt werden soll.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Das Feld dient (True) / dient nicht (False) als Primärschlüssel <b>Standardwert:</b> False

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
Dim isPrimaryKey As Boolean

dataTable = VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Id")
dataTableField.PrimaryKey = True
VcNet1.DataTableCollection.Update()
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Id");
dataTableField.PrimaryKey = true;
vcNet1.DataTableCollection.Update();
```

## RelationshipFieldIndex

### Eigenschaft von VcDataTableField

Mit dieser Eigenschaft verbinden Sie ein Datenfeld und seine Beschreibung. Dazu setzen Sie hier den Index des Datensatzfeldes, auf welches sich die gesetzten Eigenschaften des Datentabellenfeldes beziehen sollen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Index des Datensatzfeldes, auf das sich die Datendefinition des Datentabellenfeldes bezieht. <b>Standardwert:</b> -1

**Code-Beispiel VB.NET**

```

Dim dataTableTask As VcDataTable
Dim dataTaskFieldId As VcDataTableField
Dim dataTaskFieldName As VcDataTableField
Dim dataTableOperation As VcDataTable
Dim dataOperationFieldId As VcDataTableField
Dim dataOperationFieldName As VcDataTableField
Dim dataOperationFieldTaskId As VcDataTableField

'Create table Task
dataTableTask = VcNet1.DataTableCollection.Add("Task")
dataTaskFieldId = dataTableTask.DataTableFieldCollection.Add("Id")
dataTaskFieldId.PrimaryKey = True
dataTaskFieldName = dataTableTask.DataTableFieldCollection.Add("Name")
dataTaskFieldName.Type = VcDataTableFieldType.vcDataTableFieldStringType

'Create table Operation
dataTableOperation = VcNet1.DataTableCollection.Add("Operation")
dataOperationFieldId = dataTableOperation.DataTableFieldCollection.Add("Id")
dataOperationFieldId.PrimaryKey = True
dataOperationFieldName = dataTableOperation.DataTableFieldCollection.Add("Name")
dataOperationFieldName.Type = VcDataTableFieldType.vcDataTableFieldStringType
dataOperationFieldTaskId =
dataTableOperation.DataTableFieldCollection.Add("TaskId")
dataOperationFieldTaskId.Type = VcDataTableFieldType.vcDataTableFieldIntegerType

'Node tables Task and Operations
dataOperationFieldTaskId.RelationshipFieldIndex =
VcNet1.DetectFieldIndex("Task", "Id")
VcNet1.DataTableCollection.Update()

```

**Code-Beispiel C#**

```

//Create table Task
VcDataTable dataTableTask = vcNet1.DataTableCollection.Add("Task");
VcDataTableField dataTaskFieldId =
dataTableTask.DataTableFieldCollection.Add("Id");
dataTaskFieldId.PrimaryKey = true;
VcDataTableField dataTaskFieldName =
dataTableTask.DataTableFieldCollection.Add("Name");
dataTaskFieldName.Type = VcDataDefinitionFieldType.vcDefFieldStringType;

//Create table Operation
VcDataTable dataTableOperation = vcNet1.DataTableCollection.Add("Operation");
VcDataTableField dataOperationFieldId =
dataTableOperation.DataTableFieldCollection.Add("Id");
dataOperationFieldId.PrimaryKey = true;
VcDataTableField dataOperationFieldName =
dataTableOperation.DataTableFieldCollection.Add("Name");
dataOperationFieldName.Type = VcDataDefinitionFieldType.vcDefFieldStringType;
VcDataTableField dataOperationFieldTaskId =
dataTableOperation.DataTableFieldCollection.Add("TaskId");
dataOperationFieldTaskId.Type = VcDataDefinitionFieldType.vcDefFieldIntegerType;

//Node tables Task and Operation
dataOperationFieldTaskId.RelationshipFieldIndex =
vcNet1.DetectFieldIndex("Task", "Id");
vcNet1.DataTableCollection.Update();

```

## Type

### Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie den Datentyp des Feldes setzen oder erfragen.

**Hinweis:** Durch Setzen der Eigenschaft **Type** kann sich die Eigenschaft **DateFormat** ändern. Durch Setzen des Eigenschaftswertes auf **vcDataTableAlphanumeric** oder **vcDataTableFieldInteger** wird ein eventuell eingestelltes Datumsformat auf "" gesetzt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcDataTableFieldType	Datentyp des Feldes, kann maximal 512 Zeichen enthalten <b>Standardwert:</b> vcDataTableFieldIntegerType

### Code-Beispiel VB.NET

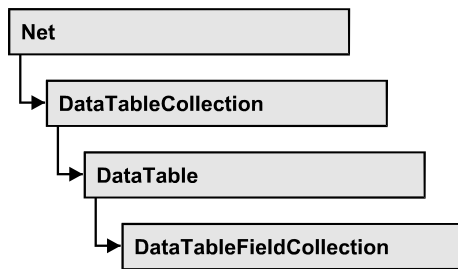
```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

VcNet1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
VcNet1.DataTableCollection.Update()
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
vcNet1.DataTableCollection.Update();
```

## 7.18 VcDataTableFieldCollection



In einem Objekt vom Typ `VcDataTableFieldCollection` sind die Datentabellenfelder einer Datentabelle zusammengefasst. Mit der Eigenschaft **Count** kann die Anzahl der Felder im Collection-Objekt erfragt werden; mit dem Enumerator-Objekt und den Methoden **FirstDataTableField** und **NextDataTableField** können Sie iterativ auf die Felder zugreifen sowie mit **DataFieldByName** und **DataFieldByIndex** auf einzelne Felder; die Methoden **Add** und **Copy** ermöglichen das Hinzufügen und Kopieren von Feldern.

### Eigenschaften

- Count

### Methoden

- Add
- Copy
- DataTableFieldByIndex
- DataTableFieldByName
- FirstDataTableField
- GetEnumerator
- NextDataTableField

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von `VcDataTableFieldCollection`

Mit dieser Eigenschaft können Sie die Anzahl der Datentabellenfelder in der `DataTableField`-Auflistung erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Anzahl der Datentabellenfelder im Collection-Objekt

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable

dataTable = VcNet1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataTableFieldCollection.Count.ToString())
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
MessageBox.Show(dataTable.DataTableFieldCollection.Count.ToString());
```

---

## Methoden

### Add

**Methode von VcDataTableFieldCollection**

Mit dieser Methode können Sie ein neues Datentabellenfeld in der DataTableFieldCollection anlegen. Wenn der Name noch nicht verwendet wurde, wird das neue Datenfeld zurückgegeben, andernfalls "Nothing" (Visual Basic) oder "0" (andere Sprachen). Maximal 9.999 Felder können angelegt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ dataTableFieldName	System.String	Name des neuen Datentabellenfeldes
<b>Rückgabewert</b>	VcDataTableField	Neu angelegtes Datentabellenfeld

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Add("Priority")
VcNet1.DataTableCollection.Update()
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Priority");
vcNet1.DataTableCollection.Update();
```

## Copy

### Methode von VcDataTableFieldCollection

Mit dieser Methode können Sie ein Datentabellenfeld kopieren. Die Identifizierung des Feldes erfolgt über den Namen.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataTableFieldName	System.String	Name des zu kopierenden Datentabellenfeldes (Quellfeld)
⇒ newDataTableFieldName	System.String	Name des neu zu erstellenden Datentabellenfeldes (Zielfeld)
<b>Rückgabewert</b>	VcDataTableField	Neu angelegtes Datentabellenfeld

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Copy("Name", "NewName")
VcNet1.DataTableCollection.Update()
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Copy("Name", "NewName");
vcNet1.DataTableCollection.Update();
```

## DataTableFieldByIndex

### Methode von VcDataTableFieldCollection

Mit dieser Methode können Sie auf einen einzelnen Datensatz über seinen Index zugreifen. Existiert kein Datensatz an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ index	System.Int16	Index des Datentabellenfeldes
<b>Rückgabewert</b>	VcDataTableField	Ermitteltes Datentabellenfeld

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByIndex(1)
MsgBox(dataTableField.Name)
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByIndex(1);
MessageBox.Show(dataTableField.Name);
```

**DataTableFieldByName****Methode von VcDataTableFieldCollection**

Mit dieser Methode können Sie auf ein einzelnes Datentabellenfeld über seinen Namen zugreifen. Existiert kein Feld unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataTableFieldName	System.String	Name des Datentabellenfeldes
<b>Rückgabewert</b>	VcDataTableField	Ermitteltes Datentabellenfeld

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Name")
dataTableField.Editable = False
VcNet1.DataTableCollection.Update()
```

**Code-Beispiel C#**

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Name");
dataTableField.Editable = false;
vcNet1.DataTableCollection.Update();
```

**FirstDataTableField****Methode von VcDataTableFieldCollection**

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Datenfeld der DataTableField-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextDataTableField** über die nachfolgenden Datenfelder



zu iterieren. Existiert kein Datenfeld in der DataTableField-Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataTableField	Erstes Datentabellenfeld

#### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.FirstDataTableField()
```

#### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.FirstDataTableField();
```

## GetEnumerator

#### Methode von VcDataTableFieldCollection

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Tabellendatenfeld-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Enumerator-Objekt

#### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcNet1.DataTableCollection.FirstDataTable()
For Each dataTableField In dataTable.DataTableFieldCollection
    ListBox1.Items.Add(dataTableField.Name)
Next
```

#### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
foreach (VcDataTableField dataTableField in dataTable.DataTableFieldCollection)
    listBox1.Items.Add(dataTableField.Name);
```

## NextDataTableField

### Methode von VcDataTableFieldCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Datentabellenfelder des DataTableFieldCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstDataTableField** den Initialwert erfasst haben. Sind alle Felder durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataTable	Nachfolgendes Datentabellenfeld

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableFieldCltn As VcDataTableFieldCollection
Dim dataTableField As VcDataTableField
Dim i As Integer

dataTable = VcNet1.DataTableCollection.FirstDataTable()
dataTableFieldCltn = dataTable.DataTableFieldCollection
dataTableField = dataTableFieldCltn.FirstDataTableField
For i = 1 To dataTableFieldCltn.Count
    ListBox1.Items.Add(dataTableField.Name)
    dataTableField = dataTableFieldCltn.NextDataTableField()
Next
```

### Code-Beispiel C#

```
VcDataTable dataTable = vcNet1.DataTableCollection.FirstDataTable();
VcDataTableFieldCollection dataTableFieldCltn =
dataTable.DataTableFieldCollection;
VcDataTableField dataTableField = dataTableFieldCltn.FirstDataTableField();
for (int i=0; i<dataTableFieldCltn.Count; i++)
{
    listBox1.Items.Add(dataTableField.Name);
    dataTableField = dataTableFieldCltn.NextDataTableField();
}
```

## 7.19 VcFilter



Ein Objekt vom Typ VcFilter enthält Filterbedingungen (VcFilterSubCondition), z. B. zulässige Werte für die Datenfelder eines Knotens oder einer Verbindung. Abhängig von den Daten trifft die Filterbedingung für einen Vorgang zu oder nicht. Filter werden verwendet, um z. B. einem Knoten ein bestimmtes Format zuzuweisen.

Nur wenn der Filter nach Änderungen der Filterbedingungen gültig ist, werden diese wirksam. Andernfalls bleiben die bisherigen Filterbedingungen wirksam (prüfbar über die Methoden VcFilter.IsValid und VcFilterSubCondition.IsValid).

### Eigenschaften

- DataDefinitionTable
- DatesWithHourAndMinute
- Name
- Specification
- StringsCaseSensitive
- SubCondition
- SubConditionCount

### Methoden

- AddSubCondition
- CopySubCondition
- Evaluate
- GetEnumerator
- IsValid
- RemoveSubCondition

## Eigenschaften

### DataDefinitionTable

Eigenschaft von VcFilter

Diese Eigenschaft gibt zurück, ob es sich um einen Filter für Knoten (vcMainData) oder für Verbindungen (vcRelations) handelt. Ändern kann man diese Eigenschaft nur, wenn der Filter keine Bedingungen enthält.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcDataTableType  <b>Mögliche Werte:</b> .vcMainData 0 .vcMaindata 0 .vcRelations 1 .vcRelations 1	Typ der Datendefinitionstabelle  Definition von Knotendaten Tabellentyp <b>vcMaindata</b> (für Knoten) Definition von Verbindungsdaten Tabellentyp <b>vcRelations</b> (für Verbindungen)

### DatesWithHourAndMinute

Eigenschaft von VcFilter

Diese Eigenschaft entscheidet, ob beim Vergleich von Datums-Filterbedingungen Stunden und Minuten berücksichtigt werden. Die Einstellung kann nur geändert werden, wenn mindestens eine Unterbedingung mit einem Datumsvergleich vorhanden ist. Ansonsten ist der Eigenschaftswert immer False.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Stunden und Minuten werden berücksichtigt (True)/ nicht berücksichtigt (False)

### Name

Eigenschaft von VcFilter

Mit dieser Eigenschaft können Sie den Namen des Filters erfragen oder setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Filters

### Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection

For Each filter In filterCltn
    ListBox1.Items.Add(filter.Name)
Next
```

### Code-Beispiel C#

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;

foreach (VcFilter filter in filterCltn)
{
    ListBox.Items.Add(filter.Name);
}
```

## Specification

### Nur-Lese-Eigenschaft von VcFilter

Mit dieser Eigenschaft können Sie die Spezifikation dieses Filters auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Erzeugung eines Filters mit der Methode **VcFilterCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Filterspezifikation

### Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filterCltn.FirstFilter
MsgBox(filter.Specification)
```

### Code-Beispiel C#

```
VcFilterCollection boxCltn = vcNet1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();
MessageBox.Show(filter.Specification);
```

## StringsCaseSensitive

### Eigenschaft von VcFilter

Diese Eigenschaft entscheidet, ob bei String-Filterbedingungen der Vergleich mit Unterscheidung von Groß- und Kleinschreibung stattfindet.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Vergleich mit Unterscheidung von Groß- und Kleinschreibung findet statt (True)/findet nicht statt (False)

## SubCondition

**Nur-Lese-Eigenschaft von VcFilter**

Mit dieser Eigenschaft können Sie auf ein VcFilterSubCondition-Objekt per Index zugreifen.

Die Eigenschaft SubCondition ist eine indizierte Eigenschaft, die in C# über die Methode get\_SubCondition (index) angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index der Filterbedingung {0 ... VcFilter.SubConditionCount-1}
<b>Eigenschaftswert</b>	VcFilterSubCondition	Filterbedingungsobjekt

## SubConditionCount

**Nur-Lese-Eigenschaft von VcFilter**

Mit dieser Eigenschaft können Sie die Anzahl der Filterbedingungen erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Anzahl der Filterbedingungen

## Methoden

### AddSubCondition

Methode von VcFilter

Mit dieser Methode können Sie eine neue Filterbedingung an der angegebenen Stelle in der Collection der bestehenden Filterbedingungen erzeugen.

Das entsprechende VcFilterSubCondition-Objekt wird zurückgegeben. Die Eigenschaften dieses Objekt sind standardmäßig folgendermaßen gesetzt:

- DataFieldIndex: -1
- Operator: vcInvalidOp
- ComparisonValueAsString: "<INVALID>"
- ConnectionOperator: vcInvalidConnOp.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ atIndex	System.Int16	Index der neuen Filterbedingung {0 to VcFilter.SubConditionCount and -1 for "at the end of the Collection" (identical with the value VcFilter.SubConditionCount)}
<b>Rückgabewert</b>	VcFilterSubCondition	Filterbedingungsobjekt

### CopySubCondition

Methode von VcFilter

Mit dieser Methode können Sie eine Filterbedingung mit Hilfe der Indexangabe kopieren. Die neue Filterbedingung wird an der angegebenen Stelle in der Collection der bestehenden Filterbedingungen eingefügt und als VcFilterSubCondition-Objekt zurückgegeben.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fromIndex	System.Int16	Index der zu kopierenden Filterbedingung

⇒ atIndex	System.Int16	Index der neuen Filterbedingung  {0 to VcFilter.SubConditionCount and -1 for "at the end of the Collection" (identical with the value VcFilter.SubConditionCount)}
<b>Rückgabewert</b>	VcFilterSubCondition	Filterbedingungsobjekt

## Evaluate

### Methode von VcFilter

Mit dieser Methode kann für einen bestimmten Datensatz geprüft werden, ob der gesetzte Filter zutrifft oder nicht. Es sollten sinnvollerweise nur Objekte übergeben werden, die intern mit Datensätzen der Datentabellen verbunden sind. Dies sind: **VcNode**, **VcLink**, **VcGroup**, **VcDataRecord**. Wird ein hier nicht aufgeführter Objekttyp übergeben, wird eine Exception ausgelöst.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ dataObjectParam	Variant	Datensatzobjekt
<b>Rückgabewert</b>	Boolean	Filter trifft für Datensatz zu (True)/nicht zu (False)

## GetEnumerator

### Methode von VcFilter

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Bedingungs-Objekte iterieren.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcObject	Referenzobjekt

### Code-Beispiel VB.NET

```
Dim filter As VcFilter
Dim filterCond As VcFilterSubCondition

filter = VcNet1.FilterCollection.FirstFilter

For Each filterCond In filter
    Debug.Write(filterCond.Index)
Next
```



### Code-Beispiel C#

```
VcFilter filter = vcNet1.FilterCollection.FirstFilter();
foreach(VcFilterSubCondition filterCond in filter)
{
    Console.WriteLine(filterCond.Index);
}
```

## IsValid

### Methode von VcFilter

Diese Methode prüft, ob alle Filterbedingungen korrekt formuliert sind. Nur wenn das der Fall ist, werden geänderte Filterbedingungen überhaupt wirksam. Andernfalls bleiben die bisherigen Filterbedingungen wirksam.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Filterbedingungen korrekt (True)/ nicht korrekt (False)

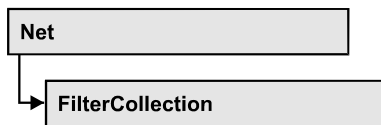
## RemoveSubCondition

### Methode von VcFilter

Mit dieser Methode können Sie eine Filterbedingung mit Hilfe der Indexangabe löschen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index der zu löschenden Filterbedingung

## 7.20 VcFilterCollection



In einem Objekt vom Typ `VcFilterCollection` sind automatisch alle verfügbaren Filter zusammengefasst. Über **For Each filter In FilterCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Filter zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **FilterByName** und **FilterByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Filter kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Filtern.

### Eigenschaften

- Count
- MarkedNodesFilter

### Methoden

- Add
- AddBySpecification
- Copy
- FilterByIndex
- FilterByName
- FirstFilter
- GetEnumerator
- NextFilter
- Remove

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von `VcFilterCollection`

Mit dieser Eigenschaft können Sie die Anzahl der Filterobjekte in der Filter-Auflistung abfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Filter

**Code-Beispiel VB.NET**

```
Dim filterCltn As VcFilterCollection
Dim numberOfFilters As Integer

filterCltn = VcNet1.FilterCollection
numberOfFilters = filterCltn.Count
```

**Code-Beispiel C#**

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
int numberOfFilters = filterCltn.Count;
```

**MarkedNodesFilter****Nur-Lese-Eigenschaft von VcFilterCollection**

Mit dieser Eigenschaft können Sie einen konstanten Pseudo-Filter holen, der nur bei **ActiveNodeFilter** eingesetzt werden kann und dort das Filtern auf die gerade markierten Knoten auslöst (Teildiagramm).

	Datentyp	Beschreibung
Eigenschaftswert	VcFilter	Pseudo-Filter

**Code-Beispiel VB.NET**

```
VcNet1.ActiveNodeFilter = VcNet1.FilterCollection.MarkedNodesFilter
```

**Code-Beispiel C#**

```
vcNet1.ActiveNodeFilter = vcNet1.FilterCollection.MarkedNodesFilter;
```

**Methoden****Add****Methode von VcFilterCollection**

Mit dieser Methode können Sie einen neuen Filter in der FilterCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Filterobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Die referenzierte Datendefinitionstabelle des neuen Filters ist automatisch vcMainData (siehe VcFilter.DataDefinitionTable). Sie kann

auf vcRelations geändert werden, solange der Filter noch keine Unterbedingungen enthält.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ newName	System.String	Name des Filters
<b>Rückgabewert</b>	VcFilter	Neues Filterobjekt

#### Code-Beispiel VB.NET

```
newFilter = VcNet1.FilterCollection.Add("foo")
```

#### Code-Beispiel C#

```
newFilter = vcNet1.FilterCollection.Add("foo");
```

## AddBySpecification

### Methode von VcFilterCollection

Mit dieser Methode können Sie einen Filter über eine Filter-Spezifikation erzeugen. Dies dient der Persistenz von Filterobjekten. Die Spezifikation eines Filters kann erfragt (siehe VcFilter-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann der gleiche Filter mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ filterSpecification	System.String	Filterspezifikation
<b>Rückgabewert</b>	VcFilter	Neues Filterobjekt

## Copy

### Methode von VcFilterCollection

Mit dieser Methode können Sie einen Filter kopieren. Wenn der Filter mit dem angegebenen Namen existiert und der Name des neuen Filters noch nicht verwendet wird, wird das neue Filterobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fromName	System.String	Name des zu kopierenden Filters

⇒ newName	System.String	Name des neuen Filters
<b>Rückgabewert</b>	VcFilter	Filterobjekt

## FilterByIndex

### Methode von VcFilterCollection

Mit dieser Methode können Sie auf einen einzelnen Filter über ihren Index zugreifen. Existiert kein Filter an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ index	System.Int16	Index des Filters
<b>Rückgabewert</b>	VcFilter	Ermitteltes Filterobjekt

## FilterByName

### Methode von VcFilterCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf einen bestimmten Filter zugreifen. Existiert kein Filter unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ filterName	System.String	Name des Filters
<b>Rückgabewert</b>	VcFilter	Filter

### Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filterCltn.FilterByName("Department A")
```

### Code-Beispiel C#

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
VcFilter filter = filterCltn.FilterByName("Department A");
```

## FirstFilter

### Methode von VcFilterCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Filter des FilterCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextFilter** über die nachfolgenden Filter zu iterieren. Existiert kein Filter im FilterCollection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcFilter	Erster Filter

### Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filtercltn.FirstFilter
```

### Code-Beispiel C#

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();
```

## GetEnumerator

### Methode von VcFilterCollection

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt verwendet. Mit diesem Objekt können Sie über alle enthaltenen Filter-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

### Code-Beispiel VB.NET

```
Dim filter As VcFilter
Dim filterCond As VcFilterSubCondition

filter = VcNet1.FilterCollection.FirstFilter

For Each filterCond In filter
    Debug.Write(filterCond.FilterName)
Next
```

**Code-Beispiel C#**

```
VcFilter filter = vcNet1.FilterCollection.FirstFilter();
foreach(VcFilterSubCondition filterCond in filter)
{
    Console.Write(filterCond.FilterName);
}
```

**NextFilter****Methode von VcFilterCollection**

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Filter des FilterCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstFilter** den Initialwert erfasst haben. Sind alle Filter durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcFilter	Nächster Filter

**Code-Beispiel VB.NET**

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcNet1.FilterCollection
filter = filterCltn.FirstFilter

While Not filter Is Nothing
    ListBox1.Items.Add(filter.Name)
    filter = filterCltn.NextFilter
End While
```

**Code-Beispiel C#**

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();

while (filter != null)
{
    ListBox.Items.Add(filter.Name);
    filter = filterCltn.NextFilter();
}
```

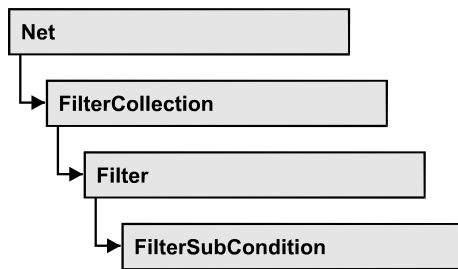
**Remove****Methode von VcFilterCollection**

Mit dieser Methode können Sie einen Filter löschen. Wenn der Filter noch in irgendeinem anderen Objekt benutzt wird, kann er nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ name	System.String	Name des Filters
<b>Rückgabewert</b>	System.Boolean	Filter gelöscht (True)/nicht gelöscht (False)



## 7.21 VcFilterSubCondition



Ein Objekt vom Typ `VcFilterSubCondition` enthält eine einzelne Filterbedingung. Eine Filterbedingung hat im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, der ihre Position im Filter bestimmt.

Im Dialog **Filter bearbeiten** gibt es für jede Filterbedingung eine eigene Zeile. Die dort zur Designzeit dargestellten Eigenschaften sind mit der API hier zur Laufzeit nachträglich veränderbar.

### Eigenschaften

- `ComparisonValueAsString`
- `ConnectionOperator`
- `DataFieldIndex`
- `FilterName`
- `Index`
- `Operator`

### Methoden

- `IsValid`

---

## Eigenschaften

### ComparisonValueAsString

Eigenschaft von `VcFilterSubCondition`

Mit dieser Eigenschaft können Sie den Vergleichswert erfragen oder setzen. Dieser String muss einem bestimmten Format entsprechen:

- `String`: wird in doppelte Anführungszeichen eingeschlossen. Beispiel in VB: `""Aachen""`; Beispiel in C/C++: `"\Aachen\"`

- Datum: wird in #-Zeichen eingeschlossen. Beispiel: "#18.06.2015;12:34:56;#" (das Datumsformat ist immer "DD.MM.YYYY;hh:mm:ss;", da es sich hierbei um das interne Standardformat, unabhängig vom Betriebssystem und dessen lokalen Einstellungen, handelt). Ein spezieller Datums-Vergleichswert ist "<TODAY>".
- Datenfeld: wird in eckige Klammern eingeschlossen. Beispiel: "[ID]"
- Zahl: wird direkt angegeben. Beispiel: "52076"
- Liste: bei einem der vc...In-Operatoren: wird in geschweifte Klammern eingeschlossen. Die enthaltenen Werte müssen dann alle vom gleichen Typ (String, Datum oder Zahl) sein und können alle obigen Formate besitzen. Beispiel: "{"NETRONIC", [Name]}"
- Ungültig (z. B. nach Neuerzeugen einer Unterbedingung): "<INVALID>"

Der Typ des Vergleichswerts muss dem Datenfeldtyp und dem Typ des Operators entsprechen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Vergleichswert

## ConnectionOperator

### Eigenschaft von VcFilterSubCondition

Mit dieser Eigenschaft können Sie den Operator für die Verknüpfung mit der folgenden Unterbedingung erfragen oder setzen. Dabei bindet **vcAnd** stärker als **vcOr**.

	Datentyp	Beschreibung
Eigenschaftswert	VcConnectionOperator  <b>Mögliche Werte:</b> .vcAnd 1 .vcInvalidConnOp 0 .vcOr 2	Operator für die Verknüpfung mit der folgenden Filterbedingung  Und-Operator ungültiger Operator Oder-Operator

## DataFieldIndex

### Eigenschaft von VcFilterSubCondition

Mit dieser Eigenschaft können Sie den Index des Datenfeldes, dessen Inhalt verglichen werden soll, erfragen oder setzen. Der Datenfeldtyp muss dem Typ des Vergleichswerts und des Operators entsprechen.

**Sonderwert:** -1: kein Datenfeld (ungültig)

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, dessen Inhalt verglichen werden soll

## FilterName

### Nur-Lese-Eigenschaft von VcFilterSubCondition

Mit dieser Eigenschaft können Sie den Namen des Filters erfragen, zu dem diese Filterbedingung gehört.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Filters

## Index

### Nur-Lese-Eigenschaft von VcFilterSubCondition

Mit dieser Eigenschaft können Sie den Index dieser Filterbedingung im zugehörigen Filter erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index der Filterbedingung im zugehörigen Filter

## Operator

### Eigenschaft von VcFilterSubCondition

Mit dieser Eigenschaft können Sie den Operator für den Vergleich erfragen oder setzen. Die über die API verfügbaren Operatoren entsprechen den Operatoren im Dialog **Filter bearbeiten**. Der Typ des Operators muss dem Datenfeldtyp des Vergleichswerts entsprechen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcOperator	Vergleichsoperator
	<b>Mögliche Werte:</b>	
	.vcDateEarlier 27	Datum früher als
	.vcDateEarlierOrEqual 28	Datum früher als oder gleich
	.vcDateEqual 25	Datum gleich
	.vcDateIn 31	Datum in
	.vcDateLater 29	Datum später als
	.vcDateLaterOrEqual 30	Datum später als oder gleich
	.vcDateNotEqual 26	Datum ungleich
	.vcDateNotIn 32	Datum nicht in
	.vcIntEqual 9	Integer gleich
	.vcIntGreater 13	Integer größer
	.vcIntGreaterOrEqual 14	Integer größer oder gleich
	.vcIntIn 15	Integer in
	.vcIntLess 11	Integer kleiner als
	.vcIntLessOrEqual 12	Integer kleiner als oder größer
	.vcIntNotEqual 10	Integer ungleich
	.vcIntNotIn 16	Integer nicht in
	.vcInvalidOp 0	ungültiger Operator
	.vcStringBeginsWith 3	String beginnt mit
	.vcStringContains 5	String enthält
	.vcStringEqual 1	String gleich
	.vcStringIn 7	String enthält
	.vcStringNotBeginsWith 4	String beginnt nicht mit
	.vcStringNotContains 6	String enthält nicht
	.vcStringNotEqual 2	String nicht gleich
	.vcStringNotIn 8	String nicht enthalten in

## Methoden

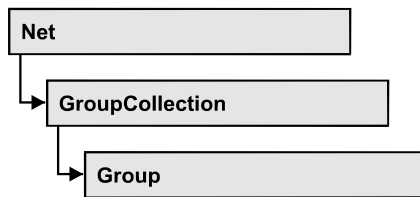
### IsValid

Methode von VcFilterSubCondition

Diese Methode prüft, ob die Filterbedingung korrekt formuliert ist.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Filterbedingung korrekt (True)/ nicht korrekt (False)

## 7.22 VcGroup



Eine Gruppe enthält alle Knoten, die im Gruppierfeld denselben Wert haben. Dieser Wert kann als Gruppenname abgefragt werden. Auf die Knoten, die eine Gruppe bilden, können Sie über die Eigenschaft **NodeCollection** zugreifen.

### Eigenschaften

- BackgroundColor
- LineColor
- LineThickness
- LineType
- Name
- NodeCollection
- Title
- TitleLineCount
- X
- Y

### Methoden

- SetXY

---

## Eigenschaften

### BackgroundColor

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie die Hintergrundfarbe einer Gruppe setzen oder erfragen. Die Standard-Farbe ist weiß.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte {0...255},{0...255},{0...255}

**Code-Beispiel VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup

group.BackColor = RGB(128, 128, 128)
```

**Code-Beispiel C#**

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup;
group.BackColor = RGB(128, 128, 128);
```

## LineColor

**Nur-Lese-Eigenschaft von VcGroup**

Mit dieser Eigenschaft können Sie die Linienfarbe für die Randlinie einer Gruppe erfragen oder festlegen. Sie können diese Eigenschaft auch auf der Eigenschaftenseite <bIGrupppierung festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte {0...255},{0...255},{0...255}

## LineThickness

**Nur-Lese-Eigenschaft von VcGroup**

Mit dieser Eigenschaft können Sie die Stärke der Randlinie der Gruppe erfragen oder festlegen.

Wenn Sie diese Eigenschaft auf Werte zwischen 1 und 4 setzen, wird damit eine absolute Liniendicke in Pixel definiert, d.h. die Linien haben unabhängig vom Zoomfaktor immer die gleiche feste Linienstärke in Pixeln. Dies wird jedoch aufgrund der besseren Lesbarkeit beim Drucken in eine vom Zoomfaktor abhängige Liniendicke umgewandelt:

Wert	Punkte	mm
1	1/2 Punkt	0,09 mm
2	1 Punkt	0,18 mm
3	3/2 Punkt	0,26 mm
4	2 Punkt	0,35 mm

Ein Punkt ist 1/72 Zoll groß und stellt die Maßeinheit für Schriftgrößen dar.

Wenn Sie diese Eigenschaft auf Werte zwischen 5 und 1.000 setzen, wird damit eine Linienstärke in 1/100 mm definiert, d.h. die Linien bekommen eine tatsächliche Dicke in Pixeln, die abhängig vom Zoomfaktor ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm <b>Standardwert:</b> As defined in the dialog

## LineType

Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie den (Rand-)Linientyp einer Gruppe erfragen oder festlegen. Sie können diese Eigenschaft auch im auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLineType	Linientyp <b>Standardwert:</b> vcSolid
	<b>Mögliche Werte:</b> .vcDashed 4 .vcDashed 4 .vcDashedDotted 5 .vcDashedDotted 5 .vcDotted 3 .vcDotted 3 .vcLineType0 100  .vcLineType1 101  .vcLineType10 110  .vcLineType11 111	Linientyp <b>gestrichelt</b> Linientyp <b>gestrichelt</b> Linientyp <b>gestrichelt-gepunktet</b> Linientyp <b>gestrichelt-gepunktet</b> Linientyp <b>gepunktet</b> Linientyp <b>gepunktet</b> Linientyp 0  Linientyp 1  Linientyp 10  Linientyp 11

.vcLineType12 112	Linientyp 12
.vcLineType13 113	Linientyp 13
.vcLineType14 114	Linientyp 14
.vcLineType15 115	Linientyp 15
.vcLineType16 116	Linientyp 16
.vcLineType17 117	Linientyp 17
.vcLineType18 118	Linientyp 18
.vcLineType2 102	Linientyp 2
.vcLineType3 103	Linientyp 3
.vcLineType4 104	Linientyp 4
.vcLineType5 105	Linientyp 5
.vcLineType6 106	Linientyp 6
.vcLineType7 107	Linientyp 7
.vcLineType8 108	Linientyp 8
.vcLineType9 109	Linientyp 9
.vcNone 1	Kein Linientyp zugewiesen
.vcNone 1	Kein Linientyp
.vcNotSet -1	Kein Linientyp zugewiesen
.vcSolid 2	Linientyp <b>durchgezogen</b>
.vcSolid 2	Linientyp <b>durchgezogen</b>

## Name

### Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie den Namen der Gruppe (= Wert des Gruppierfeldes GroupField) erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Gruppenname

### Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
groupName = group.Name
```



### Code-Beispiel C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
string groupName = group.Name;
```

## NodeCollection

**Nur-Lese-Eigenschaft von VcGroup**

Über diese Eigenschaft haben Sie Zugriff auf alle Knoten, die zu einer Gruppe gehören.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcNodeCollection	NodeCollection Objekt

### Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
nodeCltn = group.NodeCollection
```

### Code-Beispiel C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
VcNodeCollection nodeCltn = group.NodeCollection;
```

## Title

**Eigenschaft von VcGroup**

Mit dieser Eigenschaft können Sie den Gruppentitel erfragen oder festlegen. Der Titel wird in der obersten Zeile der Gruppe ausgegeben. Wenn Sie diese Eigenschaft nicht festlegen, keinen Dateinamen bei VcNet.Group-DescriptionName angeben und kein Feld bei VcNet.GroupTitleField angeben, so wird in der obersten Zeile der Gruppenname ausgegeben.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Gruppentitel

### Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroupMsgBox(group.Title)
```

**Code-Beispiel C#**

```
VcGroupCollection groupCltn = Dummyobject2.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
MessageBox.Show(group.Title);
```

**TitleLineCount****Eigenschaft von VcGroup**

Mit dieser Eigenschaft können Sie bei der aktuellen Gruppe die Anzahl der Zeilen für den Titeltext einstellen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16 1 ... 5	Anzahl der Zeilen für den Titeltext <b>Standardwert:</b> 1

**Code-Beispiel VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
group.TitleLineCount = 5
```

**Code-Beispiel C#**

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
group.TitleLineCount = 5;
```

**X****Nur-Lese-Eigenschaft von VcGroup**

Diese Eigenschaft gibt die aktuelle x-Position der Gruppe in Bandnummern an.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	X-Koordinate

**Y****Nur-Lese-Eigenschaft von VcGroup**

Diese Eigenschaft gibt die aktuelle y-Position der Gruppe in Bandnummern an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Y-Koordinate

## Methoden

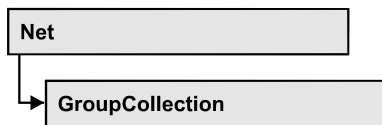
### SetXY

Methode von VcGroup

Mit dieser Methode können Sie die Position der Gruppe setzen. Diese Methode kann nur im Visualisierungsmodus Clusterung (GroupMode = vcGMClustering) angewendet werden, und nur wenn die Gruppe kollabiert ist oder wenn diese Methode im **OnGroupCreate**-Ereignis aufgerufen wird.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ x	System.Int32	X-Koordinate (in Bandnummern)
⇒ y	System.Int32	Y-Koordinate (in Bandnummern)
<b>Rückgabewert</b>	System.Boolean	Koordinaten erfolgreich gesetzt (True)/nicht erfolgreich gesetzt (False)

## 7.23 VcGroupCollection



In einem Objekt vom Typ `VcGroupCollection` sind die bestehenden Gruppen zusammengefasst, sofern Knoten gruppiert worden sind. Über **For Each group In GroupCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Gruppen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaft **GroupByName**. Die Anzahl der im Auflistungsobjekt vorhandenen Gruppen kann über die Eigenschaft **Count** erfragt werden.

### Eigenschaften

- `Count`

### Methoden

- `FirstGroup`
- `GetEnumerator`
- `GroupByName`
- `NextGroup`

---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcGroupCollection**

Mit dieser Eigenschaft kann die Anzahl der Gruppen in der Group-Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Gruppen

### Code-Beispiel VB.NET

```

Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim numberOfGroups As Integer

groupCltn = VcNet1.GroupCollection
numberOfGroups = groupCltn.Count
  
```

**Code-Beispiel C#**

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
int numberOfGroups = groupCltn.Count;
```

---

## Methoden

### FirstGroup

**Methode von VcGroupCollection**

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Gruppe des GroupCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextGroup** über die nachfolgenden Gruppen zu iterieren. Existiert keine Gruppe im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcGroup	Erste Gruppe der GroupCollection

**Code-Beispiel VB.NET**

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
```

**Code-Beispiel C#**

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
```

### GetEnumerator

**Methode von VcGroupCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Gruppen-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

## GroupByName

### Methode von VcGroupCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf eine bestimmte Gruppe zugreifen. Zuvor muss mit der Methode **SelectGroups** die gewünschte Gruppe ausgewählt worden sein. Existiert keine Gruppe unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ Rückgabewert	VcGroup	Gruppe
⇒ groupName	System.String	Gruppenbezeichnung
<b>Rückgabewert</b>	VcGroup	Gruppe

### Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.GroupByName("Group A")
```

### Code-Beispiel C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
```

## NextGroup

### Methode von VcGroupCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Gruppen des GroupCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstGroup** den Initialwert erfasst haben. Sind alle Gruppen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcGroup	Folgegruppe

## 438 API Referenz: VcGroupCollection

### Code-Beispiel VB.NET

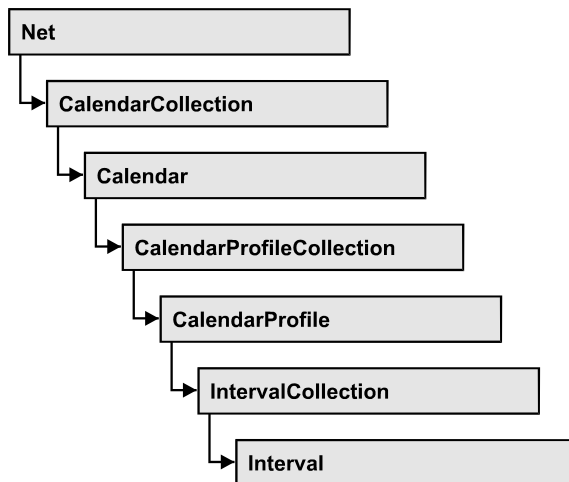
```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcNet1.GroupCollection
group = groupCltn.FirstGroup
While Not group Is Nothing
    ListBox1.Items.Add(group.Name)
    group = groupCltn.NextGroup
End While
```

### Code-Beispiel C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
while (group != null)
{
    listBox1.Items.Add(group.Name);
    group = groupCltn.NextGroup();
}
```

## 7.24 VcInterval



Das Objekt **VcInterval** bietet die Möglichkeit, Zeitintervalle zu definieren, die als Arbeitszeit oder Nicht-Arbeitszeit interpretiert werden. Die Unterscheidung zwischen den beiden Ausprägungen erfolgt durch die speziellen Setzungen **<WORK>** und **<NONWORK>** bei der Eigenschaft **CalendarProfileName**. Ein Intervall kann sich durch die Eigenschaft **CalendarProfileName** auch auf andere bereits definierte Kalenderprofile beziehen.

Je nach vorliegendem Intervalltyp (**vcCalendarInterval**, **vcDayProfileInterval**, **vcWeekProfileInterval**, **vcYearProfileInterval** oder **vcShiftProfileInterval**), der nicht explizit gesetzt wird, sondern sich aus dem Verwendungszusammenhang ergibt, sind nur bestimmte Eigenschaften des Objektes wirksam.

Die folgende Tabelle listet auf, welche Eigenschaften bei den einzelnen Intervalltypen einsetzbar sind:

<b>vcCalendar-Interval</b>	<b>vcYearProfile-Interval</b>	<b>vcWeekProfile-Interval</b>	<b>vcDayProfile-Interval</b>	<b>vcShift-Interval</b>
StartDateTime	StartMonth	StartWeekday	StartTime	Duration
EndDateTime	EndMonth	EndWeekday	EndTime	TimeUnit
	DayInEndMonth			
	DayInStartMonth			

Ein **CalendarInterval** beschreibt eine einmalige Zeitspanne innerhalb eines genau spezifizierten Zeitraums. Beispiel: 5.5.2010 11:30 Uhr bis 15.9.2010 17:00 Uhr.



Ein **YearProfileInterval** erlaubt die Vereinbarung eines jährlich wiederkehrenden Tages oder einer wiederkehrenden Zeitspanne. Beispiel: 1. 5. oder 24.12-26.12.

Ein **WeekProfileInterval** bezieht sich auf einzelne oder mehrere zusammenhängende Tage einer Woche. Beispiel: Samstag oder Montag bis Freitag

Ein **DayProfileProfileInterval** bezieht sich auf die Angabe von Zeiten innerhalb eines Tages. Beispiel: 8.00 bis 17.00 Uhr

Ein **ShiftProfile** beschreibt eine Zeitspanne, die eine bestimmte Dauer in der angegebenen Einheit **vcDay**, **vcHours**, **vcMinute** oder **vcSeconds** ohne Terminbezug darstellt. Beispiel: 4 Stunden

### Eigenschaften

- CalendarProfileName
- DayInEndMonth
- DayInStartMonth
- EndDateTime
- EndMonth
- EndTime
- EndWeekday
- Name
- Specification
- StartDateTime
- StartMonth
- StartTime
- StartWeekday
- Type

### Methoden

- PutInOrderAfter

## Eigenschaften

### CalendarProfileName

**Eigenschaft von VcInterval**

Mit dieser Eigenschaft können Sie dem Intervall ein Kalenderprofil zuweisen oder das verwendete erfragen. Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalenderprofils

### DayInEndMonth

**Eigenschaft von VcInterval**

Mit dieser Eigenschaft können Sie den Tag des letzten Monats des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcYearProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Tag im letzten Monat

### DayInStartMonth

**Eigenschaft von VcInterval**

Mit dieser Eigenschaft können Sie den Tag des ersten Monats des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcYearProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Tag im ersten Monat

## EndTime

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie Enddatum und -zeit des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcCalendar**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Enddatum und -zeit des Intervalls

## EndMonth

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie Endmonat des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcYearProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcMonth  <b>Mögliche Werte:</b> .vcApril 4 .vcAugust 8 .vcDecember 12 .vcFebruary 2 .vcJanuary 1 .vcJuly 7 .vcJune 6 .vcMarch 3 .vcMay 5 .vcNovember 11 .vcOktober 10 .vcSeptember 9	Endmonat des Intervalls  <b>April</b> <b>August</b> <b>Dezember</b> <b>Februar</b> <b>Januar</b> <b>Juli</b> <b>Juni</b> <b>März</b> <b>Mai</b> <b>November</b> <b>Oktober</b> <b>September</b>

## EndTime

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie Endezeit des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcDayProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Endezeit des Intervalls

## EndWeekday

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie den letzten Wochentag des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcWeekProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcWeekday  <b>Mögliche Werte:</b> .vcFriday 5 .vcMonday 1 .vcSaturday 6 .vcSunday 7 .vcThursday 4 .vcTuesday 2 .vcWednesday 3	Letzter Wochentag des Intervalls  Wochentag <b>Freitag</b> Wochentag <b>Montag</b> Wochentag <b>Samstag</b> Wochentag <b>Sonntag</b> Wochentag <b>Donnerstag</b> Wochentag <b>Dienstag</b> Wochentag <b>Mittwoch</b>

## Name

### Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie den Namen eines Intervalls erfragen. Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Intervalls

## Specification

### Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie die Spezifikation dieses Intervalls erfragen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Intervalls mit der Methode **VcIntervalCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Spezifikation des Intervalls

## StartDateTime

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie Startdatum und -zeit des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcCalendar**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Startdatum und -zeit des Intervalls

## StartMonth

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie Startmonat des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcYearProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcMonth  <b>Mögliche Werte:</b> .vcApril 4 .vcAugust 8 .vcDecember 12 .vcFebruary 2 .vcJanuary 1 .vcJuly 7 .vcJune 6 .vcMarch 3 .vcMay 5 .vcNovember 11 .vcOktober 10 .vcSeptember 9	Startmonat des Intervalls  <b>April</b> <b>August</b> <b>Dezember</b> <b>Februar</b> <b>Januar</b> <b>Juli</b> <b>Juni</b> <b>März</b> <b>Mai</b> <b>November</b> <b>Oktober</b> <b>September</b>

## StartTime

### Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie Startzeit des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcDayProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Startzeit des Intervalls

## StartWeekday

Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie den ersten Wochentag des Intervalls setzen oder erfragen (nur für Profile vom Typ **vcWeekProfile**). Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcWeekday  <b>Mögliche Werte:</b> .vcFriday 5 .vcMonday 1 .vcSaturday 6 .vcSunday 7 .vcThursday 4 .vcTuesday 2 .vcWednesday 3	Erster Wochentag des Intervalls  Wochentag <b>Freitag</b> Wochentag <b>Montag</b> Wochentag <b>Samstag</b> Wochentag <b>Sonntag</b> Wochentag <b>Donnerstag</b> Wochentag <b>Dienstag</b> Wochentag <b>Mittwoch</b>

## Type

Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie den Typ des Intervalls erfragen. Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcIntervalType	Typ des Intervalls

## Methoden

### PutInOrderAfter

Methode von VcInterval

Mit dieser Methode können Sie dieses Intervall in der Auflistung aller Intervalle hinter das durch den Namen angegebene setzen. Wenn als Name "" angegeben wird, wird das Intervall an die erste Stelle gesetzt. Die Reihenfolge der Intervalle in der Auflistung entscheidet darüber, in welcher Reihenfolge sie in Kalendern angewendet werden.

## 446 API Referenz: VcInterval

	Datentyp	Beschreibung
<b>Parameter:</b> refName	System.String	Name des Intervalls, hinter das das aktuelle Intervall gesetzt werden soll
<b>Rückgabewert</b>	Void	

### Code-Beispiel VB.NET

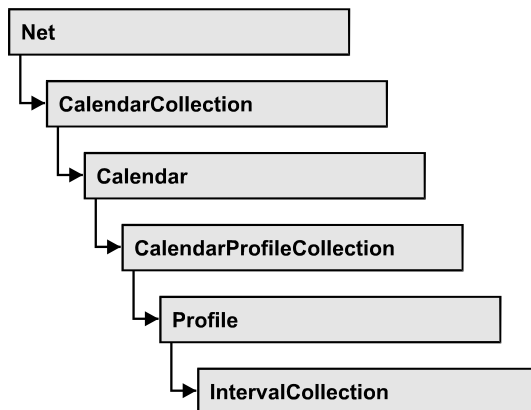
```
Dim intvlCltn As VcIntervalCollection
Dim intvl1 As VcInterval
Dim intvl2 As VcInterval

intvlCltn = VcGantt1.IntervalCollection()
intvl1 = intvlCltn.Add("intvl1")
intvl2 = intvlCltn.Add("intvl2")
intvl1.PutInOrderAfter("intvl2")
intvlCltn.Update()
```

### Code-Beispiel C#

```
VcIntervalCollection intvlCltn = vcGantt1.IntervalCollection;
VcInterval intvl1 = intvlCltn.Add("intvl1");
VcInterval intvl2 = intvlCltn.Add("intvl2");
intvl1.PutInOrderAfter("intvl2");
intvlCltn.Update();
```

## 7.25 VcIntervalCollection



Im VcInterval-Auflistungsobjekt sind alle verfügbaren Intervalle zusammengefasst. Über **For Each interval In IntervalCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Formate zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **IntervalByName** und **IntervalByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Intervalle kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add...**, **Copy...** und **Remove...** ermöglichen das Hinzufügen, Kopieren und Löschen von Intervallen.

### Eigenschaften

- Count

### Methoden

- Add
- AddBySpecification
- Copy
- FirstInterval
- IntervalByIndex
- IntervalByName
- NextInterval
- Remove
- Update



---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcIntervalCollection**

Mit dieser Eigenschaft können Sie die Anzahl der Intervallobjekte in der Intervall-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Intervallobjekte

---

## Methoden

### Add

**Methode von VcIntervalCollection**

Mit dieser Methode können Sie ein neues Intervall in der IntervalCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Intervallobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ intervalName	System.String	Name des Intervalls
<b>Rückgabewert</b>	VcInterval	Neues Intervallobjekt

### AddBySpecification

**Methode von VcIntervalCollection**

Mit dieser Methode können Sie ein Intervall über eine Intervall-Spezifikation erzeugen. Dies dient der Persistenz von Intervallobjekten. Die Spezifikation eines Intervallobjektes kann erfragt (siehe VcInterval-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Intervall mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Specification	System.String	Intervallspezifikation
<b>Rückgabewert</b>	VcInterval	Neues Intervallobjekt

## Copy

### Methode von VcIntervalCollection

Mit dieser Methode können Sie ein Intervall kopieren. Wenn das Intervall mit dem angegebenen Namen existiert und der Name des neuen Intervalls noch nicht verwendet wird, wird das neue Intervallobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ intervalName	System.String	Name des zu kopierenden Intervalls
⇒ newIntervalName	System.String	Name des neuen Intervalls
<b>Rückgabewert</b>	VcInterval	Intervallobjekt

## FirstInterval

### Methode von VcIntervalCollection

Mit dieser Methode können Sie auf den Initialwert, d. h. das erste Intervall der Intervall-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextIntervall** über die nachfolgenden Intervalle zu iterieren. Existiert kein Intervall in der Auflistung, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcInterval	Erstes Intervallobjekt

## IntervalByIndex

### Methode von VcIntervalCollection

Mit dieser Methode können Sie auf ein einzelnes Intervall über seinen Index zugreifen. Existiert kein Intervall unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Index	System.Int16	Index des Intervalls
<b>Rückgabewert</b>	VcInterval	Ermitteltes Intervallobjekt

## IntervalByName

### Methode von VcIntervalCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Intervall zugreifen. Existiert kein Intervall unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ intervalName	System.String	Name des Intervallobjekts
<b>Rückgabewert</b>	VcInterval	Zurückgegebenes Interval-Objekt

## NextInterval

### Methode von VcIntervalCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Intervalle der Intervall-Auflistung zugreifen, nachdem Sie mit der Methode **FirstInterval** den Initialwert erfasst haben. Sind alle Intervalle durchlaufen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcInterval	Nachfolgendes Intervallobjekt

## Remove

### Methode von VcIntervalCollection

Mit dieser Methode können Sie ein Intervall löschen. Wenn das Intervall noch in irgendeinem anderen Objekt verwendet wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ intervalName	System.String	Name des Intervalls
<b>Rückgabewert</b>	System.Boolean	Intervall gelöscht (True)/nicht gelöscht (False)

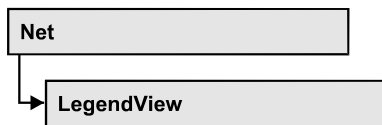
## Update

### Methode von VcIntervalCollection

Mit dieser Methode können Sie eine Intervall-Collection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

## 7.26 VcLegendView



Ein Objekt vom Typ **VcLegendView** bezeichnet das Legendenansicht-Fenster.

### Eigenschaften

- Border
- Height
- HeightActualValue
- Left
- LeftActualValue
- ScrollBarMode
- Top
- TopActualValue
- Visible
- Width
- WidthActualValue
- WindowMode

### Methoden

- Update

---

## Eigenschaften

### Border

**Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann gesetzt oder erfragt werden, ob die Legendenansicht einen Rahmen besitzt (nicht im Modus **vcPopupWindow**). Die Rahmenfarbe ist **Color.Black**. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Rahmen um die Legendenansicht (True)/kein Rahmen um die Legendenansicht (False) <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed
VcNet1.LegendView.Border = True
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed;
vcNet1.LegendView.Border = true;
```

## Height

**Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die vertikale Ausdehnung der Legendenansicht erfragt werden. In den Modi **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** und **vcPopupWindow** kann sie außerdem gesetzt werden.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Höhe der Legendenansicht <b>Standardwert:</b> 100

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Height = 100
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Height = 100;
```

## HeightActualValue

**Nur-Lese-Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die tatsächlich dargestellte vertikale Ausdehnung der Legendenansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Tatsächliche Höhe der Legendenansicht  {0, ...}

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Height = 300
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Height = 100;
```

**Left****Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die linke Position der Legendenansicht erfragt werden. In den Modi **vcNotFixed** und **vcPopupWindow** kann sie außerdem gesetzt werden.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Linke Position der Legendenansicht <b>Standardwert: 0</b>

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Left = 200
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Left = 200;
```

**LeftActualValue****Nur-Lese-Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die tatsächlich dargestellte linke Position der Legendenansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Tatsächliche linke Position der Legendenansicht  {0, ...}

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.LeftActualValue = 150
```

**Code-Beispiel C#**

```
vcNet1.LegendView.LeftActualValue = 150;
```

## ScrollBarMode

**Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann der Scrollbarmodus der Legendenansicht erfragt oder gesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLegendViewScrollBarMode	Scrollbarmodus <b>Standardwert:</b> NoScrollBar
	<b>Mögliche Werte:</b>	
	.vcAutomaticScrollBar 3	Anzeige einer horizontalen oder vertikalen Bildlaufleiste, wenn nötig.
	.vcHorizontalScrollBar 1	Anzeige einer horizontalen Bildlaufleiste, wenn nötig.
	.vcNoScrollBar 0	Es wird immer das vollständige Diagramm ohne Bildlaufleisten angezeigt.
	.vcVerticalScrollBar 2	Anzeige einer vertikalen Bildlaufleiste, wenn nötig.

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.ScrollBarMode = vcAutomaticScrollbar
```

**Code-Beispiel C#**

```
vcNet1.LegendView.ScrollBarMode = vcAutomaticScrollBar;
```

## Top

**Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die obere Position der Legendenansicht erfragt werden. In den Modi **vcNotFixed** und **vcPopupWindow** kann sie außerdem gesetzt werden.



Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Obere Position der Legendenansicht <b>Standardwert: 0</b>

#### Code-Beispiel VB.NET

```
VcNet1.LegendView.Top = 20
```

#### Code-Beispiel C#

```
vcNet1.LegendView.Top = 20;
```

## TopActualValue

**Nur-Lese-Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die tatsächlich dargestellte obere Position der Legendenansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Tatsächliche obere Position der Legendenansicht {0, ...}

#### Code-Beispiel VB.NET

```
VcNet1.LegendView.TopActualValue = 40
```

#### Code-Beispiel C#

```
vcNet1.LegendView.TopActualValue = 40;
```

## Visible

**Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann festgelegt oder erfragt werden, ob die Legendenansicht sichtbar ist. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Legendenansicht sichtbar (True)/unsichtbar (False) <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Visible = True
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Visible = true;
```

## Width

**Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die horizontale Ausdehnung der Legendenansicht erfragt werden. In den Modi **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** und **vcPopupWindow** kann diese Eigenschaft außerdem gesetzt werden.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Horizontale Ausdehnung der Legendenansicht <b>Standardwert:</b> 100

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Width = 200
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Width = 200;
```

## WidthActualValue

**Nur-Lese-Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann die tatsächlich dargestellte horizontale Ausdehnung der Legendenansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Tatsächliche horizontale Ausdehnung der Legendenansicht  {0, ...}

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.WidthActualValue = 600
```

**Code-Beispiel C#**

```
vcNet1.LegendView.WidthActualValue = 600;
```

**WindowMode****Eigenschaft von VcLegendView**

Mit dieser Eigenschaft kann der Modus der Legendenansicht erfragt oder gesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLegendViewWindowMode	Modus der Gesamtansicht  <b>Standardwert:</b> vcPopupWindow
	<b>Mögliche Werte:</b>	
	.vcFixedAtBottom 4	Die Legendenansicht wird unten im Fenster des VARCHART .NET Steuerelements angezeigt. Dann kann nur die Höhe festgelegt werden, während Position und Breite vorgegeben sind.
	.vcFixedAtLeft 1	Die Legendenansicht wird links im Fenster des VARCHART .NET Steuerelements angezeigt. Dann kann nur die Breite festgelegt werden, während Position und Höhe vorgegeben sind.
	.vcFixedAtRight 2	Die Legendenansicht wird rechts im Fenster des VARCHART .NET Steuerelements angezeigt. Dann kann nur die Breite festgelegt werden, während Position und Höhe vorgegeben sind.
	.vcFixedAtTop 3	Die Legendenansicht wird oben im Fenster des VARCHART .NET Steuerelements angezeigt. Dann kann nur die Höhe festgelegt werden, während Position und Breite vorgegeben sind.
	.vcNotFixed 5	Die Legendenansicht ist ein untergeordnetes Kindfenster des aktuellen Vaterfensters des VARCHART .NET Steuerelements und kann an beliebiger Position mit beliebiger Ausdehnung angeordnet werden. Das Vaterfenster kann bei Bedarf über die Eigenschaft
	.vcPopupWindow 6	<b>VcLegendView.ParentHWnd</b> geändert werden. Die Legendenansicht ist ein Popup-Fenster, das einen eigenen Rahmen besitzt und vom Benutzer in Position und Größe verändert werden kann. Es kann über das Standard-Kontextmenü ein- bzw. ausgeschaltet oder über die <b>Schließen</b> -Schaltfläche in der Titelleiste ausgeschaltet werden.

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed
```

**Code-Beispiel C#**

```
vcNet1.LegendView.Mode = VcLegendViewMode.vcNotFixed;
```

---

## Methoden

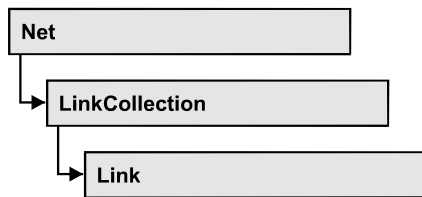
### Update

Methode von VcLegendView

Mit dieser Methode wird die Legende aktualisiert.

	Datentyp	Beschreibung

## 7.27 VcLink



Ein Verbindungsobjekt stellt die logische und grafische Verbindung zwischen zwei Knoten dar. Das Aussehen wird durch diejenigen LinkAppearance-Objekte bestimmt, deren Filter auf die Verbindungsdaten zutreffen. Sie können Verbindungen entweder interaktiv oder über die Methode **InsertLinkRecord** des **VcNet**-Objektes erzeugen.

### Eigenschaften

- AllData
- DataField
- ID
- Marked
- PredecessorNode
- SuccessorNode

### Methoden

- DataRecord
- Delete
- RelatedDataRecord
- Update

---

## Eigenschaften

### AllData

Eigenschaft von VcLink

Mit dieser Eigenschaft können alle Daten für die Verbindung gesetzt oder erfragt werden. Beim Setzen ist ein CSV-String (Semikolon als Trennzeichen) oder ein Datenfeld erlaubt, beim Erfragen wird ein String zurückgegeben. (siehe auch **InsertLinkRecord**.)

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Alle Daten der Verbindung

**Code-Beispiel VB.NET**

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

linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
allDataOfLink = link.AllData
```

**Code-Beispiel C#**

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();
string allDataOfLink = link.AllData.ToString();
```

**DataField****Eigenschaft von VcLink**

Mit dieser Eigenschaft kann ein einzelnes Datenfeld einer Verbindung gesetzt oder erfragt werden. Die Werte, die den Vorgänger- oder Nachfolgerknoten identifizieren, dürfen nicht verändert werden.

Die Eigenschaft DataField ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set\_DataField (index, pvn) und get\_DataField (index) angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Datenfeldes
<b>Eigenschaftswert</b>	System.Object	Inhalt des Datenfelds

**Code-Beispiel VB.NET**

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

linkCltn = VcNet1.LinkCollection
For Each link In linkCltn
    message = "Delete link from " + link.DataField(1) + " to " +
    link.DataField(2) + " ?"
    If MsgBox(message, MsgBoxStyle.OKCancel, "Delete Link") = MsgBoxResult.OK
    Then
        link.Delete()
    End If
Next
```

### Code-Beispiel C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;

foreach (VcLink link in linkCltn)
{
    DialogResult retVal = MessageBox.Show("Delete link from " +
link.get_DataField(1) + " to " + link.get_DataField(2) + " ?", "Deleting curve
point", MessageBoxButtons.OKCancel);
    if (retVal == DialogResult.OK)
        link.Delete();
}
```

## ID

**Nur-Lese-Eigenschaft von VcLink**

Mit dieser Eigenschaft können Sie die ID einer Verbindung erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Verbindungs-ID

## Marked

**Nur-Lese-Eigenschaft von VcLink**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob diese Verbindung markiert ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	Boolean	Verbindung markiert (True)/nicht markiert (False)

## PredecessorNode

**Nur-Lese-Eigenschaft von VcLink**

Mit dieser Eigenschaft kann der Vorgängerknoten einer Verbindung identifiziert werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcNode	Vorgängerknoten

**Code-Beispiel VB.NET**

```

Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
node = link.PredecessorNode
nodeName = node.DataField(1)

```

**Code-Beispiel C#**

```

VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();
VcNode node = link.PredecessorNode;
string nodeName = node.get_DataField(1).ToString();

```

**SuccessorNode****Nur-Lese-Eigenschaft von VcLink**

Mit dieser Eigenschaft kann der Nachfolgerknoten einer Verbindung identifiziert werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcNode	Nachfolgerknoten

**Code-Beispiel VB.NET**

```

Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
node = link.SuccessorNode
nodeName = node.DataField(1)

```

**Code-Beispiel C#**

```

VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();

VcNode node = link.SuccessorNode;
string nodeName = node.get_DataField(1).ToString();

```



## Methoden

### DataRecord

Methode von VcLink

Mit dieser Eigenschaft können Sie die Verbindung als Datensatzobjekt erfragen. Über die Eigenschaften des Datensatzobjektes haben Sie auch Zugriff auf die entsprechende Datentabelle und Tabellenauflistung.

	Datentyp	Beschreibung
Rückgabewert	VcDataRecord	Zurückgegebener Datensatz

### Delete

Methode von VcLink

Mit dieser Methode können Sie eine Verbindung löschen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Verbindung erfolgreich /nicht erfolgreich gelöscht

#### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcLinksRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksRightClicking
    Dim message As String
    message = "Delete link: " + e.LinkCollection.FirstLink.AllData

    If MsgBox(message, MsgBoxStyle.OKCancel, "Delete link") = MsgBoxResult.OK Then
        e.LinkCollection.FirstLink.Delete()
    End If
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

#### Code-Beispiel C#

```
private void vcNet1_VcLinksRightClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
{
    string message = "Delete link: " + e.LinkCollection.FirstLink().AllData;
    DialogResult retVal = MessageBox.Show(message, "Deleting link",
MessageBoxButtons.OKCancel);
    if (retVal == DialogResult.OK)
        e.LinkCollection.FirstLink().Delete();
    else
        e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

## RelatedDataRecord

### Methode von VcLink

Mit dieser Methode können Sie einen Datensatz aus einer verknüpften Tabelle erfragen, der dem Datensatz der Verbindungsdatentabelle zugeordnet ist. Der im Parameter übergebene Index bezeichnet das Feld im Datensatz, in dem der Schlüssel des zugeordneten Datensatzes steht.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Datenfeldes, das den Schlüssel enthält
<b>Rückgabewert</b>	VcDataRecord	Zurückgegebener zugeordneter Datensatz

## Update

### Methode von VcLink

Wenn ein Datenfeld einer Verbindung mit der Eigenschaft **DataField** verändert wurde, so kann mit der Methode **Update** die Darstellung aktualisiert werden.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Verbindung erfolgreich /nicht erfolgreich aktualisiert

### Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

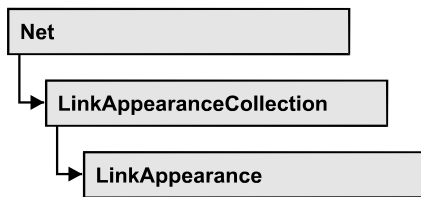
linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
link.DataField(2) = 10
link.Update()
```

### Code-Beispiel C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();

link.set_DataField(2, 10);
link.Update();
```

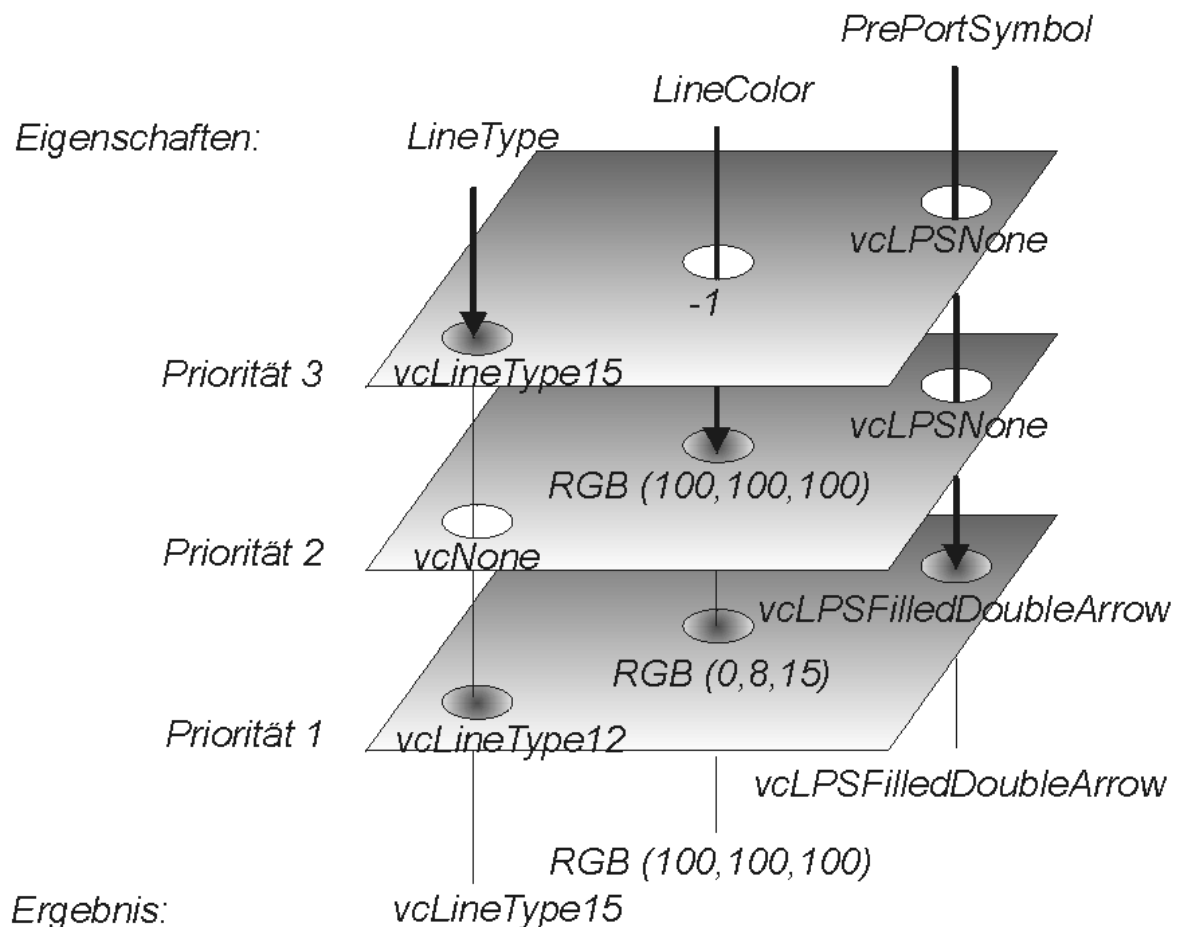
## 7.28 VcLinkAppearance



Ein LinkAppearance-Objekt bestimmt das Aussehen aller Verbindungen, deren Daten die Filterbedingungen erfüllen, die dem LinkAppearance-Objekt zugeordnet sind. Verschiedene LinkAppearance-Objekte können auf der Eigenschaftenseite **Verbindungen** in der **Aussehen**-Tabelle voreingestellt werden.

Die Skizze zeigt, wie sich die Eigenschaften von LinkAppearance-Objekten auf das Aussehen einer Verbindung auswirken. Die auf die Verbindung zutreffenden LinkAppearances sind nach Priorität absteigend dargestellt.

Nicht gesetzte Eigenschaften bei LinkAppearance-Objekten führen dazu, daß die Eigenschaft des nächsttieferen LinkAppearance-Objekts übernommen wird.



## Eigenschaften

- FilterName
- FormatName
- LineColor
- LineThickness
- LineType
- Name
- PredecessorPortSymbol
- RoutingType
- Specification
- SuccessorPortSymbol
- Visible

## Methoden

- PutInOrderAfter

---

## Eigenschaften

### FilterName

Nur-Lese-Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie den Filter erfragen, der für ein bestimmtes Verbindungsausssehen verwendet wird. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Verbindungen** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Filtername

#### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim filterOfLinkApp As String

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
filterOfLinkApp = linkAppearance.FilterName
```

#### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
string filterOfLinkApp = linkAppearance.FilterName;
```

## FormatName

### Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie das Format des LinkAppearance-Objekts setzen oder erfragen. Bei **Nothing** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren LinkAppearance-Objektes zum Tragen, dessen Filterbedingungen für die Verbindung ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **Nothing** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name eines Verbindungsformat-Objekts oder leere Zeichenkette

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
MsgBox (nodeAppearance.FormatName)
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
MessageBox.Show (nodeAppearance.FormatName);
```

## LineColor

### Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie die Farbe eines LinkAppearance-Objekts erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialogfeld **Linienart**, das Sie über die Eigenschaftenseite **Verbindungen** erreichen, festlegen

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color RGB {(0...255},{0...255},{0...255)}	RGB-Farbwerte

### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName ("Blue")
linkAppearance.LineColor = Color.Blue
```

**Code-Beispiel C#**

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
linkAppearance.LineColor = Color.LightSteelBlue;
```

**LineThickness****Eigenschaft von VcLinkAppearance**

Mit dieser Eigenschaft können Sie die Linienstärke eines LinkAppearance-Objekts erfragen oder festlegen.

Wenn Sie diese Eigenschaft auf Werte zwischen 1 und 4 setzen, wird damit eine absolute Liniendicke in Pixel definiert, d.h. die Linien haben unabhängig vom Zoomfaktor immer die gleiche feste Linienstärke in Pixeln. Dies wird jedoch aufgrund der besseren Lesbarkeit beim Drucken in eine vom Zoomfaktor abhängige Liniendicke umgewandelt:

Wert	Punkte	mm
1	1/2 Punkt	0,09 mm
2	1 Punkt	0,18 mm
3	3/2 Punkt	0,26 mm
4	2 Punkt	0,35 mm

Ein Punkt ist 1/72 Zoll groß und stellt die Maßeinheit für Schriftgrößen dar.

Wenn Sie diese Eigenschaft auf Werte zwischen 5 und 1.000 setzen, wird damit eine Linienstärke in 1/100 mm definiert, d.h. die Linien bekommen eine tatsächliche Dicke in Pixeln, die abhängig vom Zoomfaktor ist.

Sie können diese Eigenschaft auch im Dialogfeld **Linienart**, das Sie über die Eigenschaftenseite **Verbindungen** erreichen, festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm <b>Standardwert:</b> As defined on property page

## Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
linkAppearance.LineThickness = 4
```

## Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Standard");
linkAppearance.LineThickness = 4;
```

## LineType

### Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie den Linientyp eines LinkAppearance-Objekts erfragen oder festlegen.

Sie können diese Eigenschaft auch im Dialogfeld **Linienart**, das Sie über die Eigenschaftenseite **Verbindungen** erreichen, festlegen.

Eigenschaftswert	Datentyp	Beschreibung
	VcLineType	Linientyp <b>Standardwert:</b> vcSolid
	<b>Mögliche Werte:</b>	
	.vcDashed 4	Linientyp <b>gestrichelt</b>
	.vcDashed 4	Linientyp <b>gestrichelt</b>
	.vcDashedDotted 5	Linientyp <b>gestrichelt-gepunktet</b>
	.vcDashedDotted 5	Linientyp <b>gestrichelt-gepunktet</b>
	.vcDotted 3	Linientyp <b>gepunktet</b>
	.vcDotted 3	Linientyp <b>gepunktet</b>
	.vcLineType0 100	Linientyp 0 _____
	.vcLineType1 101	Linientyp 1 - - - - -
	.vcLineType10 110	Linientyp 10 _ _ _ _ _
	.vcLineType11 111	Linientyp 11 _ . . . .
	.vcLineType12 112	Linientyp 12 _ _ _ . .
	.vcLineType13 113	Linientyp 13 _ _ _ _ .
	.vcLineType14 114	Linientyp 14 _ _ _ _ _
	.vcLineType15 115	Linientyp 15 _ . . . .
	.vcLineType16 116	Linientyp 16 _ _ _ . .
	.vcLineType17 117	Linientyp 17 _ . . . .
	.vcLineType18 118	Linientyp 18 _ _ _ . .

.vcLineType2 102	Linientyp 2 .....
.vcLineType3 103	Linientyp 3 -----
.vcLineType4 104	Linientyp 4 -----
.vcLineType5 105	Linientyp 5 -----
.vcLineType6 106	Linientyp 6 -----
.vcLineType7 107	Linientyp 7 -----
.vcLineType8 108	Linientyp 8 -----
.vcLineType9 109	Linientyp 9 -----
.vcNone 1	Kein Linientyp zugewiesen
.vcNone 1	Kein Linientyp
.vcNotSet -1	Kein Linientyp zugewiesen
.vcSolid 2	Linientyp <b>durchgezogen</b>
.vcSolid 2	Linientyp <b>durchgezogen</b>

**Code-Beispiel VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineType = 5
```

**Code-Beispiel C#**

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
linkAppearance.LineType = VcLineType.vcLineType5;
```

**Name****Nur-Lese-Eigenschaft von VcLinkAppearance**

Mit dieser Eigenschaft können Sie den Namen der LinkAppearance erfragen.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Verbindungsansiehens



### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
nameLinkApp = linkAppearance.Name
```

### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
string nameLinkApp = linkAppearance.Name;
```

## PredecessorPortSymbol

### Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie einer Verbindung ein Portsymbol, das die Einmündung zum Vorgängerknoten kennzeichnet, zuweisen oder erfragen.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkPredecessorPortSymbol	Symbol am Vorgängerknoten <b>Standardwert:</b> vcLPSNone

### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSDoubleSemiCircle
```

### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSFilledDoubleSemiCircle;
```

## RoutingType

### Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Verbindungen im Diagramm nur waagrecht und senkrecht (und damit

orthogonal) verlaufen dürfen, oder ob sie auch schräg (d.h. durch Objekte hindurchschneidend) verlaufen dürfen.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcRoutingType	Routentyp <b>Standardwert:</b> vcLRTOρθogonal

## Specification

### Nur-Lese-Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie die Spezifikation dieses Verbindungsaussehens auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Verbindungsaussehens mit der Methode **VcLinkAppearanceCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Spezifikation des Verbindungsaussehens

### Code-Beispiel C#

```
VcBoxCollection boxCltn =vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

## SuccessorPortSymbol

### Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie einer Verbindung ein Portsymbol, das die Einmündung zum Nachfolgerknoten kennzeichnet, zuweisen oder erfragen.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden.

## 474 API Referenz: VcLinkAppearance

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkSuccessorPortSymbol	Symbol am Nachfolgerknoten <b>Standardwert:</b> vcLSSNone

### Code-Beispiel VB.NET

```
VcLinkAppearanceCollection linkAppearanceCltn = VcNet1.LinkAppearanceCollection;  
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();  
linkAppearance.SuccessorPortSymbol =  
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;  
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();  
linkAppearance.SuccessorPortSymbol =  
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

## Visible

### Eigenschaft von VcLinkAppearance

Mit der Eigenschaft **Visible** können Sie erfragen oder festlegen, ob die Verbindungen - nicht jedoch die Phantomlinien für Links bei Interaktionen - sichtbar sein sollen oder nicht.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden, gilt dort jedoch auch für die Phantomlinien.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft/nicht in Kraft <b>Standardwert:</b> True

### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection  
Dim linkAppearance As VcLinkAppearance  
  
linkAppearanceCltn = VcNet1.LinkAppearanceCollection  
linkAppearance = linkAppearanceCltn.FirstLinkAppearance  
linkAppearance.Visible = False
```

### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;  
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();  
linkAppearance.Visible = false;
```

## Methoden

### PutInOrderAfter

Methode von VcLinkAppearance

Mit dieser Methode können Sie dieses Verbindungsaussehen in der Auflistung aller Verbindungsaussehen hinter das durch den Namen angegebene setzen. Wenn als Name "" angegeben wird, wird das Verbindungsaussehen an die erste Stelle gesetzt. Die Reihenfolge der Verbindungsaussehen in der Auflistung entscheidet darüber, in welcher Reihenfolge sie auf die Verbindungen angewendet werden.

	Datentyp	Beschreibung
<b>Parameter:</b> refLinkAppearanceName	System.String	Name des Verbindungsaussehens, hinter das das aktuelle Verbindungsaussehen gesetzt werden soll

#### Code-Beispiel VB.NET

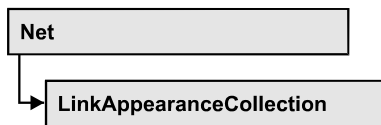
```
Dim linkAppCltn As VcLinkAppearanceCollection
Dim linkApp1 As VcLinkAppearance
Dim linkApp2 As VcLinkAppearance

linkAppCltn = VcGantt1.LinkAppearanceCollection()
linkApp1 = linkAppCltn.Add("linkApp1")
linkApp2 = linkAppCltn.Add("linkApp2")
linkApp1.PutInOrderAfter("linkApp2")
linkAppCltn.Update()
```

#### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppCltn = vcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkApp1 = linkAppCltn.Add("linkApp1");
VcLinkAppearance linkApp2 = linkAppCltn.Add("linkApp2");
linkApp1.PutInOrderAfter("linkApp2");
linkAppCltn.Update();
```

## 7.29 VcLinkAppearanceCollection



In einem Objekt vom Typ **VcLinkAppearanceCollection** sind automatisch alle definierten LinkAppearance-Objekte zusammengefasst. Über **For Each linkAppearance In LinkAppearanceCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Verbindungsaussichten zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **LinkAppearanceByName** und **LinkAppearanceByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Verbindungsaussichten kann über die Eigenschaft **Count** erfragt werden.

### Eigenschaften

- Count

### Methoden

- Add
- AddBySpecification
- Copy
- FirstLinkAppearance
- GetEnumerator
- LinkAppearanceByIndex
- LinkAppearanceByName
- NextLinkAppearance
- Remove
- Update

---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcLinkAppearanceCollection**

Mit dieser Eigenschaft kann die Anzahl der LinkAppearance-Objekte in der LinkAppearance-Auflistung erfragt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Anzahl der LinkAppearance-Objekte

**Code-Beispiel VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim numberOfLinkAppearance As Integer
```

```
linkAppearanceCltn = VcNet1.LinkAppearanceCollection
numberOfLinkAppearance = linkAppearanceCltn.Count
```

**Code-Beispiel C#**

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
int numberOfLinkAppearance = linkAppearanceCltn.Count;
```

---

## Methoden

### Add

**Methode von VcLinkAppearanceCollection**

Mit dieser Methode können Sie ein neues Verbindungsaussehen in der LinkAppearance-Auflistung anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue VcLinkAppearance-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Bei dem neuen Verbindungsaussehen sind standardmäßig alle Eigenschaften auf transparent gesetzt.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ newName	System.String	Name des LinkAppearance-Objekts
<b>Rückgabewert</b>	VcLinkAppearance	Neues LinkAppearance-Objekt

**Code-Beispiel VB.NET**

```
newLinkAppearance = VcNet1.LinkAppearanceCollection.Add("linkapp1")
```

**Code-Beispiel C#**

```
newLinkAppearance = vcNet1.LinkAppearanceCollection.Add("linkapp1");
```

## AddBySpecification

### Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie ein Verbindungsaussehen über eine Verbindungsaussehen-Spezifikation erzeugen. Dies ermöglicht die Persistenz von Verbindungsaussehen-Objekten. Die Spezifikation eines Verbindungsaussehens kann erfragt (siehe VcLinkAppearance-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Verbindungsaussehen mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ linkAppearanceSpecification	System.String	Verbindungsaussehen-Spezifikation
<b>Rückgabewert</b>	VcLinkAppearance	Neues Verbindungsaussehen-Objekt

## Copy

### Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie ein Verbindungsaussehen kopieren. Wenn das Verbindungsaussehen mit dem angegebenen Namen existiert und der Name des neuen Verbindungsaussehens noch nicht verwendet wird, wird das neue Verbindungsaussehen-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ fromName	System.String	Name des zu kopierenden Verbindungsaussehens
⇒ newName	System.String	Name des neuen Verbindungsaussehens
<b>Rückgabewert</b>	VcLinkAppearance	LinkAppearance-Objekt

## FirstLinkAppearance

### Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste LinkAppearance-Objekt der LinkAppearanceCollection zugreifen, um anschließend in einer Schleife mit der Methode **NextLinkAppearance** über die nachfolgenden Objekte zu iterieren. Existiert kein LinkAppearance-

Objekt in der LinkAppearanceCollection, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLinkAppearance	Erstes LinkAppearance-Objekt

#### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
```

#### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
```

## GetEnumerator

### Methode von VcLinkAppearanceCollection

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Verbindungsausssehen-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

## LinkAppearanceByIndex

### Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie auf ein einzelnes LinkAppearance-Objekt über seinen Index zugreifen. Existiert kein LinkAppearance-Objekt an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des LinkAppearance-Objektes
<b>Rückgabewert</b>	VcLinkAppearance	Ermitteltes LinkAppearance-Objekt



## LinkAppearanceByName

Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes LinkAppearance-Objekt zugreifen. Existiert kein LinkAppearance-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ linkAppearanceName	System.String	Name des LinkAppearance-Objektes
<b>Rückgabewert</b>	VcLinkAppearance	LinkAppearance-Objekt

### Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
```

### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Standard");
```

## NextLinkAppearance

Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden LinkAppearance-Objekte der LinkAppearanceCollection zugreifen, nachdem Sie mit der Methode **FirstLinkAppearance** den Initialwert erfasst haben. Sind alle LinkAppearance-Objekte durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcLinkAppearance	Nachfolgendes LinkAppearance-Objekt

**Code-Beispiel VB.NET**

```

Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
While Not linkAppearance Is Nothing
    linkAppearance.Visible = False
    ListBox1.Items.Add("Name: " + linkAppearance.Name)
    linkAppearance = linkAppearanceCltn.NextLinkAppearance
End While

```

**Code-Beispiel C#**

```

VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
while (linkAppearance != null)
{
    linkAppearance.Visible = false;
    listBox1.Items.Add("Name: " + linkAppearance.Name);
    linkAppearance = linkAppearanceCltn.NextLinkAppearance();
}

```

**Remove****Methode von VcLinkAppearanceCollection**

Mit dieser Methode können Sie ein Verbindungsaussehen löschen. Wenn das Verbindungsaussehen noch irgendwo verwendet wird, kann es nicht gelöscht werden. In diesem Fall wird **False** zurückgegeben, sonst **True**.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ name	System.String	Name des Verbindungsaussehens
<b>Rückgabewert</b>	System.Boolean	Verbindungsaussehen gelöscht (True)/nicht gelöscht (False)

**Update****Methode von VcLinkAppearanceCollection**

Mit dieser Methode können Sie die Auflistung von Verbindungsaussehen aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Auflistung erfolgreich /nicht erfolgreich aktualisiert

## 482 API Referenz: VcLinkAppearanceCollection

### Code-Beispiel VB.NET

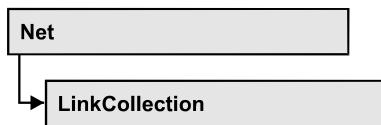
```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcNet1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0)
linkAppearanceCltn.Remove(linkAppearance.Name)
linkAppearanceCltn.Update()
```

### Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0);
linkAppearanceCltn.Remove(linkAppearance.Name);
linkAppearanceCltn.Update();
```

## 7.30 VcLinkCollection



In einem Objekt vom Typ **VcLinkCollection** sind automatisch alle definierten Verbindungsobjekte zusammengefasst. Über **For Each link In-LinkCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Verbindungen zugreifen. Die Anzahl der im Auflistungsobjekt vorhandenen Verbindungen kann über die Eigenschaft **Count** erfragt werden.

### Eigenschaften

- Count

### Methoden

- FirstLink
- GetEnumerator
- NextLink
- SelectLinks

---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcLinkCollection**

Mit dieser Eigenschaft kann die Anzahl der Verbindungen in der Link-Auflistung abgefragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Verbindungen

### Code-Beispiel VB.NET

```

Dim linkCltn As VcLinkCollection
Dim numberOfLinks As Integer

linkCltn = VcNet1.LinkCollection
numberOfLinks = linkCltn.Count
  
```

**Code-Beispiel C#**

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
int numberOfLinks = linkCltn.Count;
```

## Methoden

### FirstLink

**Methode von VcLinkCollection**

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Verbindung des LinkCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextLink** über die nachfolgenden Verbindungen zu iterieren. Existiert keine Verbindung in der LinkCollection, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLink	Erste Verbindung

**Code-Beispiel VB.NET**

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
```

```
linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
```

**Code-Beispiel C#**

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();
```

### GetEnumerator

**Methode von VcLinkCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Verbindungsobjekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

## NextLink

### Methode von VcLinkCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Verbindungen des LinkCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstLink** den Initialwert erfasst haben. Sind alle Verbindungen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLink	Folgeverbindung

### Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

linkCltn = VcNet1.LinkCollection
link = linkCltn.FirstLink
While Not link Is Nothing
    ListBox1.Items.Add(link.AllData)
    link = linkCltn.NextLink
End While
```

### Code-Beispiel C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
VcLink link = linkCltn.FirstLink();

while (link != null)
{
    listBox1.Items.Add(link.AllData);
    link = linkCltn.NextLink();
}
```

## SelectLinks

### Methode von VcLinkCollection

Mit dieser Methode können Sie festlegen, welche Verbindungen im LinkCollection-Objekt verfügbar sein sollen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ selectionType	VcSelectionType  <b>Mögliche Werte:</b> .vcAll 0 .vcAllLinksCausingCycles 7	Auszuwählende Verbindungen  Alle Objekte im Diagramm werden ausgewählt. Wird diese Selektion gewählt, dann befinden sich in der LinkCollection alle Verbindungen, die tatsächlich Zyklen verursachen, d.h. würde diese minimale Anzahl Verbindungen gelöscht, gäbe es keine Zyklen mehr.

## 486 API Referenz: VcLinkCollection

	<code>.vcAllLinksInCycles 6</code> <code>.vcAllVisible 1</code> <code>.vcMarked 2</code>	Wird dieser Selektionstyp gewählt, dann befinden sich in der LinkCollection alle Verbindungen, die Zyklen bilden. Zyklen sind geschlossene Ketten von Knoten und Verbindungen. Alle sichtbaren Objekte werden ausgewählt. Alle markierten Objekte werden ausgewählt.
<b>Rückgabewert</b>	System.Int32	Anzahl ausgewählter Verbindungen

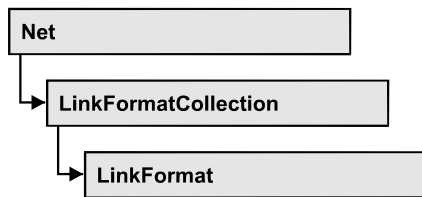
### Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection  
linkCltn = VcNet1.LinkCollection  
linkCltn.SelectGroups (vcAllMarked)
```

### Code-Beispiel C#

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;  
linkCltn.SelectGroups (vcAllMarked);
```

## 7.31 VcLinkFormat



Ein Objekt vom Typ VcLinkFormat legt Inhalt und Erscheinungsbild einer Verbindung fest. Verbindungsformate werden zur Designzeit im Dialogfeld **Verbindungsformate verwalten**, das Sie über die Eigenschaftenseite **Verbindungen** erreichen, verwaltet und bearbeitet.

### Eigenschaften

- FormatField
- FormatFieldCount
- Name
- Specification

### Methoden

- CopyFormatField
- GetEnumerator
- RemoveFormatField

---

## Eigenschaften

### FormatField

**Nur-Lese-Eigenschaft von VcLinkFormat**

Mit dieser Eigenschaft können Sie ein VcLinkFormatField-Objekt per Index holen. Der Index muss im Bereich von 0 bis FormatFieldCount-1 liegen.

	Datentyp	Beschreibung
<b>Parameter:</b> index	System.Int16	Index des Verbindungsformatfeldes
<b>Eigenschaftswert</b>	VcNodeFormatField	Verbidnungsformatfeld



## FormatFieldCount

**Nur-Lese-Eigenschaft von VcLinkFormat**

Mit dieser Eigenschaft können Sie die Anzahl der Felder eines Verbindungsformats ermitteln.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Anzahl der Felder im Verbindungsformat

### Code-Beispiel VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.FormatFieldCount)
```

### Code-Beispiel C#

```
VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.FormatFieldCount.ToString());
```

## Name

**Eigenschaft von VcLinkFormat**

Mit dieser Eigenschaft können Sie den Namen des Verbindungsformats erfragen oder setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Verbindungsformats

### Code-Beispiel VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.Name)
```

### Code-Beispiel C#

```
VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.Name);
```

## Specification

**Nur-Lese-Eigenschaft von VcLinkFormat**

Mit dieser Eigenschaft können Sie die Spezifikation dieses Verbindungsformats auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht

Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Verbindungsformats mit der Methode **VcLinkFormatCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Spezifikation des Verbindungsformats

#### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

#### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

## Methoden

### CopyFormatField

Methode von VcLinkFormat

Mit dieser Methode können Sie ein Verbindungsformatfeld kopieren. Das neue VcNodeLinkField-Objekt wird zurückgegeben. Es erhält den nächsten, noch nicht vergebenen Index.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ position	VcFormatFieldPosition  <b>Mögliche Werte:</b> .vcAbove 1 .vcBelow 3 .vcLeftOf 0 .vcOutsideAbove 9 .vcOutsideBelow 11 .vcOutsideLeftOf 8 .vcOutsideRightOf 12 .vcRightOf 4	Position des neuen Verbindungsformatfeldes  oberhalb unterhalb links von außerhalb, oberhalb außerhalb, unterhalb außerhalb, links von außerhalb, rechts von rechts von
⇒ refIndex	System.Int16	Index des Referenz-Verbindungsformatfeldes
<b>Rückgabewert</b>	VcLinkFormatField	Neu angelegtes Verbindungsformatfeld-Objekt

## GetEnumerator

Methode von VcLinkFormat

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Knotenformatfelder iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

### Code-Beispiel VB.NET

```
Dim format As VcNodeFormat
For Each format In VcNet1.NodeFormatCollection
    Debug.Write(format.Name)
Next
```

### Code-Beispiel C#

```
foreach (VcNodeFormat format in vcNet1.NodeFormatCollection)
    Console.Write(format.Name);
```

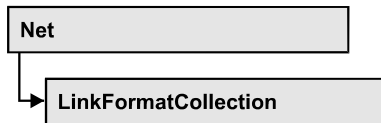
## RemoveFormatField

Methode von VcLinkFormat

Mit dieser Methode können Sie ein Verbindungsformatfeld über den angegebenen Index löschen. Anschließend wird ggf. der Index aller Verbindungsformatfelder neu festgesetzt, so dass sie wieder fortlaufend numeriert sind.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des zu löschenden Verbindungsformatfeldes

## 7.32 VcLinkFormatCollection



In einem Objekt vom Typ `VcLinkFormatCollection` sind automatisch alle verfügbaren Verbindungsformate zusammengefasst. Über **For Each format InLinkFormatCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Verbindungsformate zugreifen. Sie haben Zugriff auf bestimmte Formate über die Eigenschaft **FormatByName**. Die Anzahl der im Auflistungsobjekt vorhandenen Verbindungsformate kann über die Eigenschaft **Count** erfragt werden.

### Eigenschaften

- `Count`

### Methoden

- `Add`
- `AddBySpecification`
- `Copy`
- `FirstFormat`
- `FormatByIndex`
- `FormatByName`
- `GetEnumerator`
- `NextFormat`
- `Remove`

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von `VcLinkFormatCollection`

Mit dieser Eigenschaft können Sie die Anzahl der Verbindungsformatobjekte in der Verbindungsformat-Auflistung abfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Verbindungsformate

### Code-Beispiel VB.NET

```
Dim formatCltn As VcLinkFormatCollection
Dim numberOfFormats As Integer

formatCltn = VcNet1.LinkFormatCollection
numberOfFormats = formatCltn.Count
```

### Code-Beispiel C#

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
int numberOfFormats = formatCltn.Count;
```

---

## Methoden

### Add

#### Methode von VcLinkFormatCollection

Mit dieser Methode können Sie ein neues Verbindungsformat in der LinkFormatCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue VcLinkFormat-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

Das Verbindungsformat besitzt standardmäßig folgende Eigenschaften:

- ein einziges Feld
- WidthOfExteriorSurrounding: 3 mm

Das Feld hat folgende Eigenschaften:

- Type: vcFFTText
- TextDataFieldIndex: in der Eigenschaftenseite **Allgemeines** festgelegte IDMinimumWidth: 3000
- Alignment: vcFFACenter
- BackColor: -1 (transparent)
- TextFontColor: RGB(0,0,0) (schwarz)
- TextFont: Arial, 10, normal
- LeftMargin, RightMargin, TopMargin, BottomMargin: 0,3 mm

- MinimumTextLineCount, MaximumTextLineCount: 1

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ newName	System.String	Name des Verbindungsformats
<b>Rückgabewert</b>	VcLinkFormat	Verbindungsformat-Objekt

#### Code-Beispiel VB.NET

```
newLinkFormat = vcNet1.LinkFormatCollection.Add("linkformat1")
```

#### Code-Beispiel C#

```
newLinkFormat = vcNet1.LinkFormatCollection.Add("linkformat1");
```

## AddBySpecification

### Methode von VcLinkFormatCollection

Mit dieser Methode können Sie ein Verbindungsformat über eine Verbindungsformat-Spezifikation erzeugen. Dies ermöglicht die Persistenz von Verbindungsformat-Objekten. Die Spezifikation eines Verbindungsformats kann erfragt (siehe VcLinkFormat-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Verbindungsformat mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ formatSpecification	System.String	Verbindungsformat-Spezifikation
<b>Rückgabewert</b>	VcLinkFormat	Neues Verbindungsformat-Objekt

## Copy

### Methode von VcLinkFormatCollection

Mit dieser Methode können Sie ein Verbindungsformat kopieren. Wenn das Verbindungsformat mit dem angegebenen Namen existiert und der Name des neuen Verbindungsformats noch nicht verwendet wird, wird das neue Verbindungsformat-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ fromName	System.String	Name des zu kopierenden Verbindungsformats
⇒ newName	System.String	Name des neuen Verbindungsformats
<b>Rückgabewert</b>	VcLinkFormat	NodeFormat-Objekt

## FirstFormat

### Methode von VcLinkFormatCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Verbindungsformat des LinkFormatCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextFormat** über die nachfolgenden Verbindungsformate zu iterieren. Existiert kein Verbindungsformat im LinkFormatCollection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcLinkFormat	Erstes Verbindungsformat

### Code-Beispiel VB.NET

```
Dim format As VcLinkFormat
format = VcNet1.LinkFormatCollection.FirstFormat
```

### Code-Beispiel C#

```
VcLinkFormat format = vcNet1.LinkFormatCollection.FirstFormat;
```

## FormatByIndex

### Methode von VcLinkFormatCollection

Mit dieser Methode können Sie auf ein einzelnes Format über seinen Index zugreifen. Existiert kein Format an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ index	System.Int16	Index des Verbindungsformats
<b>Rückgabewert</b>	VcLinkFormat	Ermitteltes Verbindungsformat

**Code-Beispiel VB.NET**

```
Dim formatCltn As VcLinkFormatCollection

formatCltn = VcNet1.LinkFormatCollection
format = formatCltn.FormatByIndex(0)
format.WidthOfExteriorSurrounding = 2
```

**Code-Beispiel C#**

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
VcLinkFormat format = formatCltn.FormatByIndex(0);
format.WidthOfExteriorSurrounding = 2;
```

## FormatByName

**Methode von VcLinkFormatCollection**

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Verbindungsformat zugreifen. Existiert kein Verbindungsformat unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ formatName	System.String	Name des Verbindungsformats
<b>Rückgabewert</b>	VcLinkFormat	Verbindungsformat

**Code-Beispiel VB.NET**

```
Dim formatCltn As VcLinkFormatCollection
Dim format As VcLinkFormat

formatCltn = VcNet1.LinkFormatCollection
format = formatCltn.FormatByName("Standard")
```

**Code-Beispiel C#**

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
VcLinkFormat format = formatCltn.FormatByName("Standard");
```

## GetEnumerator

**Methode von VcLinkFormatCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Knotenformat-Objekte iterieren.



	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

**Code-Beispiel VB.NET**

```
Dim format As VcLinkFormat

For Each format In VcNet1.LinkFormatCollection
    Debug.Write( format.Name)
Next
```

**Code-Beispiel C#**

```
foreach (VcLinkFormat format In vcNet1.LinkFormatCollection)
    Console.Write(format.Name);
```

**NextFormat****Methode von VcLinkFormatCollection**

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Verbindungsformate des LinkFormatCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstFormat** den Initialwert erfasst haben. Sind alle Formate durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLinkFormat	Folgendes Verbindungsformat

**Code-Beispiel VB.NET**

```
Dim formatCltn As VcLinkFormatCollection
Dim format As VcLinkFormat

formatCltn = VcNet1.LinkFormatCollection
format = formatCltn.Firstformat

While Not format Is Nothing
    ListBox1.Items.Add(format.Name)
    format = formatCltn.NextFormat
End While
```

**Code-Beispiel C#**

```
VcLinkFormatCollection formatCltn = vcNet1.LinkFormatCollection;
VcLinkFormat format = formatCltn.FirstFormat;

while (format != null)
{
    listBox1.Items.Add(format.Name);
    format = formatCltn.NextFormat();
}
```

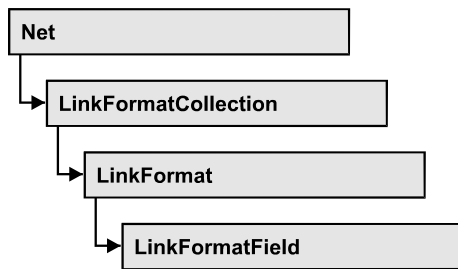
## Remove

### Methode von VcLinkFormatCollection

Mit dieser Methode können Sie ein Verbindungsformat löschen. Wenn das Verbindungsformat noch irgendwo verwendet wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ name	System.String	Name des Verbindungsformats
<b>Rückgabewert</b>	System.Boolean	Verbindungsformat gelöscht (True)/nicht gelöscht (False)

## 7.33 VcLinkFormatField



Ein Objekt vom Typ **VcLinkFormatField** stellt ein Verbindungsformatfeld, also ein Feld eines VcLinkFormat-Objekts dar. Ein Verbindungsformatfeld besitzt im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, unter dem es im Linkformat untergebracht ist.

### Eigenschaften

- Alignment
- ConstantText
- FormatName
- Index
- MinimumWidth
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextLineCount

---

## Eigenschaften

### Alignment

**Nur-Lese-Eigenschaft von VcLinkFormatField**

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Linkformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFormatFieldAlignment	Ausrichtung des Feldinhalts
	<b>Mögliche Werte:</b> .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25	unten unten links unten rechts unten mittig

.vcFFALeft 24	links
.vcFFARight 26	rechts
.vcFFATop 22	oben
.vcFFATopLeft 21	oben links
.vcFFATopRight 23	oben rechts

## ConstantText

### Nur-Lese-Eigenschaft von VcLinkFormatField

Mit dieser Eigenschaft können Sie einen konstanten Text in dem Linkformatfeld ausgeben, falls der Typ des Linkformatfeldes auf **vcFFTText** und falls die Eigenschaft **TextDataFieldIndex** auf **-1** gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	konstanter Text

## FormatName

### Nur-Lese-Eigenschaft von VcLinkFormatField

Mit dieser Eigenschaft können Sie den Namen des Linkformats erfragen, zu dem dieses Linkformatfeld gehört.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name eines Linienformat-Objektes

## Index

### Nur-Lese-Eigenschaft von VcLinkFormatField

Mit dieser Eigenschaft können Sie den Index des Linkformatfelds im zugehörigen Linkformat erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Tabellenformatfeldes

## MinimumWidth

Nur-Lese-Eigenschaft von VcLinkFormatField

Mit dieser Eigenschaft können Sie die minimale Breite des Linkformatfeldes in mm festlegen oder erfragen. Die Breite des Feldes kann sich vergrößern, wenn unter oder über dem Feld andere Felder größere minimale Breiten besitzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Integer	minimale Breite des Layerformatfeldes in mm 0 ... 99

## TextDataFieldIndex

Nur-Lese-Eigenschaft von VcLinkFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTText** den Index des Datenfelds, dessen Inhalt in dem Linkformatfeld dargestellt werden soll, erfragen oder setzen. Wenn Sie den Index auf -1 setzen, wird stattdessen der Inhalt der Eigenschaft **ConstantText** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes

## TextFont

Nur-Lese-Eigenschaft von VcLinkFormatField

Mit dieser Eigenschaft können Sie die Schriftart des Linkformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFTText** gesetzt wurde. Wenn über die Eigenschaft **TextFontMapName** eine Zuordnungstabelle angegeben wurde, steuert diese die Schriftart in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftart des Tabellenformatfelds

## TextFontColor

**Nur-Lese-Eigenschaft von VcLinkFormatField**

Mit dieser Eigenschaft können Sie die Schriftfarbe des Linkformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFTText** gesetzt wurde. Wenn über die Eigenschaft **TextFontMapName** eine Zuordnungstabelle angegeben wurde, steuert diese die Schriftfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	Schriftfarbe des Tabellenformatfelds <b>Standardwert:</b> -1

## TextLineCount

**Nur-Lese-Eigenschaft von VcLinkFormatField**

Mit dieser Eigenschaft können Sie die Zeilenanzahl erfragen oder setzen, wenn die Größe des Beschriftungsfeldes mehrere zulässt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Integer	Zeilenzahl

## 7.34 VcMap



Eine Zuordnungstabelle (Map) legt über Datenfeldeinträge bestimmte Eigenschaften von Knoten, beispielsweise die Hintergrundfarbe, datenfeldabhängig fest.

Sie können in einer Zuordnungstabelle maximal 150 Zuordnungen festlegen. Über **For Each mapEntry In Map** können Sie in einer Schleife auf alle Verbindungen zugreifen.

### Eigenschaften

- ConsiderFilterEntries
- Count
- Name
- Specification
- Type

### Methoden

- CreateEntry
- DeleteEntry
- FirstMapEntry
- GetMapEntry
- NextMapEntry

---

## Eigenschaften

### ConsiderFilterEntries

**Nur-Lese-Eigenschaft von VcMap**

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob bei der Zuordnung von Datenfeldeinträgen zu einer Zuordnungstabelle Filter berücksichtigt werden, um so auch Wertebereiche als Schlüsselwerte angeben zu können.

	Datentyp	Beschreibung
--	----------	--------------

## Count

**Nur-Lese-Eigenschaft von VcMap**

Mit dieser Eigenschaft kann die Anzahl der Einträge in der Zuordnungstabelle (Map) abgefragt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Anzahl der Mapeinträge

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim numberOfEntries As Integer

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
numberOfEntries = map.Count
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
int numberOfEntries = map.Count;
```

## Name

**Nur-Lese-Eigenschaft von VcMap**

Mit dieser Eigenschaft können Sie den Namen der Zuordnungstabelle (Map) abfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name der Zuordnungstabelle

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapName As String

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
mapName = map.Name
```



**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
string mapName = map.Name;
```

**Specification****Nur-Lese-Eigenschaft von VcMap**

Mit dieser Eigenschaft können Sie die Spezifikation dieser Zuordnungstabelle auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Erzeugung einer Zuordnungstabelle mit der Methode **VcMapCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Spezifikation der Zuordnungstabelle

**Code-Beispiel VB.NET**

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcNet1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

**Code-Beispiel C#**

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

**Type****Eigenschaft von VcMap**

Mit dieser Eigenschaft können Sie den Typ der Zuordnungstabelle (Map) abfragen oder setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcMapType  <b>Mögliche Werte:</b> .vcAnyMap 0 .vcColorMap 1 .vcFontMap 8 .vcGraphicsFileMap 7 .vcMillimeterMap 9 .vcNumberMap 10 .vcPatternMap 3	Typ der Zuordnungstabelle  <b>Beliebig</b> (nur zum Selektieren verwendet) <b>Farben</b> <b>Schriften</b> <b>Grafikdatei</b> <b>Millimeter</b> <b>Zahlen</b> <b>Schraffuren</b>

.vcTextMap 6

Text

**Code-Beispiel VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
map = mapCltn.MapByName ("Map1")
map.Type = VcMapType.vcPatternMap
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName ("Map1");
map.Type = VcMapType.vcPatternMap;
```

## Methoden

### CreateEntry

Methode von VcMap

Mit dieser Methode kann ein neuer Eintrag (= eine neue Zeile) für die Zuordnungstabelle (Map) erzeugt werden. Damit der neue Eintrag in der Zuordnungstabelle wirksam wird, sollte anschließend die Methode **MapCollection.Update()** aufgerufen werden.

	Datentyp	Beschreibung
Rückgabewert	VcMapEntry	Map-Eintrag

**Code-Beispiel VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
map = mapCltn.MapByName ("Map1")
mapEntry = map.CreateEntry
mapCltn.Update
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName ("Map1");
VcMapEntry mapEntry = map.CreateEntry();
mapCltn.Update;
```

## DeleteEntry

Methode von VcMap

Mit dieser Methode kann ein Eintrag (= eine Zeile) der Zuordnungstabelle (Map) gelöscht werden. Damit die Löschung in der Zuordnungstabelle wirksam wird, sollte anschließend die Methode **MapCollection.Update()** aufgerufen werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ mapEntry	VcMapEntry	Eintrag der Map
<b>Rückgabewert</b>	System.Boolean	Map-Eintrag erfolgreich/nicht erfolgreich gelöscht

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
map = mapCltn.MapByName ("Map1")
mapEntry = map.FirstMapEntry
map.DeleteEntry (mapEntry)
mapCltn.Update
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName ("Map1");
VcMapEntry mapEntry = map.FirstMapEntry ();
map.DeleteEntry (mapEntry);
mapCltn.Update;
```

## FirstMapEntry

Methode von VcMap

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Eintrag der Zuordnungstabelle (Map) zugreifen, um anschließend in einer Schleife mit der Methode **NextMapEntry** über die nachfolgenden Einträge zu iterieren. Existiert kein Eintrag im Map-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcMapEntry	Erster Map-Eintrag

**Code-Beispiel VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)

map = mapCltn.FirstMap
mapEntry = map.FirstMapEntry
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry();
```

**GetMapEntry****Methode von VcMap**

Diese Methode liefert den entsprechenden Eintrag der Zuordnungstabelle (Map) zu dem im Datenfeld angegebenen Wert.

	Datentyp	Beschreibung
Rückgabewert	System.String	Map-Eintrag entsprechend Feldinhalt

**NextMapEntry****Methode von VcMap**

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Einträge (Zeilen) des Map-Objekts zugreifen, nachdem Sie mit der Methode **FirstMapEntry** den Initialwert erfasst haben. Sind alle Einträge durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcMapEntry	Nachfolgender Map-Eintrag

### Code-Beispiel VB.NET

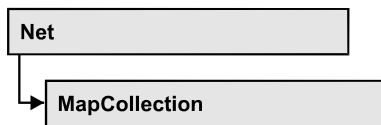
```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
mapEntry = map.FirstMapEntry
While Not mapEntry Is Nothing
    ListBox1.Items.Add(mapEntry.LegendText)
    mapEntry = map.NextMapEntry
End While
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry()
while (mapEntry != null)
{
    listBox1.Items.Add(mapEntry.LegendText);
    mapEntry= map.NextMapEntry();
}
```

## 7.35 VcMapCollection



In einem Objekt des Typs VcMapCollection sind die Zuordnungstabellen (Maps) zusammengefasst, die dem Auflistungsobjekt über die Methode **SelectMaps** zugewiesen wurden. Über **For Each map InMapCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Zuordnungstabellen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **MapByName** und **MapByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Zuordnungstabellen kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Zuordnungstabellen.

### Eigenschaften

- Count

### Methoden

- Add
- AddBySpecification
- Copy
- FirstMap
- GetEnumerator
- MapByIndex
- MapByName
- NextMap
- Remove
- SelectMaps
- Update

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von VcMapCollection

Mit dieser Eigenschaft kann die Anzahl der Zuordnungstabellen (Maps) in der Map-Auflistung abgefragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Maps

#### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim numberOfMaps As Integer

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
numberOfMaps = mapCltn.Count
```

#### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
int numberOfMaps = mapCltn.Count;
```

## Methoden

### Add

Methode von VcMapCollection

Mit dieser Methode können Sie eine neue Zuordnungstabelle in der MapCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Zuordnungstabellenobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ mapName	System.String	Name der Zuordnungstabelle
<b>Rückgabewert</b>	VcMap	Neues Zuordnungstabellenobjekt

#### Code-Beispiel VB.NET

```
newMap = VcNet1.MapCollection.Add("Map1")
```

**Code-Beispiel C#**

```
VcMap newMap = vcNet1.MapCollection.Add("Map1");
```

**AddBySpecification****Methode von VcMapCollection**

Mit dieser Methode können Sie eine Zuordnungstabelle über eine Zuordnungstabellen-Spezifikation erzeugen. Dies dient der Persistenz von Zuordnungstabellen-Objekten. Die Spezifikation einer Zuordnungstabelle kann erfragt werden (siehe VcMap-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann die gleiche Zuordnungstabelle mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ specification	System.String	Zuordnungstabellenspezifikation
<b>Rückgabewert</b>	VcMap	Neues Zuordnungstabellenobjekt

**Copy****Methode von VcMapCollection**

Mit dieser Methode können Sie eine Zuordnungstabelle kopieren. Wenn die Zuordnungstabelle mit dem angegebenen Namen existiert und der Name der neuen Zuordnungstabelle noch nicht verwendet wird, wird das neue Zuordnungstabellenobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ mapName	System.String	Name der zu kopierenden Zuordnungstabelle
⇒ newMapName	System.String	Name der neuen Zuordnungstabelle
<b>Rückgabewert</b>	VcMap	Zuordnungstabellenobjekt



## FirstMap

Methode von VcMapCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Zuordnungstabelle (Map) des MapCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextMap** über die nachfolgenden Maps zu iterieren. Existiert keine Zuordnungstabelle (Map) in der MapCollection, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**). Zuvor muss mit der Methode **SelectMaps** die gewünschte Map-Auswahl getroffen worden sein.

	Datentyp	Beschreibung
Rückgabewert	VcMap	Erste Map

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
map = mapCltn.FirstMap
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
```

## GetEnumerator

Methode von VcMapCollection

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Mit diesem Objekt können Sie über alle enthaltenen Zuordnungstabellen iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

## MapByIndex

Methode von VcMapCollection

Mit dieser Methode können Sie auf eine einzelne Zuordnungstabelle über ihren Index zugreifen. Existiert keine Zuordnungstabelle an dem

angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index der Zuordnungstabelle
<b>Rückgabewert</b>	VcMap	Ermitteltes Zuordnungstabellenobjekt

## MapByName

### Methode von VcMapCollection

Mit dieser Methode können Sie unter Verwendung des Namens der Zuordnungstabelle (Map) auf eine bestimmte Zuordnungstabelle zugreifen. Zuvor muss mit der Methode **SelectMaps** die gewünschte Auswahl der Zuordnungstabelle getroffen worden sein. Existiert kein Map-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ mapName	System.String	Name der Map
<b>Rückgabewert</b>	VcMap	Map

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
```

## NextMap

### Methode von VcMapCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Zuordnungstabellen (Maps) des MapCollection-Objekts zugreifen, nachdem

Sie mit der Methode **FirstMap** den Initialwert erfasst haben. Sind alle Maps durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcMap	Nachfolgende Map

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps (VcMapType.vcAnyMap)
map = mapCltn.FirstMap
While Not map Is Nothing
    ListBox1.Items.Add (map.Name)
    map = mapCltn.NextMap
End While
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps (VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
while (map != null)
{
    listBox1.Items.Add (map.Name);
    map = mapCltn.NextMap();
}
```

## Remove

### Methode von VcMapCollection

Mit dieser Methode können Sie eine Zuordnungstabelle löschen. Wenn die Zuordnungstabelle noch in irgendeinem anderen Objekt benutzt wird, kann sie nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ mapName	System.String	Name der Zuordnungstabelle
<b>Rückgabewert</b>	System.Boolean	Zuordnungstabelle gelöscht (True)/nicht gelöscht (False)

## SelectMaps

### Methode von VcMapCollection

Mit dieser Methode können Sie steuern, welche Typen von Zuordnungstabellen (Maps) in Ihr MapCollection-Objekt aufgenommen werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ selectionType	VcMapType  <b>Mögliche Werte:</b> .vcAnyMap 0 .vcColorMap 1 .vcFontMap 8 .vcGraphicsFileMap 7 .vcMillimeterMap 9 .vcNumberMap 10 .vcPatternMap 3 .vcTextMap 6	Auszuwählender Map-Typ  <b>Beliebig</b> (nur zum Selektieren verwendet) <b>Farben</b> <b>Schriften</b> <b>Grafikdatei</b> <b>Millimeter</b> <b>Zahlen</b> <b>Schraffuren</b> <b>Text</b>
<b>Rückgabewert</b>	System.Int32	Anzahl der Ausgewählten Maps

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
```

## Update

### Methode von VcMapCollection

Mit dieser Methode können Sie die Darstellung aller Objekte, die durch die verwendeten Zuordnungstabellen (Maps) bestimmt werden, aktualisieren. Wenn Sie diese Methode nicht aufrufen, werden die Änderungen der Zuordnungstabellen (Maps) zur Laufzeit nicht ausgeführt. Sie sollten diese Methode erst am Ende des Codes zur Festlegung der Zuordnungstabellen und des MapCollection-Objekts verwenden, damit die Aktualisierung nicht schon ausgeführt wird, bevor alle Festlegungen der Zuordnungstabellen ausgeführt worden sind.

## 516 API Referenz: VcMapCollection

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
While Not mapEntry.DataFieldValue = "A"
    mapEntry = map.NextMapEntry
End While

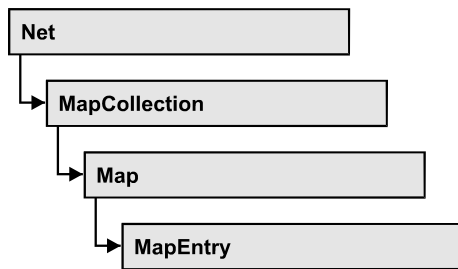
mapEntry.Color = Color.Blue
mapCltn.Update()
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
while (mapEntry.DataFieldValue != "A")
    mapEntry = map.NextMapEntry();

mapEntry.Color = Color.LightSteelBlue;
mapCltn.Update();
```

## 7.36 VcMapEntry



Ein Objekt vom Typ VcMapEntry ist ein Eintrag einer Zuordnungstabelle (Map) und damit das Element einer Zuordnungstabelle. Ein Zuordnungstabelleneintrag enthält die Kombination aus einem Datenfeldinhalt des Vorgangsdatensatzes sowie bestimmten Attributen (z. B. Farbe, Grafikdatei, Schriftattribute).

Sie können in einer Zuordnungstabelle maximal 150 Zuordnungen festlegen. Falls Sie noch mehr Zuordnungen benötigen, erstellen Sie einfach eine neue Zuordnungstabelle, z. B. als Kopie der bereits vorhandenen.

### Eigenschaften

- Color
- DataFieldValue
- FontBody
- FontName
- FontSize
- GraphicsFileName
- Number
- Pattern

---

## Eigenschaften

### Color

**Eigenschaft von VcMapEntry**

*Für Farben-Zuordnungstabellen:* Mit dieser Eigenschaft können Sie die Farbe für den Zuordnungstabelleneintrag festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von

0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim colorOfMapEntry As Color

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcColorMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
colorOfMapEntry = mapEntry.Color
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcColorMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
Color colorOfMapEntry = mapEntry.Color;
```

## DataFieldValue

### Eigenschaft von VcMapEntry

Mit dieser Eigenschaft können Sie den Inhalt des Datenfeldes des Zuordnungstabelleneintrags erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Inhalt des Datenfelds

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim dataFieldValue As String

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
string dataFieldValue = mapEntry.DataFieldValue;
```

**FontBody****Eigenschaft von VcMapEntry**

*für Schriften-Zuordnungstabellen:* Mit dieser Eigenschaft können Sie den Schriftgrad für den Zuordnungstabelleneintrag erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFontBody	Schriftgrad

**Code-Beispiel VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontBodyOfMapEntry As VcFontBody

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontBodyOfMapEntry = VcFontBody.vcBold
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
VcFontBody fontBodyOfMapEntry = VcFontBody.vcBold;
```

**FontName****Eigenschaft von VcMapEntry**

*für Schriften-Zuordnungstabellen:* Mit dieser Eigenschaft können Sie die Schriftart für den Zuordnungstabelleneintrag erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Schriftart



### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontNameOfMapEntry As String

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontNameOfMapEntry = "Arial"
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
string fontNameOfMapEntry = "Arial";
```

## FontSize

**Eigenschaft von VcMapEntry**

*für Schriften-Zuordnungstabellen:* Mit dieser Eigenschaft können Sie die Schriftgröße für den Zuordnungstabelleneintrag erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Schriftgröße

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontSizeOfMapEntry As Integer

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontSizeOfMapEntry = 14
```

### Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
int fontSizeOfMapEntry = 14;
```

## GraphicsFileName

Eigenschaft von VcMapEntry

*Für Grafikdateien-Zuordnungstabellen:* Mit dieser Eigenschaft können Sie den Namen der Grafikdatei des Zuordnungstabelleneintrags erfragen oder festlegen. Mögliche Formate:

- \*.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- \*.GIF (Graphics Interchange Format)
- \*.JPG (Joint Photographic Experts Group)
- \*.PNG (Portable Network Graphics)
- \*.TIF (Tagged Image File Format)
- \*.VMF (Viewer Metafile)
- \*.WMF (Microsoft Windows Metafile, ggf. mit eingebautem EMF)

Nur EMF, EMF+, VMF und WMF sind Vektorformate, in denen das Diagramm auflösungsunabhängig gespeichert werden kann. Die übrigen Formate sind pixelorientiert und bieten damit nicht beliebige Auflösungen.

Das VMF-Format wird in der Zukunft nicht mehr weiterentwickelt, aus Kompatibilitätsgründen für bestehende Anwendungen aber zunächst noch weiter unterstützt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name der Grafikdatei

### Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim exeName As String
Dim exeDir As String

mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcGraphicsFileMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry

exeName = System.Environment.GetCommandLineArgs(0)
exeDir = System.IO.Path.GetDirectoryName(exeName)
mapEntry.GraphicsFileName = exeDir + "\Bitmaps\picture1.bmp"
```

## Code-Beispiel C#

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcGraphicsFileMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();

String exeName = Environment.GetCommandLineArgs()[0];
mapEntry.GraphicsFileName = System.IO.Path.GetDirectoryName(exeName) +
@"\..\Bitmaps\picture1.bmp";
```

## Number

### Eigenschaft von VcMapEntry




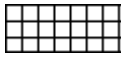
*Für numerische Zuordnungstabellen:* Mit dieser Eigenschaft können Sie eine Zahl als Zuordnungstabelleneintrag festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Numerischer Wert

## Pattern









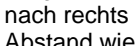

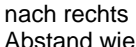
### Eigenschaft von VcMapEntry

*Für Schraffuren-Zuordnungstabellen (vcPatternMap):* mit dieser Eigenschaft können Sie den Schraffurtyp des Zuordnungstabelleneintrags erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern  <b>Mögliche Werte:</b> .vc05PercentPattern... vc90PercentPattern 01 - 11  .vcAeroGlassPattern 44  .vcBDiagonalPattern 5  .vcCrossPattern 6	Mustertyp  Punkte in Vordergrundfarbe auf Hintergrundfarbe; mit steigender Prozentzahl Vordergrundfarbe immer dichter  Vertikaler Farbverlauf in der Füllmusterfarbe  Diagonale Linien von links unten nach rechts oben  Kreuzschraffur 

.vcDarkDownwardDiagonalPattern 2014	Diagonale Linien von links oben nach rechts unten, 50 % näher zusammen als vcFDiagonalPattern und mit doppelter Liniendicke	
.vcDarkHorizontalPattern 2023	Horizontale Linien mit 50% geringerem Abstand als vcHorizontalPattern und doppelter Liniendicke	
.vcDarkUpwardDiagonalPattern 2015	Diagonale Linien von links unten nach rechts oben; mit 50%geringerem Abstand als vcBDiagonalPattern und mit doppelter Liniendicke	
.vcDarkVerticalPattern 2022	Vertikale Linien mit 50% geringerem Abstand als vcVerticalPattern und doppelter Liniendicke	
.vcDashedDownwardDiagonalPattern 2024	Gestrichelte diagonale Linien von links oben nach rechts unten	
.vcDashedHorizontalPattern 2026	Horizontale gestrichelte Linien	
.vcDashedUpwardDiagonalPattern 2025	Unterbrochene diagonale Linien von links unten nach rechts oben	
.vcDashedVerticalPattern 2027	Vertikale gestrichelte Linien	
.vcDiagCrossPattern 7	Diagonale Kreuzschraffur, klein	
.vcDiagonalBrickPattern 2032	Diagonales Backsteinmuster	
.vcDivotPattern 2036	Grassoden-Muster	
.vcDottedDiamondPattern 2038	Diagonale Kreuzschraffur aus punktierten Linien	
.vcDottedGridPattern 2037	Kreuzschraffur aus punktierten Linien	
.vcFDiagonalPattern 4	Diagonale Linien von links oben nach rechts unten	
.vcHorizontalBrickPattern 2033	Horizontales Backsteinmuster	
.vcHorizontalGradientPattern 52	Horizontaler Farbverlauf	
.vcHorizontalPattern 3	Horizontale Linien	

.vcLargeCheckerboardPattern 2044	Schachbrettmuster mit doppelt so großen Quadraten wie vcSmall-CheckerBoardPattern
.vcLargeConfettiPattern 2029	Konfetti -Muster, groß
.vcLightDownwardDiagonalPattern 2012	Diagonale Linien von links oben nach rechts unten; mit 50% geringerem Abstand als vcB-DiagonalPattern
.vcLightHorizontalPattern 2019	Horizontale Linien mit 50% geringerem Abstand als bei vc-HorizontalPattern
.vcLightUpwardDiagonalPattern 2013	Diagonale Linien von links unten nach rechts oben; 50 % näher zusammen als vcBDiagonalPattern
.vcLightVerticalPattern 2018	Vertikale Linien mit 50% geringerem Abstand als bei vcVerticalPattern
.vcNarrowHorizontalPattern 2021	Horizontale Linien mit 75% geringerem Abstand als vc-HorizontalPattern
.vcNarrowVerticalPattern 2020	Vertikale Linien mit 75% geringerem Abstand als vcVerticalPattern
.vcNoPattern 1276	Kein Füllmuster
.vcOutlinedDiamondPattern 2045	Diagonale Kreuzschraffur, groß
.vcPlaidPattern 2035	Schottenstoff-Muster
.vcShinglePattern 2039	Diagonales Dachschindel-Muster
.vcSmallCheckerBoardPattern 2043	Schachbrettmuster
.vcSmallConfettiPattern 2028	Konfetti-Muster
.vcSmallGridPattern 2042	Kreuzschraffur mit 50 % geringerem Abstand als vcCrossPattern
.vcSolidDiamondPattern 2046	Schachbrettmuster mit diagonalen Quadraten
.vcSpherePattern 2041	Kugeln schachbrettartig angeordnet
.vcTrellisPattern 2040	Spalier-Muster

.vcVerticalBottomLightedConvexPattern 43	Vertikaler Farbverlauf von dunkel nach hell 
.vcVerticalConcavePattern 40	Vertikaler Farbverlauf von dunkel über hell nach dunkel 
.vcVerticalConvexPattern 41	Vertikaler Farbverlauf von hell über dunkel nach hell 
.vcVerticalGradientPattern 62	Vertikaler Farbverlauf 
.vcVerticalPattern 2	Vertikale Linien 
.vcVerticalTopLightedConvexPattern 42	Vertikaler Farbverlauf von hell nach dunkel 
.vcWavePattern 2031	Horizontales Wellenmuster 
.vcWeavePattern 2034	Muster mit verwebten Streifen 
.vcWideDownwardDiagonalPattern 2016	Diagonale Linien von links oben nach rechts unten, mit demselben Abstand wie vcFDiagonalPattern, aber mit dreifacher Liniendicke 
.vcWideUpwardDiagonalPattern 2017	Diagonale Linien von links oben nach rechts unten, mit demselben Abstand wie vcBDiagonalPattern, aber mit dreifacher Liniendicke 
.vcZigZagPattern 2030	Horizontale Zick-Zacklinien 

**Code-Beispiel VB.NET**

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim pattern As VcFillPattern

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcPatternMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
pattern = VcFillPattern.vcBDiagonalPattern
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcPatternMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
VcFillPattern pattern = VcFillPattern.vcBDiagonalPattern;
```

## 7.37 VcNet

Net
-----

Ein Objekt vom Typ VcNet bezeichnet das VARCHAR-XNet-Steuer-element selbst. Über Ereignisse können Sie dessen Interaktionen kontrollieren. Das VcNet-Objekt kann durch eine Vielzahl von Eigenschaften und Methoden den Anforderungen entsprechend konfiguriert werden.

### Eigenschaften

- ActiveNodeFilter
- BorderArea
- BoxCollection
- BoxFormatCollection
- CalendarCollection
- CalendarProfileCollection
- CtrlCXVProcessingEnabled
- DataTableCollection
- DateOutputFormat
- DiagramBackgroundColor
- DialogFont
- DoubleOutputFormat
- Enabled
- ExtendedDataTablesEnabled
- FilePath
- FilterCollection
- FontAntiAliasingEnabled
- GroupCollection
- GroupHorizontalMargin
- GroupingActivated
- GroupingDataFieldIndex
- GroupingTitlesFullyVisible
- GroupingType
- GroupInteractionsAllowed
- GroupSortingDataFieldIndex
- GroupSortMode
- GroupTitleDataFieldIndex
- GroupTitlesFileName
- GroupVerticalMargin
- InbuiltMouseCursorWhileDraggingEnabled

- InFlowGroupingActivated
- InFlowGroupingDataFieldIndex
- InFlowGroupSeparationLineColor
- InFlowGroupSeparationLineType
- InFlowGroupTimeInterval
- InFlowGroupTitleDataFieldIndex
- InFlowGroupTitlesBackgroundColor
- InFlowGroupTitlesFileName
- InFlowGroupTitlesFont
- InFlowGroupTitlesVisibleAtBottomOrRight
- InFlowGroupTitlesVisibleAtTopOrLeft
- InFlowGroupTitleTimeFormat
- InFlowGroupVerticalCaptionWidth
- InPlaceEditingAllowed
- InteractionMode
- InterfaceNodesShown
- LegendView
- LinkAnnotationColumnNumberDataFieldIndex
- LinkAnnotationRowNumberDataFieldIndex
- LinkAppearanceCollection
- LinkCollection
- LinkCreationWithDialog
- LinkFormatCollection
- LinkPredecessorDataFieldIndex
- LinksDataTableName
- LinkSuccessorDataFieldIndex
- LinkTypeDataFieldIndex
- MapCollection
- MinimumColumnWidth
- MinimumRowHeight
- MouseProcessingEnabled
- MovingCollapsedClustersAllowed
- NodeAndLinkCreationAllowed
- NodeAppearanceCollection
- NodeCalendarNameDataFieldIndex
- NodeChangeRankToPredecessorRankDataFieldIndex
- NodeCollection
- NodeColumnNumberDataFieldIndex
- NodeCreationWithDialog
- NodeFormatCollection



- NodeRowNumberDataFieldIndex
- NodesDataTableName
- NodesUseCalendars
- NodeToolTipTextDataFieldIndex
- ObliqueTracksOnLinks
- Orientation
- PhantomDrawingWhileDraggingEnabled
- Printer
- RoundedLinkSlantsEnabled
- Scheduler
- ShortenedLinks
- StraightLinkDrawing
- TextEntrySupplyingEventEnabled
- TimeUnit
- ToolTipChangeDuration
- ToolTipDuration
- ToolTipPointerDuration
- ToolTipShowAfterClick
- ToolTipTextSupplyingEventEnabled
- UngroupedNodesAllowed
- ViewXCoordinate
- ViewYCoordinate
- WaitCursorEnabled
- WorldView
- ZoomFactor
- ZoomingPerMouseWheelAllowed

### **Methoden**

- Arrange
- Clear
- CompleteViewMode
- CopyNodesIntoClipboard
- CutNodesIntoClipboard
- DeleteLinkRecord
- DeleteNodeRecord
- DetectDataTableFieldName
- DetectDataTableName
- DetectFieldIndex
- DumpConfiguration
- EndLoading

- ExportGraphicsToFileEx
- GetAValueFromARGB
- GetBValueFromARGB
- GetGValueFromARGB
- GetLinkByID
- GetLinkByNodeIDs
- GetNodeByID
- GetRValueFromARGB
- IdentifyFormatField
- IdentifyObjectAt
- ImportConfiguration
- InsertLinkRecord
- InsertNodeRecord
- Load
- MakeARGB
- PasteNodesFromClipboard
- PixelsToRaster
- PrintEx
- PrintToFile
- Reset
- SaveAsEx
- ScheduleProject
- ScrollToNode
- SetImageResource
- ShowAboutDialog
- ShowExportGraphicsDialog
- ShowLinkEditDialog
- ShowNodeEditDialog
- ShowPageSetupDialog
- ShowPrintDialog
- ShowPrinterSetupDialog
- ShowPrintPreviewDialog
- SuspendUpdate
- UpdateLinkRecord
- UpdateNodeRecord
- Zoom
- ZoomOnMarkedNodes

## **Ereignisse**

- VcBoxLeftClicking

- VcBoxLeftDoubleClicking
- VcBoxModified
- VcBoxModifying
- VcBoxRightClicking
- VcDataModified
- VcDataRecordCreated
- VcDataRecordCreating
- VcDataRecordDeleted
- VcDataRecordDeleting
- VcDataRecordModified
- VcDataRecordModifying
- VcDataRecordNotFound
- VcDiagramLeftClicking
- VcDiagramLeftDoubleClicking
- VcDiagramRightClicking
- VcDragCompleting
- VcDragStarting
- VcErrorOccurring
- VcFieldSelecting
- VcGiveFeedbackOnNodeCreating
- VcGroupCreated
- VcGroupDeleting
- VcGroupLeftClicking
- VcGroupLeftDoubleClicking
- VcGroupModified
- VcGroupModifying
- VcGroupRightClicking
- VcHelpRequested
- VcInPlaceEditorShowing
- VcLegendViewClosed
- VcLinkCreated
- VcLinkCreating
- VcLinkDeleted
- VcLinkDeleting
- VcLinkModified
- VcLinkModifying
- VcLinksLeftClicking
- VcLinksLeftDoubleClicking
- VcLinksMarked
- VcLinksMarking

- VcLinksRightClicking
- VcMouseDoubleClicking
- VcMouseDown
- VcMouseMove
- VcMouseUp
- VcNodeCreated
- VcNodeCreating
- VcNodeDeleted
- VcNodeDeleting
- VcNodeLeftClicking
- VcNodeLeftDoubleClicking
- VcNodeModifiedEx
- VcNodeModifying
- VcNodeRightClicking
- VcNodesMarked
- VcNodesMarking
- VcStatusLineTextShowing
- VcTextEntrySupplying
- VcToolTipTextSupplying
- VcWorldViewClosed
- VcZoomFactorModified

---

## Eigenschaften

### ActiveNodeFilter

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Filter festlegen oder erfragen, der die darzustellenden Knoten selektiert.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFilter	Filterobjekt <b>Standardwert:</b> Nothing

#### Code-Beispiel VB.NET

```
VcNet1.ActiveNodeFilter = VcNet1.FilterCollection.FilterByName("Milestone")
```

#### Code-Beispiel C#

```
vcNet1.ActiveNodeFilter = vcNet1.FilterCollection.FilterByName("Milestone");
```

## BorderArea

Nur-Lese-Eigenschaft von VcNet

Diese Eigenschaft ermöglicht den Zugriff auf das BorderArea-Objekt, also auf den Titel- und Legendenbereich.

	Datentyp	Beschreibung
Eigenschaftswert	VcBorderArea	Titel- und Legendenbereich

### Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
borderArea = VcNet1.BorderArea
```

### Code-Beispiel C#

```
VcBorderArea borderArea = vcNet1.BorderArea;
```

## BoxCollection

Nur-Lese-Eigenschaft von VcNet

Diese Eigenschaft ermöglicht den Zugriff auf die BoxAuflistung und damit auf die verwendeten Boxen.

	Datentyp	Beschreibung
Eigenschaftswert	VcBoxCollection	BoxCollection-Objekt

### Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
boxCltn = VcNet1.BoxCollection
```

### Code-Beispiel C#

```
VcBoxCollection boxCltn = vcNet1.BoxCollection;
```

## BoxFormatCollection

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft haben Sie Zugriff auf die BoxFormat-Auflistung, in der alle zur Verfügung stehenden Box-Formate enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcBoxFormatCollection	BoxFormatCollection-Objekt

## CalendarCollection

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft haben Sie Zugriff auf die Kalender-Auflistung, in der alle zur Verfügung stehenden Kalender zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendarCollection	CalendarCollection-Objekt

### Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
calendarCltn = VcNet1.CalendarCollection
```

### Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcNet1.CalendarCollection;
```

## CalendarProfileCollection

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft haben Sie Zugriff auf die Kalenderprofilauflistung, in der alle zur Verfügung stehenden Kalenderprofile enthalten sind.

	Datentyp	Beschreibung

## CtrlCXVProcessingEnabled

Eigenschaft von VcNet

Mit dieser Eigenschaft werden die Tastenkombinationen Strg+C, Strg+X und Strg+V automatisch in die Zwischenablage-Operationen **CopyNodesToClipboard**, **CutNodesToClipboard** bzw. **PasteNodesFromClipboard** übersetzt. Dieses Verhalten kann abgeschaltet werden, damit in Visual Basic kein Konflikt mit Bearbeitungsmethoden für Menüpunkte entsteht, die die gleichen Tastaturkombinationen benutzen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Tastenkombinationen werden/werden nicht in Zwischenablage-Operationen übersetzt <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.CtrlCXVProcessing = False
```

**Code-Beispiel C#**

```
vcNet1.CtrlCXVProcessing = false;
```

**DataTableCollection****Eigenschaft von VcNet**

Diese Eigenschaft ermöglicht den Zugriff auf die DataTable-Auflistung und auf die darin verwendeten Datentabellen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcDataTableCollection	Ermitteltes Auflistungsobjekt der Datentabellen

**Code-Beispiel VB.NET**

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable
```

```
dataTableCltn = VcNet1.DataTableCollection
For Each dataTable In dataTableCltn
    ListBox1.Items.Add(dataTable.Name)
Next
```

**Code-Beispiel C#**

```
VcDataTableCollection dataTableCltn = vcNet1.DataTableCollection;
foreach(VcDataTable dataTable in dataTableCltn)
    listBox1.Items.Add(dataTable.Name);
```

**DateOutputFormat****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie das Ausgabeformat von Terminen einstellen oder erfragen. Für das Datum stehen folgende Kürzel zur Verfügung:

- D: Wochentagsname erster Buchstabe
- TD: Wochentagsname (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)

- DD: Tagesnummer zweistellig: 01-31 (nicht anpassbar)
- DDD: die ersten drei Buchstaben des Wochentagsnamens (nicht anpassbar)
- M: erster Buchstabe des Monatsnamens (nicht anpassbar)
- TM: Monatsname (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- MM: zweistellige Monatsnummer: 01-12
- MMM: erste drei Buchstaben des Monatsnamens (nicht anpassbar)
- YY: zweistellige Jahreszahl
- YYYY: vierstellige Jahreszahl
- WW: zweistellige Nummer der Kalenderwoche: 01-53
- TW: Text für "Kalenderwoche" (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- Q: einstellige Quartalsnummer: 1-4
- TQ: Quartalsname (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- hh: Stunde zweistellig im 24-Stunden-Format: 00-23
- HH: Stunde zweistellig im 12-Stunden-Format: 01-12
- Th: Text für "o' clock" (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- TH: "am" oder "pm" (kann über das Ereignis **VcTextEntrySupplying** angepasst werden)
- mm: Minute zweistellig: 00-59
- ss: Sekunde zweistellig: 00-59
- TS: kurzes Datumsformat, wie in der Windows-Systemsteuerung über die Regions- und Sprachoptionen definiert
- TL: langes Datumsformat, wie in der Windows-Systemsteuerung über die Regions- und Sprachoptionen definiert
- TT: Zeitformat, wie in der Windows-Systemsteuerung über die Regions- und Sprachoptionen definiert

**Hinweis:** Bitte setzen Sie allen Zeichen, die nicht als Datumsbestandteile interpretiert werden sollen, einen Backslash '\' voran. '\\' z.B. ergibt '\' in der



Ausgabe. Die Sonderzeichen: **:**, **/**, **-** und **Leerzeichen** brauchen nicht durch **\** gekennzeichnet zu werden.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Datumsformat

#### Code-Beispiel VB.NET

```
VcNet1.DateOutputFormat = "DD.MM.YY"
```

#### Code-Beispiel C#

```
vcNet1.DateOutputFormat = "DD.MM.YY";
```

## DiagramBackgroundColor

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die Hintergrundfarbe Ihres Netzdiagramms setzen oder erfragen. Die Standard-Farbe ist weiß.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte {0...255},{0...255},{0...255}

#### Code-Beispiel VB.NET

```
VcNet1.DiagramBackgroundColor = RGB(255, 204, 204)
```

#### Code-Beispiel C#

```
vcNet1.DiagramBackgroundColor = RGB(255, 204, 204);
```

## DialogFont

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die Schriftgröße und den Schrifttyp der zur Laufzeit erscheinenden Dialogfelder im VARCHART XNet erfragen oder festlegen. Als Objekt wird ein Font-Objekt Ihrer Programmierumgebung erwartet, z. B. bei Visual Basic ein Objekt der Klasse **StdFont**.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Schriftname

**Code-Beispiel VB.NET**

```
Dim newFont As Font
newFont = Newfont ("Verdana", 14)
VcNet1.DialogFont = newFont
```

**Code-Beispiel C#**

```
Font newFont = newFont ("Verdana", 14);
vcNet1.DialogFont = newFont;
```

**DoubleOutputFormat****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie das Ausgabeformat von Zahlen als double-Wert im Netz-Diagramm einstellen oder erfragen. Das Format kann über folgende Zeichen dargestellt werden:

- Text
- I
- D

sowie die Trennzeichen Komma und Punkt. Dabei steht **Text** für einen beliebigen Text, **I** für die Ziffern vor dem Dezimaltrenner und **D** für je eine Ziffer hinter dem Dezimaltrenner. Die erlaubte, generelle Reihenfolge ist **Text I D Text**, wobei Komma und Punkt an beliebiger Stelle eingefügt werden können. Als Beispiel sei die Zahl -284901,3458 gegeben. Über das Format **I,DDDD ppm** wird sie als **-284901,3458 ppm** dargestellt. Über das Format **\$I,III.DD** wird sie als **\$-284,901.35** dargestellt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Zeichenfolge, die das Double-Format beschreibt, z.B. "I,DDDD ppm"

**Code-Beispiel VB.NET**

```
VcNet1.DoubleOutputFormat = "I,DDDD ppm"
```

**Code-Beispiel C#**

```
vcNet1.DoubleOutputFormat = "$I,III.DD";
```

## Enabled

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie das VARCHART-XNet-Steuerelement so abschalten, dass es auf Maus- und Tastenbefehle nicht reagiert.

	Datentyp	Beschreibung

### Code-Beispiel VB.NET

```
VcNet1.Enabled = False
```

### Code-Beispiel C#

```
vcNet1.Enabled = false;
```

## ExtendedDataTablesEnabled

Eigenschaft von VcNet

Mit dieser Eigenschaft könne Sie zwischen der Beschränkung auf zwei Datentabellen (Maindata und Relations) und der weiterentwickelten Verwendung von bis zu 90 Datentabellen wechseln. Die Verwendung der zweiten Option wird empfohlen. Diese Eigenschaft muss zu Beginn des Programms gesetzt werden, bevor Datatabellen und Datensätze angelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	<b>false:</b> nur zwei Datentabellen (Maindata und Relations)  <b>true:</b> bis zu 99 Datentabellen  <b>Standardwert:</b> false

### Code-Beispiel VB.NET

```
VcNet1.ExtendedDataTablesEnabled = True
```

### Code-Beispiel C#

```
vcNet1.ExtendedDataTablesEnabled = true;
```

## FilePath

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie einen Verzeichnispfad setzen, damit auch Grafik- und Gruppentitel-Dateien, bei denen nur ein relativer Dateiname

angegeben ist, in dem angegebenen Verzeichnis gefunden werden. Andernfalls wird die Datei in dem gerade aktiven Arbeitsverzeichnis der Applikation und im Installationsverzeichnis des VARCHART-Windows-Forms-Steuerelements gesucht.

Es empfiehlt sich, diese Eigenschaft beim Start der Applikation während des Initialisierens des VARCHART-Windows-Forms-Steuerelements zu setzen. Der Einfachheit halber sollte als Verzeichnispfad der Verzeichnispfad der Applikation oder ein Unterverzeichnis davon verwendet werden. Dies hat den Vorteil, dass die Applikation in einem beliebigen Verzeichnis installiert sein kann.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Verzeichnispfad <b>Standardwert:</b> " "

#### Code-Beispiel VB.NET

```
Dim exeName As String
Dim exeDir As String

exeName = System.Environment.GetCommandLineArgs(0)
exeDir = System.IO.Path.GetDirectoryName(exeName)
VcNet1.FilePath = exeDir + "\Bitmaps"
```

#### Code-Beispiel C#

```
string exeName = Environment.GetCommandLineArgs()[0];
vcNet1.FilePath = System.IO.Path.GetDirectoryName(exeName) + @"..\Bitmaps";
```

## FilterCollection

**Nur-Lese-Eigenschaft von VcNet**

Diese Eigenschaft ermöglicht den Zugriff auf die Filter-Auflistung, die alle vorhandenen Filter enthält.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFilterCollection	FilterCollection-Objekt

#### Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
filterCltn = VcNet1.FilterCollection
```

#### Code-Beispiel C#

```
VcFilterCollection filterCltn = vcNet1.FilterCollection;
```

## FontAntiAliasingEnabled

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob Schriftzeichen optisch per GDI+ geglättet werden sollen. Wenn dies bei bestimmten Schriftarten, insbesondere bei nichtlateinischen Zeichen, zu verminderter Lesefähigkeit führt, sollte die Eigenschaft auf **False** gesetzt werden.

Die Glättung per GDI+ bewirkt zusätzlich, dass die Texte bei jeder Zoomstufe die gleiche relative Ausdehnung haben, so dass immer dieselbe Anzahl Zeichen z.B. in ein Knotenfeld passt. Wenn diese Eigenschaft aber auf **False** steht, dann wird stattdessen die Einstellung des Betriebssystems übernommen (einstellbar in der **Systemsteuerung**, Dialogfeld **Anzeige**, Reiter **Darstellung: Effekte**). Wenn dort eine Kantenglättung eingeschaltet ist, dann werden also Texte weiterhin geglättet. Es kann dann aber sein, dass bei manchen Zoomstufen mehr Text sichtbar ist als bei anderen, weil die systemeigene Kantenglättung dies nicht garantiert.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Textzeichen werden optisch geglättet / nicht geglättet. <b>Standardwert:</b> true

## GroupCollection

Nur-Lese-Eigenschaft von VcNet

Diese Eigenschaft ermöglicht bei gruppierter Darstellung den Zugriff auf das GroupCollection-Objekt, das alle vorhandenen Gruppen enthält.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcGroupCollection	GroupCollection-Objekt

### Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
groupCltn = VcNet1.GroupCollection
```

### Code-Beispiel C#

```
VcGroupCollection groupCltn = vcNet1.GroupCollection;
```

## GroupHorizontalMargin

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den linken und rechten Rand von Gruppen einstellen. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Single 0 ... 9.9 mm	Breite des linken/rechten Randes von Gruppen in mm <b>Standardwert: 0</b>

### Code-Beispiel VB.NET

```
VcNet1.GroupHorizontalMargin = 1.1
```

### Code-Beispiel C#

```
VcNet1.GroupHorizontalMargin = 1.1;
```

## GroupingActivated

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie das Gruppieren von Knoten ein- oder ausschalten. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppierung aktiviert (True) / nicht aktiviert (False)

### Code-Beispiel VB.NET

```
VcNet1.Grouping = True
```

### Code-Beispiel C#

```
vcNet1.Grouping = true;
```

## GroupingDataFieldIndex

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie festlegen oder erfragen, welches Feld aus der Datendefinitionstabelle das Gruppierkriterium enthalten soll. Die Sortierung der Gruppen erfolgt automatisch in alphabetischer, numerisch aufsteigender oder absteigender Reihenfolge (s. auch **GroupNodeSortingData-**

**FieldIndex**). Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

Die Eigenschaft **GroupingDataFieldIndex** ist eine indizierte Eigenschaft, die in C# über die beiden Methoden `set_GroupingDataFieldIndex (groupingLevel, pvn)` und `get_GroupingDataFieldIndex (groupingLevel)` angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ groupingLevel	System.Int16	Gruppierenebene (beginnend mit 0)
<b>Eigenschaftswert</b>	System.Int16	Datendefinitions-Feldindex

#### Code-Beispiel VB.NET

```
Dim definitionTable As VcDataDefinitionTable
definitionTable =
VcNet1.DataDefinition.DefinitionTable (VcDataTableType.vcMaindata)
VcNet1.GroupingDataFieldIndex (0) =
definitionTable.DataDefinitionFieldByName ("Code 1").ID
VcNet1.GroupNodes (True)

VcNet1.GroupField = dataDefinitionField.ID
```

#### Code-Beispiel C#

```
VcDataDefinitionTable definitionTable =
vcNet1.DataDefinition.get_DefinitionTable (VcDataTableType.vcMaindata);
vcNet1.set_GroupingDataFieldIndex (0, "Code 1");
vcNet1.GroupNodes (true);
```

## GroupingTitlesFullyVisible

**Nur-Lese-Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Gruppentitel sichtbar bleiben, auch wenn der dargestellte Ausschnitt des Diagramms per Rollbalken verschoben wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Rückgabewert	System.Boolean	Gruppentitel sichtbar (True)/ unsichtbar (False)
<b>Eigenschaftswert</b>	System.Boolean	Sichtbar (True)/ nicht sichtbar (False) <b>Standardwert:</b> False

## GroupingType

Eigenschaft von VcNet

Diese Eigenschaft legt den Visualisierungsmodus der Gruppen fest:

- **Gruppierung:** normale Gruppendarstellung (die Breite und Höhe jeder Gruppe wird durch die Positionen der Knoten bestimmt; jede Gruppe nimmt jeweils die gesamte Breite bzw. Höhe des Netzdiagramms ein)
- **Clusterung:** Die Gruppen umschließen die darin enthaltenen Knoten möglichst platzsparend, wobei die Gruppen frei im Netzdiagramm verteilt sind.

Die Eigenschaft sollte nicht umgeschaltet werden, wenn bereits Gruppen im Chart sichtbar sind. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Gruppierung** eingestellt werden.

Weitere Informationen finden Sie im Kapitel "Wichtige Begriffe: Gruppierung".

	Datentyp	Beschreibung
Eigenschaftswert	VcGroupMode	Visualisierungsmodus Standardwert: 0

### Code-Beispiel VB.NET

```
VcNet1.GroupMode = vcGMClustering
```

### Code-Beispiel C#

```
vcNet1.GroupMode = vcGMClustering;
```

## GroupInteractionsAllowed

Eigenschaft von VcNet

Diese Eigenschaft legt fest, ob die Gruppen interaktiv kollabiert bzw. expandiert werden können (durch das Plus- bzw. Minus-Zeichen neben dem Gruppentitel).

Beim interaktiven Kollabieren/Expandieren wird das Ereignis **VcGroup-Modifying** ausgelöst.

Diese Eigenschaft sollte nicht umgeschaltet werden, wenn bereits Gruppen im Chart sichtbar sind.



Diese Eigenschaft kann auch auf der Eigenschaftenseite **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft (True)/nicht in Kraft (False) <b>Standardwert:</b> True

#### Code-Beispiel VB.NET

```
VcNet1.GroupInteractionsAllowed = False
```

#### Code-Beispiel C#

```
vcNet1.GroupInteractionsAllowed = false;
```

## GroupSortingDataFieldIndex

**Eigenschaft von VcNet**

Mit dieser Eigenschaft stellen Sie ein, welches Feld der Datendefinitionstabelle zum Sortieren der Gruppen verwendet werden soll. Standardmäßig werden die Gruppen in alphabetischer Reihenfolge nach dem **GroupField** sortiert. Durch Verwendung von **GroupNodeSortingDataFieldIndex** können Sie jede beliebige Reihenfolge erzielen. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Datendefinitions-Feldindex

#### Code-Beispiel VB.NET

```
VcNet1.GroupSortingDataFieldIndex(0) = 12
VcNet1.GroupSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcNet1.SortGroups()
```

```
VcNet1.GroupNodeSortingDataFieldIndex = dataDefinitionField.ID
```

#### Code-Beispiel C#

```
vcNet1.set_GroupSortingDataFieldIndex(0, 12);
vcNet1.set_GroupSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcNet1.SortGroups();
```

## GroupSortMode

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Sortierreihenfolge der Gruppen festlegen oder erfragen. Als Standard ist die Reihenfolge "aufsteigend nach

dem Gruppenfeld" (vcAscending) eingestellt. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcGroupSortMode	Sortierreihenfolge der Gruppen <b>Standardwert:</b> vcAscending

#### Code-Beispiel VB.NET

```
VcNet1.GroupSortMode = vcDescending
```

#### Code-Beispiel C#

```
vcNet1.GroupSortMode = vcDescending;
```

## GroupTitleDataFieldIndex

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können sie den Datendefinitions-Feldindex aus dem Knotendatensatz erfragen oder festlegen, aus dem der Gruppentitel verwendet werden soll. Ein Gruppentitel dient der Beschriftung einer Gruppe und wird in der obersten Zeile ausgegeben. Es empfiehlt sich, bei allen Knoten mit gleichem Gruppennamen denselben Titel zu verwenden, da sonst nicht absehbar ist, welcher Titel angezeigt wird. Wenn allerdings ein Dateiname in der Eigenschaft **GroupDescriptionName** steht, wird der dort vermerkte Titel benutzt.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Datendefinitions-Feldindex

#### Code-Beispiel VB.NET

```
VcNet1.GroupTitleDataFieldIndex = VcNet1.DetectFieldIndex("Maindata", "Code 2")
```

#### Code-Beispiel C#

```
vcNet1.GroupTitleDataFieldIndex = vcNet1.DetectFieldIndex("Maindata", "Code 2");
```

## GroupTitlesFileName

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie den Namen einer Datei setzen oder erfragen, aus der die Gruppentitel den vorhandenen Gruppen von Knoten

zugeordnet werden sollen. Es ist eine einfache Textdatei, in der pro Zeile eine Zuordnung gemacht wird. Am Anfang der Zeile steht der Gruppenname, dahinter folgt, getrennt durch ein Leerzeichen, der Titel. Innerhalb eines Namens sind keine Leerzeichen erlaubt. Ein leerer Gruppenname wird durch "" gekennzeichnet. Die Standard-Zuordnung für einen Namen ist "".

Wird ein relativer Dateiname angegeben, so wird die Datei zur Laufzeit zuerst in dem Verzeichnis gesucht, das in der VARCHART-ActiveX-Eigenschaft **FilePath** gesetzt ist. Wird sie dort nicht gefunden, wird sie zuerst im gerade aktiven Arbeitsverzeichnis der Applikation und dann im Installationsverzeichnis des VARCHART-ActiveX-Steuerelements gesucht.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name

#### Code-Beispiel VB.NET

```
VcNet1.GroupDescriptionName = "C:\varchart\xnet\samples\net.des"
```

#### Code-Beispiel C#

```
vcNet1.GroupDescriptionName = "C:\varchart\xnet\samples\net.des";
```

## GroupVerticalMargin

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den oberen und unteren Rand von Gruppen einstellen. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Gruppierung** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Single 0 ... 9.9 mm	Höhe des oberen/unteren Randes von Gruppen in mm <b>Standardwert:</b> 0

#### Code-Beispiel VB.NET

```
VcNet1.GroupVerticalMargin = 0.9
```

#### Code-Beispiel C#

```
VcNet1.GroupVerticalMargin = 0.9;
```

## InbuiltMouseCursorWhileDraggingEnabled

Nur-Lese-Eigenschaft von VcNet

Diese Eigenschaft bestimmt, ob während eines OLE-Drag-Vorgangs der Cursor in der Zielkomponente gesetzt werden soll. Bei OLE Drag & Drop ist es möglich, den Cursor in der Quellkomponente über das Ereignis **OLEGiveFeedback** zu setzen. Daher würde ein Setzen durch die Zielkomponente zu einem Flimmern der konkurrierenden Cursor führen, was man über diese Eigenschaft beeinflussen kann.

Außerdem können bei eingeschaltetem Cursor und der auf **vcOLEDropManual** gesetzten Eigenschaft Objekte außerhalb der Anlagerungsstellen eines Knotens nicht fallengelassen werden, während dies bei ausgeschaltetem Cursor möglich ist.

Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Knoten** setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Mauszeiger wird/wird nicht in der Zielkomponente gesetzt <b>Standardwert:</b> True

### Code-Beispiel VB.NET

```
VcNet1.OLEDragWithOwnMouseCursor = False
```

### Code-Beispiel C#

```
vcNet1.OLEDragWithOwnMouseCursor = false;
```

## InFlowGroupingActivated

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die In-Flow-Gruppierung aktivieren bzw. deaktivieren. Es wird dann automatisch eine neue Layoutberechnung für das Netzdiagramm durchgeführt (siehe VcNet-Methode **Arrange**). Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	In-Flow-Gruppierung aktiviert (True)/ deaktiviert (False) <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.InFlowGroupingEnabled = True
```

**Code-Beispiel C#**

```
vcNet1.InFlowGroupingEnabled = true;
```

## InFlowGroupingDataFieldIndex

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie das Datenfeld, das die In-Flow-Gruppierung bestimmt, erfragen oder setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Rückgabewert	System.Int16	Feld, nach dem gruppiert wird
<b>Eigenschaftswert</b>	System.Int.16	Feld, nach dem gruppiert wird <b>Standardwert:</b> -1

## InFlowGroupSeparationLineColor

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Farbe der Trennlinien der In-Flow-Gruppen erfragen bzw. setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte {0...255},{0...255},{0...255}

**Code-Beispiel VB.NET**

```
VcNet1.InFlowGroupSeparationLineColor = RGB(255, 204, 204)
```

**Code-Beispiel C#**

```
vcNet1.InFlowGroupSeparationLineColor = RGB(255, 204, 204);
```

## InFlowGroupSeparationLineType

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Typ der Trennlinien der In-Flow-Gruppen erfragen bzw. setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

Eigenschaftswert	Datentyp	Beschreibung
	VcLineType	Typ der Trennlinien der In-Flow-Gruppen
	<b>Mögliche Werte:</b>	
.vcDashed 4		Linientyp <b>gestrichelt</b>
.vcDashed 4		Linientyp <b>gestrichelt</b>
.vcDashedDotted 5		Linientyp <b>gestrichelt-gepunktet</b>
.vcDashedDotted 5		Linientyp <b>gestrichelt-gepunktet</b>
.vcDotted 3		Linientyp <b>gepunktet</b>
.vcDotted 3		Linientyp <b>gepunktet</b>
.vcLineType0 100		Linientyp 0 _____
.vcLineType1 101		Linientyp 1 - - - - -
.vcLineType10 110		Linientyp 10 _____
.vcLineType11 111		Linientyp 11 - . - . - .
.vcLineType12 112		Linientyp 12 _____
.vcLineType13 113		Linientyp 13 _____
.vcLineType14 114		Linientyp 14 _____
.vcLineType15 115		Linientyp 15 _____
.vcLineType16 116		Linientyp 16 - - - - -
.vcLineType17 117		Linientyp 17 - - - - -
.vcLineType18 118		Linientyp 18 - . - . - .
.vcLineType2 102		Linientyp 2 .....
.vcLineType3 103		Linientyp 3 - - - - -
.vcLineType4 104		Linientyp 4 - - - - -
.vcLineType5 105		Linientyp 5 - - - - -
.vcLineType6 106		Linientyp 6 - - - - -
.vcLineType7 107		Linientyp 7 - - - - -
.vcLineType8 108		Linientyp 8 - - - - -
.vcLineType9 109		Linientyp 9 _____
.vcNone 1		Kein Linientyp zugewiesen
.vcNone 1		Kein Linientyp
.vcNotSet -1		Kein Linientyp zugewiesen
.vcSolid 2		Linientyp <b>durchgezogen</b>

.vcSolid 2

Linientyp **durchgezogen**

## InFlowGroupTimeInterval

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie das Intervall der In-Flow-Gruppierung (z. B. 1 Sekunde, 1 Minute, 1 Stunde, 1 Tag, 2 Monate, 1 Jahr) festlegen bzw. erfragen. Diese Eigenschaft kann auch im Dialog **In-Flow-Gruppierung bearbeiten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcTimeOrientationInterval	Intervall der In-Flow-Gruppierung
	<b>Mögliche Werte:</b>	
	.vcDay 5	Tag
	.vcFifteenMinutes 1848	15 Minuten
	.vcFifteenSeconds 1845	15 Sekunden
	.vcFourHours 1853	4 Stunden
	.vcHalfYear 1242	halbes Jahr
	.vcHour 6	Stunde
	.vcMinute 7	Minute
	.vcMonth 3	Monat
	.vcQuarter 2	Quartal (3 Monate)
	.vcSecond 8	Sekunde
	.vcSixHours 1854	6 Stunden
	.vcThirtyMinutes 1849	30 Minuten
	.vcThirtySeconds 1846	30 Sekunden
	.vcThreeHours 1852	3 Stunden
	.vcTwelveHours 1855	12 Stunden
	.vcTwoHours 1851	2 Stunden
	.vcTwoWeeks 1238	zwei Wochen
	.vcTwoYears 1245	zwei Jahre
	.vcWeek 4	Woche
	.vcYear 1	Jahr

### Code-Beispiel VB.NET

```
VcNet1.InFlowGroupTimeInterval = vcDay
```

### Code-Beispiel C#

```
vcNet1.InFlowGroupTimeInterval = vcDay;
```

## InFlowGroupTitleDataFieldIndex

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie das Datenfeld, das die Titel der In-Flow-Gruppen enthält, erfragen oder setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeld, das die Titel der In-Flow-Gruppen enthält

**Code-Beispiel VB.NET**

```
VcNet1.InFlowGroupTitleField = 1
```

**Code-Beispiel C#**

```
VcNet1.InFlowGroupTitleField = 1;
```

## InFlowGroupTitlesBackgroundColor

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Hintergrundfarbe der Titel der In-Flow-Gruppen erfragen oder setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Hintergrundfarbe der Titel der In-Flow-Gruppen

## InFlowGroupTitlesFileName

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Datei, die die Gruppentiteltexte für die In-Flow-Gruppierung enthält, erfragen oder setzen.

Es ist eine einfache Textdatei, in der pro Zeile eine Zuordnung gemacht wird. Am Anfang der Zeile steht der Gruppenname, dahinter folgt, getrennt durch ein Leerzeichen, der Titel. Innerhalb eines Namens sind keine Leerzeichen erlaubt. Ein leerer Gruppenname wird durch "" gekennzeichnet. Die Standard-Zuordnung für einen Namen ist "".

Wird ein relativer Dateiname angegeben, so wird die Datei zur Laufzeit zuerst in dem Verzeichnis gesucht, das in der Eigenschaft **FilePath** gesetzt ist. Wird sie dort nicht gefunden, wird sie zuerst im gerade aktiven Arbeitsverzeichnis der Applikation und dann im In-stallationsverzeichnis des VARCHART-Steuerelements gesucht. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.



	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Datei, die die Titeltaxe für die In-Flow-Gruppierung enthält

## InFlowGroupTitlesFont

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die Schriftattribute der Titel der In-Flow-Gruppen erfragen oder setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	StdFont	Schriftart der Titel der In-Flow-Gruppen

## InFlowGroupTitlesVisibleAtBottomOrRight

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Titel der In-Flow-Gruppen am unteren bzw. rechten Rand der Grafik sichtbar sind. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Sichtbar (True)/ nicht sichtbar (False)

## InFlowGroupTitlesVisibleAtTopOrLeft

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Titel der In-Flow-Gruppen am oberem bzw. linken Rand der Grafik sichtbar sind. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Sichtbar (True)/ nicht sichtbar (False)

## InFlowGroupTitleTimeFormat

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie das Datums-/Zeit-Ausgabeformat für die In-Flow-Gruppierung nach Datumfeld erfragen bzw. setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Datums-/Zeit-Ausgabeformat für die In-Flow-Gruppierung nach Datumfeld <b>Standardwert:</b> DD.MM.YYYY

## InFlowGroupVerticalCaptionWidth

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die Breite der Titelleisten bei Orientierung von oben nach unten in mm erfragen bzw. setzen. Sie können diese Eigenschaft auch im Dialog **In-Flow-Gruppierung bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite der Titelleisten bei Orientierung von oben nach unten in mm <b>Standardwert:</b> 50

### Code-Beispiel VB.NET

```
VcNet1.InFlowGroupVerticalCaptionWidth = 30
```

### Code-Beispiel C#

```
VcNet1.InFlowGroupVerticalCaptionWidth = 30;
```

## InPlaceEditingAllowed

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob zur Laufzeit das direkte Editieren von Datenfeldern in der Tabelle, in Boxen und in Layern möglich ist. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

**Hinweis:** Falls einzelne Datenfelder nicht editierbar sein sollen, darf im Dialog **Datentabellen verwalten** das Attribut **editierbar** nicht ausgewählt sein.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Direktes Editieren möglich (True) / nicht möglich (False) <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcGantt1.InPlaceEditingAllowed = True
```

**Code-Beispiel C#**

```
vcGantt1.InPlaceEditingAllowed = true;
```

**InteractionMode****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie einen der verfügbaren Interaktionsmodi einstellen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcInteractionMode	Interaktionsmodus <b>Standardwert:</b> vcPointer
	<b>Mögliche Werte:</b> .vcCreateLink 4 .vcCreateNodesAndLinks 1 .vcPointer 0	Erzeugemodus für Verbindungen Erzeugemodus für Knoten und Verbindungen Selektiermodus

**Code-Beispiel VB.NET**

```
VcNet1.InteractionMode = vcCreateNodesAndNodes
```

**Code-Beispiel C#**

```
vcNet1.InteractionMode = vcCreateNodesAndNodes;
```

**InterfaceNodesShown****Eigenschaft von VcNet**

Mit Hilfe dieser Eigenschaft können Sie festlegen, ob bei der Erstellung eines Teildiagramms auch die Anschlussknoten dargestellt werden sollen (True) oder nicht (False). Das Aussehen dieser Anschlussknoten kann man im Dialog **Knotenaussehen bearbeiten** bestimmen, indem man als Spezialfilter <InterfaceNodes> angibt. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft/nicht in Kraft <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.InterfaceNodesShown = False
```

**Code-Beispiel C#**

```
vcNet1.InterfaceNodesShown = false;
```

## LegendView

**Nur-Lese-Eigenschaft von VcNet**

Diese Eigenschaft ermöglicht den Zugriff auf das LegendView-Objekt, das die Legendenansicht des Diagramms definiert.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLegendView	LegendView-Objekt

**Code-Beispiel VB.NET**

```
Dim legendview As VcLegendView
legendview = VcNet1.LegendView
legendview.Visible = True
```

**Code-Beispiel C#**

```
VcLegendView legendview = vcNet1.LegendView;
legendview.Visible = true;
```

## LinkAnnotationColumnNumberDataFieldIndex

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, in das die Spaltennummer jeder Verbindungsbeschriftung geschrieben werden soll. Das Setzen dieser Eigenschaft ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Index des Datenfeldes, das die Spaltennummer jeder Verbindungsbeschriftung enthält

## LinkAnnotationRowIndex

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, in das die Zeilennummer jeder Verbindungsbeschriftung geschrieben werden soll. Das Setzen dieser Eigenschaft ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Index des Datenfeldes, das die Zeilennummer jeder Verbindungsbeschriftung enthält

### Code-Beispiel VB.NET

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Handles MyBase.Load

    VcNet1.NodeRowIndex =
    VcNet1.DetectFieldIndex("NodeDataTable", "SortNumber")

    'Load data
    LoadData()

    VcNet1.UpdateRowIndex()
    VcNet1.SaveAsEx("C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding)
End Sub
```

### Code-Beispiel C#

```
private void Form1_Load(object sender, System.EventArgs e)
{
    vcNet1.NodeRowIndex =
    vcNet1.DetectFieldIndex("NodeDataTable", "SortNumber");

    // Load data
    loadData();

    vcNet1.UpdateRowIndex();
    vcNet1.SaveAsEx(@"C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding);
}
```

## LinkAppearanceCollection

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft haben Sie Zugriff auf das LinkAppearanceCollection-Objekt, in dem alle vorhandenen LinkAppearance-Objekte zusammengefasst sind.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkAppearanceCollection	LinkAppearanceCollection-Objekt

**Code-Beispiel VB.NET**

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
linkAppearanceCltn = VcNet1.LinkAppearanceCollection
```

**Code-Beispiel C#**

```
VcLinkAppearanceCollection linkAppearanceCltn = vcNet1.LinkAppearanceCollection;
```

## LinkCollection

**Nur-Lese-Eigenschaft von VcNet**

Diese Eigenschaft ermöglicht den Zugriff auf das LinkCollection-Objekt, das alle vorhandenen Verbindungen enthält.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkCollection	LinkCollection-Objekt

**Code-Beispiel VB.NET**

```
Dim linkCltn As VcLinkCollection
linkCltn = VcNet1.LinkCollection
```

**Code-Beispiel C#**

```
VcLinkCollection linkCltn = vcNet1.LinkCollection;
```

## LinkCreationWithDialog

**Eigenschaft von VcNet**

Mit dieser Eigenschaft wird festgelegt, ob bei der Erzeugung einer neuen Verbindung der Dialog **Daten bearbeiten** erscheinen soll. Die Eigenschaft **AllowNewNodesAndLinks** muss auf True gesetzt sein, damit interaktiv eine neue Verbindung erzeugt werden kann.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Der <b>Verbindung bearbeiten</b> -Dialog erscheint (True) / erscheint nicht (False) beim Anlegen

**Code-Beispiel VB.NET**

```
VcNet1.EditNewLink = False
```

**Code-Beispiel C#**

```
vcNet1.EditNewLink = false;
```

## LinkFormatCollection

Nur-Lese-Eigenschaft von VcNet

Diese Eigenschaft ermöglicht den Zugriff auf die LinkFormat-Auflistung und damit auf die verwendeten Verbindungsformate.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkFormatCollection	LinkFormatCollection-Objekt

### Code-Beispiel VB.NET

```
Dim formatCollection As VcLinkFormatCollection
Set formatCollection = VcNet1.LinkFormatCollection
```

## LinkPredecessorDataFieldIndex

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, in dem die Identifizierung des Vorgängerknotens der Verbindung enthalten ist. Dies ist nur möglich, solange noch keine Daten geladen wurden.

Die Eigenschaft `LinkPredecessorDataFieldIndex` ist eine indizierte Eigenschaft, die in C# über die beiden Methoden `set_LinkPredecessorDataFieldIndex (identifizierIndex, pvn)` und `get_LinkPredecessorDataFieldIndex (identifizierIndex)`.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ identifizierIndex	System.Int32	Index des Vorgängerknotens {0...2}
<b>Eigenschaftswert</b>	System.Int32	Feldindex aus der Datendefinitionstabelle

**Code-Beispiel VB.NET**

```

Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcNet1.DataTableCollection.Update()

VcNet1.LinkPredecessorDataFieldIndex(0) =
VcNet1.DetectFieldIndex("LinkDataTable", "Id")
VcNet1.LinkSuccessorDataFieldIndex(0) = VcNet1.DetectFieldIndex("LinkDataTable",
"Id")

'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading()

```

**Code-Beispiel C#**

```

VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcNet1.DataTableCollection.Update();

vcNet1.set_LinkPredecessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));
vcNet1.set_LinkSuccessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcNet1.EndLoading();

```

**LinksDataTableName****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie den Namen der Tabelle setzen oder erfragen, die die Felder für die Verbindungen enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name der Datentabelle aus der die Verbindungen kommen



**Code-Beispiel VB.NET**

```

Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcNet1.DataTableCollection.Update()

VcNet1.LinkPredecessorDataFieldIndex(0) =
VcNet1.DetectFieldIndex("LinkDataTable", "Id")
VcNet1.LinkSuccessorDataFieldIndex(0) = VcNet1.DetectFieldIndex("LinkDataTable",
"Id")

'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading()

```

**Code-Beispiel C#**

```

VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcNet1.DataTableCollection.Update();

vcNet1.set_LinkPredecessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));
vcNet1.set_LinkSuccessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcNet1.EndLoading();

```

**LinkSuccessorDataFieldIndex****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, in dem die Identifizierung des Nachfolgerknotens der Verbindung enthalten ist. Dies ist nur möglich, solange noch keine Daten geladen wurden.

Die Eigenschaft **LinkSuccessorDataFieldIndex** ist eine indizierte Eigenschaft, die in C# über die beiden Methoden `set_LinkSuccessorDataFieldIndex (identifizierIndex, pvn)` und `get_LinkSuccessorDataFieldIndex (identifizierIndex)` angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ identifizierIndex	System.Int32	Index des Nachfolgerknotens {0...2}
<b>Eigenschaftswert</b>	System.Int32	Feldindex aus der Datendefinitionstabelle

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcNet1.DataTableCollection.Update()

VcNet1.LinkPredecessorDataFieldIndex(0) =
VcNet1.DetectFieldIndex("LinkDataTable", "Id")
VcNet1.LinkSuccessorDataFieldIndex(0) = VcNet1.DetectFieldIndex("LinkDataTable",
"Id")

'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcNet1.EndLoading()
```

### Code-Beispiel C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcNet1.DataTableCollection.Update();

vcNet1.set_LinkPredecessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));
vcNet1.set_LinkSuccessorDataFieldIndex(0,
vcNet1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcNet1.EndLoading();
```

## LinkTypeDataFieldIndex

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das den Verbindungstyp enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Index des Datenfeldes, das die Verbindungstypen enthält

**Code-Beispiel VB.NET**

```
Dim dataTable As VcDataTable

'create Link DataTable
dataTable = VcNet1.DataTableCollection.Add("LinkDataTable")
VcNet1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
dataTable.DataTableFieldCollection.Add("LinkType")
VcNet1.DataTableCollection.Update()
```

**Code-Beispiel C#**

```
VcDataTable dataTable;

//create Link DataTable
dataTable = vcNet1.DataTableCollection.Add("LinkDataTable");
vcNet1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
dataTable.DataTableFieldCollection.Add("LinkType");
vcNet1.DataTableCollection.Update();
```

**MapCollection****Nur-Lese-Eigenschaft von VcNet**

Diese Eigenschaft ermöglicht den Zugriff auf das MapCollection-Objekt, in dem eine bestimmte Menge von Zuordnungstabellen zusammengefasst ist. Die Menge der Zuordnungstabellen wird durch die Methode **VcMapCollection.SelectMaps** definiert.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcMapCollection	MapCollection-Objekt

**Code-Beispiel VB.NET**

```
Dim mapCltn As VcMapCollection
mapCltn = VcNet1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

**Code-Beispiel C#**

```
VcMapCollection mapCltn = vcNet1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
```

## MinimumColumnWidth

Eigenschaft von VcNet

Mit dieser Eigenschaft wird die minimale Breite einer Spalte (Einheit: mm) eingestellt. Die eingestellte Breite sollte etwa der mittleren Breite eines Knotens entsprechen. Damit Verbindungen bei der Orientierung von links nach rechts weniger Platz auf dem Bildschirm einnehmen, kann die Breite weiter herabgesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32 {1...1 000}	Mindestspaltenbreite in mm <b>Standardwert: 1</b>

### Code-Beispiel VB.NET

```
VcNet1.MinimumColumnWidth = 100
```

### Code-Beispiel C#

```
vcNet1.MinimumColumnWidth = 100;
```

## MinimumRowHeight

Eigenschaft von VcNet

Mit dieser Eigenschaft wird die minimale Höhe einer Zeile (Einheit: 1/100 mm) eingestellt. Die eingestellte Höhe sollte etwa der mittleren Höhe eines Knotens entsprechen. Damit Verbindungen bei der Orientierung von oben nach unten weniger Platz auf dem Bildschirm einnehmen, kann die Höhe weiter herabgesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

Die minimale Höhe ist nur dann wirksam, wenn sich kein Vorgang oder nur Vorgänge mit geringerer grafischer Höhe in der Zeile befinden. In allen anderen Fällen wird die Zeilenhöhe entsprechend dem erforderlichen Platzbedarf automatisch vergrößert. Der einstellbaren Werte liegen zwischen 2 und 1000.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32 {1...1 000}	Mindestzeilenhöhe in mm <b>Standardwert: 1</b>

### Code-Beispiel VB.NET

```
VcNet1.MinimumRowHeight = 100
```

**Code-Beispiel C#**

```
vcNet1.MinimumRowHeight = 100;
```

**MouseProcessingEnabled****Eigenschaft von VcNet**

Diese Eigenschaft kann dazu genutzt werden, Ihre eigene Verarbeitung von Mausereignissen zu ermöglichen. Wenn Sie eine eigene Verarbeitung vom Ereignis **VcMouseDown** bis zum **VcMouseUp** durchführen möchten, setzen Sie die Eigenschaft **MouseProcessingEnabled** für diese Zeit auf **False**. Dann ignoriert VARCHART XNet bis zum Zurücksetzen auf **True** alle Mausbewegungen und Klicks.

Diese Eigenschaft kann auch in den VcMouse-Ereignissen gesetzt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft (True)/ nicht in Kraft (False) <b>Standardwert:</b> True

**MovingCollapsedClustersAllowed****Eigenschaft von VcNet**

Mit dieser Eigenschaft wird dem Anwender bei Clusterung das Verschieben kollabierter Cluster erlaubt (True) bzw. verboten (False). Diese Eigenschaft kann auch auf der Eigenschaftenseite **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Verschieben kollabierter Cluster erlaubt (True)/nicht erlaubt (False) <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
Dim boole As Boolean
boole = VcNet1.GroupMovingAllowed
```

**Code-Beispiel C#**

```
Boolean boole = vcNet1.GroupMovingAllowed;
```

## NodeAndLinkCreationAllowed

Eigenschaft von VcNet

Mit dieser Eigenschaft wird dem Anwender das Anlegen neuer Knoten und Verbindungen erlaubt (True) oder gesperrt (False). Wird diese Eigenschaft auf False gesetzt, kann der Erzeugemodus nicht eingeschaltet werden.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft (True)/nicht in Kraft (False) <b>Standardwert:</b> True

### Code-Beispiel VB.NET

```
Dim boole As Boolean

boole = VcNet1.AllowNewNodesAndLinks
```

### Code-Beispiel C#

```
Boolean boole = vcNet1.AllowNewLinksAndNodes;
```

## NodeAppearanceCollection

Nur-Lese-Eigenschaft von VcNet

Mit dieser Eigenschaft haben Sie Zugriff auf das NodeAppearanceCollection-Objekt, in dem alle zur Verfügung stehenden NodeAppearance-Objekte zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeAppearanceCollection	NodeAppearanceCollection-Objekt

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.BackgroundColor = Color.LightBlue
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.BackgroundColor = Color.LightBlue;
```

## NodeCalendarNameDataFieldIndex

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das den Namen des für einen Knoten zu verwendenden Kalenders enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das den Namen des für den Knoten zu verwendenden Kalenders enthält

## NodeChangeRankToPredecessorRankDataFieldIndex

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, in das der Rang des Vorläuferknotens eingetragen wird. Dies ist nur möglich, solange noch keine Daten geladen wurden.

Wählen Sie dann das

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das die Rangnummer des Vorläuferknotens enthält

## NodeCollection

Nur-Lese-Eigenschaft von VcNet

Diese Eigenschaft ermöglicht den Zugriff auf das NodeCollection-Objekt, in dem abhängig von **SelectNodes** alle Knoten (vcAll), nur die markierten (vcMarked) oder nur die sichtbaren (vcAllVisible) Knoten zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeCollection	NodeCollection-Objekt

### Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes (VcSelectionType.vcAll
```

**Code-Beispiel C#**

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes (VcSelectionType.vcAll);
```

**NodeColumnNumberDataFieldIndex****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, in das die Spaltennummer jedes Vorgangs geschrieben werden soll. Das Setzen dieser Eigenschaft ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Index des Datenfeldes, das die Spaltennummer jedes Vorgangs enthält

**NodeCreationWithDialog****Eigenschaft von VcNet**

Mit dieser Eigenschaft wird festgelegt, ob bei der Erzeugung eines neuen Knotens der Dialog **Vorgänge bearbeiten** erscheinen soll. Dabei muss die Eigenschaft **AllowNewNodesAndLinks** auf True gesetzt sein, damit überhaupt ein neuer Knoten erzeugt werden kann.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	<b>Vorgänge bearbeiten</b> -Dialog erscheint (True) / erscheint nicht (False) beim Anlegen

**Code-Beispiel VB.NET**

```
VcNet1.NodeCreationWithDialog = False
```

**Code-Beispiel C#**

```
vcNet1.NodeCreationWithDialog = false;
```

**NodeFormatCollection****Nur-Lese-Eigenschaft von VcNet**

Diese Eigenschaft ermöglicht den Zugriff auf das NodeFormatCollection-Objekt und damit auf die verwendeten Knotenformate.



	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcNodeFormatCollection	NodeFormatCollection-Objekt

**Code-Beispiel VB.NET**

```
Dim formatCtln As VcNodeFormatCollection
formatCtln = VcNet1.NodeFormatCollection
```

**Code-Beispiel C#**

```
VcNodeFormatCollection formatCtln = vcNet1.NodeFormatCollection;
```

**NodeRowIndex****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, in das die Zeilennummer jedes Vorgangs geschrieben werden soll. Das Setzen dieser Eigenschaft ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Index des Datenfeldes, das die Zeilennummer jedes Vorgangs enthält

**Code-Beispiel VB.NET**

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    VcNet1.NodeRowIndex =
VcNet1.DetectFieldIndex("NodeDataTable", "SortNumber")

    'Load data
    LoadData()

    VcNet1.UpdateRowIndexFields()
    VcNet1.SaveAsEx("C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding)
End Sub
```

**Code-Beispiel C#**

```
private void Form1_Load(object sender, System.EventArgs e)
{
    vcNet1.NodeRowIndex =
vcNet1.DetectFieldIndex("NodeDataTable", "SortNumber");

    // Load data
    loadData();

    vcNet1.UpdateRowIndexFields();
    vcNet1.SaveAsEx(@"C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding);
}
```

## NodesDataTableName

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie den Namen der Tabelle setzen oder erfragen, die die Felder für die Darstellung der Knoten zur Verfügung stellt.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Datentabelle aus der die Knoten kommen

### Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Node DataTable
dataTable = VcNet1.DataTableCollection.Add("NodeDataTable")
VcNet1.NodesDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
'Load Data
dataTable = VcNet1.DataTableCollection.DataTableByName("NodeDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;Node One;")
dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;")
VcNet1.EndLoading()
```

### Code-Beispiel C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Node DataTable
dataTable = vcNet1.DataTableCollection.Add("NodeDataTable");
vcNet1.NodesDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
//Load Data
dataTable = vcNet1.DataTableCollection.DataTableByName("NodeDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;Node One;");
dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;");
vcNet1.EndLoading();
```

## NodesUseCalendars

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie festlegen, ob den Vorgängen ein Kalender zugewiesen werden soll. Die Kalenderzuweisung wirkt sich beim Verschieben von Vorgängen aus: Anfang und Ende der Vorgänge werden nicht auf arbeitsfreie Tage gelegt. Außerdem werden beim Berechnen der Dauer von Vorgängen die arbeitsfreien Zeiten berücksichtigt. Standardmäßig ist ein Fünf-Tage-Kalender definiert, Sie können aber auch eigene Kalender definieren. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Es werden Kalender verwendet (True) / nicht verwendet (False) <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.NodesUseCalendars = False
```

**Code-Beispiel C#**

```
vcNet1.NodesUseCalendars = false;
```

## NodeToolTipTextDataFieldIndex

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie erfragen bzw. festlegen, welches Datenfeld eines Knotens für Tooltips in VMF-Dateien genutzt werden soll. Drückt man im WebViewer die rechte Maustaste, erscheint der zugeordnete Text. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Index des Knoten-Datenfelds für Tooltiptexte <b>Standardwert:</b> 4

**Code-Beispiel VB.NET**

```
VcNet1.NodeToolTipTextDataFieldIndex = 1
```

**Code-Beispiel C#**

```
vcNet1.NodeToolTipTextDataFieldIndex = 1;
```

## ObliqueTracksOnLinks

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Verbindungslinien direkt an den horizontalen Linienstücken ansetzen und schräg verlaufen oder ob die Verbindungslinien orthogonal dargestellt werden. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean {True, False}	schräge Verbindungslinien (True)/orthogonale Verbindungslinien (False) <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.ObliqueTracksOnLinks = True
```

**Code-Beispiel C#**

```
vcNet1.ObliqueTracksOnLinks = true;
```

## Orientation

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Flussrichtung des Diagramms einstellen oder erfragen. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Allgemeines** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLayoutOrientation  <b>Mögliche Werte:</b> .vcLeftToRight 0  .vcTopToBottom 1	Von links nach rechts/von oben nach unten  Flussrichtung im Netzdiagramm <b>von links nach rechts</b> Flussrichtung im Netzdiagramm von oben nach unten

**Code-Beispiel VB.NET**

```
VcNet1.Orientation = vcLeftToRight
```

**Code-Beispiel C#**

```
vcNet1.Orientation = vcLeftToRight;
```

## PhantomDrawingWhileDraggingEnabled

**Eigenschaft von VcNet**

Diese Eigenschaft bestimmt, ob während eines OLE-Drag-Vorgangs ein Phantom erscheinen soll oder nicht. Das Abschalten des Phantoms ist für Anwendungen gedacht, die beim Hineindragen eines Objekts kein neues Objekt erzeugen, sondern z. B. den Knoten, auf dem dann ein Objekt fallengelassen wird, nur neu attributieren.

Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Knoten** setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Phantom erscheint/erscheint nicht <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.OLEDragWithPhantom = False
```

**Code-Beispiel C#**

```
vcNet1.OLEDragWithPhantom = false;
```

## Printer

**Eigenschaft von VcNet**

Mit diesem Objekt können Sie die Eigenschaften des aktuell verwendeten Druckers erfragen oder setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcPrinter	Druckerobjekt

**Code-Beispiel VB.NET**

```
Dim printerZoomfactor As Integer
Dim printerCuttingMarks As String
```

```
printerZoomfactor = VcNet1.Printer.ZoomFactor
printerCuttingMarks = VcNet1.Printer.CuttingMarks
```

**Code-Beispiel C#**

```
int printerZoomfactor = vcNet1.Printer.ZoomFactor;
bool printerCuttingMarks = vcNet1.Printer.CuttingMarks;
```

## RoundedLinkSlantsEnabled

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob die Schrägen bei Verbindungen vom Routing-Typ **vcLRTOrthogonal** als Viertelkreise statt als gerade Linien dargestellt werden sollen. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Schrägen bei Verbindungen werden/werden nicht als Viertelkreise dargestellt <b>Standardwert:</b> false

**Code-Beispiel VB.NET**

```
VcNet1.RoundedLinkSlantsEnabled = True
```

**Code-Beispiel C#**

```
vcNet1.RoundedLinkSlants.Enabled = true;
```

## Scheduler

**Nur-Lese-Eigenschaft von VcNet**

Diese Eigenschaft gibt das VcScheduler-Objekt zurück.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcScheduler	Gibt das VcScheduler-Objekt zurück

## ShortenedLinks

**Eigenschaft von VcNet**

Diese Eigenschaft wirkt sich auf das Layout eines Netzdiagramms aus und wird von der Methode **Arrange** berücksichtigt. Wird die Eigenschaft auf **True** gesetzt, werden alle Knoten so nahe wie möglich an ihre Nachfolgerknoten herangeschoben, um die Verbindungen möglichst kurz zu halten. Wird sie auf **False** gesetzt, werden Knoten grundsätzlich so weit links bzw. oben wie möglich angeordnet. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Allgemeines** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft/nicht in Kraft

**Code-Beispiel VB.NET**

```
VcNet1.ShortenedNodes = False
VcNet1.Arrange
```

**Code-Beispiel C#**

```
vcNet1.ShortenedLinks = false
vcNet1.Arrange;
```

## StraightLinkDrawing

**Eigenschaft von VcNet**

Ist diese Eigenschaft auf **True** gesetzt, können Verbindungen zwischen Knoten direkt, das heißt, nicht orthogonal mit Knicken, gezeichnet werden.

Diese Eigenschaft macht die Eigenschaft **ObliqueTracksOnLinks** unwirksam, wenn sie gesetzt wird.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Direktes Zeichnen von Verbindungen eingeschaltet (True) / ausgeschaltet (False) <b>Standardwert:</b> false

## TextEntrySupplyingEventEnabled

Eigenschaft von VcNet

Mit dieser Eigenschaft aktivieren Sie das **VcTextEntrySupplying**-Ereignis, über das Sie die Texte der VARCHART-XNet-Komponente verändern können, z. B. um sie in eine andere Sprache zu übersetzen. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft/nicht in Kraft

### Code-Beispiel VB.NET

```
VcNet1.TextEntrySupplyingEventEnabled = True
```

### Code-Beispiel C#

```
vcNet1.TextEntrySupplyingEventEnabled = true;
```

## TimeUnit

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die Zeiteinheit für die Berechnung der Dauer (siehe "Layer") und für das interaktive Anlegen und Verändern Ihrer Knoten in der Darstellung setzen oder erfragen. Haben Sie beispielsweise die Zeiteinheit Tage gewählt, lassen sich Knoten nur in Sprüngen von ganzen Tagen anlegen und verändern, und die Dauer von Knoten wird ebenfalls in Tagen berechnet. Die Eigenschaft **TimeUnit** kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

Änderungen der Eigenschaft sollten vor dem Einlesen von Daten, d.h. am besten beim Start erfolgen, weil nachträgliche Änderungen keine Auswirkung haben.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcTimeUnit	Zeiteinheit <b>Standardwert:</b> vcDay

**Code-Beispiel VB.NET**

```
Dim timeUnit As VcTimeUnit
timeUnit = VcNet1.TimeUnit
```

**Code-Beispiel C#**

```
VcTimeUnit timeUnit = vcNet1.TimeUnit;
```

**ToolTipChangeDuration****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Zeitdauer setzen, die vergeht, bevor das nächste ToolTip-Fenster auf dem Bildschirm erscheint, wenn der Mauszeiger auf das nächste Objekt gesetzt wird. Einheit: Millisekunden. Mit der Ziffer -1 stellen Sie den Standardwert von 98 Millisekunden (für Windows XP) ein.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Zeitdauer in Millisekunden. Maximalwert = 32767 ms <b>Standardwert:</b> -1

**ToolTipDuration****Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie die Zeitdauer setzen, während der das ToolTip-Fenster sichtbar bleiben soll (sofern der Mauszeiger innerhalb des umgebenden Rechtecks eines Objektes unbewegt bleibt). Einheit: Millisekunden. Mit der Ziffer -1 stellen Sie den Standardwert von 5.000 Millisekunden ein.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Zeitdauer in Millisekunden. Maximalwert = 32767 ms <b>Standardwert:</b> -1



## ToolTipPointerDuration

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie die Zeitdauer setzen, während der der Mauszeiger innerhalb des umgebenden Rechtecks eines Objektes unbewegt bleiben muss, damit das ToolTip-Fenster erscheint. Einheit: Millisekunden. Mit der Ziffer -1 stellen Sie den Standardwert von 480 Millisekunden (für Windows XP) ein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Zeitdauer in Millisekunden <b>Standardwert:</b> -1

## ToolTipShowAfterClick

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie einstellen, ob das angezeigte ToolTip-Fenster beim Anklicken des Objektes verschwinden soll (Standard-Verhalten) oder entsprechend seiner eingestellten Zeiten weiterhin angezeigt werden soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	ToolTip-Fenster verschwindet (false) oder bleibt (true) <b>Standardwert:</b> False

## ToolTipTextSupplyingEventEnabled

Eigenschaft von VcNet

Mit dieser Eigenschaft können Sie das Ereignis **VcToolTipTextSupplying** aktivieren oder deaktivieren. Mit Hilfe dieses Ereignisses können Sie die Texte der Tooltips bestimmen.

Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Allgemeines** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft/nicht in Kraft <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.ToolTipTextSupplyingEventEnabled = True
```

**Code-Beispiel C#**

```
vcNet1.ToolTipTextSupplyingEventEnabled = true;
```

## UngroupedNodesAllowed

**Eigenschaft von VcNet**

Diese Eigenschaft legt fest, ob Knoten, deren Gruppencode der leere String "" ist, nicht innerhalb einer Gruppe oder in einer eigenen Gruppe für Vorgänge ohne Gruppencode-Angabe dargestellt werden.

Diese Eigenschaft ist nur beim Gruppierungsmodus Clustering (GroupMode = vcGMClustering) aktiv.

Sie sollte nicht umgeschaltet werden, wenn bereits Gruppen im Chart sichtbar sind.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Eigenschaft in Kraft (True)/nicht in Kraft (False) <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.UngroupedNodesAllowed = True
```

**Code-Beispiel C#**

```
vcNet1.UngroupedNodesAllowed = true;
```

## ViewXCoordinate

**Eigenschaft von VcNet**

Mit dieser Eigenschaft kann der aktuelle Scroll-Offset des angezeigten Ausschnitts in x-Richtung gespeichert werden und bei einem neuen Start der

gleichen Anwendung wieder gesetzt werden. Letzteres setzt voraus, dass zuvor auch der Zoomfaktor auf dieselbe Weise gesetzt wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Scroll-Offset in x-Richtung

## ViewYCoordinate

Eigenschaft von VcNet

Mit dieser Eigenschaft kann der aktuelle Scroll-Offset des angezeigten Ausschnitts in y-Richtung gespeichert werden und bei einem neuen Start der gleichen Anwendung wieder gesetzt werden. Letzteres setzt voraus, dass zuvor auch der Zoomfaktor auf dieselbe Weise gesetzt wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Scroll-Offset in y-Richtung

## WaitCursorEnabled

Eigenschaft von VcNet

Mit dieser Eigenschaft kann man steuern, ob bei zeitkritischen Aufrufen (wie ScheduleProject) ein Wartecursor gesetzt werden soll oder nicht.

Die Eigenschaft kann auch auf der Eigenschaftseite **Allgemeines** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Wartecursor wird/wird nicht gesetzt <b>Standardwert:</b> False

## WorldView

Nur-Lese-Eigenschaft von VcNet

Über diese Eigenschaft erhalten Sie Zugriff auf das VcWorldView-Objekt, das die Komplettansicht des Diagramms definiert.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcWorldView	Komplettansicht-Objekt

**Code-Beispiel VB.NET**

```
Dim worldview As VcWorldView

worldview = VcNet1.WorldView
worldview.Visible = True
```

**Code-Beispiel C#**

```
VcWorldView worldview = vcNet1.WorldView;
worldview.Visible = true;
```

## ZoomFactor

**Eigenschaft von VcNet**

Mit dieser Eigenschaft kann der absolute Zoomfaktor der Bildschirmdarstellung in % angegeben oder erfragt werden (Zoomfaktor = 100: Originalgröße, Zoomfaktor > 100: Vergrößerung, Zoomfaktor < 100: Verkleinerung).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16 {0...1000}	Zoomfaktor

**Code-Beispiel VB.NET**

```
VcNet1.ZoomFactor = 150
```

**Code-Beispiel C#**

```
vcNet1.ZoomFactor = 150;
```

## ZoomingPerMouseWheelAllowed

**Eigenschaft von VcNet**

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das Zoomen per Mausrad zugelassen ist. Um zu zoomen, muss der Anwender dann die Strg-Taste festhalten und das Mausrad drehen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Zoomen erlaubt (True)/nicht erlaubt (False)

**Code-Beispiel VB.NET**

```
VcNet1.ZoomingPerMouseWheelAllowed = False
```

**Code-Beispiel C#**

```
vcNet1.ZoomingPerMouseWheelAllowed = false;
```

---

## Methoden

### Arrange

**Methode von VcNet**

Mit dieser Methode wird eine neue Layoutberechnung für das Netzdiagramm durchgeführt. Dabei wird die Eigenschaft **ShortenedLinks** berücksichtigt.

	Datentyp	Beschreibung
Rückgabewert	Void	

**Code-Beispiel VB.NET**

```
VcNet1.Arrange
```

**Code-Beispiel C#**

```
vcNet1.Arrange;
```

### Clear

**Methode von VcNet**

Mit dieser Methode werden alle grafisch darstellbaren Objekte aus dem Diagramm gelöscht. Das Ergebnis dieser Methode ist ein "leeres" Chart (wie beim ersten Start gemäß den Einstellungen in der INI-Datei), in das nun wieder Objekte geladen werden können.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Knoten erfolgreich gelöscht {True}

**Code-Beispiel VB.NET**

```
VcNet1.Clear
```

**Code-Beispiel C#**

```
vcNet1.Clear;
```

## CompleteViewMode

Methode von VcNet

Mit dieser Methode können Sie die Darstellung so einstellen, dass stets das komplette Diagramm angezeigt wird. Der Zoomfaktor passt sich bei jeder Änderung des Charts automatisch an. Der maximale Zoomfaktor von 100% wird nicht überschritten, die Knoten werden also höchstens in Originalgröße dargestellt. Siehe auch Eigenschaft **ZoomFactor** und Methode **Zoom**.

	Datentyp	Beschreibung
Rückgabewert	Void	

### Code-Beispiel VB.NET

```
VcNet1.CompleteViewMode
```

### Code-Beispiel C#

```
vcNet1.CompleteViewMode;
```

## CopyNodesIntoClipboard

Methode von VcNet

Mit dieser Methode werden die markierten Knoten vom Netzdiagramm in den Zwischenspeicher kopiert. Siehe auch die Methoden **CutNodesIntoClipboard** und **PasteNodesFromClipboard**.

	Datentyp	Beschreibung
Rückgabewert	Void	

### Code-Beispiel VB.NET

```
VcNet1.CopyNodesIntoClipboard
```

### Code-Beispiel C#

```
vcNet1.CopyNodesIntoClipboard;
```

## CutNodesIntoClipboard

Methode von VcNet

Mit dieser Methode werden die im Netzdiagramm markierten Knoten ausgeschnitten und in den Zwischenspeicher abgelegt. Siehe auch **CopyNodesIntoClipboard** und **PasteNodesFromClipboard**.

	Datentyp	Beschreibung
Rückgabewert	Void	

**Code-Beispiel VB.NET**

```
VcNet1.CutNodesIntoClipboard
```

**Code-Beispiel C#**

```
vcNet1.CutNodesIntoClipboard;
```

## DeleteLinkRecord

**Methode von VcNet**

Mit dieser Methode können Sie eine Verbindung löschen, indem Sie den Datensatz der Verbindung angeben (s. auch Objekt **VcLink**, Methode **Delete**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ linkRecord	System.Object	Datensatz der Verbindung
<b>Rückgabewert</b>	System.Boolean	Datensatz der Verbindung erfolgreich/nicht erfolgreich gelöscht

**Code-Beispiel VB.NET**

```
VcNet1.DeleteLinkRecord "A100;A105;;"
```

**Code-Beispiel C#**

```
vcNet1.DeleteLinkRecord "A100;A105;;";
```

## DeleteNodeRecord

**Methode von VcNet**

Mit dieser Methode können Sie einen Knoten löschen. Der Knoten wird durch den Primärschlüssel im Datensatz identifiziert. Welches Datenfeld als Identifizierung verwendet wird, wird im Dialog **Datentabellen verwalten** festgelegt.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ nodeRecord	System.Object	Datensatz des Knotens
<b>Rückgabewert</b>	System.Boolean	Datensatz des Knotens erfolgreich/nicht erfolgreich gelöscht

**Code-Beispiel VB.NET**

```
VcNet1.DeleteNodeRecord "A100;;;;;"
```

**Code-Beispiel C#**

```
vcNet1.DeleteNodeRecord "A100;;;;;"
```

## DetectDataTableFieldName

**Methode von VcNet**

Mit dieser Eigenschaft können Sie über den Index eines Tabellendatenfeldes seinen Namen erfragen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fieldIndex	System.Int32	Index des Datentabellenfeldes, dessen Name ermittelt werden soll
<b>Rückgabewert</b>	System.String	Zurückgegebener Name des Datentabellenfeldes

**Code-Beispiel VB.NET**

```
'Find the name of a DataTableField
Dim fieldName As String

fieldName = VcNet1.DetectDataTableFieldName(0)
```

**Code-Beispiel C#**

```
//Find the name of a DataTableField
string fieldName = vcNet1.DetectDataTableFieldName(0);
```

## DetectDataTableName

**Methode von VcNet**

Mit dieser Eigenschaft können Sie über den Index einer Datentabelle ihren Namen erfragen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fieldIndex	System.Int32	Index der Datentabelle, deren Name ermittelt werden soll
<b>Rückgabewert</b>	System.String	Zurückgegebener Name der Datentabelle



**Code-Beispiel VB.NET**

```
'Find the name of a DataTable
Dim tableName As String

tableName = VcNet1.DetectDataTableName(0)
```

**Code-Beispiel C#**

```
//Find the name of a DataTable
string tableName = vcNet1.DetectDataTableName(0);
```

**DetectFieldIndex****Methode von VcNet**

Mit dieser Eigenschaft können Sie über den Namen einer Datentabelle und den Feldnamen den Index eines Tabellendatenfeldes erfragen.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ dataTableName	System.String	Name der Datentabelle, in der sich das Feld befindet, dessen Index ermittelt werden soll
⇒ dataTableFieldName	System.String	Name des Datentabellenfeldes, dessen Index ermittelt werden soll
<b>Rückgabewert</b>	System.Int32	Zurückgegebener Index des Datentabellenfeldes

**Code-Beispiel VB.NET**

```
'Find the index of a DataTableField
Dim fieldIndex As Integer

fieldIndex = VcNet1.DetectFieldIndex("Maindata", "Name")
```

**Code-Beispiel C#**

```
//Find the index of a DataTableField
int fieldIndex = vcNet1.DetectFieldIndex("Maindata", "Name");
```

**DumpConfiguration****Methode von VcNet**

Mit dieser Methode können Sie die Konfiguration, bestehend aus .INI und .IFD-Datei, speichern.

Die Methode sollte nur zu Diagnosezwecken genutzt werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ FileName	System.String	Dateiname, ggf. mit Pfad.

⇒ encoding	VcEncoding  <b>Mögliche Werte:</b> .vcUnicodeEncoding 2	Art der Kodierung  Wird eine Datei in Unicode-Kodierung gespeichert, ist sie unabhängig von irgendwelchen Einstellungen. Dies Verfahren sollte, wenn möglich, bevorzugt werden. Eine in Unicode-Kodierung gespeicherte Datei erfordert jedoch in Visual Basic 6 eine spezielle Behandlung, wenn sie dort unabhängig vom VARCHAR-Steurelement eingelesen werden soll.
<b>Rückgabewert</b>	System.Boolean	Datei erfolgreich (True)/nicht erfolgreich (False) abgespeichert.

## EndLoading

Methode von VcNet

Hiermit wird das Ende des Ladevorgangs via **InsertNodeRecord** oder **InsertLinkRecord** angezeigt. Dadurch wird eine Aktualisierung der Grafik ausgelöst.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Ladevorgang beendet  {True}

### Code-Beispiel VB.NET

```
VcNet1.EndLoading ()
```

### Code-Beispiel C#

```
vcNet1.EndLoading ();
```

## ExportGraphicsToFileEx

Methode von VcNet

Mit dieser Methode können Sie ein Netz-Diagramm in einer Datei abspeichern, ohne einen **Speichern unter**-Dialog zu erzeugen. Mögliche Formate:

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- \*.GIF (Graphics Interchange Format)

- \*.JPG (Joint Photographic Experts Group)
- \*.PNG (Portable Network Graphics)
- \*.TIF (Tagged Image File Format)
- \*.VMF (Viewer Metafile)
- \*.WMF (Microsoft Windows Metafile, ggf. mit eingebauten EMF)

Nur EMF, EMF+, VMF und WMF sind Vektorformate, in denen das Diagramm auflösungsunabhängig gespeichert werden kann. Die übrigen Formate sind pixelorientiert und bieten damit nicht beliebige Auflösungen.

Das VMF-Format wird in der Zukunft nicht mehr weiterentwickelt, aus Kompatibilitätsgründen für bestehende Anwendungen aber zunächst noch weiter unterstützt.

Beim Exportieren in Bitmapgrafikformate kann durch Angabe einer 0 bei der gewünschten Pixelzahl in X- oder Y-Richtung eine verzerrungsfreie Grafik exportiert werden. Sind beide Pixelanzahlen 0, dann wird die Größe der exportierten Grafik in Pixeln von VARCHAR XNet wie folgt berechnet:

- PNG: Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter SizeX ein Wert  $\leq -50$  angegeben, so wird die absolute Zahl als DPI-Vorgabe genommen. Die DPI-Zahl wird auch in der ausgegebenen PNG-Datei abgelegt, so dass Anzeigeprogramme die richtige Anzeigegröße bei gegebenem Zoomfaktor ermitteln können.
- GIF, TIFF, BMP, JPEG: Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter SizeX ein Wert  $\leq -50$  angegeben, so wird die absolute Zahl als DPI-Vorgabe genommen. Es gibt aber zusätzlich eine interne Begrenzung auf 50 MB Größe für die im Speicher für das Exportieren benötigte, nicht komprimierte Ausgangsbitmap, so dass größere Grafiken eine kleinere Auflösung bekommen als gewünscht.

Bei den Vektorgrafikformaten kann keine Pixelanzahl vorgegeben werden, sondern es werden folgende Koordinatenräume benutzt:

- WMF: Es wird eine feste Auflösung angenommen, bei der die größere Ausdehnung die Koordinaten von 0 bis 10.000 benutzt. Die kleinere Ausdehnung benutzt entsprechend einen kleineren Maximalwert für die Koordinaten für eine verzerrungsfreie Darstellung.
- EMF/EMF+: Es wird die volle Auflösung mit Koordinaten in 1/100 mm Abstand in beiden Richtungen X und Y verwendet.

Detaillierte Erläuterungen zu den einzelnen Grafik-Formaten finden Sie im Kapitel: "Wichtige Konzepte: Grafikformate".

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ fileName	System.String	Dateiname, ggf. mit Pfad.
⇒ printOutputFormat	PrintOutputFormat	Format der abzuspeichernden Datei
	<b>Mögliche Werte:</b> .vcBMP 2 .vcEMF 9 .vcEMFPlus 12  .vcEMFWithEMFPlusIncluded 11  .vcEPS 3 .vcGIF 4 .vcJPG 5 .vcPCX 6 .vcPNG 7 .vcTIF 8 .vcVMF 0 .vcWMF 1 .vcWMFWithEMFIncluded 10	Datei wird im Format BMP geschrieben. Datei wird im Format EMF geschrieben. Datei wird als *.EMF-Datei geschrieben, beinhaltet aber nur das EMF+-Format. Datei wird als *.EMF-Datei geschrieben, beinhaltet aber zusätzlich das EMF+-Format. Wird nicht mehr unterstützt. Datei wird im Format GIF geschrieben. Datei wird im Format JPG geschrieben. Wird nicht mehr unterstützt. Datei wird im Format PNG geschrieben. Datei wird im Format TIF geschrieben. Datei wird im Format VMF geschrieben. Datei wird im Format WMF geschrieben. Datei wird als *.WMF-Datei geschrieben, beinhaltet aber zusätzlich das EMF-Format.
⇒ SizeX	System.Int16	Breite des exportierten Diagramms in Pixeln. Nur bei Pixelformaten möglich. Bei Angabe von 0 wird der Wert unter Beachtung des Seitenverhältnisses berechnet.
⇒ SizeY	System.Int16	Höhe des exportierten Diagramms in Pixeln. Nur bei Pixelformaten möglich. Bei Angabe von 0 wird der Wert unter Beachtung des Seitenverhältnisses berechnet.
<b>Rückgabewert</b>	System.Boolean	Datei erfolgreich (true) / nicht erfolgreich (false) abgespeichert.

#### Code-Beispiel VB.NET

```
VcNet1.ExportGraphicsToFile "C:\temp\export", vcVMF, 0, 0
```

#### Code-Beispiel C#

```
VcNet1.ExportGraphicsToFile (@"c:\Tmp\test.vmf", VcPrintOutputFormat.vcVMF, 0, 0);
```

## GetAValueFromARGB

Methode von VcNet

Ein Farbwert setzt sich aus vier Teilen zusammen: A (Alpha), R (Rot), G (Grün) und B (Blau). Der Alpha-Wert 0 bedeutet Volltransparenz und 255 ist ohne Transparenz. Die Farbwerte für R, G und B werden mit höherer Zahl heller, d.h. R,G,B von 0, 0, 0 ist schwarz und 255,255,255 ist weiß. Diese Methode erfragt den Alpha-Wert eines ARGB-Wertes.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ argb	System.Int32	ARGB- Wert, aus dem der Alpha-Wert ermittelt werden soll
<b>Rückgabewert</b>	SystemInt.32	Ermittelter Alpha-Wert

### Code-Beispiel VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha, red, green, blue)
alpha = VcNet1.GetAValueFromARGB(argb)
```

### Code-Beispiel C#

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha, red, green, blue);
alpha = vcNet1.GetAValueFromARGB(argb);
```

## GetBValueFromARGB

Methode von VcNet

Ein Farbwert setzt sich aus vier Teilen zusammen: A (Alpha), R (Rot), G (Grün) und B (Blau). Der Alpha-Wert 0 bedeutet Volltransparenz und 255 ist ohne Transparenz. Die Farbwerte für R, G und B werden mit höherer Zahl heller, d.h. R,G,B von 0, 0, 0 ist schwarz und 255,255,255 ist weiß. Diese Methode erfragt den Blau-Wert eines ARGB-Wertes.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ argb	System.Int32	ARGB- Wert, aus dem der Blau-Wert ermittelt werden soll
<b>Rückgabewert</b>	SystemInt.32	Ermittelter Blau-Wert

**Code-Beispiel VB.NET**

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha, red, green, blue)
blue = VcNet1.GetBValueFromARGB(argb)
```

**Code-Beispiel C#**

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha, red, green, blue);
blue = vcNet1.GetBValueFromARGB(argb);
```

**GetGValueFromARGB****Methode von VcNet**

Ein Farbwert setzt sich aus vier Teilen zusammen: A (Alpha), R (Rot), G (Grün) und B (Blau). Der Alpha-Wert 0 bedeutet Volltransparenz und 255 ist ohne Transparenz. Die Farbwerte für R, G und B werden mit höherer Zahl heller, d.h. R,G,B von 0, 0, 0 ist schwarz und 255,255,255 ist weiß. Diese Methode erfragt den Grün-Wert eines ARGB-Wertes.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ argb	System.Int32	ARGB- Wert, aus dem der Grün-Wert ermittelt werden soll
<b>Rückgabewert</b>	SystemInt.32	Ermittelter Grün-Wert

**Code-Beispiel VB.NET**

```

Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha, red, green, blue)
green = VcNet1.GetRValueFromARGB(argb)

```

**Code-Beispiel C#**

```

int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha, red, green, blue);
green = vcNet1.GetGValueFromARGB(argb);

```

**GetLinkByID****Methode von VcNet**

Mit dieser Methode können Sie auf eine einzelne Verbindung über ihre Identifikation zugreifen, die im Dialog **Datentabellen verwalten** festgelegt wurde. Wenn die Identifikation aus mehreren Feldern besteht (zusammengesetzter Primärschlüssel), muss diese mehrteilige ID folgendermaßen angegeben werden:

**ID=ID1|ID2|ID3**

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ linkID	System.Object	Identifikation der Verbindung
<b>Rückgabewert</b>	VcLink	Verbindung

**Code-Beispiel VB.NET**

```

Dim link As VcLink

link = VcNet1.GetLinkByID(" 5")

```

**Code-Beispiel C#**

```

VcLink link = vcNet1.GetLinkByID(" 5");

```

## GetLinkByNodeIDs

Methode von VcNet

Mit dieser Methode können Sie auf eine einzelne Verbindung über die Identifikation ihres Vorgänger- und ihres Nachfolgerknotens zugreifen. Wenn die Identifikation aus mehreren Feldern besteht (zusammengesetzter Primärschlüssel), muss diese mehrteilige ID folgendermaßen angegeben werden:

**ID=ID1|ID2|ID3**

	Datentyp	Beschreibung

### Code-Beispiel VB.NET

```
Dim link As VcLink
link = VcNet1.GetLinkByNodeIDs(" 2", " 3")
```

### Code-Beispiel C#

```
VcLink link = vcNet1.GetLinkByNodeIDs(" 2", " 3");
```

## GetNodeByID

Methode von VcNet

Mit dieser Methode können Sie auf einen einzelnen Knoten über seine Identifikation zugreifen, die im Dialog **Datentabellen verwalten** festgelegt wurde. Wenn die Identifikation aus mehreren Feldern besteht (zusammengesetzter Primärschlüssel), muss diese mehrteilige ID folgendermaßen angegeben werden:

**ID=ID1|ID2|ID3**

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ nodeID	System.Object	Identifikation des Knotens
<b>Rückgabewert</b>	VcNode	Knoten

### Code-Beispiel VB.NET

```
Dim node As VcNode
node = VcNet1.GetNodeByID("10")
```

### Code-Beispiel C#

```
VcNode node = vcNet1.GetNodeByID("10");
```



## GetRValueFromARGB

Methode von VcNet

Ein Farbwert setzt sich aus vier Teilen zusammen: A (Alpha), R (Rot), G (Grün) und B (Blau). Der Alpha-Wert 0 bedeutet Volltransparenz und 255 ist ohne Transparenz. Die Farbwerte für R, G und B werden mit höherer Zahl heller, d.h. R,G,B von 0, 0, 0 ist schwarz und 255,255,255 ist weiß. Diese Methode erfragt den Rot-Wert eines ARGB-Wertes.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ argb	System.Int32	ARGB- Wert, aus dem der Rot-Wert ermittelt werden soll
<b>Rückgabewert</b>	SystemInt.32	Ermittelter Rot-Wert

### Code-Beispiel VB.NET

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = alpha + 11
red = red + 11
green = green + 11
blue = blue + 11
argb = VcNet1.MakeARGB(alpha, red, green, blue)
red = VcNet1.GetRValueFromARGB(argb)
```

### Code-Beispiel C#

```
int alpha;
int red;
int green;
int blue;
long argb;
alpha = alpha + 11;
red = red + 11;
green = green + 11;
blue = blue + 11;
argb = vcNet1.MakeARGB(alpha, red, green, blue);
red = vcNet1.GetRValueFromARGB(argb);
```

## IdentifyFormatField

Methode von VcNet

Mit dieser Methode können Sie das aktuell verwendete Format des angegebenen Knotens sowie den Index des an der bezeichneten Position befindlichen Formatfeldes erfragen. Falls sich an der bezeichneten Position ein Feld befindet, wird **True** zurückgegeben, andernfalls **False**.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ x	System.Int32	X-Koordinate der Position
⇒ y	System.Int32	Y-Koordinate der Position
⇒ node	VcNode	Referenzknoten
⇐ format	VcNodeFormat	Identifiziertes Format
⇐ formatFieldIndex	System.Int16	Format-Feldindex
<b>Rückgabewert</b>	System.Boolean	Ein Formatfeld befindet sich/befindet sich nicht an der angegebenen Position

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
    Dim foundFlag As Boolean
    Dim format As VcNodeFormat
    Dim formatFieldIndex As Integer
    foundFlag = VcNet1.IdentifyFormatField(e.X, e.Y, e.Node, format,
formatFieldIndex)
    If foundFlag Then
        MsgBox("You hit the field with the index " + CStr(formatFieldIndex))
    End If
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeLeftClicking(object sender, VcNodeClickingEventArgs e)
{
    bool foundFlag;
    VcNodeFormat format = null;
    short formatFieldIndex = new short();
    foundFlag = vcNet1.IdentifyFormatField(e.X, e.Y, e.Node, ref format, ref
formatFieldIndex);
    if (foundFlag)
        MessageBox.Show("You hit the field with the index " +
formatFieldIndex.ToString());
}
```

## IdentifyObjectAt

Methode von VcNet

Mit dieser Methode können Sie das Objekt, das sich an einer bestimmten Position des Diagramms befindet, identifizieren. Der Typ des Objekts wird zurückgegeben.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers

↔ identifiedObject	System.Object	Erkanntes Objekt
↔ identifiedObjectType	VcObjectType  <b>Mögliche Werte:</b> .vcObjTypeBox 15 .vcObjTypeGroup 7 .vcObjTypeLinkCollection 3 .vcObjTypeNode 2 .vcObjTypeNone 0	Typ des erkannten Objekts  Objekttyp <b>Box</b> Objekttyp <b>Gruppe</b> Objekttyp <b>LinkCollection</b> Objekttyp <b>Knoten</b> kein Objekt
<b>Rückgabewert</b>	System.Boolean	Objekt identifiziert/Objekt nicht identifiziert

**Code-Beispiel VB.NET**

```

Private Sub VcNet1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles VcNet1.MouseMove

    Dim identifiedObject As Object = Nothing
    Dim identifiedObjectType As VcObjectType = VcObjectType.vcObjTypeNone
    Dim node As VcNode = Nothing
    Dim identifiedLayer As VcLayer = Nothing

    VcNet1.IdentifyObjectAt(e.X, e.Y, identifiedObject, identifiedObjectType)

    Select Case identifiedObjectType
        Case VcObjectType.vcObjTypeNodeInDiagram
            node = identifiedObject

            VcNet1.IdentifyLayerAt(e.X, e.Y, node, identifiedLayer)

            If identifiedLayer IsNot Nothing Then
                Labell1.Text = "X = " & e.X & "    Y = " & e.Y & vbCrLf & _
                    "Node ID = " & node.DataField(0) & vbCrLf & _
                    "Layer Name = " & identifiedLayer.Name
            End If

        Case Else
            Labell1.Text = ""
        End Select

    End Select

End Sub

```

**Code-Beispiel C#**

```
private void VcNet1_MouseMove(object sender, MouseEventArgs e)
{
    object identifiedObject = null;
    VcObjectType identifiedObjectType = VcObjectType.vcObjTypeNone;
    VcNode node = null;
    VcLayer identifiedLayer = null;

    VcNet1.IdentifyObjectAt(e.X, e.Y, ref identifiedObject, ref
identifiedObjectType);

    switch (identifiedObjectType)
    {
        case VcObjectType.vcObjTypeNodeInDiagram:
            {
                node = (VcNode)identifiedObject;

                VcNet1.IdentifyLayerAt(e.X, e.Y, node, ref identifiedLayer);

                if (identifiedLayer != null)
                    labell1.Text = "X = " + e.X + "    Y = " + e.Y +
                        "\nNode ID = " + node.get_DataField(0) +
                        "\nLayer Name = " + identifiedLayer.Name;

                break;
            }
        default:
            {
                labell1.Text = "";
                break;
            }
    }
}
```

**ImportConfiguration****Methode von VcNet**

Mit dieser Methode können Sie eine Konfigurationsdatei (\*.ini) laden, aus der alle relevanten Einstellungen übernommen werden. Das schließt eine zugehörige (ggf. andere) Datenschnittstelle (\*.ifd) ein.

Als Konfigurationsdatei können Sie eine lokale Datei mit Pfad oder eine URL angeben.

**Hinweis:** Beim Einlesen einer neuen Konfigurationsdatei gehen die bestehenden Daten verloren und müssen ggf. wieder eingelesen werden.

	Datentyp	Beschreibung

**Code-Beispiel VB.NET**

```
VcNet1.ImportConfiguration ( "c:\VARCHART\XNet\sample.ini")
'or
VcNet1.ImportConfiguration
("http://members.tripod.de/netronic_te/xnet_sample.ini)
```

**Code-Beispiel C#**

```
vcNet1.ImportConfiguration (@"c:\VARCHART\XNet\sample.ini");
// or
vcNet1.ImportConfiguration
(@"http://members.tripod.de/netronic_te/xnett_sample.ini");
```

**InsertLinkRecord****Methode von VcNet**

Mit dieser Methode wird eine neue Verbindung erzeugt. Die Daten werden als CSV-String (mit Trennzeichen Semikolon) gemäß der auf der Eigenschaftenseite **Datendefinition** festgelegten Struktur übergeben. Die Methode **EndLoading** sollte am Ende des kompletten Ladevorgangs (Verbindungen und Knoten) einmal aufgerufen werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ linkRecord	data field/string	Datensatz der Verbindung
<b>Rückgabewert</b>	VcLink	Verbindung

**Code-Beispiel VB.NET**

```
VcNet1.InsertNodeRecord "A100;Activity 1;12.09.14;17.09.14;5;Planning"
VcNet1.InsertNodeRecord "A105;Activity 5;13.09.14;18.09.14;7;Testing"
VcNet1.InsertLinkRecord "A100;A105;FS;0"
```

```
VcNet1.EndLoading
```

**Code-Beispiel C#**

```
vcNet1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning");
vcNet1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing");
vcNet1.InsertLinkRecord("A100;A105;FS;0");
```

**InsertNodeRecord****Methode von VcNet**

Mit dieser Methode wird ein neuer Knoten erzeugt. Die Daten werden als CSV-String (mit Trennzeichen Semikolon) gemäß der auf der Eigenschaftenseite **Datendefinition** festgelegten Struktur übergeben. Die Methode **EndLoading** sollte am Ende des kompletten Ladevorgangs (Verbindungen und Knoten) einmal aufgerufen werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ nodeRecord	data field/string	Datensatz des Knotens

Rückgabewert	VcNode	Knoten
--------------	--------	--------

### Code-Beispiel VB.NET

```
' data format: "Number;Name;Start date;Finish date;Group code;Group name"
VcNet1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning")
VcNet1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing")
```

### Code-Beispiel C#

```
//data format: "Number;Name;Start date;Finish date;Group code;Group name"
vcNet1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning");
vcNet1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing");
```

## Load

### Methode von VcNet

Mit dieser Methode werden die Daten aus der angegebenen Datei geladen. In der Datei müssen die Daten gemäß der auf der Eigenschaftenseite **Datendefinition** festgelegten Struktur im CSV-Format (mit Semikolon als Trennzeichen) gespeichert sein. Zuerst werden Knotendaten gelesen, nach einer Zeile mit 4 Sternchen (\*\*\*\*) werden Verbindungsdaten gelesen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fileName	System.String	Dateiname
<b>Rückgabewert</b>	System.Boolean {True}	Ohne Bedeutung

### Code-Beispiel VB.NET

```
VcNet1.Open "C:\ProjectData.net"
```

### Code-Beispiel C#

```
vcNet1.Open "C:\ProjectData.net";
```

## MakeARGB

### Methode von VcNet

Mit dieser Methode können Sie aus den vier Einzelwerten einer Farbe einen ARGB-Wert bilden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ alpha	System.Int.32	Alpha- Wert
⇒ red	System.Int.32	Rot- Wert

⇒ green	SystemInt.32	Grün- Wert
⇒ blue	SystemInt.32	Blau- Wert
<b>Rückgabewert</b>	System.Int32	Zurückgegebener ARGB- Wert

**Code-Beispiel VB.NET**

```
Dim alpha As Integer
Dim red As Integer
Dim green As Integer
Dim blue As Integer
Dim argb As Long
alpha = FF
red = A0
green = 34
blue = AB
argb = VcNet1.MakeARGB(alpha, red, green, blue)
```

**Code-Beispiel C#**

```
long argb;
int alpha = FF;
int red = A0;
int green = 34;
int blue = AB;
argb = vcNet1.MakeARGB(alpha, red, green, blue);
```

**PasteNodesFromClipboard**

Methode von VcNet

Mit dieser Methode werden die im Zwischenspeicher vorhandenen Knoten in das Diagramm eingefügt. Siehe auch **CopyNodesIntoClipboard** und **CutNodesIntoClipboard**.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	Void	

**Code-Beispiel VB.NET**

```
Dim nodeCltn As VcNodeCollection
nodeCltn = VcNet1.NodeCollection
nodecollection.SelectNodes(VcSelectionType.vcMarked)
If nodecollection.Count = 1 Then
VcNet1.PasteNodesFromClipboard(nodecollection.FirstNode,
VcPastePosition.vcPasteAsLastChild)
End If
```

**Code-Beispiel C#**

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcMarked);
if (nodeCltn.Count == 1)
vcNet1.PasteNodesFromClipboard(nodeCltn.FirstNode(),
VcPastePosition.vcPasteAsLastChild);
```

## PixelsToRaster

Methode von VcNet

Mit dieser Methode konvertieren Sie die angegebenen Fensterkoordinaten, wie sie z. B. von Ereignissen zurückgegeben werden, in Bandnummern für horizontale und vertikale Richtung. Falls die Bandnummern außerhalb des Diagramms liegen, wird der Funktionswert **False** zurückgegeben. Siehe auch **RasterToPixels**.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ x	System.Int32	X-Koordinate in Pixeln
⇒ y	System.Int32	Y-Koordinate in Pixeln
⇐ xBandNo	System.Int32	X-Koordinate in Bandnummern
⇐ yBandNo	System.Int32	Y-Koordinate in Bandnummern
<b>Rückgabewert</b>	System.Boolean	Konvertierung erfolgreich/nicht erfolgreich durchgeführt

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
    Dim lineNo As Long
    Dim columnNo As Long
    'change a data field of the node to the line number
    VcNet1.PixelsToRaster(e.X, e.Y, columnNo, lineNo)
    e.Node.DataField(19) = lineNo
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeLeftClicking(object sender, VcNodeClickingEventArgs e)
{
    int lineNo = new int();
    int columnNo = new int();
    //change a data field of the node to the line number
    vcNet1.PixelsToRaster(e.X, e.Y, ref columnNo, ref lineNo);
    e.Node.set_DataField(19, lineNo);
}
```

## PrintEx

Methode von VcNet

Mit dieser Methode können Sie das Diagramm direkt ausdrucken, ohne dass zuvor ein Dialogfeld erscheint. Der Rückgabewert soll bei nicht erfolgreichem Druck Aufschluss über die Ursache liefern. Dies kann z.B. auch ein Eintrag in einer Logdatei sein.



	Datentyp	Beschreibung																		
<b>Rückgabewert</b>	VcPrintResultStatus	<b>Mögliche Werte:</b> <table border="1"> <thead> <tr> <th>Name</th> <th>Parameterposition</th> <th>Beschreibung</th> </tr> </thead> <tbody> <tr> <td>vcPrintingSucceeded</td> <td>0</td> <td>Druck verlief erfolgreich</td> </tr> <tr> <td>vcNoPrinterInstalled</td> <td>1</td> <td>Es wurde weder der über VcPrinter.PrinterName angegebene noch ein im Windows-Betriebssystem als Standarddrucker gekennzeichnete Drucker gefunden.</td> </tr> <tr> <td>vcPrintingAbortedByUser</td> <td>2</td> <td>Der Druck wurde durch den Anwender abgebrochen.</td> </tr> <tr> <td>vcPrintingAbortedByDriver</td> <td>3</td> <td>Der Druck wurde durch den Windows-Druckertreiber abgebrochen.</td> </tr> <tr> <td>vcUnprintablePageLayout</td> <td>4</td> <td>Es konnte nicht gedruckt werden, weil das Seitenlayout zusammen mit den Druckereigenschaften wie Papiergröße und Ränder zu einem nicht druckbaren Layout führten.</td> </tr> </tbody> </table>	Name	Parameterposition	Beschreibung	vcPrintingSucceeded	0	Druck verlief erfolgreich	vcNoPrinterInstalled	1	Es wurde weder der über VcPrinter.PrinterName angegebene noch ein im Windows-Betriebssystem als Standarddrucker gekennzeichnete Drucker gefunden.	vcPrintingAbortedByUser	2	Der Druck wurde durch den Anwender abgebrochen.	vcPrintingAbortedByDriver	3	Der Druck wurde durch den Windows-Druckertreiber abgebrochen.	vcUnprintablePageLayout	4	Es konnte nicht gedruckt werden, weil das Seitenlayout zusammen mit den Druckereigenschaften wie Papiergröße und Ränder zu einem nicht druckbaren Layout führten.
Name	Parameterposition	Beschreibung																		
vcPrintingSucceeded	0	Druck verlief erfolgreich																		
vcNoPrinterInstalled	1	Es wurde weder der über VcPrinter.PrinterName angegebene noch ein im Windows-Betriebssystem als Standarddrucker gekennzeichnete Drucker gefunden.																		
vcPrintingAbortedByUser	2	Der Druck wurde durch den Anwender abgebrochen.																		
vcPrintingAbortedByDriver	3	Der Druck wurde durch den Windows-Druckertreiber abgebrochen.																		
vcUnprintablePageLayout	4	Es konnte nicht gedruckt werden, weil das Seitenlayout zusammen mit den Druckereigenschaften wie Papiergröße und Ränder zu einem nicht druckbaren Layout führten.																		

### Code-Beispiel C#

```
VcPrintResultStatus status = VcNet1.PrintDirectEx();
if (status != VcPrintResultStatus.vcPrintingSucceeded)
    System.Diagnostics.Trace.WriteLine("Printing failed: "+status.ToString);
```

## PrintToFile

### Methode von VcNet

Mit dieser Methode können Sie das Diagramm direkt in eine Datei drucken. Ob dies gelingt, hängt, da viele PDF-Druckertreiber keine Dateinamen akzeptieren, vom Druckertreiber ab.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fileName	System.String	Name der Datei
<b>Rückgabewert</b>	Void	

## Reset

Methode von VcNet

Mit dieser Methode wird je nach eingestelltem Wert von resetAction der komplette Inhalt sämtlicher Datentabellen gelöscht bzw. der zur Entwurfszeit auf den Eigenschaftenseiten eingestellte Zustand wiederhergestellt

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ resetAction	VcResetAction  <b>Mögliche Werte:</b> .vcEmptyAllDataTables 4  .vcReloadConfiguration 2	Objekte, die gelöscht oder neu initialisiert werden  Der komplette Inhalt sämtlicher Datentabellen wird gelöscht, die Datentabellen selbst bleiben bestehen. Komplette Neuinitialisierung mit der INI-Datei. Alle Einstellungen und erzeugten Objekte verfallen.
<b>Rückgabewert</b>	System.Boolean	Objekte im Diagramm erfolgreich gelöscht  {True}

### Code-Beispiel VB.NET

```
VcNet1.Reset(VcResetAction.vcReloadConfiguration)
```

### Code-Beispiel C#

```
vcNet1.Reset(VcResetAction.vcReloadConfiguration);
```

## SaveAsEx

Methode von VcNet

Mit dieser Methode werden die aktuellen Daten in die angegebene Datei im CSV-Format gespeichert. Dabei wird die auf der Eigenschaftenseite **Objekte** unter **Datentabellen** festgelegte Struktur verwendet. Datentabellen, die keine Datensätze enthalten, werden nicht gespeichert. Ist kein Name angegeben, wird die zuletzt bei **Open** angegebene Datei überschrieben (entspricht der üblichen **Save**-Funktion).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fileName	System.String	Dateiname
⇒ encoding	VcEncoding  <b>Mögliche Werte:</b>	Art der Kodierung

	.vcUnicodeEncoding 2	Wird eine Datei in Unicode-Kodierung gespeichert, ist sie unabhängig von irgendwelchen Einstellungen. Dies Verfahren sollte, wenn möglich, bevorzugt werden. Eine in Unicode-Kodierung gespeicherte Datei erfordert jedoch in Visual Basic 6 eine spezielle Behandlung, wenn sie dort unabhängig vom VARCHAR-Steurelement eingelesen werden soll.
<b>Rückgabewert</b>	System.Boolean	Speicherung erfolgreich/nicht erfolgreich erfolgt

**Code-Beispiel VB.NET**

```
VcNet1.SaveAs "C:\ProjectData.net"
```

**Code-Beispiel C#**

```
vcNet1.SaveAs "C:\ProjectData.net";
```

**ScheduleProject****Methode von VcNet**

Mit dieser Methode können Sie eine Vorwärts- und Rückwärtsberechnung des aktuellen Projekts durchführen. Bei Übergabe des Starttermins wird zunächst eine Vorwärtsberechnung, dann eine Rückwärtsberechnung durchgeführt. Bei Übergabe des Endtermins wird zunächst eine Rückwärtsberechnung, dann eine Vorwärtsberechnung durchgeführt. Es können auch Anfangs- und Enddatum übergeben werden, die Vorgänge erhalten dann entsprechende Pufferzeiten. Das Fehlen beider Daten führt zu einer Fehlermeldung. Falls ein Zyklus der Knoten und Verbindungen festgestellt wird, werden diese automatisch markiert.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ startDate	Date/Time	Startdatum oder 0
⇒ endDate	Date/Time	Enddatum oder 0
<b>Rückgabewert</b>	System.Boolean	Berechnung erfolgreich/nicht erfolgreich durchgeführt

**Code-Beispiel VB.NET**

```
VcNet1.ScheduleProject "21.06.04", 0
```

**Code-Beispiel C#**

```
vcNet1.ScheduleProject "21.06.04", 0;
```

## ScrollToNode

Methode von VcNet

Mit dieser Methode können Sie zu der Zeile eines bestimmten Knotens scrollen, so dass dieser sichtbar wird.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ node	VcNode	Knoten, zu dessen Zeile gescrollt werden soll
<b>Rückgabewert</b>	System.Boolean	Scrollen erfolgreich/nicht erfolgreich durchgeführt

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
    'scroll the diagram so that the node is completely on screen
    VcNet1.ScrollToNode(e.Node)
End Sub
```

### Code-Beispiel C#

```
object[] objDataRecord = new object[5];

vcNet1.ExtendedDataTablesEnabled = true;
vcNet1.MinimumRowHeight = 1000;

vcNet1.TimeScaleEnd = new DateTime(2010, 8, 1);
vcNet1.TimeScaleStart = new DateTime(2010, 6, 1);

objDataRecord[2] = new DateTime(2010, 6, 3);
objDataRecord[3] = new DateTime(2010, 6, 10);
objDataRecord[4] = 5;

VcDataRecordCollection dataRecordCol =
vcNet1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection;
for (int i = 1; i < 100; i++)
{
    objDataRecord[0] = i;
    objDataRecord[1] = "Node " + i.ToString();

    dataRecordCol.Add(objDataRecord);
}
vcNet1.EndLoading();
vcNet1.ScrollToNode(vcNet1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned);
```

## SetImageResource

Methode von VcNet

Diese Methode ordnet zur Laufzeit einem angegebenen Namen ein in der Anwendung vorhandenes Image-Objekt zu. Dies ist eine Alternative zur bisherigen Funktionalität, bei der in den XNet-Eigenschaftenseiten angegebene Image-Namen immer dazu führen, dass ein Image-Objekt aus der angesprochenen Datei gelesen wird. Die Methode sollte zu Beginn einer

Anwendung, z.B. in der Methode **Form\_Load**, ausgeführt werden. Die Image-Namen sind frei wählbar. Zur Unterscheidung von Dateinamen können hier auch Zeichen verwendet werden, die sonst für Dateinamen verboten sind, z.B. das Sternchen (\*) . Alle Image-Objekte sind erlaubt: Bitmaps (Formate BMP, JPG, GIF, PNG, TIFF) und Metafiles (Formate WMF, EMF). Wenn der Parameter **image** auf null gesetzt ist, werden vorherige Zuordnungen aufgehoben.

Beispiel: Man fügt einer Anwendung eine Image- oder Datei-Ressource hinzu (im Visual Studio unter den Projekteigenschaften: Ressourcen/Ressource hinzufügen/Vorhandene Datei hinzufügen) und setzt dann in Form\_Load beispielsweise folgenden Code ein:

Für Bitmap-Ressourcen:

```
vcNet1.SetImageResource("*PlusImage",
<namespace>.Properties.Resources.plusImage);
```

für Metafile-Ressourcen:

```
vcNet1.SetImageResource("*MinusImage", new Metafile(new
MemoryStream(<namespace>.Properties.Resources.minusImage)));
```

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ imageName	System.String	Name, der dem Image-Objekt zugeordnet ist
<b>Rückgabewert</b>	System.Drawing.Image	Image-Objekt

## ShowAboutDialog

Methode von VcNet

Mit dieser Methode können Sie die **About**-Box aufrufen. Sie enthält eine Übersicht über die jeweils verwendeten Programm- und Bibliotheksdateien mit absolutem Pfad und Versionsnummer. Dies dient der vereinfachten Hotline-Unterstützung. Die Übersicht kann mit der Maus selektiert und mit Strg+C herauskopiert werden, um sie z. B. mit Strg+V in eine Mail einzufügen.

	Datentyp	Beschreibung
Rückgabewert	Void	

**Code-Beispiel VB.NET**

```
VcNet1.ShowAboutDialog()
```

**Code-Beispiel C#**

```
vcNet1.ShowAboutDialog();
```

## ShowExportGraphicsDialog

Methode von VcNet

Mit dieser Methode können Sie das Export-Dialogfeld **Speichern unter** aufrufen, um die Darstellung abzuspeichern. Verfügbare Formate sind:

- \*.BMP (Microsoft Windows Bitmap)
- \*.EMF (Enhanced Metafile oder Enhanced Metafile Plus)
- \*.GIF (Graphics Interchange Format)
- \*.JPG (Joint Photographic Experts Group)
- \*.PNG (Portable Network Graphics)
- \*.TIF (Tagged Image File Format)
- \*.VMF (Viewer Metafile)
- \*.WMF (Microsoft Windows Metafile, ggf. mit eingebautem EMF)

Nur EMF, EMF+, VMF und WMF sind Vektorformate, in denen das Diagramm auflösungsunabhängig gespeichert werden kann. Die übrigen Formate sind pixelorientiert und bieten damit nicht beliebige Auflösungen.

Das VMF-Format wird in der Zukunft nicht mehr weiterentwickelt, aus Kompatibilitätsgründen für bestehende Anwendungen aber zunächst noch weiter unterstützt.

Detaillierte Erläuterungen zu den einzelnen Grafik-Formaten lesen Sie bitte im Kapitel: **Wichtige Konzepte: Grafikformate**.

Beim Exportieren wird die Größe des exportierten Diagramms in Pixeln wie folgt berechnet:

- **PNG:** Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter SizeX ein Wert  $\leq -50$  angegeben, so wird die absolute Zahl als DPI-Vorgabe genommen.
- **GIF, TIFF, BMP, JPEG:** Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter SizeX ein Wert  $\leq -50$  angegeben, so wird die absolute Zahl als DPI-Vorgabe genommen. Es gibt aber zusätzlich eine interne Begrenzung auf 50 MB Größe für die im Speicher für das Exportieren benötigte, nicht komprimierte Ausgangsbitmap, so dass größere Grafiken eine kleinere Auflösung bekommen als gewünscht.
- **WMF:** Es wird eine feste Auflösung angenommen, bei der die größere Ausdehnung die Koordinaten von 0 bis 10.000 benutzt. Die kleinere Ausdehnung benutzt entsprechend einen kleineren Maximalwert für die Koordinaten für eine verzerrungsfreie Darstellung.
- **EMF/EMF+:** Es wird die volle Auflösung mit Koordinaten in 1/100 mm Abstand verwendet.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Grafik erfolgreich (true) /nicht erfolgreich (false) exportiert

#### Code-Beispiel VB.NET

```
VcNet1.ShowExportGraphicsDialog()
```

#### Code-Beispiel C#

```
vcNet1.ShowExportGraphicsDialog();
```

## ShowLinkEditDialog

Methode von VcNet

Mit dieser Methode wird der Dialog **Verbindung bearbeiten** für die angegebene Verbindung aufgerufen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ link	VcLink	Verbindung, deren Daten editiert werden sollen

<b>Rückgabewert</b>	System.Boolean	Verbindungsdaten wurden editiert/Editierung abgebrochen
---------------------	----------------	---

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinksLeftClicking(ByVal sender As System.Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksLeftClicking
    If e.LinkCollection.Count = 1 Then
        VcNet1.ShowLinkEditDialog(e.LinkCollection.FirstLink)
    End If
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinksLeftClicking(object sender, VcLinksClickingEventArgs
e)
{
    if (e.LinkCollection.Count == 1)
        vcNet1.ShowLinkEditDialog(e.LinkCollection.FirstLink());
}
```

**ShowNodeEditDialog****Methode von VcNet**

Mit dieser Methode wird der Dialog **Vorgänge bearbeiten** für den angegebenen Knoten aufgerufen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ node	VcNode	Knoten, dessen Daten editiert werden sollen
<b>Rückgabewert</b>	System.Boolean	Knotendaten wurden editiert/Editierung abgebrochen

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
    VcNet1.ShowNodeEditDialog(node)
End Sub
```

**Code-Beispiel C#**

```
private void vcvcNet1_VcNodeLeftClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
{
    vcNet1.ShowNodeEditDialog(e.Node);
}
```

**ShowPageSetupDialog****Methode von VcNet**

Mit dieser Methode rufen Sie den Dialog **Seite einrichten** auf.



	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Ohne Bedeutung {True}

**Code-Beispiel VB.NET**

```
VcNet1.ShowPageLayoutDialog()
```

**Code-Beispiel C#**

```
vcNet1.ShowPageLayoutDialog();
```

## ShowPrintDialog

Methode von VcNet

Mit dieser Methode wird der **Drucken**-Dialog aufgerufen. Dabei werden die im Dialog **Seite einrichten** gesetzten Parameter verwendet.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Ohne Bedeutung {True}

**Code-Beispiel VB.NET**

```
VcNet1.ShowPrintDialog()
```

**Code-Beispiel C#**

```
vcNet1.ShowPrintDialog();
```

## ShowPrinterSetupDialog

Methode von VcNet

Mit dieser Methode rufen Sie den Windows-Dialog **Drucker einrichten** auf.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Ohne Bedeutung {True}

**Code-Beispiel VB.NET**

```
VcNet1.ShowPrinterSetupDialog()
```

**Code-Beispiel C#**

```
vcNet1.ShowPrinterSetupDialog();
```

## ShowPrintPreviewDialog

Methode von VcNet

Mit dieser Methode wird die Druckvorschau aufgerufen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Ohne Bedeutung {True}

### Code-Beispiel VB.NET

```
VcNet1.ShowPrintPreviewDialog()
```

### Code-Beispiel C#

```
vcNet1.ShowPrintPreviewDialog();
```

## SuspendUpdate

Methode von VcNet

Bei größeren Datenmengen kann es unter Umständen zu lange dauern, wenn man bei einer großen Anzahl von Knoten dieselbe Aktion durchführt. Dies kann man mit Hilfe der Methode **SuspendUpdate** beschleunigen. Klammern Sie den Code für die wiederholte Aktion wie im Code-Beispiel durch **SuspendUpdate (True)** und **SuspendUpdate (False)** ein. Dann wird das Update nicht für jeden Knoten einzeln, sondern für alle gemeinsam durchgeführt, wodurch die Performance erhöht wird.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	SuspendUpdate(True): Beginn der SuspendUpdate Methode/ SuspendUpdate(False): Ende der SuspendUpdate Methode

### Code-Beispiel VB.NET

```
VcNet1.SuspendUpdate (True)

    If updateFlag Then
        For Each node In nodeCltn
            If node.DataField(2) < "07.09.14" Then
                node.DataField(13) = "X"
                node.Update
                counter = counter + 1
            End If
        Next node
    Else
        For Each node In nodeCltn
            If node.DataField(2) < "07.09.14" Then
                node.DataField(13) = ""
                node.Update
                counter = counter + 1
            End If
        Next node
    End If

VcNet1.SuspendUpdate (False)
```

### Code-Beispiel C#

```
bool updateFlag = true;
VcNodeCollection nodeCltn = vcvcNet1.NodeCollection;
int counter = 0;

vcvcNet1.SuspendUpdate(true);
if (updateFlag == true)
{
    foreach (VcNode node in nodeCltn)
    {
        if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.14")) < 0)
        {
            node.set_DataField(13, "X");
            node.Update();
            counter = counter + 1;
        }
    }
}
else
{
    foreach(VcNode node in nodeCltn)
    {
        if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.14")) < 0)
        {
            node.set_DataField(13, "");
            node.Update();
            counter = counter + 1;
        }
    }
}
vcNet1.SuspendUpdate(false);
```

## UpdateLinkRecord

Methode von VcNet

Mit dieser Methode können die Daten eines bestehenden Datensatzes einer Verbindung verändert werden. Der Datensatz wird durch die auf der Eigenschaftenseite **Datendefinition** festgelegte ID identifiziert. Diese Methode wird verwendet, wenn externe Änderungen in der Grafik auf dem Bildschirm nachvollzogen werden sollen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ linkRecord	System.Object	Datensatz der Verbindung
<b>Rückgabewert</b>	VcLink	Aktualisierte Verbindung

### Code-Beispiel VB.NET

```
VcNet1.UpdateLinkRecord ("A100;A105;FS;0")
```

### Code-Beispiel C#

```
vcNet1.UpdateLinkRecord( "A100;A105;FS;0");
```

## UpdateNodeRecord

Methode von VcNet

Mit dieser Methode können die Daten eines bestehenden Datensatzes eines Knotens verändert werden. Der Datensatz wird durch die auf der Eigenschaftenseite **Datendefinition** festgelegte ID identifiziert. Diese Methode wird verwendet, wenn externe Änderungen in der Grafik auf dem Bildschirm nachvollzogen werden sollen.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ nodeRecord	System.Object	Datensatz des Knotens
<b>Rückgabewert</b>	VcNode	Aktualisierter Knoten

### Code-Beispiel VB.NET

```
VcNet1.UpdateNodeRecord ("A100;Activity 1;12.09.14;18.09.14;6;Planning")
```

### Code-Beispiel C#

```
vcNet1.UpdateNodeRecord ("A100;Activity 1;12.09.14;18.09.14;6;Planning");
```

## Zoom

Methode von VcNet

Mit dieser Methode kann die Bildschirmdarstellung um den angegebenen Prozent-Faktor vergrößert (`zoomFactor > 100`) oder verkleinert (`zoomFactor < 100`) werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ <code>zoomFactor</code>	<code>System.Int16 {1...1000}</code>	Zoomfaktor {11...999}, anderer Werte bleiben unberücksichtigt
<b>Rückgabewert</b>	<code>System.Boolean</code>	{True}

### Code-Beispiel VB.NET

```
VcNet1.Zoom 120
```

### Code-Beispiel C#

```
vcNet1.Zoom 120;
```

## ZoomOnMarkedNodes

Methode von VcNet

Mit dieser Methode können Sie auf die markierten Knoten zoomen.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	Void	

### Code-Beispiel VB.NET

```
VcNet1.ZoomOnMarkedNodes
```

### Code-Beispiel C#

```
vcNet1.ZoomOnMarkedNodes;
```

## Ereignisse

### VcBoxLeftClicking

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Box klickt. Das getroffene VcBox-Objekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter zurückgegeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ box	VcBox	Getroffene Box
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

#### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcBoxLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.Xnet.VcBoxClickingEventArgs) Handles VcNet1.VcBoxLeftClicking
    TextBox1.Text = e.Box.FieldText(1)
End Sub
```

#### Code-Beispiel C#

```
private void vcNet1_VcBoxLeftClicking(object sender,
NETRONIC.XNetVcBoxClickingEventArgs e)
{
    textBox1.Text = e.Box.get_FieldText(1);
}
```

### VcBoxLeftDoubleClicking

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Box doppelklickt. Das getroffene VcBox-Objekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter zurückgegeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ box	VcBox	Getroffene Box
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcBoxLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcBoxClickingEventArgs) Handles VcNet1.VcBoxLeftDoubleClicking
    e.Box.FieldText(0) = TextBox1.Text
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcBoxLeftDoubleClicking(object sender,
VcNetLib.VcBoxClickingEventArgs e)
{
    e.Box.set_FieldText(1, textBox1.Text);
}
```

## VcBoxModified

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn die Modifizierung der Box abgeschlossen ist. Das veränderte VcBox-Objekt und der Modifikationstyp werden als Parameter mitgegeben.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcBoxModifiedEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcBoxModifiedEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ box	VcBox	Veränderte Box

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcBoxModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcBoxModifiedEventArgs) Handles VcNet1.VcBoxModified
    MsgBox("The box has been modified")
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcBoxModified(object sender, VcNetLib.VcBoxModifiedEventArgs
e)
{
    MessageBox.Show("The box has been modified");
}
```

**VcBoxModifying****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn eine Box interaktiv verändert wurde. Das veränderte VcBox-Objekt und der Modifikationstyp werden als Parameter zurückgegeben.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcBoxModified**.

Durch Setzen des Rückgabestatus auf **vcRetStatFalse** kann die Änderung verhindert werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcBoxModifyingEventArgs	Ereignisspezifisches Objekt

**Eigenschaften des VcBoxModifyingEventArgs-Objektes**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ box	VcBox	Veränderte Box
⇒ modificationType	VcBoxModificationTypes	Art der Veränderung
	<b>Mögliche Werte:</b>	
	.vcBMTAnything 1	beliebige Veränderung
	.vcBMTNothing 0	keine Veränderung
	.vcBMTTextModified 4	Text der Box verändert
	.vcBMTXYOffsetModified 2	Offset verändert
⇔ returnStatus	VcReturnStatus	Rückgabestatus



**Mögliche Werte:**

.vcRetStatDefault 2  
 .vcRetStatFalse 0  
 .vcRetStatNoPopup 4  
  
 .vcRetStatOK 1

Das Default-Verhalten wird nicht verändert.  
 Das Default-Verhalten wird nicht durchgeführt.  
 Das Erscheinen des Kontextmenüs wird unterdrückt.  
 Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcBoxModifying(ByVal box As NETRONIC.XNet.VcBox, _
    ByVal modificationType As _
    VcNetLib.VcBoxModificationTypes, _
    returnStatus As Variant)

    Select Case modificationType
        Case vbBMTAnything: MsgBox "Box modification"
        Case vbBMTXYOffsetModified: MsgBox "Offset changed"
        Case vbBMTTextModified: MsgBox "Box field text changed"
    End Select

End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcBoxModifying(object sender,
NETRONIC.XNet.VcBoxModifyingEventArgs e)
{
    switch (e.ModificationType)
    {
        case VcBoxModificationTypes.vcBMTAnything:
            MessageBox.Show("Box modification");
            break;
        case VcBoxModificationTypes.vcBMTXYOffsetModified:
            MessageBox.Show("Offset changed");
            break;
        case VcBoxModificationTypes.vcBMTTextModified:
            MessageBox.Show("Box field text changed");
            break;
    }
}
```

## VcBoxRightClicking

**Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf eine Box klickt. Das getroffene Boxobjekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ box	VcBox	Getroffene Box
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers

↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcBoxRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcBoxClickingEventArgs) Handles VcNet1.VcBoxRightClicking
    PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcBoxRightClicking(object sender,
NETRONIC.XNet.VcBoxClickingEventArgs e)
{
    PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
}
```

## VcDataModified

### Ereignis von VcNet

Dieses Ereignis tritt immer dann auf, wenn Daten interaktiv im Chart verändert werden, also explizit nach den folgenden Ereignissen:

- VcBoxModified
- VcLinkCreated
- VcLinkDeleted
- VcNodeCreated
- VcNodeDeleting
- VcNodeModified

Mit diesem Ereignis ist es möglich, sich im Anwenderprogramm eine Marke zu setzen, die daran erinnert, dass beim Beenden des Programms die Daten noch gespeichert werden müssen.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
↔ (no parameter)		Kein Parameter

## VcDataRecordCreated

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn das interaktive Anlegen eines Objektes beendet ist, das einen Datensatz erzeugt. Das DataRecord-Objekt, die Form des Anlegens (hier nur **vcDataRecordCreated** und **vcDataRecordCreated-ByResourceScheduling**) und die Information, ob der angelegte Datensatz der einzige Datensatz oder der letzte einer Menge ist (derzeit immer **True**), werden als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

Wenn eine Verbindung oder ein Knoten angelegt wurde, können Sie zudem auf das analoge Verbindungs- oder Knoten-Ereignis reagieren und hier vor der Weiterverarbeitung zusätzlich grafische Daten prüfen (s. **VcNodeCreated** und **VcLinkCreated**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDataRecordCreatedEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcDataRecordCreatedEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ dataRecord	VcDataRecord	Angelegtes Datensatz-Objekt
⇒ creationType	VcCreationType	Typ des Anlegens von Datensätzen
	<b>Mögliche Werte:</b>	
	.vcDataRecordCreated 6	Datensatz wurde durch Interaktion angelegt
	.vcLinkCreated 2	Verbindung wurde durch Interaktion angelegt
	.vcNodeCreated 1	Knoten durch "Stempeln" angelegt
	.vcNodesAndLinksCloned 4	selektierte Knoten wurden durch Ziehen mit der Maus bei gleichzeitigem Drücken der Strg-Taste kopiert
	.vcNodeWithLinkCreated 3	Knoten zusammen mit Verbindung angelegt
⇒ isLast	System.Boolean	<b>True:</b> Angelegter Datensatz ist der einzige oder der letzte Datensatz einer Menge <b>False:</b> Angelegter Datensatz ist nicht der einzige oder der letzte Datensatz einer Menge

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcDataRecordCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordCreatedEventArgs) Handles VcNet1.VcDataRecordCreated
    MsgBox(e.DataRecord.AllData)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcDataRecordCreated(object sender,
NETRONIC.XNet.VcDataRecordCreatedEventArgs e)
{
    MessageBox.Show(e.DataRecord.AllData.ToString());
}
```

## VcDataRecordCreating

**Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender interaktiv ein Objekt erzeugt hat, das einen Datensatz generiert. Das neu erzeugte Datensatz-Objekt wird als Parameter zurückgegeben, so dass eine Validierung und ggf. ein Datenbankeintrag vorgenommen werden kann.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcDataRecordCreated**.

Durch Setzen des Rückgabestatus kann die Anlage verhindert werden.

Wenn eine Verbindung oder ein Knoten angelegt wurde, können Sie zudem auf das analoge Verbindungs- oder Knoten-Ereignis reagieren und hier vor der Weiterverarbeitung zusätzlich grafische Daten prüfen (s. **VcNodeCreating** und **VcLinkCreating**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDataRecordCreatingEventArgs	Ereignisspezifisches Objekt

**Eigenschaften des VcDataRecordCreatingEventArgs-Objektes**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ dataRecord	VcDataRecord	Angelegtes Datensatz-Objekt
⇔ returnStatus	VcReturnStatus	Rückgabestatus

**Mögliche Werte:**  
 .vcRetStatFalse 0  
 .vcRetStatOK 1

Der Datensatz wird nicht angelegt.  
 Der Datensatz wird angelegt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcDataRecordCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordCreatedEventArgs) Handles VcNet1.VcDataRecordCreated
    MsgBox(e.DataRecord.AllData)
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcDataRecordCreated(object sender,
NETRONIC.XNet.VcDataRecordCreatedEventArgs e)
{
    MessageBox.Show(e.DataRecord.AllData.ToString());
}
```

## VcDataRecordDeleted

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn das Löschen eines Objektes, das auf einem Datensatz-Objekt basiert, beendet ist. Der Datensatz und die Information, ob der betroffene Datensatz der einzige oder der letzte Datensatz einer Menge ist, werden als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

Wenn eine Verbindung oder ein Knoten gelöscht wurde, können Sie zudem auf das analoge Verbindungs- oder Knoten-Ereignis reagieren und hier vor der Weiterverarbeitung zusätzlich grafische Daten prüfen (s. **VcNodeDeleted** und **VcLinkDeleted**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDataRecordDeletedEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcDataRecordDeletedEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ dataRecord	VcDataRecord	Gelöschter Datensatz

⇒ isLast	System.Boolean	<p><b>True:</b> Gelöschter Datensatz ist der einzige oder der letzte Datensatz einer Menge</p> <p><b>False:</b> Gelöschter Datensatz ist nicht der einzige oder der letzte Datensatz einer Menge</p>
----------	----------------	--

## VcDataRecordDeleting

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs ein Objekt löscht, das auf einem Datensatz-Objekt basiert. Der betroffene Datensatz wird als Parameter zurückgegeben, so dass Sie z. B. noch eine Überprüfung vornehmen und bei negativem Ergebnis dieser Prüfung die Löschung ggf. durch Setzen des Rückgabestatus verhindern können.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDataRecordDeletingEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcDataRecordDeletingEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ dataRecord	VcDataRecord	Gelöschtes Datensatz-Objekt
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<p><b>Mögliche Werte:</b>                      .vcRetStatFalse 0                      .vcRetStatOK 1</p>	<p>Der Datensatz wird nicht gelöscht.                      Der Datensatz wird gelöscht.</p>

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcDataRecordDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordDeletingEventArgs) Handles VcNet1.VcDataRecordDeleting
    'deny deletion of data record with a certain value
    If e.DataRecord.DataField(0) = "1" Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcDataRecordDeleting(object sender,
NETRONIC.XNet.VcDataRecordDeletingEventArgs e)
{
    // deny deletion of data record with a certain value
    if (e.DataRecord.get_DataField(0).Equals("1"))
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

**VcDataRecordModified****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn die Modifizierung des Datensatzes abgeschlossen ist.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDataRecordModifiedEventArgs	Ereignisspezifisches Objekt

**Eigenschaften des VcDataRecordModifiedEventArgs-Objektes**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ dataRecord	VcDataRecord	Veränderter Datensatz

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcDataRecordModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDataRecordModifiedEventArgs) Handles VcNet1.VcDataRecordModified
    MsgBox("The data record has been modified")
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcDataRecordModified(object sender,
NETRONIC.XNet.VcDataRecordModifiedEventArgs e)
{
    MessageBox.Show("The data record has been modified");
}
```

**VcDataRecordModifying****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn ein Objekt, dem ein Datensatz zu Grunde liegt, interaktiv verändert wurde. Das veränderte VcDataRecord-Objekt und der Modifikationstyp werden als Parameter zurückgegeben.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcDataRecordModified**.

Durch Setzen des Rückgabestatus kann die Änderung verhindert werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDataRecordModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordModifyingEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ dataRecord	VcDataRecord	Veränderter Datensatz
⇒ modificationType	VcModificationTypes	Art der Veränderung
	<b>Mögliche Werte:</b>	
	.vcAnything 1	Änderungstyp nicht näher bestimmt
	.vcChangedGroup 16	Zuordnung des Knotens zu einer Gruppe wurde verändert (nur für Knoten).
	.vcMoved 8	Objekt wurde verschoben.
	.vcNothing 0	Keine Änderung
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatFalse 0	Die Veränderung wird rückgängig gemacht.
	.vcRetStatOK 1	Die Veränderung wird durchgeführt.

## VcDataRecordNotFound

**Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn ein abhängiger Datensatz nicht gefunden wurde. Der Index des Feldes im aktuellen Datensatz, in dem der Schlüsselwert des abhängigen Datensatzes steht, wird zurückgegeben und bietet so Informationen über den nicht gefundenen Datensatz.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	
⇒ e	VcDataRecordNotFoundEventArgs	



	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
↔ index	System.Int32	Index des Feldes, das den Schlüssel des abhängigen Datensatzes enthält

## VcDiagramLeftClicking

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste im Diagrammbereich klickt und dabei kein Objekt trifft. Die Position (x,y-Koordinaten des Mauszeigers) wird als Parameter zurückgegeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDiagramClickingEventArgs) Handles VcNet1.VcDiagramLeftClicking
    MsgBox("x: " + e.X.ToString() + " y: " + e.Y.ToString())
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcDiagramLeftClicking(object sender,
NETRONIC.XNet.VcDiagramClickingEventArgs e)
{
    MessageBox.Show("x: " + e.X.ToString() + " y: " + e.Y.ToString());
}
```

## VcDiagramLeftDoubleClicking

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste im Diagrammbereich doppelklickt und dabei kein Objekt trifft. Die Position (x,y-Koordinaten des Mauszeigers) wird als Parameter zurückgegeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcDiagramLeftDoubleClicking(ByVal sender As Object, ByVal e
NETRONIC.XNet.VcDiagramClickingEventArgs) Handles
VcNet1.VcDiagramLeftDoubleClicking
    VcNet1.Zoom(90)
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcDiagramLeftDoubleClicking(object sender,
NETRONIC.XNet.VcDiagramClickingEventArgs e)
{
    vcNet1.Zoom(90);
}
```

## VcDiagramRightClicking

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste im Diagrammbereich klickt und dabei kein Objekt trifft. Die Position (x,y-Koordinaten des Mauszeigers) wird als Parameter übergeben. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ x	System.Int32	X-Koordinate

⇒ y	System.Int32	Y-Koordinate
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcDiagramRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcDiagramClickingEventArgs) Handles VcNet1.VcDiagramRightClicking
    PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
    e.ReturnStatus = VcNetLib.VcReturnStatus.vcRetStatNoPopup
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcDiagramRightClicking(object sender,
NETRONIC.XNet.VcDiagramClickingEventArgs e)
{
    PopupMenu.Show(vcNet1, new Point(e.X, e.Y));
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

## VcDragCompleting

### Ereignis von VcNet

Dieses Ereignis wird zum Abschluss eines Drag-Vorgangs bei der Quellkomponente ausgelöst und gibt den Drop-Effekt bekannt.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDragCompletingEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcDragCompletingEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ DropEffect	System.Windows.Forms.DragDropEffects	Auswirkungen einer Drag & Drop-Operation

## VcDragStarting

Ereignis von VcNet

Mit diesem Ereignis ist es möglich, die erlaubten DropEffects beim Start eines Drag-Vorgangs zu bestimmen und damit ggf. einzuschränken. Gleichzeitig muss die Eigenschaft **LeavingControlWhileDraggingAllowed** auf **True** gesetzt sein. Die Eigenschaft ist mit dem kombinierten Wert **DragDropEffects.Copy Or DragDropEffects.Move** vorbesetzt. Wenn z.B. Knoten beim Herausziehen aus dem Steuerelement in jedem Fall kopiert und niemals verschoben werden sollen, dann setzt man die Eigenschaft auf **DragDropEffects.Copy**.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcDragStartingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDragStartingEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ allowedEffects	System.Windows.Forms.AllowedEffects	Erlaubte DropEffects

## VcErrorOccurring

Ereignis von VcNet

Dieses Ereignis tritt nur dann auf, wenn ein unvorhergesehener Fehler im Code des VARCHART XNet entdeckt wird. NETRONIC ist bemüht, jeden dieser Fehler zu beseitigen. Um sie auf einfache Art beim Kunden, z. B. in einer Datei, protokollieren zu können, werden sie nun über dieses Ereignis nach außen bekanntgegeben. Das Parameterprofil ist durch den ActiveX-Standard vorgegeben. Dadurch sind die übergebenen Parameter teilweise konstant. Die Nummer sollte im Ereignis immer gegengeprüft werden, um bei zukünftigen Erweiterungen nicht alle Fehlerarten pauschal abzublocken.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ description	System.String	Fehlerbeschreibungstext
⇒ scode	System.Int32	&h800a402f (konstant)

⇒ source	System.String	Name des Controls (konstant)
⇒ helpFile	System.String	Hilfedatei: "" (konstant)
⇒ helpContext	System.Int32	Hilfekontext: 0 (konstant)
⇐ cancelDisplay	System.Boolean	Beim Wert True wird kein normaler Fehler mit der Nummer 71 mehr ausgegeben, der über das 'VcErrorOccurring'-Konstrukt abfangbar wäre.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcErrorOccuring(ByVal sender As System.Object, _
                                   ByVal e As
NETRONIC.XNet.VcErrorOccuringEventArgs) _
    Handles VcNet1.VcErrorOccuring
    MsgBox(e.Code + " - " + e.Text)
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcErrorOccuring(object sender, VcErrorOccuringEventArgs e)
{
    MessageBox.Show(e.Code + " - " + e.Text);
}
```

## VcFieldSelecting

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn ein Feld einer Box selektiert wird. Die Selektion kann durch Setzen des Rückgabestatus verhindert werden.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcFieldSelectingEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcFieldSelectingEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ editObject	VcObject	Editiertes Objekt
⇒ editObjectType	VcObjectType	Objektyp
	<b>Mögliche Werte:</b>	
	.vcObjTypeBox 15	Objektyp <b>Box</b>
	.vcObjTypeGroup 7	Objektyp <b>Gruppe</b>
	.vcObjTypeLinkCollection 3	Objektyp <b>LinkCollection</b>
	.vcObjTypeNode 2	Objektyp <b>Knoten</b>
	.vcObjTypeNone 0	kein Objekt

⇒ fieldIndex	System.Int32	Feldindex
⇒ objRectComplete	VcRect	Komplettes Rechteck des getroffenen Objekts
⇒ objRectVisible	VcRect	sichtbares Rechteck des getroffenen Objekts
⇒ fldRectComplete	VcRect	Komplettes Rechteck des getroffenen Feldes
⇒ fldRectVisible	VcRect	sichtbares Rechteck des getroffenen Feldes
returnStatus	VcReturnStatus	
	<b>Mögliche Werte:</b> .vcRetStatFalse 0 .vcRetStatOK 1	Das Feld wird nicht ausgewählt. Das Feld wird ausgewählt.

## VcGiveFeedbackOnNodeCreating

Ereignis von VcNet

Dieses Ereignis wird aufgerufen, wenn der Knotenerzeugemodus eingeschaltet ist. X und Y bezeichnen die Position des Mauszeigers relativ zum Koordinatenursprung des Steuerelements. Wird der Defaultwert **True** von **creationAllowed** nicht geändert, können an dieser Mausposition Knoten erzeugt werden. Wenn **creationAllowed** auf **False** gesetzt wird, ist das Erzeugen von Knoten nicht möglich. Dies kann genutzt werden, um eine Knotenerzeugung in bestimmten Teilen des Diagramms (z.B. in Bereichen ohne Gruppen wie im Codebeispiel unten) von vornherein zu unterbinden.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ X	System.Int32	X-Koordinate des Mauszeigers
⇒ Y	System.Int32	Y-Koordinate des Mauszeigers
⇔ CreationAllowed	System.Boolean	Rückgabestatus

### Code-Beispiel C#

```
private void vcNet1_VcGiveFeedbackForNodeCreating(object sender,
VcGiveFeedbackForNodeCreatingEventArgs e)
{
    object obj = null;
    VcObjectType objType = VcObjectType.vcObjTypeNone;
    vcNet1.IdentifyObjectAt(e.X, e.Y, ref obj, ref objType);
    if (objType == VcObjectType.vcObjTypeNone)
        e.CreationAllowed = false;
}
```

## VcGroupCreated

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn eine Gruppe erzeugt wird, also wenn der erste Knoten mit einem neuen Gruppencode im ActiveX-Steuerelement erzeugt wird. Das neu erzeugte Objekt wird als Parameter übergeben, so dass eine Validierung und ggf. ein Datenbank-Eintrag vorgenommen werden kann.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Angelegte Gruppe
↔ returnStatus	VcReturnStatus	Rückgabestatus: z.Z. ohne Funktion
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

## VcGroupDeleting

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender interaktiv den letzten Knoten einer Gruppe löscht oder verschiebt, so dass diese leer wird und daher gelöscht wird. Dabei wird das Gruppenobjekt als Parameter übergeben. Derzeit kann das Löschen einer Gruppe nicht durch Setzen des Rückgabestatus verhindert werden.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Getroffene Gruppe
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcGroupDeleting(ByVal sender As Object, ByVal e As
NETRONIC.Xnet.VcGroupDeletingEventArgs) Handles VcNet1.VcGroupDeleting
    If e.Group.Name = "A" Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        MsgBox("Group A cannot be deleted")
    End If
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcGroupDeleting(object sender,
NETRONIC.XNet.VcGroupDeletingEventArgs e)
{
    if (e.Group.Name == "A")
    {
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
        MessageBox.Show("Group A cannot be deleted");
    }
}
```

**VcGroupLeftClicking**

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender auf eine Gruppe mit der linken Maustaste klickt. Das getroffene Group-Objekt wird zusammen mit der Position (x,y-Koordinaten des Mauszeigers) als Parameter übergeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Getroffene Gruppe
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcGroupLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupClickingEventArgs) Handles VcNet1.VcGroupLeftClicking
    MsgBox(e.Group.SubGroups.Count)
End Sub
```



**Code-Beispiel C#**

```
private void vcNet1_VcGroupLeftClicking(object sender,
NETRONIC.XNet.VcGroupClickingEventArgs e)
{
    MessageBox.Show(e.Group.SubGroups.Count.ToString());
}
```

**VcGroupLeftDoubleClicking**

Ereignis von VcNet

**Dieses Ereignis tritt ein, wenn der Anwender auf eine Gruppe mit der linken Maustaste doppelklickt. Das getroffene Group-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter übergeben.**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Getroffene Gruppe
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcGroupLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupClickingEventArgs) Handles VcNet1.VcGroupLeftDoubleClicking
    MsgBox(e.Group.Name)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcGroupLeftDoubleClicking(object sender,
NETRONIC.XNet.VcGroupClickingEventArgs e)
{
    MessageBox.Show(e.Group.Name);
}
```

**VcGroupModified**

Ereignis von VcNet

**Dieses Ereignis tritt ein, wenn das interaktive Kollabieren oder Expandieren einer Gruppe im Modus Clusterung abgeschlossen wurde.**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Veränderte Gruppe
⇒ modificationType	VcGroupModificationTypes	Art der Veränderung
	<b>Mögliche Werte:</b> .vcGMTCollapsing 2 .vcGMTExpanding 4	Gruppe kollabiert Gruppe expandiert

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcGroupModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupModifiedEventArgs) Handles VcNet1.VcGroupModified
    MsgBox("The group has been modified.")
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcGroupModified(object sender,
NETRONIC.XNet.VcGroupModifiedEventArgs e)
{
    MessageBox.Show("The group has been modified");
}
```

## VcGroupModifying

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn eine Gruppe im Modus Clustering interaktiv kollabiert (`modificationType = vcGMTCollapsing`) oder expandiert (`vcGMTExpanding`) wird. Das getroffene Group-Objekt, die Art der Veränderung und der Rückgabestatus werden als Parameter mitgegeben. Durch Setzen des Rückgabestatus auf `vcRetStatFalse` wird die Aktion abgebrochen.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Veränderte Gruppe
⇒ modificationType	VcGroupModificationTypes	Art der Veränderung
	<b>Mögliche Werte:</b> .vcGMTCollapsing 2 .vcGMTExpanding 4	Gruppe kollabiert Gruppe expandiert
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcGroupModifying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupModifyingEventArgs) Handles VcNet1.VcGroupModifying
    Select Case e.ModificationType
        Case VcGroupModificationTypes.vcGMTNothing
            MsgBox("No modification")
        Case VcGroupModificationTypes.vcGMTAnything
            MsgBox("Any modification")
        Case VcGroupModificationTypes.vcGMTMinusPressed
            MsgBox("Collapsing group:" + e.Group.Name)
        Case VcGroupModificationTypes.vcGMTPlusPressed
            MsgBox("Expanding group" + e.Group.Name)
    End Select
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcGroupModifying(object sender,
NETRONIC.XNet.VcGroupModifyingEventArgs e)
{
    switch (e.ModificationType)
    {
        case VcGroupModificationTypes.vcGMTNothing:
            MessageBox.Show("No modification");
            break;
        case VcGroupModificationTypes.vcGMTAnything:
            MessageBox.Show("Any modification");
            break;
        case VcGroupModificationTypes.vcGMTMinusPressed:
            MessageBox.Show("Collapsing group: " + e.Group.Name);
            break;
        case VcGroupModificationTypes.vcGMTPlusPressed:
            MessageBox.Show("Expanding group: " + e.Group.Name);
            break;
    }
}
```

**VcGroupRightClicking****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender auf eine Gruppe mit der rechten Maustaste klickt. Das getroffene Gruppenobjekt wird zusammen mit der Position (x,y-Koordinaten) als Parameter übergeben. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ group	VcGroup	Getroffene Gruppe
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate

⇒ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcGroupRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcGroupClickingEventArgs) Handles VcNet1.VcGroupRightClicking
    PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
    e.ReturnStatus = VcNetLib.VcReturnStatus.vcRetStatNoPopup
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcGroupRightClicking(object sender,
NETRONIC.XNet.VcGroupClickingEventArgs e)
{
    PopupMenu.Show(vcNet1, new Point(e.X, e.Y));
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

**VcHelpRequested**

**Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender in einem zur Laufzeit angezeigten Dialog die **F1**-Taste drückt. Die Applikation erhält damit die Möglichkeit, ihr eigenes Hilfesystem aufzurufen, um dialog- und anwendungsbezogene Hilfe anbieten zu können.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcHelpRequestedEventArgs	Ereignisspezifisches Objekt

**Eigenschaften des VcHelpRequestedEventArgs-Objektes**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ DialogType	VcDialogType	Dialog, für den Hilfe angefordert wurde
	<b>Mögliche Werte:</b> .vcEditDataRecordDialog 5400 .vcPageSetupDialog 4097	Hilfe wurde für den <b>Datensatz bearbeiten</b> Dialog angefordert Hilfe wurde für den <b>Seite einrichten</b> Dialog angefordert

.vcPrintPreviewDialog 4096

Hilfe wurde für den **Druckvorschau** Dialog angefordert

## VcInPlaceEditorShowing

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der im Programm implementierte Editor gestartet wird.

Das Ereignis wird erst aktiviert, wenn die Eigenschaft **InPlaceEditing-Allowed** auf True gesetzt ist.

Wenn Sie den Rückgabestatus auf **False** setzen, können Sie den Start des integrierten Editors verhindern und an den übergebenen Koordinaten ein eigener Editor hochbringen.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ editObject	VcObject	Editiertes Objekt
⇒ editObjectType	VcObjectType	Objekttyp
	<b>Mögliche Werte:</b>	
	.vcObjTypeBox 15	Objekttyp <b>Box</b>
	.vcObjTypeGroup 7	Objekttyp <b>Gruppe</b>
	.vcObjTypeLinkCollection 3	Objekttyp <b>LinkCollection</b>
	.vcObjTypeNode 2	Objekttyp <b>Knoten</b>
	.vcObjTypeNone 0	kein Objekt
⇒ fieldIndex	System.Int32	Feldindex
⇒ objRectComplete	VcRect	Komplettes Rechteck des getroffenen Objekts
⇒ objRectVisible	VcRect	Sichtbares Rechteck des getroffenen Objekts
⇒ fldRectComplete	VcRect	Komplettes Rechteck des getroffenen Feldes
⇒ fldRectVisible	VcRect	Sichtbares Rechteck des getroffenen Feldes
returnStatus	VcReturnStatus	
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```

Private Sub VcNet1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcInPlaceEditorShowingEventArgs) Handles
VcNet1.VcInPlaceEditorShowing

    Dim node As VcNode
    node = e.EditObject
    If e.EditObjectType = VcObjectType.vcObjTypeNodeInTable Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        Select Case e.FieldIndex
            Case 1 'Name
                TextBox1.Left = e.FldRectVisible.Left + VcNet1.Left
                TextBox1.Top = e.FldRectVisible.Top + VcNet1.Top
                TextBox1.Width = e.FldRectVisible.Width
                TextBox1.Height = e.FldRectVisible.Height
                TextBox1.Text = node.DataField(0)
                TextBox1.Visible = True
                TextBox1.Focus()
            Case 2, 3 'Start or End
                DateTimePicker1.Left = e.FldRectVisible.Left + VcNet1.Left
                DateTimePicker1.Top = e.FldRectVisible.Top + VcNet1.Top
                DateTimePicker1.Value = node.DataField(0)
                DateTimePicker1.Visible = True
                DateTimePicker1.Focus()
            Case 13 'Employee
                ComboBox1.Left = e.FldRectVisible.Left + VcNet1.Left
                ComboBox1.Top = e.FldRectVisible.Top + VcNet1.Top
                ComboBox1.Width = e.FldRectVisible.Width
                ComboBox1.Height = e.FldRectVisible.Height
                ComboBox1.Text = node.DataField(0)
                ComboBox1.Visible = True
                ComboBox1.Focus()
        End Select
    End If

```

**Code-Beispiel C#**

```

private void vcNet1_VcInPlaceEditorShowing(object sender,
NETRONIC.XNet.VcInPlaceEditorShowingEventArgs e)
{
    VcNode node = (VcNode)e.EditObject;
    if (e.EditObjectType == VcObjectType.vcObjTypeNodeInTable)
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
    switch (e.FieldIndex)
    {
        case 1: //Name
            textBox1.Left = e.FldRectVisible.Left + vcNet1.Left;
            textBox1.Top = e.FldRectVisible.Top + vcNet1.Top;
            textBox1.Width = e.FldRectVisible.Width;
            textBox1.Height = e.FldRectVisible.Height;
            textBox1.Text = Convert.ToString(node.get_DataField(0));
            textBox1.Visible = true;
            textBox1.Focus();
            break;
        case 2: //Start or end
            dateTimePicker1.Left = e.FldRectVisible.Left + vcNet1.Left;
            dateTimePicker1.Top = e.FldRectVisible.Top + vcNet1.Top;
            dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
            dateTimePicker1.Visible = true;
            dateTimePicker1.Focus();
            break;
        case 13: //Employee
            comboBox1.Left = e.FldRectVisible.Left + vcNet1.Left;
            comboBox1.Top = e.FldRectVisible.Top + vcNet1.Top;
            comboBox1.Width = e.FldRectVisible.Width;
            comboBox1.Height = e.FldRectVisible.Height;
            comboBox1.Text = Convert.ToString(node.get_DataField(0));
            comboBox1.Visible = true;
            comboBox1.Focus();
            break;
    }
}

```

**VcLegendViewClosed****Ereignis von VcNet**

Dieses Ereignis wird aufgerufen, wenn das Popup-Fenster der Legendenansicht geschlossen wird.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	
⇒ e	VcLEgendViewClosedEventArgs	

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇐ (no parameter)		

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLegendViewClosed(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLegendViewClosedEventArgs) Handles VcNet1.VcLegendViewClosed
    MsgBox("Do you want to close the legend view window?", MsgBoxStyle.OKCancel)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLegendViewClosed(object sender,
NETRONIC.XNet.VcLegendViewClosedEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Do you want to close the legend view
window?", "Closing legend view window", MessageBoxButtons.OKCancel);
}
```

**VcLinkCreated****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn das interaktive Anlegen einer Verbindung zwischen zwei Knoten beendet ist. Das Verbindungs-Objekt, der Typ des Anlegens der Verbindung und die Information, ob die angelegte Verbindung die einzige Verbindung bzw. die letzte Verbindung einer Menge ist, werden als Parameter übergeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ link	VcLink	Angelegte Verbindung
⇒ creationType	VcCreationType	Typ des Knotenanlegens
	<b>Mögliche Werte:</b>	
	.vcDataRecordCreated 6	Datensatz wurde durch Interaktion angelegt
	.vcLinkCreated 2	Verbindung wurde durch Interaktion angelegt
	.vcNodeCreated 1	Knoten durch "Stempeln" angelegt
	.vcNodesAndLinksCloned 4	selektierte Knoten wurden durch Ziehen mit der Maus bei gleichzeitigem Drücken der Strg-Taste kopiert
	.vcNodeWithLinkCreated 3	Knoten zusammen mit Verbindung angelegt
⇒ isLast	System.Boolean	Angelegte Verbindung ist/ist nicht die einzige bzw. die letzte Verbindung einer Menge

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinkCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinkCreatedEventArgs) Handles VcNet1.VcLinkCreated
    MsgBox(e.Link.AllData)
End Sub
```



**Code-Beispiel C#**

```
private void vcNet1_VcLinkCreated(object sender,
NETRONIC.XNet.VcLinkCreatedEventArgs e)
{
    MessageBox.Show(e.Link.AllData.ToString());
}
```

**VcLinkCreating****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender interaktiv eine Verbindung zwischen zwei Knoten erzeugt hat. Das neu erzeugte Objekt wird als Parameter übergeben, so dass eine Validierung und ggf. ein Datenbank-Eintrag vorgenommen werden kann.

Dieses Ereignis sollte nur verwendet werden, um Daten der aktuellen Verbindung auszulesen. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcLinkCreated**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ link	VcLink	Angelegte Verbindung
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinkCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinkCreatedEventArgs) Handles VcNet1.VcLinkCreated
    MsgBox(e.Link.AllData)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinkCreated(object sender, VcNetLib.VcLinkCreatedEventArgs
e)
{
    MessageBox.Show(e.Link.AllData.ToString());
}
```

## VcLinkDeleted

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn das Löschen einer Verbindung beendet ist. Die getroffene Verbindung und die Information, ob die angelegte Verbindung die einzige Verbindung bzw. die letzte Verbindung einer Menge ist, werden als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ link	VcLink	Gelöschte Verbindung
⇒ isLast	System.Boolean	Gelöschte Verbindung ist/ist nicht die einzige bzw. die letzte Verbindung einer Menge.

## VcLinkDeleting

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender per Taste oder mit Hilfe des Kontextmenüs eine Verbindung löscht. Die getroffene Verbindung wird als Parameter übergeben, so dass noch eine Überprüfung vorgenommen werden kann. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Verbindung nicht gelöscht.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ link	VcLink	Gelöschte Verbindung
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcLinkDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinkDeletingEventArgs) Handles VcNet1.VcLinkDeleting
    'deny deletion of link with a certain predecessor
    If e.Link.PredecessorNode.DataField(0) = "1" Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinkDeleting(object sender,
NETRONIC.XNet.VcLinkDeletingEventArgs e)
{
    // deny deletion of link with a certain predecessor
    if (e.Link.PredecessorNode.get_DataField(0).Equals("1"))
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

**VcLinkModified****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn die Modifizierung der angegebenen Verbindung abgeschlossen ist.

Das Knoten-Objekt und die Information, ob der angelegte Knoten der einzige Knoten bzw. der letzte Knoten einer Menge ist (derzeit immer **True**), werden als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ link	VcLink	Angelegte Verbindung
⇒ isLast	System.Boolean	Angelegte Verbindung ist/ist nicht die einzige bzw. die letzte Verbindung einer Menge

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinkModified(ByVal sender As System.Object, _
ByVal e As NETRONIC.XNet.VcLinkModifiedEventArgs) _
Handles VcNet1.VcLinkModified
    'modify a record in the underlying database of the application
    modifyDataRecord(e.Link.AllData)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinkModified(object sender, VcLinkModifiedEventArgs e)
{
    //modify a record in the underlying database of the application
    modifyDataRecord(e.Link.AllData);
}
```

**VcLinkModifying****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender interaktiv eine Verbindung verändert hat. Dabei kann die Verbindung verschoben oder ein Wert im Dialog **Daten bearbeiten** verändert worden sein. Wenn Sie den

Rückgabestatus auf **vcRetStatFalse** setzen, wird die Veränderung rückgängig gemacht.

Dieses Ereignis sollte nur verwendet werden, um Daten der aktuellen Verbindung auszulesen. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcLinkModified**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ link	VcLink	Verbindung nach der Veränderung
⇒ oldlink	VcLink	Verbindung vor der Veränderung
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

#### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcLinkModifying(ByVal sender As System.Object, _
    ByVal e As NETRONIC.XNet.VcLinkModifyingEventArgs) _
    Handles VcNet1.VcLinkModifying
    'deny any modification
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

#### Code-Beispiel C#

```
private void vcNet1_VcLinkModifying(object sender, VcLinkModifyingEventArgs e)
{
    //deny any modification
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcLinksLeftClicking

#### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Verbindung bzw. mehrere sich überlagernde Verbindungen klickt. Ein LinkCollection-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter übergeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ linkCltn	VcLinkCollection	Getroffenes LinkCollection-Objekt
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate

⇒ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksLeftClicking
    Dim linkCltn As VcLinksCollection
    Dim link As VcLink
    linkCltn = VcNet1.LinkCollection
    'set certain data field of all links
    For Each link In linkCltn
        link.DataField(2) = "A"
    Next
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcLinksLeftClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
{
    VcLinkCollection linkCltn = vcNet1.LinkCollection;
    // set certain data field of all links
    foreach (VcLink link in linkCltn)
        link.set_DataField(2, "A");
}
```

## VcLinksLeftDoubleClicking

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Verbindung bzw. mehrere sich überlagernde Verbindungen doppelt klickt. Ein LinkCollection-Objekt wird zusammen mit der Position (x,y-Koordinaten) als Parameter übergeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ linkCltn	VcLinkCollection	Getroffenes LinkCollection-Objekt
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksLeftClicking
    Dim linkCltn As VcLinkCollection
    Dim link As VcLink
    linkCltn = VcNet1.LinkCollection
    'set certain data field of all links
    For Each link In linkCltn
        link.DataField(2) = "A"
    Next
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinksLeftClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
{
    VcLinkCollection inkCltn = vcNet1.LinkCollection;
    // set certain data field of all links
    foreach (VcLink link in linkCltn)
        node.set_DataField(2, "A");
}
```

**VcLinksMarked****Ereignis von VcNet**

Mit diesem Ereignis wird das Ende einer Markier- oder Demarkieroperation bei Verbindungen angezeigt.

	Datentyp	Beschreibung
<b>Eigenschaften:</b> ↔ (no parameter)		Kein Parameter

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinksMarked(ByVal sender As System.Object, _
ByVal e As
NETRONIC.XNet.VcLinksMarkedEventArgs)_
Handles
VcNet1.VcLinksMarked
    MsgBox("Links have been successfully marked.")
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinksMarked(object sender, VcLinksMarkedEventArgs e)
{
    MessageBox.Show("Links have been successfully marked.");
}
```

## VcLinksMarking

### Ereignis von VcNet

Mit diesem Ereignis wird bekanntgegeben, dass der Benutzer Verbindungen zum Markieren ausgewählt oder markierte Verbindungen durch einen Klick in den leeren Diagrammbereich demarkiert hat. In der Verbindungen-Auflistung (LinkCollection-Objekt) sind die beim Markieren ausgewählten Verbindungen verzeichnet. Falls durch einen Klick ins leere Diagramm demarkiert wurde, ist die LinkCollection leer.

Wenn Sie den Rückgabestatus auf **vcRetStatFalse** setzen, muss das Markieren oder Demarkieren selbst übernommen werden.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcLinksMarked**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ linkCollection	VcLinkCollection	Verbindungsaflistung (LinkCollection), die die vom Anwender selektierten Verbindungen enthält. Wenn in das Diagramm geklickt wurde, ist die Auflistung leer.
⇒ button	System.Int16	Zahl, die angibt, auf welche Weise markiert wurde: <b>0</b> : über die Tastatur, <b>1</b> : linke Maustaste, <b>2</b> : rechte Maustaste, <b>4</b> : mittlere Maustaste
⇒ shift	System.Int16	Zahl, die den Zustand der Modifizierungstasten zu dem Zeitpunkt angibt, zu dem Daten über das Drop-Ziel gezogen werden. Die gültigen Modifizierungstasten sind die <Umschalt>-, <Strg>- und <Alt>-Tasten, die den Zahlen <b>1</b> , <b>2</b> und <b>4</b> entsprechen. Der Parameter <b>shift</b> zeigt den Status dieser Tasten an; es können einige, alle oder keines der drei Zahlen gesetzt werden, was jeweils anzeigt, dass einige, alle oder keine der Tasten gedrückt wird. Wenn beispielsweise die <Strg>- und die <Alt>-Tasten gedrückt werden, ist der Wert von <b>shift</b> "6".
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinksMarking(ByVal sender As System.Object, _
                                ByVal e As
NETRONIC.XNet.VcLinksMarkingEventArgs) _
                                Handles
VcNet1.VcLinksMarking
    If MsgBox("Mark this node?", vbYesNo, "Marking nodes") = vbNo Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinksMarking(object sender, VcLinksMarkingEventArgs e)
{
    if (MessageBox.Show("Mark this node?",
                        "Marking nodes",
                        MessageBoxButtons.YesNo) ==
DialogResult.No)
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

**VcLinksRightClicking****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf eine Verbindung bzw. mehrere sich überlagernde Verbindungen klickt. Ein LinkCollection-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Cursors als Parameter übergeben. Sie können so an der betreffenden Position Ihr eigenes Kontextmenü anzeigen. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ linkCltn	VcLinkCollection	Getroffenes LinkCollection-Objekt
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.



**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcLinksRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcLinksClickingEventArgs) Handles VcNet1.VcLinksRightClicking
    PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
    e.ReturnStatus = VcNetLib.VcReturnStatus.vcRetStatNoPopup
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcLinksRightClicking(object sender,
NETRONIC.XNet.VcLinksClickingEventArgs e)
{
    PopupMenu.Show(vcNet1, new Point (e.X, e.Y));
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

**VcMouseDoubleClicking****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender eine Maustaste zweimal drückt.

Bitte beachten Sie auch die Eigenschaft **<MouseProcessingEnabled**. Wenn Sie den Status der Sondertasten **Shift**, **Strg** und **Alt** der Tastatur erfragen möchten, gibt es hierfür die statische Methode **System.Windows.Forms.Control.ModifierKeys**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ button	System.Int16	Zahl, die angibt, welche Maustasten gedrückt sind: <b>1</b> (links), <b>2</b> (rechts) oder <b>4</b> (Mitte).
⇒ shift	System.Int16	Eine Ganzzahl, die dem Zustand der Tasten Umschalt, Strg und Alt zu dem Zeitpunkt entspricht, an dem das Ereignis aufgetreten ist. Das Argument Shift ist ein Bitfeld, bei dem die niederwertigen Bits der Umschalt-Taste (Bit 0), der Strg-Taste (Bit 1) und der Alt-Taste (Bit 2) entsprechen. Diese Bits entsprechen jeweils dem Wert 1, 2 und 4. Einige, alle oder keine dieser Bits können gesetzt werden, um anzuzeigen, dass einige, alle oder keine der Tasten gedrückt sind. Wenn zum Beispiel sowohl Strg als auch Alt gedrückt werden, hat shift den Wert 6.
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers

**VcMouseDown****Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender eine Maustaste drückt.

Bitte beachten Sie auch die Eigenschaft **MouseProcessingEnabled**. Wenn Sie den Status der Sondertasten **Shift**, **Strg** und **Alt** der Tastatur erfragen möchten, gibt es hierfür die statische Methode **System.Windows.Forms.Control.ModifierKeys**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ button	System.Int16	Zahl, die angibt, welche Maustasten gedrückt sind: <b>1</b> (links), <b>2</b> (rechts) oder <b>4</b> (Mitte).
⇒ shift	System.Int16	Eine Ganzzahl, die dem Zustand der Tasten Umschalt, Strg und Alt zu dem Zeitpunkt entspricht, an dem das Ereignis aufgetreten ist. Das Argument Shift ist ein Bitfeld, bei dem die niederwertigen Bits der Umschalt-Taste (Bit 0), der Strg-Taste (Bit 1) und der Alt-Taste (Bit 2) entsprechen. Diese Bits entsprechen jeweils dem Wert 1, 2 und 4. Einige, alle oder keine dieser Bits können gesetzt werden, um anzuzeigen, dass einige, alle oder keine der Tasten gedrückt sind. Wenn zum Beispiel sowohl Strg als auch Alt gedrückt werden, hat shift den Wert 6.
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers

## VcMouseMove

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender die Maus bewegt.

Bitte beachten Sie auch die Eigenschaft **MouseProcessingEnabled**. Wenn Sie den Status der Sondertasten **Shift**, **Strg** und **Alt** der Tastatur erfragen möchten, gibt es hierfür die statische Methode **System.Windows.Forms.Control.ModifierKeys**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ button	System.Int16	Zahl, die angibt, welche Maustasten gedrückt sind: <b>1</b> (links), <b>2</b> (rechts) oder <b>4</b> (Mitte).

⇒ shift	System.Int16	Eine Ganzzahl, die dem Zustand der Tasten Umschalt, Strg und Alt zu dem Zeitpunkt entspricht, an dem das Ereignis aufgetreten ist. Das Argument Shift ist ein Bitfeld, bei dem die niederwertigen Bits der Umschalt-Taste (Bit 0), der Strg-Taste (Bit 1) und der Alt-Taste (Bit 2) entsprechen. Diese Bits entsprechen jeweils dem Wert 1, 2 und 4. Einige, alle oder keine dieser Bits können gesetzt werden, um anzuzeigen, dass einige, alle oder keine der Tasten gedrückt sind. Wenn zum Beispiel sowohl Strg als auch Alt gedrückt werden, hat shift den Wert 6.
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers

## VcMouseUp

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender eine gedrückte Maustaste wieder loslässt.

Bitte beachten Sie auch die Eigenschaft **MouseProcessingEnabled**. Wenn Sie den Status der Sondertasten **Shift**, **Strg** und **Alt** der Tastatur erfragen möchten, gibt es hierfür die statische Methode **System.Windows.Forms.Control.ModifierKeys**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ button	System.Int16	Zahl, die angibt, welche Maustasten gedrückt sind: <b>1</b> (links), <b>2</b> (rechts) oder <b>4</b> (Mitte).
⇒ shift	System.Int16	Eine Ganzzahl, die dem Zustand der Tasten Umschalt, Strg und Alt zu dem Zeitpunkt entspricht, an dem das Ereignis aufgetreten ist. Das Argument Shift ist ein Bitfeld, bei dem die niederwertigen Bits der Umschalt-Taste (Bit 0), der Strg-Taste (Bit 1) und der Alt-Taste (Bit 2) entsprechen. Diese Bits entsprechen jeweils dem Wert 1, 2 und 4. Einige, alle oder keine dieser Bits können gesetzt werden, um anzuzeigen, dass einige, alle oder keine der Tasten gedrückt sind. Wenn zum Beispiel sowohl Strg als auch Alt gedrückt werden, hat shift den Wert 6.
⇒ x	System.Int32	X-Koordinate des Mauszeigers
⇒ y	System.Int32	Y-Koordinate des Mauszeigers

## VcNodeCreated

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn das interaktive Anlegen eines Knotens abgeschlossen ist. Das Knotenobjekt, der Typ des Knotenanlegens und die Information, ob der angelegte Knoten der einzige Knoten bzw. der letzte Knoten einer Menge ist, werden als Parameter übergeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Angelegter Knoten
⇒ creationType	VcCreationType	Typ des Knotenanlegens
	<b>Mögliche Werte:</b>	
	.vcDataRecordCreated 6	Datensatz wurde durch Interaktion angelegt
	.vcLinkCreated 2	Verbindung wurde durch Interaktion angelegt
	.vcNodeCreated 1	Knoten durch "Stempeln" angelegt
	.vcNodesAndLinksCloned 4	selektierte Knoten wurden durch Ziehen mit der Maus bei gleichzeitigem Drücken der Strg-Taste kopiert
	.vcNodeWithLinkCreated 3	Knoten zusammen mit Verbindung angelegt
⇒ isLast	System.Boolean	Angelegter Knoten ist/ist nicht der einzige Knoten bzw. der letzte Knoten einer Menge

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeCreated(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeCreatedEventArgs) Handles VcNet1.VcNodeCreated
    MsgBox(e.Node.AllData)
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeCreated(object sender,
NETRONIC.XNet.VcNodeCreatedEventArgs e)
{
    MessageBox.Show(e.Node.AllData.ToString());
}
```

## VcNodeCreating

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender interaktiv einen Knoten erzeugt hat. Das Knotenobjekt wird als Parameter übergeben, so dass eine Datenvalidierung vorgenommen werden kann. Dies kann wichtig sein, wenn der Benutzer bei aktiviertem Dialog **Daten bearbeiten** eigene Daten eingegeben hat.

Dieses Ereignis sollte nur verwendet werden, um Daten des aktuellen Knotens auszulesen. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcNodeCreated**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Anzulegender Knoten
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeCreating(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeCreatingEventArgs) Handles VcNet1.VcNodeCreating
    MsgBox("Show your own dialog")
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeCreating(object sender,
NETRONIC.XNet.VcNodeCreatingEventArgs e)
{
    MessageBox.Show("Show your own dialog");
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcNodeDeleted

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn das interaktive Löschen eines Knotens abgeschlossen ist. Das Knotenobjekt und die Information, ob der gelöschte Knoten der zuletzt gelöschte einer Menge ist, werden als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Gelöschter Knoten
⇒ isLast	System.Boolean	Gelöschter Knoten ist/ist nicht der letzte Knoten einer Menge

## VcNodeDeleting

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs einen Knoten löscht. Der getroffene Knoten wird als Parameter übergeben, so dass noch eine Überprüfung vorgenommen werden kann. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird der Knoten nicht gelöscht.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Knotenobjekt
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeDeletingEventArgs) Handles VcNet1.VcNodeDeleting
    'deny the deletion of the last node in the chart
    If VcNet1.NodeCollection.Count = 1 Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        MsgBox("The last node in the chart cannot be deleted.")
    End If
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeDeleting(object sender,
NETRONIC.XNet.VcNodeDeletingEventArgs e)
{
    //deny the deletion of the last node in the chart
    if (vcNet1.NodeCollection.Count == 1)
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
        MessageBox.Show("The last node in the chart cannot be deleted.");
}
}
```

## VcNodeLeftClicking

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender auf einen Knoten mit der linken Maustaste klickt. Das getroffene Knotenobjekt wird zusammen mit der Position des Mauszeigers(x,y-Koordinaten) und dem Diagrammbereich übergeben.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Knotenobjekt
⇒ location	VcLocation	Lokalisierung im Chart
	<b>Mögliche Werte:</b> .vcInDiagram 1	im Knotenbereich
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftClicking
    'change data field of the node
    e.Node.DataField(4) = 1 - Convert.ToInt64(e.Node.DataField(4))
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeLeftClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
{
    //change data field of the node
    e.Node.set_DataField(4, Convert.ToInt64(e.Node.get_DataField(4)));
}
```

## VcNodeLeftDoubleClicking

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender auf einen Knoten mit der linken Maustaste doppelt klickt. Das getroffene Knotenobjekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) und dem Diagrammbereichsparameter übergeben. Nach der Rückkehr wird der Dialog **Vorgänge bearbeiten** für diesen Knoten aufgerufen. Wenn Sie den Rückgabestatus auf **vcRetStatFalse** setzen, wird dieser Aufruf unterdrückt.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Knotenobjekt
⇒ location	VcLocation	Lokalisierung im Chart
	<b>Mögliche Werte:</b>	

	.vcInDiagram 1	im Knotenbereich
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeLeftDoubleClicking
    MsgBox("Show your own dialog")
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeLeftDoubleClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
{
    MessageBox.Show("Show your own dialog");
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcNodeModifiedEx

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn die Modifizierung des angegebenen Knotens abgeschlossen ist.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcNodeModifiedEventArgs	Ereignisspezifisches Objekt

### Eigenschaften des VcNodeModifiedEventArgs-Objektes

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Angelegter Knoten
⇒ isLast	System.Boolean	Angelegter Knoten ist/ist nicht der einzige Knoten bzw. der letzte Knoten einer Menge



**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcNodeModifiedEx(ByVal sender As System.Object, _
                                   ByVal e As
NETRONIC.XNet.VcNodeModifiedExEventArgs)
    Handles VcNet1.VcNodeModifiedEx
    'modify a record in the underlying database of the application
    modifyDataRecord(e.Node.AllData)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcNodeModifiedEx(object sender,
VcNodeModifiedExEventArgs e)
{
    //modify a record in the underlying database of the application
    modifyDataRecord(e.Node.AllData);
}
```

## VcNodeModifying

**Ereignis von VcNet**

Dieses Ereignis tritt ein, wenn der Anwender interaktiv einen Knoten verändert. Dabei kann der Knoten verschoben oder ein Wert im Dialogfeld **Vorgänge bearbeiten** verändert worden sein. Die Daten des Knotens vor und nach der Veränderung werden als Parameter zurückgegeben. Über den Parameter **modificationType** erhalten Sie nähere Informationen über die Art der Veränderung.

Durch Setzen des Rückgabestatus auf **vcRetStatFalse** kann die Änderung verhindert werden.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcNodeModified**.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcNodeModifiedEventArgs	Ereignisspezifisches Objekt

**Eigenschaften des VcNodeModifiedEventArgs-Objektes**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ oldNode	VcNode	Knoten vor der Veränderung

⇒ node	VcNode	zu verändernder Knoten
⇒ modificationType	VcModificationTypes	Art der Veränderung
	<b>Mögliche Werte:</b> .vcAnything 1 .vcChangedGroup 16  .vcMoved 8 .vcNothing 0	Änderungstyp nicht näher bestimmt Zuordnung des Knotens zu einer Gruppe wurde verändert (nur für Knoten). Objekt wurde verschoben. Keine Änderung
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeModifyingEventArgs) Handles VcNet1.VcNodeModifying
    ' revoke the modification if the node would change the group
    If e.ModificationType And VcModificationTypes.vcChangedGroup Then
        MsgBox("The node cannot be moved to a different group.")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub

End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeModifying(object sender,
NETRONIC.XNet.VcNodeModifyingEventArgs e)
{
    //revoke the modification if the node would change the group
    if (e.ModificationType == VcModificationTypes.vcChangedGroup)
    {
        MessageBox.Show("The node cannot be moved into another group.");
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
    }
}
```

## VcNodeRightClicking

### Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf einen Knoten klickt. Das getroffene Knotenobjekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter übergeben. Sie können so an der entsprechenden Position Ihr eigenes Kontextmenü anzeigen. Das integrierte Menü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

Dieses Ereignis sollte nur verwendet werden, um Daten des aktuellen Knotens auszulesen. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcNodesMarked**.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ node	VcNode	Knotenobjekt
⇒ location	VcLocation	Lokalisierung im Chart
	<b>Mögliche Werte:</b> .vcInDiagram 1	im Knotenbereich
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
↔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

#### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking
    PopupMenu.Show(VcNet1, New Point(e.X, e.Y))
    e.ReturnStatus = NETRONIC.XNet.VcReturnStatus.vcRetStatNoPopup
End Sub
```

#### Code-Beispiel C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
{
    PopupMenu.Show(vcNet1, new Point(e.X, e.Y));
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

## VcNodesMarked

Ereignis von VcNet

Mit diesem Ereignis wird das Ende einer Markier- oder Demarkieroperation eines Knotens angezeigt.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
↔ (no parameter)		Kein Parameter

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcNodesMarked(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodesMarkedEventArgs) Handles VcNet1.VcNodesMarked
    MsgBox("Nodes have been marked successfully.")
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcNodesMarked(object sender,
NETRONIC.XNet.VcNodesMarkedEventArgs e)
{
    MessageBox.Show("Nodes have been marked successfully.");
}
```

**VcNodesMarking****Ereignis von VcNet**

Mit diesem Ereignis wird bekanntgegeben, dass der Benutzer Knoten zum Markieren ausgewählt oder markierte Knoten durch einen Klick in den leeren Diagrammbereich demarkiert hat. In der Knotenaufistung (NodeCollection-Objekt) sind die beim letzten Markiervorgang ausgewählten Knoten verzeichnet. Falls durch einen Klick ins leere Diagramm demarkiert wurde, ist die NodeCollection leer.

Wenn Sie den Rückgabestatus auf **vcRetStatFalse** setzen, muss das Markieren oder Demarkieren selbst übernommen werden.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcNodesMarked**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ nodeCollection	VcNodeCollection	NodeCollection, die die vom Anwender selektierten Knoten enthält. Wenn in das Diagramm geklickt wurde, ist die Collection leer.
⇒ button	System.Int16	Zahl, die angibt, auf welche Weise markiert wurde: <b>0:</b> über die Tastatur, <b>1:</b> linke Maustaste, <b>2:</b> rechte Maustaste, <b>4:</b> mittlere Maustaste
⇒ shift	System.Int16	Zahl, die den Zustand der Modifizierungstasten zu dem Zeitpunkt angibt, zu dem Daten über das Drop-Ziel gezogen werden. Die gültigen Modifizierungstasten sind die <Umschalt>-, <Strg>- und <Alt>-Tasten, die den Zahlen <b>1</b> , <b>2</b> und <b>4</b> entsprechen. Der Parameter <b>shift</b> zeigt den Status dieser Tasten an; es können einige, alle oder keines der drei Zahlen gesetzt werden, was jeweils anzeigt, dass einige, alle oder keine der Tasten gedrückt wird. Wenn beispielsweise die <Strg>- und die <Alt>-Tasten gedrückt werden, ist der Wert von <b>shift</b> "6".

⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b> .vcRetStatDefault 2 .vcRetStatFalse 0 .vcRetStatNoPopup 4 .vcRetStatOK 1	Das Default-Verhalten wird nicht verändert. Das Default-Verhalten wird nicht durchgeführt. Das Erscheinen des Kontextmenüs wird unterdrückt. Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodesMarking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodesMarkingEventArgs) Handles VcNet1.VcNodesMarking
    If MsgBox("Mark this node?", MsgBoxStyle.YesNo, "Marking nodes") =
MsgBoxResult.No Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End If
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodesMarking(object sender,
NETRONIC.XNet.VcNodesMarkingEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Mark this node?", "Marking nodes",
MessageBoxButtons.YesNo);
    if (retVal == DialogResult.No)
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
}
```

## VcStatusLabelTextShowing

### Ereignis von VcNet

Dieses Ereignis teilt Information über einen mit dem Mauscursor berührten Knoten mit. Sie können das Ereignis verwenden, um diese Information z. B. in einer Statusleiste anzuzeigen. Die Information selbst wird aus einem Datenfeld des Knotens entnommen. Welches Datenfeld übergeben wird, ist über die Konfigurationsdatei (IFD, Feld IF\_ID2) einstellbar und ist normalerweise das Feld mit dem Index 4.

	Datentyp	Beschreibung
<b>Eigenschaften:</b> ⇔ text	System.String	Text

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcStatusLabelTextShowing(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcStatusLabelTextShowingEventArgs) Handles
VcNet1.VcStatusLabelTextShowing
    TextBox1.Text = e.Text
End Sub
```

**Code-Beispiel C#**

```
private void vcvcNet1_VcStatusLineTextShowing(object sender,
NETRONIC.XNet.VcStatusLineTextShowingEventArgs e)
{
    textBox1.Text = e.Text;
}
```

**VcTextEntrySupplying**

Ereignis von VcNet

Das Ereignis tritt auf, wenn ein Text ausgegeben werden soll. Voraussetzung ist, dass Sie die Eigenschaft **EnableSupplyTextEntryEvent** auf **True** gesetzt haben. Sie können hier alle vorgegebenen Texte von Monats- und Tagesnamen durch eigene Texte ersetzen, z. B. um sie in unterschiedliche Sprachen zu übersetzen.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ sender	VcNet	Verweis auf das ereignisauslösende Objekt
⇒ e	VcTextEntrySupplyingEventArgs	Ereignisspezifisches Objekt

**Eigenschaften des VcTextEntrySupplyingEventArgs-Objektes**

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ controllIndex	VcTextEntryIndex	Zu ersetzender Text
	<b>Mögliche Werte:</b>	
	.vcTXECtxmenArrange 2150	Text im Kontextmenü: <b>Knoten anordnen</b>
	.vcTXECtxmenArrowMode 2116	Text im Kontextmenü: <b>Selektier-Modus</b>
	.vcTXECtxmenCopyNodes 2152	Text im Kontextmenü: <b>Knoten kopieren</b>
	.vcTXECtxmenCreateNodesAndLinksMode 2117	Text im Kontextmenü: <b>Knoten und Verbindungen anlegen</b>
	.vcTXECtxmenCutNodes 2151	Text im Kontextmenü: <b>Knoten ausschneiden</b>
	.vcTXECtxmenDeleteLink 2102	Text im Kontextmenü: <b>Verbindung löschen</b>
	.vcTXECtxmenDeleteNode 2101	Text im Kontextmenü: <b>Knoten löschen</b>
	.vcTXECtxmenEditLink 2154	Text im Kontextmenü: <b>Verbindung bearbeiten</b>
	.vcTXECtxmenEditNode 2100	Text im Kontextmenü: <b>Daten bearbeiten</b>
	.vcTXECtxmenFilePrint 2122	Text im Kontextmenü: <b>Drucken</b>
	.vcTXECtxmenFilePrintPreview 2121	Text im Kontextmenü: <b>Druckvorschau</b>
	.vcTXECtxmenFilePrintSetup 2120	Text im Kontextmenü: <b>Drucker einrichten</b>
	.vcTXECtxmenFullDiagram 2156	Text im Kontextmenü <b>Gesamtnetz wiederherstellen</b>

.vcTXECtxmenGraphicExport 2123	Text im Kontextmenü: <b>Grafik exportieren</b>
.vcTXECtxmenPageLayout 2119	Text im Kontextmenü: <b>Seite einrichten</b>
.vcTXECtxmenPasteNodes 2153	Text im Kontextmenü: <b>Knoten einfügen</b>
.vcTXEDateAM 2225	Ausgabertext für <b>vormittags</b>
.vcTXEDateCW 2223	Ausgabertext für <b>Kalenderwoche</b>
.vcTXEDateDay0 2212	Ausgabertext für <b>Montag</b>
.vcTXEDateDay1 2213	Ausgabertext für <b>Dienstag</b>
.vcTXEDateDay2 2214	Ausgabertext für <b>Mittwoch</b>
.vcTXEDateDay3 2215	Ausgabertext für <b>Donnerstag</b>
.vcTXEDateDay4 2216	Ausgabertext für <b>Freitag</b>
.vcTXEDateDay5 2217	Ausgabertext für <b>Samstag</b>
.vcTXEDateDay6 2218	Ausgabertext für <b>Sonntag</b>
.vcTXEDateMonth0 2200	Ausgabertext für <b>Januar</b>
.vcTXEDateMonth1 2201	Ausgabertext für <b>Februar</b>
.vcTXEDateMonth10 2210	Ausgabertext für <b>November</b>
.vcTXEDateMonth11 2211	Ausgabertext für <b>Dezember</b>
.vcTXEDateMonth2 2202	Ausgabertext für <b>März</b>
.vcTXEDateMonth3 2203	Ausgabertext für <b>April</b>
.vcTXEDateMonth4 2204	Ausgabertext für <b>Mai</b>
.vcTXEDateMonth5 2205	Ausgabertext für <b>Juni</b>
.vcTXEDateMonth6 2206	Ausgabertext für <b>Juli</b>
.vcTXEDateMonth7 2207	Ausgabertext für <b>August</b>
.vcTXEDateMonth8 2208	Ausgabertext für <b>September</b>
.vcTXEDateMonth9 2209	Ausgabertext für <b>Oktober</b>
.vcTXEDateOClock 2224	Ausgabertext für <b>Uhr</b>
.vcTXEDatePM 2226	Ausgabertext für <b>nachmittags</b>
.vcTXEDateQuarter0 2219	Ausgabertext für <b>1. Quartal</b>
.vcTXEDateQuarter1 2220	Ausgabertext für <b>2. Quartal</b>
.vcTXEDateQuarter2 2221	Ausgabertext für <b>3. Quartal</b>
.vcTXEDateQuarter3 2222	Ausgabertext für <b>4. Quartal</b>
.vcTXEDlgLegArrangement 2046	Text im Dialog <b>Legendenattribute: Anordnung</b>
.vcTXEDlgLegBottomMargin 2052	Text im Dialog <b>Legendenattribute: Unterer Rand:</b>
.vcTXEDlgLegFixedToColumns 2048	Text im Dialog <b>Legendenattribute: nach Spaltenanzahl</b>
.vcTXEDlgLegFixedToRows 2047	Text im Dialog <b>Legendenattribute: nach Zeilenanzahl</b>
.vcTXEDlgLegFixedToRowsAndColumns 2049	Text im Dialog <b>Legendenattribute: nach Zeilen- und Spaltenanzahl</b>
.vcTXEDlgLegIldcancel 2042	Schaltfläche im Dialog <b>Legendenattribute: Abbrechen</b>
.vcTXEDlgLegIldd 2040	Dialog <b>Legendenattribute</b> Beschriftung der Titelzeile
.vcTXEDlgLegIldok 2041	Schaltflächentext im Dialog <b>Legendenattribute: OK</b>
.vcTXEDlgLegLegendElements 2045	Text im Dialog <b>Legendenattribute: Legendenelemente</b>
.vcTXEDlgLegLegendFont 2053	Schaltfläche im Dialog <b>Legendenattribute: Schriftart...</b> für Legende
.vcTXEDlgLegLegendTitleFont 2044	Schaltfläche im Dialog <b>Legendenattribute: Schriftart...</b> für Legendentitel
.vcTXEDlgLegLegendTitleVisible 2043	Text im Dialog <b>Legendenattribute: Legendentitel sichtbar</b>
.vcTXEDlgLegMargins 2050	Text im Dialog <b>Legendenattribute: Ränder</b>
.vcTXEDlgLegTopMargin 2051	Text im Dialog <b>Legendenattribute: Oberer Rand:</b>
.vcTXEDlgNedCaptionPrefix 2024	Dialog <b>Vorgänge bearbeiten</b> ,: Text für Beschriftungszeile: "Knoten"
.vcTXEDlgNedIldapply 2027	Dialog <b>Vorgänge bearbeiten</b> , "Übernehmen"-Schaltfläche

.vcTXEDlgNedIldcancel 2016	Text im Dialog <b>Vorgänge bearbeiten: Abbrechen</b>
.vcTXEDlgNedIldclose 2029	Dialog <b>Vorgänge bearbeiten: Schließen</b> -Schaltfläche
.vcTXEDlgNedIldd 2014	Überschrift des Dialogs <b>Vorgänge bearbeiten</b>
.vcTXEDlgNedIldhelp 2028	Dialog <b>Vorgänge bearbeiten: Hilfe</b> -Schaltfläche
.vcTXEDlgNedIldok 2015	Text im Dialog <b>Vorgänge bearbeiten: OK</b>
.vcTXEDlgNedNamesColStr 2018	Text im Dialog <b>Vorgänge bearbeiten: Datenfelder</b>
.vcTXEDlgNedTTGotoFirst 2032	Dialog <b>Vorgänge bearbeiten: Tooltiptext Ersten ausgewählten Vorgang anzeigen</b>
.vcTXEDlgNedTTGotoLast 2035	Dialog <b>Vorgänge bearbeiten</b> , Tooltip "Letzten ausgewählten Vorgang anzeigen"
.vcTXEDlgNedTTGotoNext 2034	Dialog <b>Vorgänge bearbeiten</b> , Tooltiptext <b>Nächsten ausgewählten Vorgang anzeigen</b>
.vcTXEDlgNedTTGotoPrev 2033	Dialog <b>Vorgänge bearbeiten</b> : Tooltiptext <b>Vorherigen ausgewählten Vorgang anzeigen</b>
.vcTXEDlgNedValuesColStr 2019	Text im Dialog <b>Vorgänge bearbeiten: Werte</b>
.vcTXEErrTxtEntryTooLong 2730	Meldungstext: "Eintrag ist zu lang, %s Zeichen sind möglich."
.vcTXEErrTxtWrongLongInteger 2729	Meldungstext: "Eintrag ist kein Integer oder zu lang."
.vcTXEPrctBtApply 2318	Schaltflächen-Text im <b>Seite einrichten</b> -Dialogs: <b>Anwenden</b>
.vcTXEPrctBtCancel 2302	Schaltflächen-Text im Druck-Info-Fenster: <b>Abbrechen</b>
.vcTXEPrctBtClose 2303	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Schließen</b>
.vcTXEPrctBtFitToPage 2308	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Einpassen</b>
.vcTXEPrctBtNext 2305	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Weiter</b>
.vcTXEPrctBtOk 2301	Schaltflächen-Text des Seitenlayout-Dialogs: <b>OK</b>
.vcTXEPrctBtPageLayout 2311	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Seite einrichten</b>
.vcTXEPrctBtPrevious 2304	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Vorher</b>
.vcTXEPrctBtPrint 2313	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Drucken</b>
.vcTXEPrctBtPrinterSetup 2312	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Drucker einrichten</b>
.vcTXEPrctBtSingle 2307	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Einzelseite</b>
.vcTXEPrctBtZoomPrint 2319	Schaltflächen-Text des <b>Druckvorschau</b> -Dialogs: <b>Ausschnitt drucken...</b>
.vcTXEPrctDtAddCuttingMarks 2514	Text des <b>Seite einrichten</b> -Dialogs: <b>Zuschnittmarken</b>
.vcTXEPrctDtAlignment 2526	Text des <b>Seite einrichten</b> -Dialogs: <b>Ausrichtung</b>
.vcTXEPrctDtAlignmentItems 2583	Text des <b>Seite einrichten</b> -Dialogs: <b>Oben links Oben Oben rechts Links Mittig Rechts Unten links Unten Unten rechts</b>

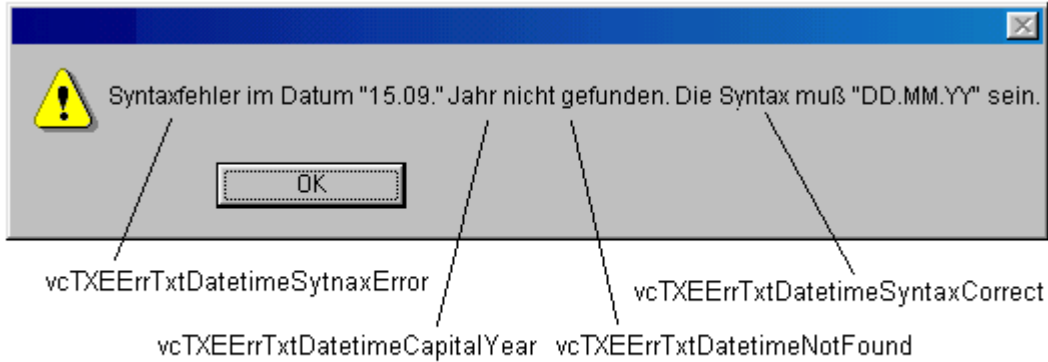


.vcTXEPrctDtApplicationName 2501	Text im Druck-Info-Fenster: <b>Name der Anwendung</b>
.vcTXEPrctDtBottom 2521	Text des <b>Seite einrichten</b> -Dialogs: <b>Unten</b>
.vcTXEPrctDtCm 2530	Text des <b>Seite einrichten</b> -Dialogs: <b>cm</b>
.vcTXEPrctDtCurrentValues 2581	Text des <b>Seite einrichten</b> -Dialogs: <b>Aktuell</b> inaktiv
.vcTXEPrctDtEnableTable 2558	Text des <b>Seite einrichten</b> -Dialogs: <b>Anpassen an Seitenzahl</b>
.vcTXEPrctDtFitToPage 2508	Text des <b>Seite einrichten</b> -Dialogs: <b>Form A Form B Form C</b>
.vcTXEPrctDtFoldingMarksItems 2577	Text des <b>Seite einrichten</b> -Dialogs: Dialogs:" <b>&amp;Faltmarkierungen (DIN 824)</b>
.vcTXEPrctDtFoldingMarksText 2576	Text des <b>Seite einrichten</b> -Dialogs: <b>Fußzeile</b>
.vcTXEPrctDtFooterGroup 2584	Text des <b>Seite einrichten</b> -Dialogs: <b>Rahmen außen</b>
.vcTXEPrctDtFrameOutside 2515	Text des <b>Seite einrichten</b> -Dialogs: <b>Zoll</b>
.vcTXEPrctDtInch 2588	Text des <b>Seite einrichten</b> -Dialogs: <b>Links</b>
.vcTXEPrctDtLeft 2520	Text des <b>Seite einrichten</b> -Dialogs: <b>Mindestgrößen für die Seitenränder</b>
.vcTXEPrctDtMargins 2529	Text des <b>Seite einrichten</b> -Dialogs: <b>Seiten</b>
.vcTXEPrctDtMaxPages 2580	Text <b>Aus</b> Dialog
.vcTXEPrctDtOff 2557	Text des <b>Seite einrichten</b> -Dialogs: <b>Seitenaufteilung</b>
.vcTXEPrctDtOptions 2528	Text des <b>Seite einrichten</b> -Dialogs: <b>Ausdruck</b>
.vcTXEPrctDtOutput 2531	Text des <b>Seite einrichten</b> -Dialogs: <b>Text</b>
.vcTXEPrctDtPageDescription 2562	Fenstertitel des <b>Seite einrichten</b> -Dialogs
.vcTXEPrctDtPageLayout 2532	Text des <b>Seite einrichten</b> -Dialogs: <b>Zeile.Spalte Spalte.Zeile Seite/Anzahl</b>
.vcTXEPrctDtPageNumberingItems 2582	Text des <b>Seite einrichten</b> -Dialogs: <b>Seiten&amp;nummerierung</b>
.vcTXEPrctDtPageNumbers 2518	Text des <b>Seite einrichten</b> -Dialogs: <b>Seiten mit Leerraum auff&amp;üllen</b>
.vcTXEPrctDtPagePadding 2585	Fenstertitel des Dialogs <b>Druckvorschau</b>
.vcTXEPrctDtPagePreview 2533	Text des <b>Seite einrichten</b> -Dialogs: <b>Maximale Höhe</b>
.vcTXEPrctDtPagesMaxHeight 2511	Text des <b>Seite einrichten</b> -Dialogs: <b>Maximale Breite</b>
.vcTXEPrctDtPagesMaxWidth 2510	Text des <b>Seite einrichten</b> -Dialogs: <b>%</b>
.vcTXEPrctDtPercent 2509	Text des <b>Seite einrichten</b> -Dialogs: <b>&amp;Druckdatum</b>
.vcTXEPrctDtPrintDate 2564	Text im Druck-Info-Fenster: <b>Seite %1 von %2 wird gedruckt auf</b>
.vcTXEPrctDtPrintingPage 2556	Text des <b>Seite einrichten</b> -Dialogs: <b>Zoomfaktor</b>
.vcTXEPrctDtReduceExpand 2507	Text des <b>Seite einrichten</b> -Dialogs: <b>Tabelle wiederholen</b>
.vcTXEPrctDtRepeatTable 2565	Text des <b>Seite einrichten</b> -Dialogs: <b>Rechts</b>
.vcTXEPrctDtRight 2522	Text des <b>Seite einrichten</b> -Dialogs: <b>Skalierung</b>
.vcTXEPrctDtScaling 2527	Text des <b>Seite einrichten</b> -Dialogs: <b>&amp;Modus:</b>
.vcTXEPrctDtScalingMode 2578	Statuszeilentext des Druckvorschau-Dialogs: <b>%1 Seiten in %2 Zeilen und %3 Spalten</b>
.vcTXEPrctDtStatusBarCurrentValues 2586	

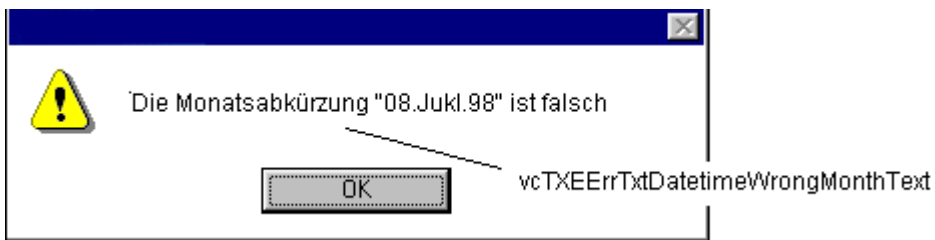
	.vcTXEPrctDtStatusBarSelectedPage 2587	<b>Statuszeilentext</b> des Druckvorschau-Dialogs: <b>Seite %1</b> selektiert (in Zeile %2, Spalte %3)
	.vcTXEPrctDtSuppressEmptyPages 2517	Text des <b>Seite einrichten</b> -Dialogs: <b>Leerseiten unterdrücken</b>
	.vcTXEPrctDtTableColumnRange 2575	Text des <b>Seite einrichten</b> -Dialogs: <b>Tabellenspalten (1-5;7)</b>
	.vcTXEPrctDtTop 2519	Text des <b>Seite einrichten</b> -Dialogs: <b>Oben</b>
	.vcTXEPrctDtZoomFactor 2579	Text des <b>Seite einrichten</b> -Dialogs: <b>&amp;Zoomfaktor:</b>
	.vcTXEPrctMtAdjustBottomAndTopMargin 2437	Meldungstext: <b>Der untere Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.</b> <b>\\nAußerdem wird der obere Rand auf %2 cm reduziert.</b>
	.vcTXEPrctMtAdjustLeftAndRightMargin 2434	Meldungstext: <b>Der linke Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.</b> <b>\\nAußerdem wird der rechte Rand auf %2 cm reduziert.</b>
	.vcTXEPrctMtAdjustRightAndLeftMargin 2435	Meldungstext: <b>Der rechte Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.</b> <b>\\nAußerdem wird der linke Rand auf %2 cm reduziert.</b>
	.vcTXEPrctMtAdjustTopAndBottomMargin 2436	Meldungstext: <b>Der obere Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.</b> <b>\\nAußerdem wird der untere Rand auf %2 cm reduziert.</b>
	.vcTXEPrctMtBottomMargin 2409	Meldungstext: <b>Unterer Rand ...</b>
	.vcTXEPrctMtIncompatibleVcVersion 2414	Meldungstext: <b>VcVersion inkompatibel</b>
	.vcTXEPrctMtLeftMargin 2406	<b>Der linke Rand ist außerhalb des Wertebereichs und wird deshalb auf %s cm reduziert.</b>
	.vcTXEPrctMtPrinterNotInstalled 2411	Meldungstext: <b>Kein Drucker installiert</b>
	.vcTXEPrctMtPrintingNotPossible 2402	Meldungstext: <b>Drucken z. Zt nicht möglich</b>
	.vcTXEPrctMtRightMargin 2408	Meldungstext: <b>Der rechte Rand ist außerhalb des Wertebereichs und wird deshalb auf %s cm reduziert.</b>
	.vcTXEPrctMtSelectPaperSize 2413	Meldungstext: <b>Gewählte Blattgröße zu klein</b>
	.vcTXEPrctMtTopMargin 2407	Meldungstext <b>Der obere Rand ist außerhalb des Wertebereichs und wird deshalb auf %s cm reduziert.</b>
	.vcTXEPrctMtValueOutOfRange 2404	Meldungstext: <b>Außerhalb des Wertebereichs %1 bis %2</b>
	.vcTXEPrctMtWillBeAdjustedTo 2410	Meldungstext: <b>Wird korrigiert auf</b>
	.vcTXERelTypeLongFF 3001	Text im Dialog <b>Verbindungen bearbeiten</b> : <b>: Ende-Ende (FF)</b>
	.vcTXERelTypeLongFS 3000	Text im Dialog <b>Verbindungen bearbeiten</b> : <b>Ende-Anfang (FS)</b>
	.vcTXERelTypeLongSF 3003	Text im Dialog <b>Verbindungen bearbeiten</b> : <b>Anfang-Ende (SF)</b>
	.vcTXERelTypeLongSS 3002	Text im Dialog <b>Verbindungen bearbeiten</b> : <b>Anfang-Anfang (SS)</b>
⇒ textEntry	System.String	Text, der den Standardtext ersetzen soll
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	

```
.vcRetStatDefault 2
.vcRetStatFalse 0
.vcRetStatNoPopup 4
.vcRetStatOK 1
```

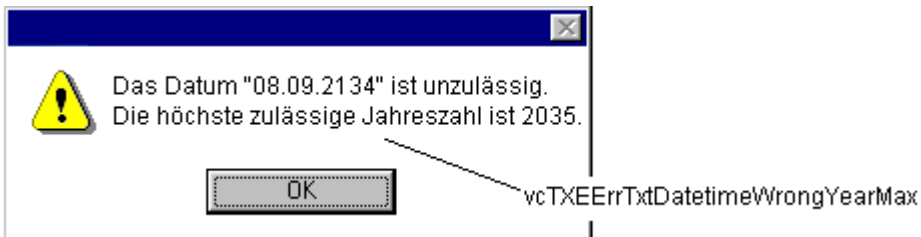
Das Default-Verhalten wird nicht verändert.  
 Das Default-Verhalten wird nicht durchgeführt.  
 Das Erscheinen des Kontextmenüs wird unterdrückt.  
 Das Default-Verhalten wird durchgeführt.



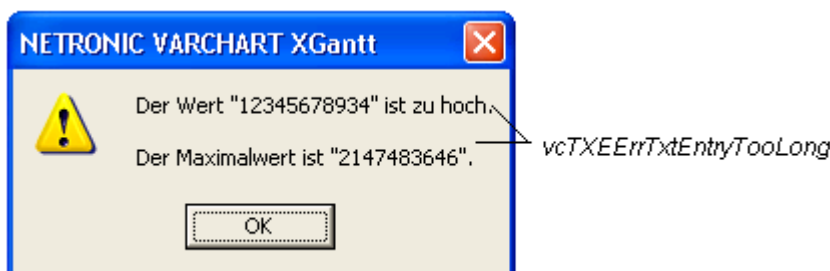
Konstanten der Fehlermeldung **Syntaxfehler im Datum**



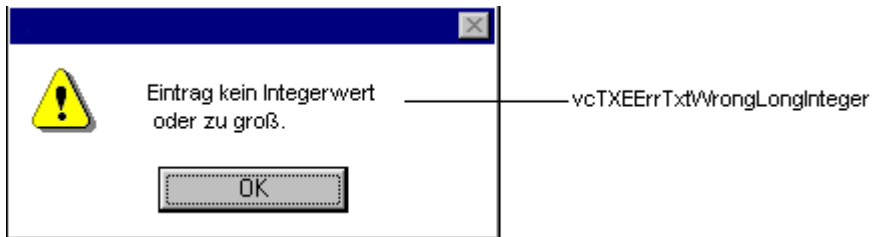
Konstanten der Fehlermeldung **Fehler im Datum, Monat falsch**



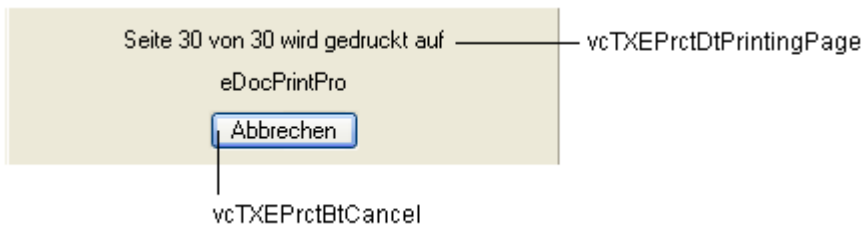
Konstante der Fehlermeldung **Fehler im Datum, Jahr zu groß**



Konstanten der Fehlermeldung **Wert zu groß**



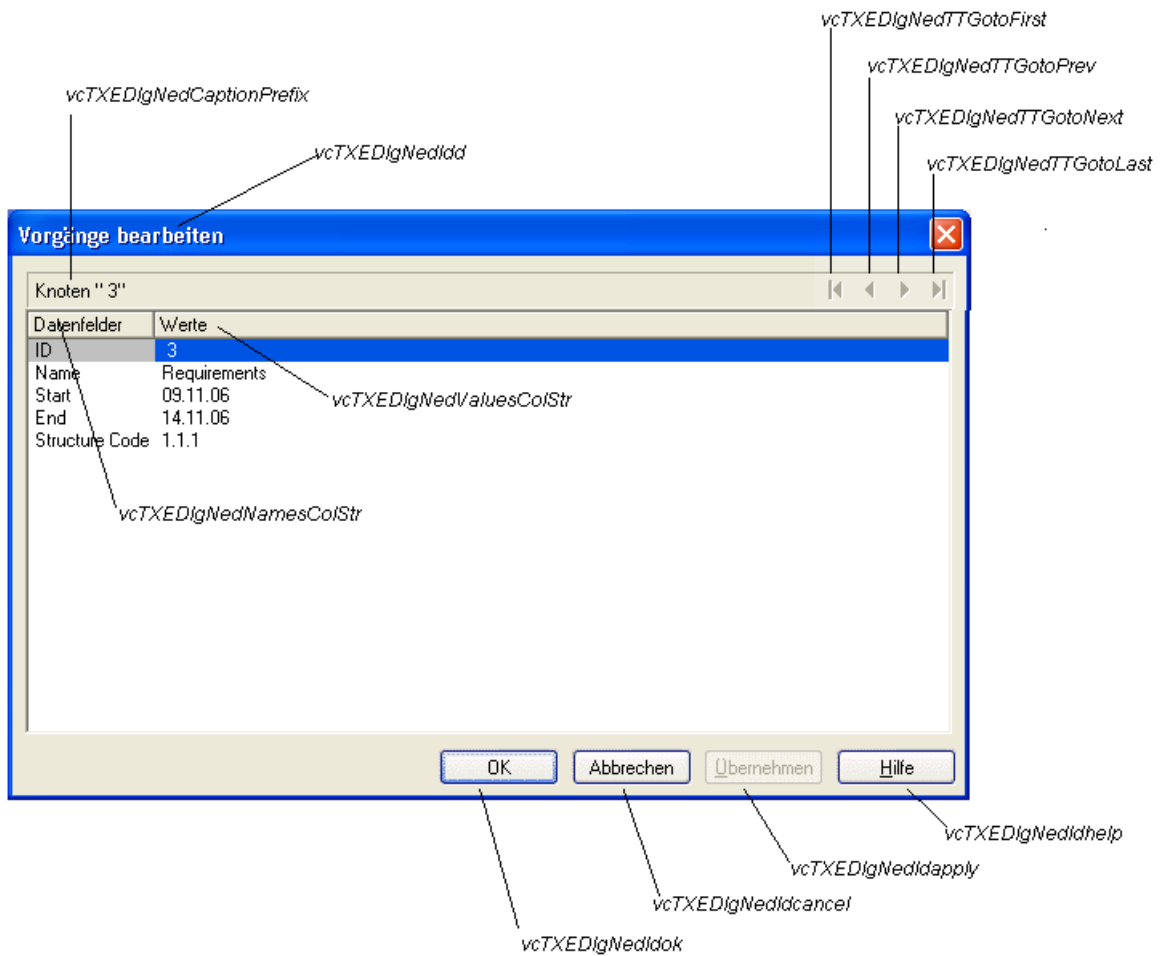
Konstanten der Fehlermeldung **Wert kein Integer**



Konstanten der Infobox **Drucken**



Konstanten der Infobox **Auflösung des Zeitskalen-Abschnitts verändern**



*Konstanten der Dialoge **Vorgänge bearbeiten** und **Verbindung bearbeiten**, hier am Beispiel des Dialogs **Vorgänge bearbeiten** dargestellt*

The 'Seite einrichten' dialog box is divided into several sections:

- Skalierung:** Includes radio buttons for 'Verkleinern / Vergrößern' (100%) and 'Anpassen' (1). The 'Anpassen' section has input fields for 'Seite(n) max. breit' (1) and 'Seite(n) max. hoch' (1).
- Seitenaufteilung:** Includes checkboxes for 'Rahmen außen', 'Knoten nicht durchtrennen', 'Leerseiten unterdrücken', 'Schnittmarkierungen', 'Seitennumerierung', 'Zusatztext', and 'Druckdatum'. A text input field is also present.
- Seitenränder:** Includes input fields for 'Oben', 'Unten', 'Links', and 'Rechts', each with a unit 'cm'.
- Ausgabe:** Includes radio buttons for 'Farbdruck', 'Graustufendruck', and 'Schwarzweißdruck'. A grid of 12 small circular icons is used for alignment and scaling.
- Buttons:** 'OK', 'Abbrechen', and 'Übernehmen'.

API constants for the dialog elements:

- vcTXEPrctDtPageLayout
- vcTXEPrctDtScaling
- vcTXEPrctDtReduceExpand
- vcTXEPrctDtPercent
- vcTXEPrctDtFitToPage
- vcTXEPrctDtPagesMaxHeight
- vcTXEPrctDtPagesMaxWidth
- vcTXEPrctDtOptions
- vcTXEPrctDtFrameOutside
- vcTXEPrctDtSinglePagesNet
- vcTXEPrctDtSuppressEmptyPages
- vcTXEPrctDtAddCuttingMarks
- vcTXEPrctDtPageNumbers
- vcTXEPrctDtPageDescription
- vcTXEPrctDtPrintDate
- vcTXEPrctDtMargins
- vcTXEPrctDtTop, vcTXEPrctDtBottom
- vcTXEPrctDtCm
- vcTXEPrctDtLeft, vcTXEPrctDtRight
- vcTXEPrctDtCm
- vcTXEPrctDtAlignment
- vcTXEPrctBtOk
- vcTXEPrctBtCancel
- vcTXEPrctBtApply
- vcTXEPrctDtColorPrint
- vcTXEPrctDtGrayShadesPrint
- vcTXEPrctDtBlackAndWhitePrint
- vcTXEPrctDtOutput

**Konstanten des Dialogs *Seite einrichten***

The toolbar for 'Druckvorschau' in 'Übersichtsmodus' includes the following buttons and their API constants:

- Schließen
- Navigation arrows (left and right)
- Übersicht (vcTXEPrctBtAll)
- Eingassen in eine Seite
- Auto (dropdown)
- Seite einrichten...
- Drucker einrichten...
- Drucken...

**Druckvorschau im *Übersichtsmodus***

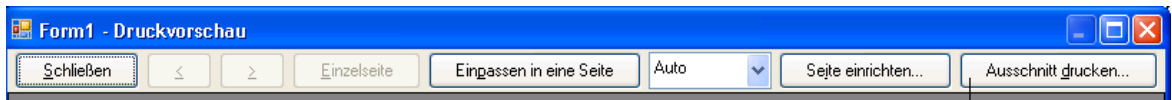
The toolbar for 'Druckvorschau' in 'Einzelansichtsmodus' includes the following buttons and their API constants:

- Schließen
- Navigation arrows (left and right)
- Einzelseite
- Eingassen in eine Seite
- Auto (dropdown)
- Seite einrichten...
- Drucker einrichten...
- Drucken...

API constants for the buttons:

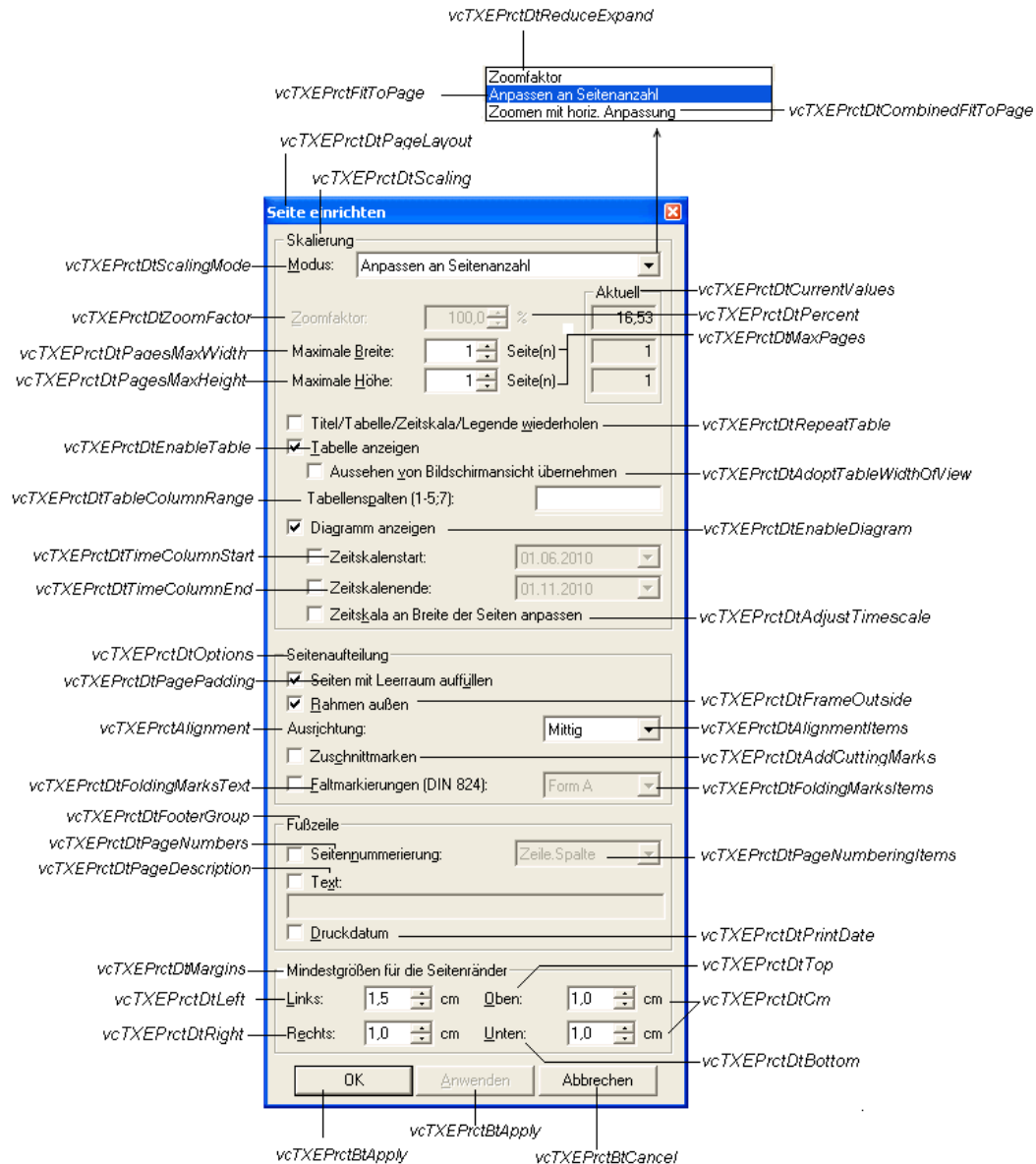
- vcTXEPrctDtPagePreview
- vcTXEPrctBtClose
- vcTXEPrctBtSingle
- vcTXEPrctBtFitToPage
- vcTXEPrctBtPrinterSetup
- vcTXEPrctBtPrevious
- vcTXEPrctBtNext
- vcTXEPrctBtPageLayout
- vcTXEPrctBtPrint
- vcTXEPrctBtPreviewZoomFactorItems

**Konstanten der Tastenbeschriftungen in der *Druckvorschau im Einzelansichtsmodus***



vcTXEPrctBtZoomPrint

**Konstanten der Tastenbeschriftungen in der Druckvorschau im Einzelansichtsmodus bei interaktiv ausgewählten Ausschnitt**



**Konstanten des Dialogs Seite einrichten**

Seite 1 selektiert (in Zeile 1, Spalte 1)	3 Seiten in 1 Zeilen und 3 Spalten
vcTXEPrctDtStatusBarSelectedPage	vcTXEPrctDtStatusBarCurrentValues

Konstanten der Statuszeile im Dialog **Druckvorschau**

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcTextEntrySupplying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcTextEntrySupplyingEventArgs) Handles VcNet1.VcTextEntrySupplying
    Select Case e.ControlIndex
        Case VcTextEntryIndex.vcTXEPrctBtNext
            e.Text = "Next page"
        Case VcTextEntryIndex.vcTXEPrctBtPrevious
            e.Text = "Previous page"
    End Select
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcTextEntrySupplying(object sender,
NETRONIC.XNet.VcTextEntrySupplyingEventArgs e)
{
    switch (e.ControlIndex)
    {
        case VcTextEntryIndex.vcTXEPrctBtNext:
            e.Text = "Next page";
            break;
        case VcTextEntryIndex.vcTXEPrctBtPrevious:
            e.Text = "Previous page";
            break;
    }
}
```

## VcToolTipTextSupplying

Ereignis von VcNet

Dieses Ereignis tritt nur auf, wenn Sie die Eigenschaft **ToolTipTextSupplyingEventEnabled** auf **True** gesetzt haben. Das Ereignis tritt auf, sobald der Cursor auf ein VcNet-Objekt bewegt wird. Es liefert Informationen über das Objekt, den Objekttyp und die Koordinaten des Cursors. Sie können mit Hilfe dieses Ereignisses die vorgegebenen Texte durch eigene Texte ersetzen, z. B. um sie in unterschiedliche Sprachen zu übersetzen. Durch Setzen des ReturnStatus auf **vcRetStatFalse** oder Leerlassen des Textstrings "" können Sie den Tooltip an dieser Stelle unterdrücken.

	Datentyp	Beschreibung
<b>Eigenschaften:</b>		
⇒ hitObject	VcObject	Objekt
⇒ hitObjectType	VcObjectType	Objekttyp
	<b>Mögliche Werte:</b>	
	.vcObjTypeBox 15	Objekttyp <b>Box</b>
	.vcObjTypeGroup 7	Objekttyp <b>Gruppe</b>



	.vcObjTypeLinkCollection 3	Objekttyp <b>LinkCollection</b>
	.vcObjTypeNode 2	Objekttyp <b>Knoten</b>
	.vcObjTypeNone 0	kein Objekt
⇒ x	System.Int32	X-Koordinate
⇒ y	System.Int32	Y-Koordinate
⇒ toolTipText	System.String	Anzuzeigender Text, ASP-Editionen: unbeschränkte Länge Übrige Editionen: maximal 1024 Zeichen lang
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	<b>Mögliche Werte:</b>	
	.vcRetStatDefault 2	Das Default-Verhalten wird nicht verändert.
	.vcRetStatFalse 0	Das Default-Verhalten wird nicht durchgeführt.
	.vcRetStatNoPopup 4	Das Erscheinen des Kontextmenüs wird unterdrückt.
	.vcRetStatOK 1	Das Default-Verhalten wird durchgeführt.

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcToolTipTextSupplying(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcToolTipTextSupplyingEventArgs) Handles
VcNet1.VcToolTipTextSupplying
```

```
    Dim node As VcNode
    If Convert.ToString(e.HitObject) = "VcNetLib.VcNode" Then
        node = DirectCast(e.HitObject, VcNode)
        Select Case e.HitObjectType
            Case VcObjectType.vcObjTypeNodeInDiagram
                e.Text = Convert.ToString(node.DataField(1))
            Case VcObjectType.vcObjTypeNodeInTable
                e.Text = Convert.ToString(node.DataField(1))
        End Select
    End If
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcToolTipTextSupplying(object sender,
VcNetLib.VcToolTipTextSupplyingEventArgs e)
{
    VcNode node;
    if (e.HitObject.ToString() == "NETRONIC.XNet.VcNode")
    {
        node = (VcNode)e.HitObject;
        switch(e.HitObjectType)
        {
            case VcObjectType.vcObjTypeNodeInDiagram:
                e.Text = Convert.ToString(node.get_DataField(1));
                break;
            case VcObjectType.vcObjTypeNodeInTable:
                e.Text = Convert.ToString(node.get_DataField(1));
                break;
        }
    }
}
```

## VcWorldViewClosed

Ereignis von VcNet

Dieses Ereignis wird aufgerufen, wenn das Popup-Fenster der Komplettansicht geschlossen wird.

	Datentyp	Beschreibung
<b>Eigenschaften:</b> ↔ (no parameter)		

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcWorldViewClosed(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcWorldViewClosedEventArgs) Handles VcNet1.VcWorldViewClosed
    MsgBox("Do you want to close the worldview window?", MsgBoxStyle.OKCancel)
End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcWorldViewClosed(object sender,
NETRONIC.XNet.VcWorldViewClosedEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Do you want to close the worldview
window?", "Closing worldview window", MessageBoxButtons.OKCancel);
}
```

## VcZoomFactorModified

Ereignis von VcNet

Dieses Ereignis tritt ein, wenn der Anwender in der Komplettansicht (WorldView) die Größe des Rechtecks verändert hat oder markierte Objekte gezoomt hat. Sie können stufenlos zoomen, indem Sie bei gedrückter Strg-Taste das Mausrad drehen. In bestimmten Schritten können Sie zoomen, indem Sie bei gedrückter Strg-Taste die Plus- bzw. Minus-Tasten des Ziffernblocks der Tastatur drücken.

	Datentyp	Beschreibung
<b>Eigenschaften:</b> ↔ (no parameter)		

### Code-Beispiel VB.NET

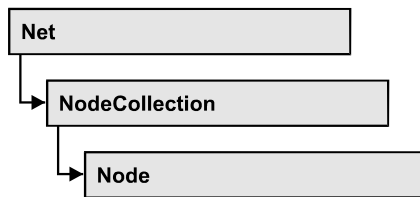
```
Private Sub VcNet1_VcZoomFactorModified(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcZoomFactorModifiedEventArgs) Handles VcNet1.VcZoomFactorModified
    MsgBox("Zoomfactor: " + VcNet1.ZoomFactor)
End Sub
```

## 674 API Referenz: VcNet

### Code-Beispiel C#

```
private void vcNet1_VcZoomFactorModified(object sender,
NETRONIC.XNet.VcZoomFactorModifiedEventArgs e)
{
    MessageBox.Show("Zoomfactor: " + vcNet1.ZoomFactor.ToString());
}
```

## 7.38 VcNode



Knoten sind Grundelemente eines Netzdiagramms. Sie lassen sich über Verbindungen zu einer Struktur verknüpfen. Das Aussehen eines Knotens wird über diejenigen NodeAppearance-Objekte bestimmt, deren Filter auf den Knoten zutreffen. Erzeugt werden Knoten über die Methode **VcNet.InsertNodeRecord** oder interaktiv.

### Eigenschaften

- AllData
- DataField
- ID
- IncomingLinks
- Marked
- OutgoingLinks
- OutgoingLinks

### Methoden

- DataRecord
- Delete
- RelatedDataRecord
- Update

---

## Eigenschaften

### AllData

**Eigenschaft von VcNode**

Mit dieser Eigenschaft können alle Daten auf einmal für den Knoten gesetzt oder erfragt werden. Beim Setzen ist ein CSV-String (Semikolon als Trennzeichen) oder ein Object erlaubt, der in einem Feld (Array) alle Datenfelder des Knotens erhält. Beim Erfragen wird ein String zurückgegeben. (Siehe auch **InsertNodeRecord**.)

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Alle Daten des Datensatzes

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcNet1.VcNodeModifying
    Dim allDataOfNode As String
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse

    allDataOfNode = e.Node.AllData
    MsgBox(allDataOfNode)
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcNodeModifying(object sender,
NETRONIC.XGantt.VcNodeModifyingEventArgs e)
{
    e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
    string allDataOfNode = e.Node.AllData.ToString();
    MessageBox.Show(allDataOfNode);
}
```

**DataField****Eigenschaft von VcNode**

Mit dieser Eigenschaft können Sie einem Datenfeld des Knotens einen Wert zuweisen oder einen gesetzten Wert erfragen. Wenn ein Knoten durch diese Methode einen neuen Wert erhalten hat, muss anschließend die Methode **Update** aufgerufen werden.

Die Eigenschaft **DataField** ist eine indizierte Eigenschaft, die in C# über die beiden Methoden **set\_DataField (index, pvn)** und **get\_DataField (index)** angesprochen wird.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Datenfeldes
<b>Eigenschaftswert</b>	System.Object	Inhalt des Datenfeldes

**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking
    If MsgBox("Delete node: " + e.Node.DataField(0), MsgBoxStyle.YesNo, "Delete
node") = MsgBoxResult.Yes Then
        e.Node.Delete()
    End If
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Delete node: " +
e.Node.get_DataField(0), "Deleting node", MessageBoxButtons.YesNo);
    if (retVal == DialogResult.Yes)
        e.Node.Delete();
    else
        e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

**ID****Nur-Lese-Eigenschaft von VcNode**

Mit dieser Eigenschaft können Sie die ID eines Knotens erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Knoten-ID

**Code-Beispiel VB.NET**

```
VcNode node = VcNet1.NodeCollection.FirstNode()
MsgBox (node.ID)
```

**Code-Beispiel C#**

```
VcNode node = vcNet1.NodeCollection.FirstNode();
MessageBox.Show (node.ID)
```

**IncomingLinks****Nur-Lese-Eigenschaft von VcNode**

Mit dieser Eigenschaft haben Sie Zugriff auf alle Verbindungen, die in einen Knoten hineinführen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkCollection	LinkCollection-Objekt

**Code-Beispiel VB.NET**

```

Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking
    Dim incomingLinks As VcLinkCollection
    Dim link As VcLink
    Dim predecessorNode As VcNode

    incomingLinks = e.Node.IncomingLinks
    For Each link In incomingLinks
        predecessorNode = link.PredecessorNode
        predecessorNode.Marked = True
    Next
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub

```

**Code-Beispiel C#**

```

private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    VcLinkCollection incomingLinks = e.Node.IncomingLinks;
    VcNode predecessorNode;
    foreach (VcLink link in incomingLinks)
    {
        predecessorNode = link.PredecessorNode;
        predecessorNode.Marked = true;
    }
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}

```

**Marked****Eigenschaft von VcNode**

Mit dieser Eigenschaft können Sie festlegen oder abfragen, ob ein Knoten markiert ist. Die gesetzte Markierung ist nur dann sichtbar, wenn auf der Eigenschaftenseite **Knoten** unter **Knotenmarkierung** nicht **Ohne** ausgewählt ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Knoten markiert/nicht markiert

**Code-Beispiel VB.NET**

```

Dim nodeCltn As VcNodeCollection
Dim node As VcNode
Dim predecessor As VcNode
Dim linkCltn As VcLinkCollection
Dim link As VcLink

nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll)

For Each node In nodeCltn
    linkCltn = node.IncomingLinks
    For Each link In linkCltn
        predecessor = link.PredecessorNode
        predecessor.Marked = True
    Next
Next

```

**Code-Beispiel C#**

```

VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
VcNode predecessorNode;
VcLinkCollection linkCltn;
foreach (VcNode node in nodeCltn)
{
    linkCltn = node.IncomingLinks;
    foreach (VcLink link in linkCltn)
    {
        predecessorNode = link.PredecessorNode;
        predecessorNode.Marked = true;
    }
}

```

**OutgoingLinks****Nur-Lese-Eigenschaft von VcNode**

Mit dieser Eigenschaft haben Sie Zugriff auf alle Verbindungen, die von einem Knoten ausgehen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkCollection	LinkCollection-Objekt

**OutgoingLinks****Nur-Lese-Eigenschaft von VcNode**

Mit dieser Eigenschaft haben Sie Zugriff auf alle Verbindungen, die von einem Knoten ausgehen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcLinkCollection	LinkCollection-Objekt



**Code-Beispiel VB.NET**

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking

    Dim outgoingLinks As VcLinkCollection
    Dim link As VcLink
    Dim successorNode As VcNode

    outgoingLinks = e.Node.OutgoingLinks
    For Each link In outgoingLinks
        successorNode = link.SuccessorNode
        successorNode.Marked = True
    Next
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

**Code-Beispiel C#**

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
{
    VcLinkCollection outgoingLinks = e.Node.OutgoingLinks;
    VcNode successorNode;
    foreach (VcLink link in outgoingLinks)
    {
        successorNode = link.SuccessorNode;
        successorNode.Marked = true;
    }
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
}
```

---

## Methoden

### DataRecord

**Methode von VcNode**

Mit dieser Eigenschaft können Sie den Knoten als Datensatzobjekt erfragen. Über die Eigenschaften des Datensatzobjektes haben Sie auch Zugriff auf die entsprechende Datentabelle und Tabellenauflistung.

	Datentyp	Beschreibung
Rückgabewert	VcDataRecord	Zurückgegebener Datensatz

### Delete

**Methode von VcNode**

Mit dieser Methode können Sie einen Knoten löschen.

	Datentyp	Beschreibung
<b>Rückgabewert</b>	System.Boolean	Knoten erfolgreich/nicht erfolgreich gelöscht

### Code-Beispiel VB.NET

```
Private Sub VcNet1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XNet.VcNodeClickingEventArgs) Handles VcNet1.VcNodeRightClicking

    If MsgBox("Delete node: " + e.Node.DataField(0), MsgBoxStyle.YesNo, "Delete
node") = MsgBoxResult.Yes Then
        e.Node.Delete()
        e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
    End If

End Sub
```

### Code-Beispiel C#

```
private void vcNet1_VcNodeRightClicking(object sender,
NETRONIC.XNet.VcNodeClickingEventArgs e)
{
    DialogResult retVal = MessageBox.Show("Delete node: " +
e.Node.get_DataField(0), "Deleting node", MessageBoxButtons.YesNo);
    if (retVal == DialogResult.Yes)
    {
        e.Node.Delete();
        e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
    }
}
```

## RelatedDataRecord

### Methode von VcNode

Mit dieser Eigenschaft können Sie einen Datensatz aus einer verknüpften Tabelle erfragen, der dem Datensatz der Knotentabellensuche zugeordnet ist. Der im Parameter übergebene Index bezeichnet das Feld im Datensatz, in dem der Schlüssel des zugeordneten Datensatzes steht.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Datenfeldes, das den Schlüssel enthält
<b>Rückgabewert</b>	VcDataRecord	Zurückgegebener zugeordneter Datensatz

## Update

### Methode von VcNode

Nachdem Sie ein oder mehrere Datenfelder eines Knotens mit der Eigenschaft **DataField** verändert haben, aktualisieren Sie die Grafik mit **Update**.

## 682 API Referenz: VcNode

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Knoten erfolgreich/nicht erfolgreich aktualisiert

### Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

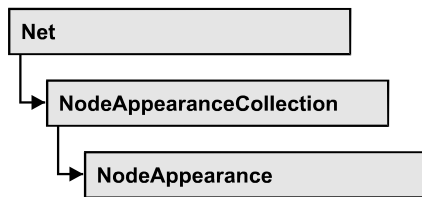
nodeCltn = VcNet1.NodeCollection
node = nodeCltn.FirstNode

node.DataField(12) = "Group A"
node.Update()
```

### Code-Beispiel C#

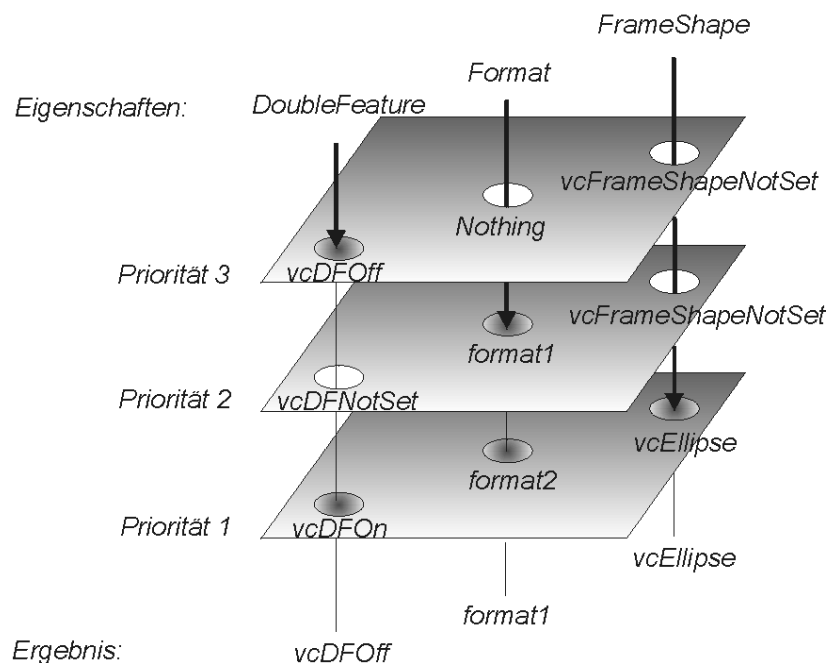
```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
node.set_DataField(12, "Group A");
node.Update();
```

## 7.39 VcNodeAppearance



Ein NodeAppearance-Objekt bestimmt das Aussehen aller Knoten, deren Daten die dem NodeAppearance-Objekt zugeordneten Filterbedingungen erfüllen. Verschiedene NodeAppearance-Objekte können im Dialogfeld **Knotenaussehen verwalten**, das Sie über die Eigenschaftenseite **Knoten** erreichen, voreingestellt werden.

Die Skizze zeigt, wie sich die Eigenschaften von NodeAppearance-Objekten auf das Aussehen eines Knotens auswirken. Die auf den Knoten zutreffenden NodeAppearances sind nach Priorität absteigend dargestellt. Nicht gesetzte Eigenschaften bei NodeAppearance-Objekten führen dazu, daß die Eigenschaft des nächsttieferen NodeAppearance-Objektes übernommen wird.



### Eigenschaften

- BackgroundColor
- LineColorDataFieldIndex
- BackgroundColorMapName
- DoubleFeature
- FilterName
- FormatName

- FrameAroundFieldsVisible
- FrameShape
- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- Pattern
- PatternColor
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- PileEffect
- Shadow
- ShadowColor
- Specification
- StrikeThrough
- StrikeThroughColor
- ThreeDEffect
- VisibleInLegend

### Methoden

- PutInOrderAfter

---

## Eigenschaften

### BackgroundColor

Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie die Hintergrundfarbe eines Knotens einstellen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt.

Bei **-1** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color RGB {0...255},{0...255},{0...255}	ARGB Farbwerte

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance

nodeAppearance.BackColor = RGB(100, 100, 100)
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance;
nodeAppearance.BackColor = RGB(100, 100, 100);
```

## BackgroundColorDataFieldIndex

Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der Verbindung mit der Eigenschaft **BackColorMapName** benötigt wird. Wenn Sie hier **-1** angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Datenfeldindex

## BackColorMapName

Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Namen einer Farbzusordnungstabelle für die Hintergrundfarbe setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **BackColorDataFieldIndex -1** angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuoordnungstabelle

## DoubleFeature

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie eine doppelte Umrahmung eines Knotens einstellen oder erfragen. Bei **vcDFNotSet** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **vcDFNotSet** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
Eigenschaftswert	VcAppearanceDoubleFeature  <b>Mögliche Werte:</b> .vcDFNotSet -1 .vcDFOff 0 .vcDFOn 1	Typen von Doppel-Linien  Schalter für <b>DoubleFeature nicht gesetzt</b> Schalter für <b>DoubleFeature aus</b> Schalter für <b>DoubleFeature an</b>

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.DoubleFrame = VcAppearanceDoubleFrame.vcDFOn
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.DoubleFrame = VcAppearanceDoubleFrame.vcDFOn;
```

## FilterName

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Filter des NodeAppearance-Objekts setzen oder erfragen. Es gibt Sonderfilter, die unveränderlich sind:

- <ALWAYS>: gilt immer (beim Standard-Aussehen immer gesetzt)
- <NEVER>: gilt niemals
- <INTERFACE-COLLAPSED>: gilt für Teildiagramm-Anschlussknoten

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Filtername

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim filterOfNodeApp As String

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
filterOfNodeApp = nodeAppearance.filtername
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("Blue");
string filterOfNodeApp = nodeAppearance.FilterName;
```

**FormatName****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie das Format des NodeAppearance-Objekts setzen oder erfragen. Bei **Nothing** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **Nothing** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name eines Knotenformat-Objekts oder leere Zeichenkette

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
MsgBox (nodeAppearance.FormatName)
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
MessageBox.Show (nodeAppearance.FormatName);
```



## FrameAroundFieldsVisible

Nur-Lese-Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft kann festgelegt werden, ob der Rahmen um die innenliegenden Felder sichtbar ist oder nicht. Die Außenrandlinie der Form ist davon nicht betroffen, daher wirkt sich diese Eigenschaft bei den möglichen Rahmenformen unterschiedlich aus und hat z.B. beim Typ **vcRectangle** keine Auswirkung.

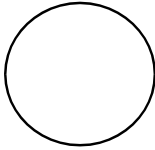

Diese Eigenschaft kann auch im Dialog **Knotenaussehen bearbeiten** gesetzt werden.





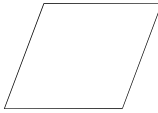


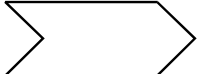

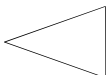

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcAppearanceFrameAroundFieldsVisible  <b>Mögliche Werte:</b> .vcFFVNotSet -1 .vcFFVOff 0 .vcFFVOn 1	Rahmen um Feld <b>Standardwert:</b> -1  Feldumrandung nicht gesetzt Schalter für Feldumrandung aus Schalter für <b>Feldumrandung</b> an

## FrameShape

Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie dem NodeAppearance-Objekt die Rahmenform zuweisen oder erfragen. Bei **vcFrameShapeNotSet** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **vcFrameShapeNotSet** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	AppearanceFrameShapeEnum  <b>Mögliche Werte:</b> .vcCircle 11   .vcEllipse 12	Rahmenform  Rahmenform kreisförmig   Rahmenform elliptisch 

.vcFile 19	Rahmenform horizontaler Zylinder 
.vcFrameShapeNotSet -1 .vcLeftArrow 17	Rahmenform nicht gesetzt Rahmenform pfeilförmig, nach links zeigend 
.vcListing 20	Rahmenform Dokument 
.vcNoFrameShape 1 .vcOval 4	keine Rahmenform Rahmenform oval 
.vcParallelogram 9	Rahmenform Parallelogramm 
.vcPointed 7	Rahmenform an den senkrechten Kanten spitz ausgezogen 
.vcRectangle 2	Rahmenform rechteckig 
.vcRightArrow 18	Rahmenform pfeilförmig, nach rechts zeigend 
.vcRounded 3	Rahmenform rechteckig gerundet 
.vcTriangleLeft 10	Rahmenform dreieckig, Spitze links 
.vcTriangleUp 13	Rahmenform dreieckig, Spitze oben 

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
```

```
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcEllipse
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcEllipse;
```

**LegendText****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie einem Knotenaussehen einen Text zuweisen oder erfragen, der in der Legende für das jeweilige Knotenaussehen angezeigt wird. Steht hier "", dann wird der Inhalt der Eigenschaft **Name** angezeigt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Legendentext des Knotenaussehens <b>Standardwert:</b> " " (content of the property <b>Name</b> )

**LineColor****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie dem NodeAppearance-Objekt eine Linienfarbe zuweisen oder erfragen. Bei **-1** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color RGB {(0...255),(0...255),(0...255)}	RGB-Farbwerte oder <b>-1</b>

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.LineColor = Color.LightBlue
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.LineColor = Color.LightBlue;
```

## LineColorDataFieldIndex

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Farbzordnungstabelle in der Eigenschaft **LineColorMapName** benötigt wird. Wenn Sie hier -1 angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

## LineColorMapName

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Namen einer Farbzordnungstabelle für die Linienfarbe setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **LineColorDataFieldIndex** -1 angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzordnungstabelle

## LineThickness

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie die Linienstärke eines NodeAppearance-Objekts erfragen oder festlegen.

Wenn Sie diese Eigenschaft auf Werte zwischen 1 und 4 setzen, wird damit eine absolute Liniendicke in Pixel definiert, d.h. die Linien haben unabhängig vom Zoomfaktor immer die gleiche feste Linienstärke in Pixeln. Dies wird jedoch aufgrund der besseren Lesbarkeit beim Drucken in eine vom Zoomfaktor abhängige Liniendicke umgewandelt:

Wert	Punkte	mm
1	1/2 Punkt	0,09 mm
2	1 Punkt	0,18 mm
3	3/2 Punkt	0,26 mm

Wert	Punkte	mm
4	2 Punkt	0,35 mm

Ein Punkt ist 1/72 Zoll groß und stellt die Maßeinheit für Schriftgrößen dar.

Wenn Sie diese Eigenschaft auf Werte zwischen 5 und 1.000 setzen, wird damit eine Linienstärke in 1/100 mm definiert, d.h. die Linien bekommen eine tatsächliche Dicke in Pixeln, die abhängig vom Zoomfaktor ist.

Wenn Sie diese Eigenschaft auf **-1** setzen, kommt die gleichnamige Eigenschaft des in der Priorität nächst niedrigeren NodeAppearance-Objekts zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm <b>Standardwert:</b> As defined on property page

#### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.NodeAppearanceByName("Standard")
nodeAppearance.LineThickness = 3
```

#### Code-Beispiel C#











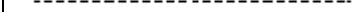





```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("Standard");
nodeAppearance.LineThickness = 3;
```

## LineType

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie dem NodeAppearance-Objekt einen Linientyp zuweisen oder erfragen. Bei **vcNotSet** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen

und dessen Eigenschaftswert nicht mit **vcNotSet** besetzt ist (s. Grafik bei VcNodeAppearance).

Eigenschaftswert	Datentyp	Beschreibung
	VcLineType	Linientyp
	<b>Mögliche Werte:</b>	Linientyp <b>gestrichelt</b>
	.vcDashed 4	Linientyp <b>gestrichelt</b>
	.vcDashed 4	Linientyp <b>gestrichelt-gepunktet</b>
	.vcDashedDotted 5	Linientyp <b>gestrichelt-gepunktet</b>
	.vcDashedDotted 5	Linientyp <b>gepunktet</b>
	.vcDotted 3	Linientyp <b>gepunktet</b>
	.vcDotted 3	Linientyp 0
	.vcLineType0 100	
	.vcLineType1 101	Linientyp 1
	.vcLineType10 110	
	.vcLineType11 111	Linientyp 10
	.vcLineType12 112	
	.vcLineType13 113	Linientyp 11
	.vcLineType14 114	
	.vcLineType15 115	Linientyp 12
	.vcLineType16 116	
	.vcLineType17 117	Linientyp 13
	.vcLineType18 118	
	.vcLineType19 119	Linientyp 14
	.vcLineType2 102	
	.vcLineType3 103	Linientyp 15
	.vcLineType4 104	
	.vcLineType5 105	Linientyp 16
	.vcLineType6 106	
	.vcLineType7 107	Linientyp 17
	.vcLineType8 108	
	.vcLineType9 109	Linientyp 18
	.vcLineType10 110	
	.vcLineType11 111	Linientyp 19
.vcLineType12 112		
.vcLineType13 113	Linientyp 2	
.vcLineType14 114		
.vcLineType15 115	Linientyp 3	
.vcLineType16 116		
.vcLineType17 117	Linientyp 4	
.vcLineType18 118		
.vcLineType19 119	Linientyp 5	
.vcNone 1		
.vcNone 1	Linientyp 6	
.vcNotSet -1	Kein Linientyp zugewiesen	
.vcSolid 2	Kein Linientyp	
.vcSolid 2	Kein Linientyp zugewiesen	
.vcSolid 2	Linientyp <b>durchgezogen</b>	
.vcSolid 2	Linientyp <b>durchgezogen</b>	

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance

nodeAppearance.LineType = vcDotted
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance;
nodeAppearance.LineType = vcDotted;
```

**Name****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie den Namen eines NodeAppearance-Objekts erfragen oder setzen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Knotenaussehens

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
Dim nameNodeApp As String

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
NodeApp = nodeAppearance.Name

nameNodeAppName = nodeAppearance.name
```




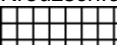

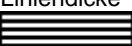
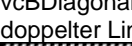

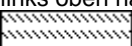
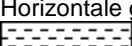
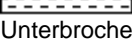
**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
string nameNodeApp = nodeAppearance.Name;
```

**Pattern****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie das Muster des Knotens festlegen oder erfragen. Wenn in der Eigenschaft **PatternMapName** eine Zuordnungstabelle angegeben ist, steuert diese das Muster in Abhängigkeit von den Daten. Bei **-1** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

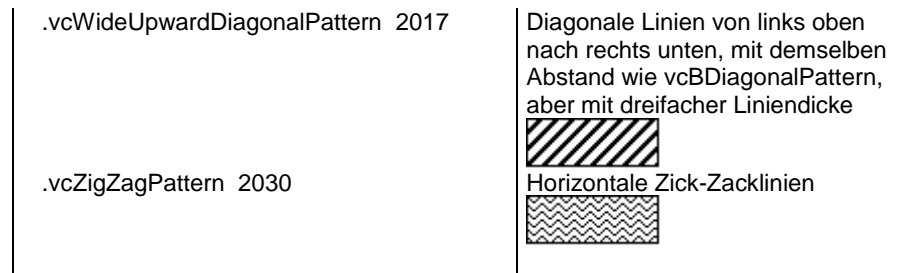
Werte von vc05PercentPattern bis vc90PercentPattern lauten richtigerweise 2001 bis 2011.

Eigenschaftswert	Datentyp	Beschreibung
	VcFillPattern	Mustertyp
	<b>Mögliche Werte:</b> .vc05PercentPattern... vc90PercentPattern 01 - 11	Punkte in Vordergrundfarbe auf Hintergrundfarbe; mit steigender Prozentzahl Vordergrundfarbe immer dichter
		
	.vcAeroGlassPattern 44	Vertikaler Farbverlauf in der Füllmusterfarbe 
	.vcBDiagonalPattern 5	Diagonale Linien von links unten nach rechts oben 
	.vcCrossPattern 6	Kreuzschraffur 
	.vcDarkDownwardDiagonalPattern 2014	Diagonale Linien von links oben nach rechts unten, 50 % näher zusammen als vcFDiagonalPattern und mit doppelter Liniendicke 
	.vcDarkHorizontalPattern 2023	Horizontale Linien mit 50% geringerem Abstand als vcHorizontalPattern und doppelter Liniendicke 
	.vcDarkUpwardDiagonalPattern 2015	Diagonale Linien von links unten nach rechts oben; mit 50%geringerem Abstand als vcBDiagonalPattern und mit doppelter Liniendicke 
	.vcDarkVerticalPattern 2022	Vertikale Linien mit 50% geringerem Abstand als vcVerticalPattern und doppelter Liniendicke 
	.vcDashedDownwardDiagonalPattern 2024	Gestrichelte diagonale Linien von links oben nach rechts unten 
	.vcDashedHorizontalPattern 2026	Horizontale gestrichelte Linien 
.vcDashedUpwardDiagonalPattern 2025	Unterbrochene diagonale Linien von links unten nach rechts oben 	



.vcDashedVerticalPattern 2027	Vertikale gestrichelte Linien
.vcDiagCrossPattern 7	Diagonale Kreuzschraffur, klein
.vcDiagonalBrickPattern 2032	Diagonales Backsteinmuster
.vcDivotPattern 2036	Grassoden-Muster
.vcDottedDiamondPattern 2038	Diagonale Kreuzschraffur aus punktierten Linien
.vcDottedGridPattern 2037	Kreuzschraffur aus punktierten Linien
.vcFDiagonalPattern 4	Diagonale Linien von links oben nach rechts unten
.vcHorizontalBrickPattern 2033	Horizontales Backsteinmuster
.vcHorizontalGradientPattern 52	Horizontaler Farbverlauf
.vcHorizontalPattern 3	Horizontale Linien
.vcLargeCheckerboardPattern 2044	Schachbrettmuster mit doppelt so großen Quadraten wie vcSmallCheckerBoardPattern
.vcLargeConfettiPattern 2029	Konfetti -Muster, groß
.vcLightDownwardDiagonalPattern 2012	Diagonale Linien von links oben nach rechts unten; mit 50% geringerem Abstand als vcB-DiagonalPattern
.vcLightHorizontalPattern 2019	Horizontale Linien mit 50% geringerem Abstand als bei vc-HorizontalPattern
.vcLightUpwardDiagonalPattern 2013	Diagonale Linien von links unten nach rechts oben; 50 % näher zusammen als vcB-DiagonalPattern
.vcLightVerticalPattern 2018	Vertikale Linien mit 50% geringerem Abstand als bei vcVerticalPattern
.vcNarrowHorizontalPattern 2021	Horizontale Linien mit 75% geringerem Abstand als vc-HorizontalPattern

.vcNarrowVerticalPattern 2020	Vertikale Linien mit 75% geringerem Abstand als vcVerticalPattern
.vcNoPattern 1276	Kein Füllmuster
.vcOutlinedDiamondPattern 2045	Diagonale Kreuzschraffur, groß
.vcPlaidPattern 2035	Schottenstoff-Muster
.vcShinglePattern 2039	Diagonales Dachschindel-Muster
.vcSmallCheckerBoardPattern 2043	Schachbrettmuster
.vcSmallConfettiPattern 2028	Konfetti-Muster
.vcSmallGridPattern 2042	Kreuzschraffur mit 50 % geringerem Abstand als vcCrossPattern
.vcSolidDiamondPattern 2046	Schachbrettmuster mit diagonalen Quadraten
.vcSpherePattern 2041	Kugeln schachbrettartig angeordnet
.vcTrellisPattern 2040	Spalier-Muster
.vcVerticalBottomLightedConvexPattern 43	Vertikaler Farbverlauf von dunkel nach hell
.vcVerticalConcavePattern 40	Vertikaler Farbverlauf von dunkel über hell nach dunkel
.vcVerticalConvexPattern 41	Vertikaler Farbverlauf von hell über dunkel nach hell
.vcVerticalGradientPattern 62	Vertikaler Farbverlauf
.vcVerticalPattern 2	Vertikale Linien
.vcVerticalTopLightedConvexPattern 42	Vertikaler Farbverlauf von hell nach dunkel
.vcWavePattern 2031	Horizontales Wellenmuster
.vcWeavePattern 2034	Muster mit verwebten Streifen
.vcWideDownwardDiagonalPattern 2016	Diagonale Linien von links oben nach rechts unten, mit demselben Abstand wie vcFDiagonalPattern, aber mit dreifacher Liniendicke



## PatternColor

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie die Musterfarbe des Knotens festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt.

Bei **-1** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

Wenn in der Eigenschaft **PatternColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Musterfarbe datenabhängig.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Wert {0...255},{0...255},{0...255},{0...255}

## PatternColorDataFieldIndex

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternColorMapName** benötigt wird. Wenn Sie hier **-1** angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

## PatternColorMapName

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Namen einer Farbzordnungstabelle (Typ `vcColorMap`) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzordnungstabelle und ein Datenfeldindex in der Eigenschaft **PatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **PatternColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzordnungstabelle

## PatternDataFieldIndex

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternMapName** benötigt wird. Wenn Sie hier **-1** angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	

## PatternMapName

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den Namen einer Musterzuordnungstabelle (Typ `vcPatternMap`) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn der Name einer Farbzordnungstabelle und ein Datenfeldindex in der Eigenschaft **PatternDataFieldIndex** angegeben sind, wird das Muster des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird das Muster aus der Eigenschaft **Pattern** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle

## PileEffect

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie die Anzahl von Knotenstapeln im Diagramm erfragen oder festlegen. Bei **-1** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl gestapelter Knoten oder <b>-1</b>

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = VcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Piles = 2
```

### Code-Beispiel C#

```
Dim nodeAppearanceCollection As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

Set nodeAppearanceCollection = vcNet1.NodeAppearanceCollection
Set nodeAppearance = nodeAppearanceCollection.FirstNodeAppearance

nodeAppearance.Piles = 2
```

## Shadow

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie dem NodeAppearance-Objekt einen Schatten zuweisen oder erfragen. Bei **vcShNotSet** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **vcShNotSet** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcAppearanceShadow  <b>Mögliche Werte:</b> .vcShNotSet -1 .vcShOff 0 .vcShOn 1	Art der Schattensetzung  Schalter für <b>Schatten nicht gesetzt</b> Schalter für <b>Schatten aus</b> Schalter für <b>Schatten an</b>

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.Shadow = VcAppearanceShadow.vcShOn
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.Shadow = VcAppearanceShadow.vcShOn;
```

**ShadowColor****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie die Schattenfarbe des Knotens festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	ARGB-Wert

**Specification****Nur-Lese-Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie die Spezifikation dieses Knotenaussehens auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Knotenaussehens mit der Methode **VcNodeAppearanceCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Knotenaussehens

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
MsgBox(nodeAppearance.Specification)
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
MessageBox.Show(nodeAppearance.Specification);
```

## StrikeThrough

**Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie das Durchstreichmuster eines NodeAppearance-Objekts setzen oder erfragen. Bei **vcStrikeThroughNotSet** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **vcStrikeThroughNotSet** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
Eigenschaftswert	VcAppearanceStrikeThrough	Durchstreichmuster

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.StrikeThrough = VcAppearanceStrikeThrough.vcBackslashed
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.StrikeThrough = VcAppearanceStrikeThrough.vcBackslashed;
```

## StrikeThroughColor

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie die Farbe des Durchstreichmusters eines NodeAppearance-Objekts setzen oder erfragen. Bei **-1** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **-1** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color RGB {0...255},{0...255},{0...255}	RGB-Farbwerte oder -1

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
nodeAppearance.StrikeThroughColor = Color.LightBlue
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
nodeAppearance.StrikeThroughColor = Color.LightBlue;
```

## ThreeDEffect

### Eigenschaft von VcNodeAppearance

Mit dieser Eigenschaft können Sie den 3D-Effekt für das NodeAppearance-Objekt erfragen oder festlegen. Bei **vc3DNotSet** kommt die gleichnamige Eigenschaft des in der Priorität nächstniedrigeren NodeAppearance-Objektes zum Tragen, dessen Filterbedingungen für den Knoten ebenfalls zutreffen und dessen Eigenschaftswert nicht mit **vc3DNotSet** besetzt ist (s. Grafik bei VcNodeAppearance).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcAppearanceThreeDEffect	Art der 3D-Effekt-Setzung

### Code-Beispiel VB.NET

```
Dim format As VcTableFormat

format = VcNet1.LeftTable.TableFormatCollection.FormatByName("StandardList")
format.ThreeDEffect = True
```



**Code-Beispiel C#**

```
VcTableFormat format =
vcNet1.LeftTable.TableFormatCollection.FormatByName("StandardList");
format.ThreeDEffect = true;
```

**VisibleInLegend****Eigenschaft von VcNodeAppearance**

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das NodeAppearance-Objekt in der Legende sichtbar ist. Diese Eigenschaft kann auch im Dialog **Knotenaussehen verwalten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Knotenaussehen in Legende sichtbar (True)/nicht sichtbar (False) <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.NodeAppearanceByName("Standard")

nodeAppearance.VisibleInLegend = False
```

**Code-Beispiel C#**

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("Standard");
nodeAppearance.VisibleInLegend = false;
```

---

**Methoden****PutInOrderAfter****Methode von VcNodeAppearance**

Mit dieser Methode können Sie dieses Knotenaussehen in der Auflistung aller Knotenaussehen hinter das durch den Namen angegebene setzen. Wenn als Name "" angegeben wird, wird das Knotenaussehen an die erste Stelle gesetzt. Die Reihenfolge der Knotenaussehen in der Auflistung entscheidet darüber, in welcher Reihenfolge sie auf die Knoten angewendet werden.

	Datentyp	Beschreibung
<b>Parameter:</b> refNodeAppearanceName	System.String	Name des Knotenaussehens, hinter das das aktuelle Knotenaussehen gesetzt werden soll

### Code-Beispiel VB.NET

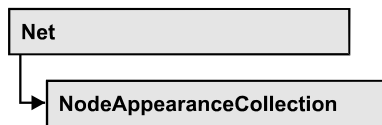
```
Dim nodeAppCltn As VcNodeAppearanceCollection
Dim nodeApp1 As VcNodeAppearance
Dim nodeApp2 As VcNodeAppearance

nodeAppCltn = VcGantt1.NodeAppearanceCollection()
nodeApp1 = nodeAppCltn.Add("nodeApp1")
nodeApp2 = nodeAppCltn.Add("nodeApp2")
nodeApp1.PutInOrderAfter("nodeApp2")
nodeAppCltn.Update()
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppCltn = vcGantt1.NodeAppearanceCollection;
VcNodeAppearance nodeApp1 = nodeAppCltn.Add("nodeApp1");
VcNodeAppearance nodeApp2 = nodeAppCltn.Add("nodeApp2");
nodeApp1.PutInOrderAfter("nodeApp2");
nodeAppCltn.Update();
```

## 7.40 VcNodeAppearanceCollection



In einem Objekt vom Typ `VcNodeAppearanceCollection` sind automatisch alle definierten `NodeAppearance`-Objekte zusammengefasst. Sie haben Zugriff auf ein Knotenaussehen über die Methode **NodeAppearanceByName**. Mit der Eigenschaft **Count** können Sie die Anzahl der `NodeAppearance`-Objekte ermitteln. Über **For Each nodeAppearance In NodeAppearanceCollection** können Sie in einer Schleife auf alle Knotenaussehen zugreifen.

### Eigenschaften

- Count

### Methoden

- Add
- AddBySpecification
- Copy
- FirstNodeAppearance
- GetEnumerator
- NextNodeAppearance
- NodeAppearanceByIndex
- NodeAppearanceByName
- Remove

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von `VcNodeAppearanceCollection`

Mit dieser Eigenschaft können Sie die Anzahl der `NodeAppearance`-Objekte erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der <code>NodeAppearance</code> -Objekte

**Code-Beispiel VB.NET**

```
MessageBox.Show(VcNet1.NodeAppearanceCollection.Count)
```

**Code-Beispiel C#**

```
MessageBox.Show(vcNet1.NodeAppearanceCollection.Count.ToString());
```

---

## Methoden

### Add

**Methode von VcNodeAppearanceCollection**

Mit dieser Methode können Sie ein neues Knotenaussehen in der NodeAppearanceCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue VcNodeAppearance-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Bei dem neuen Knotenaussehen sind standardmäßig alle Eigenschaften auf transparent gesetzt.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ newName	System.String	Name des NodeAppearance-Objekts
<b>Rückgabewert</b>	VcNodeAppearance	Neues NodeAppearance-Objekt

**Code-Beispiel VB.NET**

```
newNodeAppearance = VcNet1.NodeAppearanceCollection.Add("nodeapp1")
```

**Code-Beispiel C#**

```
newNodeAppearance = vcNet1.NodeAppearanceCollection.Add("nodeapp1");
```

### AddBySpecification

**Methode von VcNodeAppearanceCollection**

Mit dieser Methode können Sie ein Knotenaussehen über eine Knotenaussehen-Spezifikation erzeugen. Dies ermöglicht die Persistenz von Knotenaussehen-Objekten. Die Spezifikation eines Knotenaussehens kann erfragt (siehe VcNodeAppearance-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Knotenaussehen mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ nodeAppearanceSpecification	System.String	Knotenaussehen-Spezifikation
<b>Rückgabewert</b>	VcNodeAppearance	Neues Knotenaussehen-Objekt

## Copy

### Methode von VcNodeAppearanceCollection

Mit dieser Methode können Sie ein Knotenaussehen kopieren. Wenn das Knotenaussehen mit dem angegebenen Namen existiert und der Name des neuen Knotenaussehens noch nicht verwendet wird, wird das neue Knotenaussehen-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ fromName	System.String	Name des zu kopierenden Knotenaussehens
⇒ newName	System.String	Name des neuen Knotenaussehens
<b>Rückgabewert</b>	VcNodeAppearance	NodeAppearance-Objekt

## FirstNodeAppearance

### Methode von VcNodeAppearanceCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste NodeAppearance-Objekt des Collection-Objekts zugreifen, um anschließend mit der Methode **NextNodeAppearance** über die nachfolgenden Objekte zu iterieren. Existiert kein NodeAppearance-Objekt im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcNodeAppearance	Erstes NodeAppearance-Objekt

**Code-Beispiel VB.NET**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
While Not nodeAppearance Is Nothing
    MessageBox.Show(nodeAppearance.Name)
    nodeAppearance = nodeAppearanceCltn.NextNodeAppearance
End While
```

**Code-Beispiel C#**

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance
nodeAppearanceCltn = vcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
While Not nodeAppearance Is Nothing
    MessageBox.Show(nodeAppearance.Name)
    nodeAppearance = nodeAppearanceCltn.NextNodeAppearance
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
while (nodeAppearance != null)
{
    MessageBox.Show(nodeAppearance.Name);
    nodeAppearance = nodeAppearanceCltn.NextNodeAppearance();
}
```

**GetEnumerator****Methode von VcNodeAppearanceCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Knotenaussehen-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

**Code-Beispiel VB.NET**

```
Dim nodeApp As VcNodeAppearance

For Each nodeApp In VcNet1.NodeAppearanceCollection
    Debug.Print nodeApp.Name
Next
```

**Code-Beispiel C#**

```
Dim nodeApp As VcNodeAppearance

For Each nodeApp In vcNet1.NodeAppearanceCollection
    Debug.Print nodeApp.Name
Next
```

## NextNodeAppearance

### Methode von VcNodeAppearanceCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden NodeAppearance-Objekte des Collection-Objekts zugreifen, nachdem Sie mit der Methode **FirstNodeAppearance** den Initialwert erfasst haben. Sind alle Objekte durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcNodeAppearance	Nachfolgendes NodeAppearance-Objekt

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance
While Not nodeAppearance Is Nothing
    ListBox1.Items.Add("Name: " + nodeAppearance.Name)
    nodeAppearance = nodeAppearanceCltn.NextNodeAppearance
End While
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance = nodeAppearanceCltn.FirstNodeAppearance();
while (nodeAppearance != null)
{
    listBox1.Items.Add("Name: " + nodeAppearance.Name);
    nodeAppearance = nodeAppearanceCltn.NextNodeAppearance();
}
```

## NodeAppearanceByIndex

### Methode von VcNodeAppearanceCollection

Mit dieser Methode können Sie auf ein einzelnes NodeAppearance-Objekt über seinen Index zugreifen. Existiert kein NodeAppearance-Objekt unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des Knotenaussehens
<b>Rückgabewert</b>	VcNodeAppearance	Ermitteltes NodeAppearance-Objekt

## NodeAppearanceByName

### Methode von VcNodeAppearanceCollection

Mit dieser Methode können Sie ein NodeAppearance-Objekt über den Namen erfragen. Existiert kein NodeAppearance-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ nodeAppearanceName	System.String	Name des NodeAppearance-Objekts
<b>Rückgabewert</b>	VcNodeAppearance	NodeAppearance-Objekt

### Code-Beispiel VB.NET

```
Dim nodeAppearanceCltn As VcNodeAppearanceCollection
Dim nodeAppearance As VcNodeAppearance

nodeAppearanceCltn = VcNet1.NodeAppearanceCollection
nodeAppearance = nodeAppearanceCltn.NodeAppearanceByName("NodeAppearanceOne")
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcCircle
```

### Code-Beispiel C#

```
VcNodeAppearanceCollection nodeAppearanceCltn = vcNet1.NodeAppearanceCollection;
VcNodeAppearance nodeAppearance =
nodeAppearanceCltn.NodeAppearanceByName("NodeAppearanceOne");
nodeAppearance.FrameShape = VcAppearanceFrameShape.vcCircle;
```

## Remove

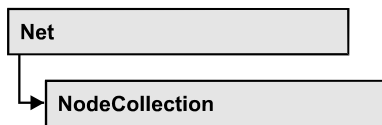
### Methode von VcNodeAppearanceCollection

Mit dieser Methode können Sie ein Knotenaussehen löschen. Wenn das Knotenaussehen noch irgendwo verwendet wird, kann es nicht gelöscht werden. In diesem Fall wird **False** zurückgegeben, sonst **True**.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ name	System.String	Name des Knotenaussehens
<b>Rückgabewert</b>	System.Boolean	Knotenaussehen gelöscht (True)/nicht gelöscht (False)



## 7.41 VcNodeCollection



Ein Objekt vom Typ `VcNodeCollection` beinhaltet alle im Diagramm vorhandenen Knoten. Mit der Methode **SelectNodes** können Sie eine Untermenge dieser Knoten selektieren. Über **For Each node InNodeCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Knoten zugreifen. Die Anzahl der im Auflistungsobjekt vorhandenen Knoten kann über die Eigenschaft **Count** erfragt werden.

### Eigenschaften

- Count

### Methoden

- FirstNode
- GetEnumerator
- NextNode
- SelectNodes

---

## Eigenschaften

### Count

**Nur-Lese-Eigenschaft von VcNodeCollection**

Mit dieser Eigenschaft können Sie die Anzahl der Knoten in der Knotenauflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl Knoten im NodeCollection-Objekt

### Code-Beispiel VB.NET

```

Dim nodeCltn As VcNodeCollection

nodeCltn = VcNet1.NodeCollection
MsgBox("Number of nodes: " + nodeCltn.Count)
  
```

**Code-Beispiel C#**

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
MessageBox.Show("Number of nodes: " + nodeCltn.Count);
```

## Methoden

### FirstNode

**Methode von VcNodeCollection**

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Knoten des NodeCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextNode** über die nachfolgenden Knoten zu iterieren. Existiert kein Knoten im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcNode	Erster Knoten

**Code-Beispiel VB.NET**

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode
```

```
nodeCltn = VcNet1.NodeCollection
node = nodeCltn.FirstNode
```

**Code-Beispiel C#**

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
```

### GetEnumerator

**Methode von VcNodeCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Knoten-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

## NextNode

### Methode von VcNodeCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Knoten des NodeCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstNode** den Initialwert erfasst haben. Sind alle Knoten durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcNode	Folgeknoten

### Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcNet1.NodeCollection
node = nodeCltn.FirstNode
While Not node Is Nothing
    node.Marked = False
    node = nodeCltn.NextNode
End While
```

### Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
while (node != null)
{
    node.Marked = false;
    node = nodeCltn.NextNode;
}
```

## SelectNodes

### Methode von VcNodeCollection

Mit dieser Methode können Sie steuern, welche Knoten in das NodeCollection-Objekt aufgenommen werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ selType	VcSelectionType  <b>Mögliche Werte:</b> .vcAll 0 .vcAllLinksCausingCycles 7  .vcAllLinksInCycles 6	Auszuwählende Knoten  Alle Objekte im Diagramm werden ausgewählt. Wird diese Selektion gewählt, dann befinden sich in der LinkCollection alle Verbindungen, die tatsächlich Zyklen verursachen, d.h. würde diese minimale Anzahl Verbindungen gelöscht, gäbe es keine Zyklen mehr. Wird dieser Selektionstyp gewählt, dann befinden sich in der LinkCollection alle Verbindungen, die Zyklen bilden. Zyklen sind geschlossene Ketten von Knoten und Verbindungen.

	.vcAllVisible 1 .vcMarked 2	Alle sichtbaren Objekte werden ausgewählt. Alle markierten Objekte werden ausgewählt.
<b>Rückgabewert</b>	System.Int32	Anzahl ausgewählter Knoten

**Code-Beispiel VB.NET**

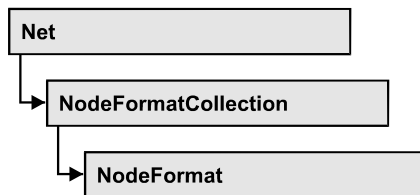
```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcNet1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcSelected)
```

**Code-Beispiel C#**

```
VcNodeCollection nodeCltn = vcNet1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcSelected);
```

## 7.42 VcNodeFormat



Ein Objekt vom Typ VcNodeFormat legt Inhalt und Erscheinungsbild eines Knotens fest. Knotenformate werden zur Designzeit im Dialogfeld **Knotenformate verwalten**, das Sie über die Eigenschaftenseite **Knoten** erreichen, verwaltet und bearbeitet.

### Eigenschaften

- FieldsSeparatedByLines
- FormatField
- FormatFieldCount
- Name
- Specification
- WidthOfExteriorSurrounding

### Methoden

- CopyFormatField
- GetEnumerator
- RemoveFormatField

---

## Eigenschaften

### FieldsSeparatedByLines

Eigenschaft von VcNodeFormat

Mit dieser Eigenschaft können Sie festlegen, ob innenliegende Felder durch sichtbare Linien getrennt werden (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Innenliegende Felder durch sichtbare Linien getrennt (True)/ nicht getrennt (False)

**Code-Beispiel VB.NET**

```
Dim nodeFormatCltn As VcNodeFormatCollection
Dim nodeFormat As VcNodeFormat

nodeFormatCltn = VcNet1.NodeFormatCollection
nodeFormat = nodeFormatCltn.FormatByName("FormatOne")
nodeFormat.FieldsSeparatedByLines = True
```

**Code-Beispiel C#**

```
VcNodeFormatCollection nodeFormatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat nodeFormat = nodeFormatCltn.FormatByName("FormatOne");
nodeFormat.FieldsSeparatedByLines = true;
```

**FormatField****Nur-Lese-Eigenschaft von VcNodeFormat**

Mit dieser Eigenschaft können Sie ein VcNodeFormatField-Objekt per Index holen. Der Index muss im Bereich von 0 bis FormatFieldCount-1 liegen.

	Datentyp	Beschreibung
<b>Parameter:</b> index	System.Int16 0 ... .FormatFieldCount-1	Index des Knotenformatfeldes
<b>Eigenschaftswert</b>	VcNodeFormatField	Knotenformatfeld

**FormatFieldCount****Nur-Lese-Eigenschaft von VcNodeFormat**

Mit dieser Eigenschaft können Sie die Anzahl der Felder eines Knotenformats ermitteln.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16	Anzahl der Felder im Knotenformat

**Code-Beispiel VB.NET**

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MsgBox(nodeFormat.FormatFieldCount)
```

**Code-Beispiel C#**

```
VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.FormatFieldCount.ToString());
```

## Name

### Eigenschaft von VcNodeFormat

Mit dieser Eigenschaft können Sie den Namen des Knotenformats erfragen oder setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Knotenformatname

### Code-Beispiel VB.NET

```
Dim nodeFormat As VcNodeFormat

nodeFormat = VcNet1.NodeFormatCollection.FirstFormat
MessageBox(nodeFormat.Name)
```

### Code-Beispiel C#

```
VcNodeFormat nodeFormat = vcNet1.NodeFormatCollection.FirstFormat();
MessageBox.Show(nodeFormat.Name);
```

## Specification

### Nur-Lese-Eigenschaft von VcNodeFormat

Mit dieser Eigenschaft können Sie die Spezifikation dieses Knotenformats auslesen. Die Spezifikation ist ein String, der nur lesbare ASCII-Zeichen im Bereich 32 bis 127 enthält und somit problemlos in Textdateien oder Datenbanken gespeichert werden kann. Dies ermöglicht Persistenz. Eine solche Spezifikation kann später zur Wiederherstellung eines Knotenformats mit der Methode **VcNodeFormatCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Knotenformats

### Code-Beispiel VB.NET

```
Dim nodeFormatCltn As VcNodeFormatCollection
Dim nodeFormat As VcNodeFormat

nodeFormatCltn = VcNet1.NodeFormatCollection
nodeFormat = nodeFormatCltn.FirstNodeFormat
MessageBox(nodeFormat.Specification)
```

### Code-Beispiel C#

```
VcNodeFormatCollection nodeFormatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat nodeFormat = nodeFormatCltn.FirstNodeFormat();
MessageBox.Show(nodeFormat.Specification);
```

## WidthOfExteriorSurrounding

### Eigenschaft von VcNodeFormat

Mit dieser Eigenschaft können Sie die Breite (in mm) des Außenbereichs des Knotenfeldes setzen, d. h. den Abstand in Millimetern, den Knoten mit diesem Knotenformat zu benachbarten Knoten und zum Rand der Darstellung halten sollen. Standardmäßig beträgt die Breite des Außenbereichs 3 mm. Bei kleineren Werten kann es gelegentlich zu Überlagerungen von grafischen Elementen kommen. Daher sollten Sie den Standardwert nur in begründeten Fällen unterschreiten.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16 0 ... 9	Breite des Außenbereichs des Knotenfeldes (in mm)

## Methoden

### CopyFormatField

#### Methode von VcNodeFormat

Mit dieser Methode können Sie ein Knotenformatfeld kopieren. Das neue VcNodeFormatField-Objekt wird zurückgegeben. Es erhält den nächsten, noch nicht vergebenen Index.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ position	VcFormatFieldPosition  <b>Mögliche Werte:</b> .vcAbove 1 .vcBelow 3 .vcLeftOf 0 .vcOutsideAbove 9 .vcOutsideBelow 11 .vcOutsideLeftOf 8 .vcOutsideRightOf 12 .vcRightOf 4	Position des neuen Knotenformatfeldes  oberhalb unterhalb links von außerhalb, oberhalb außerhalb, unterhalb außerhalb, links von außerhalb, rechts von rechts von
⇒ refIndex	System.Int16	Index des Referenz-Knotenformatfeldes
<b>Rückgabewert</b>	VcNodeFormatField	Neu angelegtes Knotenformatfeld-Objekt



## GetEnumerator

Methode von VcNodeFormat

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Knotenformatfelder iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

### Code-Beispiel VB.NET

```
Dim format As VcNodeFormat
For Each format In VcNet1.NodeFormatCollection
    Debug.Write(format.Name)
Next
```

### Code-Beispiel C#

```
foreach (VcNodeFormat format in vcNet1.NodeFormatCollection)
    Console.Write(format.Name);
```

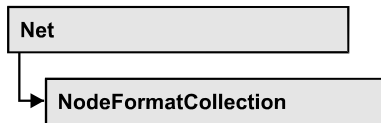
## RemoveFormatField

Methode von VcNodeFormat

Mit dieser Methode können Sie ein Knotenformatfeld über den angegebenen Index löschen. Anschließend wird ggf. der Index aller Knotenformatfelder neu festgesetzt, so dass sie wieder fortlaufend nummeriert sind.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ index	System.Int16	Index des zu löschenden Knotenformatfeldes

## 7.43 VcNodeFormatCollection



In einem Objekt vom Typ `VcNodeFormatCollection` sind automatisch alle verfügbaren Knotenformate zusammengefasst. Über **For Each format In-NodeFormatCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Knotenformate zugreifen. Sie haben Zugriff auf bestimmte Formate über die Eigenschaften **FormatByName**. Die Anzahl der im Auflistungsobjekt vorhandenen Knotenformate kann über die Eigenschaft **Count** erfragt werden.

### Eigenschaften

- Count

### Methoden

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

---

## Eigenschaften

### Count

Nur-Lese-Eigenschaft von `VcNodeFormatCollection`

Mit dieser Eigenschaft können Sie die Anzahl der Knotenformatobjekte in der `NodeFormat`-Auflistung abfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Knotenformate

### Code-Beispiel VB.NET

```
Dim formatCltn As VcNodeFormatCollection
Dim numberOfFormats As Integer

formatCltn = VcNet1.NodeFormatCollection
numberOfFormats = formatCltn.Count
```

### Code-Beispiel C#

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
int numberOfFormats = formatCltn.Count;
```

---

## Methoden

### Add

#### Methode von VcNodeFormatCollection

Mit dieser Methode können Sie ein neues Knotenformat in der NodeFormatCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue VcNodeFormat-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

Das Knotenformat besitzt standardmäßig folgende Eigenschaften:

- ein einziges Feld
- WidthOfExteriorSurrounding: 3 mm

Das Feld hat folgende Eigenschaften:

- Type: vcFFTText
- TextDataFieldIndex: in der Eigenschaftenseite **Allgemeines** festgelegte IDMinimumWidth: 3000
- Alignment: vcFFACenter
- BackColor: -1 (transparent)
- TextFontColor: RGB(0,0,0) (schwarz)
- TextFont: Arial, 10, normal
- LeftMargin, RightMargin, TopMargin, BottomMargin: 0,3 mm

- MinimumTextLineCount, MaximumTextLineCount: 1

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ newName	System.String	Name des Knotenformats
<b>Rückgabewert</b>	VcNodeFormat	Knotenformat-Objekt

#### Code-Beispiel VB.NET

```
newNodeFormat = VcNet1.NodeFormatCollection.Add("nodeformat1")
```

#### Code-Beispiel C#

```
newNodeFormat = vcNet1.NodeFormatCollection.Add("nodeformat1");
```

## AddBySpecification

### Methode von VcNodeFormatCollection

Mit dieser Methode können Sie ein Knotenformat über eine Knotenformat-Spezifikation erzeugen. Dies ermöglicht die Persistenz von Knotenformat-Objekten. Die Spezifikation eines Knotenformats kann erfragt (siehe VcNodeFormat-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Knotenformat mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ formatSpecification	System.String	Knotenformat-Spezifikation
<b>Rückgabewert</b>	VcNodeFormat	Neues Knotenformat-Objekt

## Copy

### Methode von VcNodeFormatCollection

Mit dieser Methode können Sie ein Knotenformat kopieren. Wenn das Knotenformat mit dem angegebenen Namen existiert und der Name des neuen Knotenformats noch nicht verwendet wird, wird das neue Knotenformat-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ fromName	System.String	Name des zu kopierenden Knotenformats
⇒ newName	System.String	Name des neuen Knotenformats
<b>Rückgabewert</b>	VcNodeFormat	NodeFormat-Objekt

## FirstFormat

### Methode von VcNodeFormatCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Knotenformat des NodeFormatCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextFormat** über die nachfolgenden Knotenformate zu iterieren. Existiert kein Knotenformat im NodeFormatCollection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Rückgabewert</b>	VcNodeFormat	Erstes Knotenformat

### Code-Beispiel VB.NET

```
Dim format As VcNodeFormat
format = VcNet1.NodeFormatCollection.FirstFormat
```

### Code-Beispiel C#

```
VcNodeFormat format = vcNet1.NodeFormatCollection.FirstFormat;
```

## FormatByIndex

### Methode von VcNodeFormatCollection

Mit dieser Methode können Sie auf ein einzelnes Format über seinen Index zugreifen. Existiert kein Format an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ index	System.Int16	Index des Knotenformats
<b>Rückgabewert</b>	VcNodeFormat	Ermitteltes Knotenformat

**Code-Beispiel VB.NET**

```
Dim formatCltn As VcNodeFormatCollection

formatCltn = VcNet1.NodeFormatCollection
format = formatCltn.FormatByIndex(0)
format.WidthOfExteriorSurrounding = 2
```

**Code-Beispiel C#**

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat format = formatCltn.FormatByIndex(0);
format.WidthOfExteriorSurrounding = 2;
```

**FormatByName****Methode von VcNodeFormatCollection**

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Knotenformat zugreifen. Existiert kein Knotenformat unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ formatName	System.String	Name des Knotenformats
<b>Rückgabewert</b>	VcNodeFormat	Knotenformat

**Code-Beispiel VB.NET**

```
Dim formatCltn As VcNodeFormatCollection
Dim format As VcNodeFormat

formatCltn = VcNet1.NodeFormatCollection
format = formatCltn.FormatByName("Standard")
```

**Code-Beispiel C#**

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat format = formatCltn.FormatByName("Standard");
```

**GetEnumerator****Methode von VcNodeFormatCollection**

Diese Methode liefert ein Enumerator-Objekt zur Unterstützung von Iterationen durch sprachspezifische Konstrukte. Bei Visual Basic und C# wird es implizit im For...Each-Konstrukt benutzt. Mit diesem Objekt können Sie über alle enthaltenen Knotenformat-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

**Code-Beispiel VB.NET**

```
Dim format As VcNodeFormat

For Each format In VcNet1.NodeFormatCollection
    Debug.Write( format.Name)
Next
```

**Code-Beispiel C#**

```
foreach (VcNodeFormat format In vcNet1.NodeFormatCollection)
    Console.Write(format.Name);
```

**NextFormat****Methode von VcNodeFormatCollection**

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Knotenformate des NodeFormatCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstFormat** den Initialwert erfasst haben. Sind alle Formate durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcNodeFormat	Folgendes Knotenformat

**Code-Beispiel VB.NET**

```
Dim formatCltn As VcNodeFormatCollection
Dim format As VcNodeFormat

formatCltn = VcNet1.NodeFormatCollection
format = formatCltn.Firstformat

While Not format Is Nothing
    ListBox1.Items.Add(format.Name)
    format = formatCltn.NextFormat
End While
```

**Code-Beispiel C#**

```
VcNodeFormatCollection formatCltn = vcNet1.NodeFormatCollection;
VcNodeFormat format = formatCltn.FirstFormat;

while (format != null)
{
    listBox1.Items.Add(format.Name);
    format = formatCltn.NextFormat();
}
```

## Remove

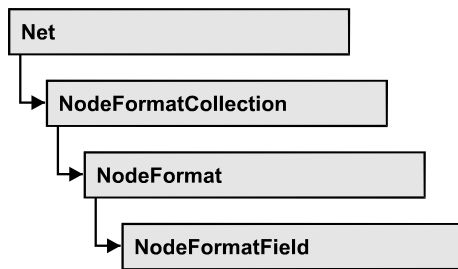
### Methode von VcNodeFormatCollection

Mit dieser Methode können Sie ein Knotenformat löschen. Wenn das Knotenformat noch irgendwo verwendet wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ name	System.String	Name des Knotenformats
<b>Rückgabewert</b>	System.Boolean	Knotenformat gelöscht (True)/nicht gelöscht (False)



## 7.44 VcNodeFormatField



Ein Objekt vom Typ VcNodeFormatField stellt ein Knotenformatfeld, also ein Feld eines VcNodeFormat-Objekts dar. Ein Knotenformatfeld besitzt im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, unter dem es im Knotenformat untergebracht ist.

### Eigenschaften

- Alignment
- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- BottomMargin
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MinimumWidth
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName

- RightMargin
- TextAndGraphicsCombined
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontDataFieldIndex
- TextFontMapName
- TopMargin
- Type

---

## Eigenschaften

### Alignment

**Eigenschaft von VcNodeFormatField**

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Knotenformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFormatFieldAlignment	Ausrichtung des Feldinhalts
	<b>Mögliche Werte:</b> .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	unten unten links unten rechts unten mittig links rechts oben oben links oben rechts

### BackgroundColor

**Eigenschaft von VcNodeFormatField**

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Knotenformatfeldes festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt.

Wählen Sie den Wert **-1**, wenn das Feld die Hintergrundfarbe des Knotenformats besitzen soll. Wenn in der Eigenschaft **BackColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Hintergrundfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte {0...255},{0...255},{0...255},{0...255}) Standardwert: -1

## BackgroundColorDataFieldIndex

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **BackColorMapName** benötigt wird. Wenn Sie hier **-1** angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

## BackColorMapName

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzusordnungstabelle (Typ vcColorMap) für die Hintergrundfarbe setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn der Name einer Farbzusordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **BackColorDataFieldIndex** angegeben ist, wird die Hintergrundfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **BackColor** benutzt.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzusordnungstabelle

## BottomMargin

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Breite des unteren Randes des Knotenformatfeldes in mm festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0...9	Breite des unteren Randes des Knotenformatfeldes in mm

## ConstantText

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie einen konstanten Text in dem Knotenformatfeld ausgeben, falls der Typ des Knotenformatfeldes auf **vcFFTText** und falls die Eigenschaft **TextDataFieldIndex** auf **-1** gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Konstanter Text

## FormatName

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Namen des Knotenformats erfragen, zu dem dieses Knotenformatfeld gehört.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Knotenformats

## GraphicsFileName

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** den Namen einer Grafikdatei setzen oder erfragen, deren Inhalt in dem Knotenformatfeld ausgegeben wird. Der Name muss eine gültige Grafikdatei bezeichnen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Grafikdatei

## GraphicsFileNameDataFieldIndex

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** den Datenfeldindex festlegen oder erfragen, der in der Eigenschaft **GraphicsFileNameMapName** benötigt wird. Beim Wert **-1** wird in dem Knotenformatfeld die Grafikdatei ausgegeben, die im entsprechenden Knotenformat angegeben ist. Ist ein gültiger Datenfeldindex angegeben und keine Zuordnungstabelle, dann wird der Grafikdateiname direkt aus dem Inhalt des angegebenen Datenfeldes entnommen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

## GraphicsFileNameMapName

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** den Namen einer Zuordnungstabelle vom Typ **vcGraphicsFileMap** oder "" setzen oder erfragen. Wenn ein Name und zusätzlich ein Datenfeldindex in der Eigenschaft **GraphicsFileNameDataFieldIndex** angegeben ist, wird eine Grafik aus der Zuordnungstabelle angezeigt. Trifft kein Datenfeldeintrag zu, wird die Grafik aus der Eigenschaft **GraphicsFileName** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Grafik-Zuordnungstabelle

## GraphicsHeight

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** die Höhe der Grafik in dem Knotenformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0 ... 99	Höhe der Grafik in mm

## Index

### Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Index des Knotenformatfelds im zugehörigen Knotenformat erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Knotenformatfeldes

## LeftMargin

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Breite des linken Randes des Knotenformatfeldes in mm festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0...9	Breite des linken Randes des Knotenformatfeldes in mm

## MaximumTextLineCount

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die maximale Anzahl der Zeilen in dem Knotenformatfeld setzen oder erfragen, falls das Knotenformatfeld vom Typ **vcFFTText** ist. Bitte sehen Sie auch die Eigenschaft **MinimumTextLineCount**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0 ... 9	Maximale Zeilenzahl

## MinimumTextLineCount

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die minimale Anzahl der Zeilen in dem Knotenformatfeld setzen oder erfragen, falls der Typ des Knotenformatfeldes auf **vcFFText** gesetzt wurde. Ist in einem Knoten mehr Text vorhanden, als in die minimale Anzahl der Zeilen hineinpasst, wird dieses Feld für diesen Knoten dynamisch bis zur maximalen angegebenen Anzahl der Zeilen ausgedehnt. Wenn Sie dieser Eigenschaft einen Wert zuweisen, sollten Sie anschließend auch erneut der Eigenschaft **MaximumTextLineCount** den gewünschten Wert setzen, sonst könnte es vorkommen, dass das Maximum durch das Minimum überschrieben wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0 ... 9	Minimale Zeilenzahl

## MinimumWidth

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die minimale Breite des Knotenformatfeldes in mm festlegen oder erfragen. Die Breite des Feldes kann sich vergrößern, wenn unter oder über dem Feld andere Felder größere minimale Breiten besitzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0 ... 99	Minimale Breite des Knotenformatfeldes in mm

## PatternBackgroundColorAsARGB

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Knotenformatfeldes festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt. Bei der Umwandlung eines RGB-Wertes in einen ARGB-Wert muss ein Alpha-Wert von 255 hinzugegeben werden.

Wählen Sie den Wert **-1**, wenn das Feld die Hintergrundfarbe des Knotenformats besitzen soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	ARGB-Farbwerte {0...255},{0...255},{0...255},{0...255}

## PatternBackgroundColorDataFieldIndex

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternBackgroundColorMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

## PatternBackgroundColorMapName

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzuoordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuoordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **PatternBackgroundColorDataFieldIndex** angegeben ist, wird die Hintergrundfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **BackColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuoordnungstabelle

## PatternColorAsARGB

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Musterfarbe des Knotenformatfeldes erfragen oder festlegen. Farbwerte haben einen Transparenz- oder Alphawert,



einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der Wert 255 eine voll deckende Farbe erzeugt. Bei der Umwandlung eines RGB-Wertes in einen ARGB-Wert muss ein Alpha-Wert von 255 hinzugegeben werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	ARGB-Farbwerte {0...255},{0...255},{0...255},{0...255}

## PatternColorDataFieldIndex

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternColorMapName** benötigt wird. Wenn Sie hier **-1** angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

## PatternColorMapName

Nur-Lese-Eigenschaft von VcNodeFormatField






Mit dieser Eigenschaft können Sie den Namen einer Farbzusordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzusordnungstabelle und ein Datenfeldindex in der Eigenschaft **PatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Kalendergitters aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **PatternColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzusordnungstabelle

## PatternEx

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie für den Hintergrund des Knotenformatfeldes ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFieldFillPattern	Mustertyp
	<b>Mögliche Werte:</b> .vcAeroGlassPattern 44	Vertikaler Farbverlauf in der Füllmusterfarbe 
	.vcFieldNoPattern 1276	Kein Füllmuster
	.vcFieldVerticalBottomLightedConvexPattern 43	Vertikaler Farbverlauf von dunkel nach hell 
	.vcFieldVerticalConcavePattern 40	Vertikaler Farbverlauf von dunkel über hell nach dunkel 
	.vcFieldVerticalConvexPattern 41	Vertikaler Farbverlauf von hell über dunkel nach hell 
	.vcFieldVerticalTopLightedConvexPattern 42	Vertikaler Farbverlauf von hell nach dunkel 

## PatternExDataFieldIndex

### Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternExMapName** benötigt wird. Wenn Sie hier **-1** angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Datenfeldindex

## PatternExMapName

Nur-Lese-Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Namen einer Muster-Zuordnungstabelle (Typ vcPatternMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Muster-Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **PatternExDataFieldIndex** angegeben ist, wird das Muster aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird das Muster aus der Eigenschaft **PatternEx** ausgegeben.

	Datentyp	Beschreibung
<b>Parameter:</b> ⇒ Rückgabewert	System.String	Name der Musterzuordnungstabelle
<b>Eigenschaftswert</b>	System.String	Name der Musterzuordnungstabelle

## RightMargin

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Breite des rechten Randes des Knotenformatfeldes in mm festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int16 0...9	Breite des rechten Randes des Knotenformatfeldes in mm

## TextAndGraphicsCombined

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das Knotenfeld ein Kombifeld ist. (Vgl. Dialog **Knotenformat bearbeiten**).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Kombifeld (True)/ kein Kombifeld (False)

## TextDataFieldIndex

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Index des Datenfelds, dessen Inhalt in dem Knotenformatfeld dargestellt werden soll, erfragen oder setzen, sofern es sich um ein Feld des Datentyps **vcFFText** handelt. Falls der Index **-1** ist, wird stattdessen der Inhalt der Eigenschaft **ConstantText** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

## TextFont

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Schriftart des Knotenformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFText** gesetzt wurde. Wenn in der Eigenschaft **TextFontMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Schriftart in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Font	Schriftart des Knotenformatfeldes

## TextFontColor

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Schriftfarbe des Knotenformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFText** gesetzt wurde. Wenn über die Eigenschaft **TextFontColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Schriftfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Schriftfarbe des Knotenformatfeldes Standardwert: -1

## TextFontDataFieldIndex

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Schrift-Zuordnungstabelle in der Eigenschaft **TextFontMapName** benötigt wird. Wenn Sie hier -1 angeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

## TextFontMapName

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Namen einer Schrift-Zuordnungstabelle (Typ vcFontMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Schrift-Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **TextFontDataFieldIndex** angegeben ist, wird die Schriftart aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Schriftart aus der Eigenschaft **TextFont** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Schrift-Zuordnungstabelle

## TopMargin

Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie die Breite des oberen Randes des Knotenformatfeldes in mm festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 0...9	Breite des oberen Randes des Knotenformatfeldes in mm

## Type

### Eigenschaft von VcNodeFormatField

Mit dieser Eigenschaft können Sie den Typ des Knotenformatfelds erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcFormatFieldType  <b>Mögliche Werte:</b> .vcFFTGraphics 64 .vcFFTText 36	Typ des Knotenformatfeldes  Grafik Text

## 7.45 VcPrinter

Das VcPrinter-Objekt stellt Ihnen Eigenschaften zur Verfügung, die die Seitengestaltung und den Druckvorgang betreffen. Sie können die Randbreiten der Seiten einstellen sowie Seitenrahmen, Seitenzahlen, Seitenbeschriftung, Schnittmarkierungen und Druckdatum setzen. Weiterhin können Sie die Anzahl der Seiten, auf die das Diagramm verteilt werden soll, sowie Vergrößerungsfaktor, Druckausrichtung, Hoch- bzw. Querformat, Papierformat und Farbmodus festlegen.

### Eigenschaften

- AbsoluteBottomMarginInInches
- AbsoluteLeftMarginInCM
- AbsoluteLeftMarginInInches
- AbsoluteRightMarginInCM
- AbsoluteRightMarginInInches
- AbsoluteTopMarginInCM
- AbsoluteTopMarginInInches
- Alignment
- CurrentHorizontalPagesCount
- CurrentVerticalPagesCount
- CurrentZoomFactor
- CuttingMarks
- DefaultPrinterName
- DocumentName
- FitToPage
- FoldingMarksType
- MarginsShownInInches
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation
- PageDescription
- PageDescriptionString
- PageFrame
- PageNumberMode
- PageNumbers
- PagePaddingEnabled
- PaperSize
- PrintDate
- PrinterName

- PrintPreviewWithFirstPage
- TitleAndLegendOnAllPages
- VcCalendarGrid
- ZoomFactorAsDouble

---

## Eigenschaften

### AbsoluteBottomMarginInInches

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie die absolute Höhe des unteren Seitenrandes in der Maßeinheit Zoll setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

**Hinweis:** Damit im Dialog **Seite einrichten** glattere Werte erzielt werden, beträgt der interne Umrechnungsfaktor 2,5 cm/Zoll statt der eigentlich korrekten 2,54 cm/Zoll (1,5 cm entsprechen dann 0,6 Zoll, 1 cm 0,4 Zoll).

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Absolute Höhe des unteren Seitenrandes in Zoll <b>Standardwert:</b> 0

#### Code-Beispiel VB.NET

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5 ' 0.5 inches
```

#### Code-Beispiel C#

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5; // 0.5 inches
```

### AbsoluteLeftMarginInCM

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie die absolute Breite des linken Seitenrandes in der Maßeinheit cm setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Breite des linken Seitenrandes in cm <b>Standardwert:</b> 0



**Code-Beispiel VB.NET**

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5 ' 2 cm
```

**Code-Beispiel C#**

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5; // 1.5 cm
```

## AbsoluteLeftMarginInInches

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die absolute Breite des linken Seitenrandes in der Maßeinheit Zoll setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

**Hinweis:** Damit im Dialog **Seite einrichten** glattere Werte erzielt werden, beträgt der interne Umrechnungsfaktor 2,5 cm/Zoll statt der eigentlich korrekten 2,54 cm/Zoll (1,5 cm entsprechen dann 0,6 Zoll, 1 cm 0,4 Zoll).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Double	Absolute Breite des linken Seitenrandes in Zoll <b>Standardwert:</b> 0

**Code-Beispiel VB.NET**

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5 ' 0.5 inches
```

**Code-Beispiel C#**

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5; // 0.5 inches
```

## AbsoluteRightMarginInCM

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die absolute Breite des rechten Seitenrandes in der Maßeinheit cm setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Double	Breite des rechten Seitenrandes in cm <b>Standardwert:</b> 0

**Code-Beispiel VB.NET**

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5 ' 2 cm
```

**Code-Beispiel C#**

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5; // 1.5 cm
```

**AbsoluteRightMarginInInches****Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die absolute Breite des rechten Seitenrandes in der Maßeinheit Zoll setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

**Hinweis:** Damit im Dialog **Seite einrichten** glattere Werte erzielt werden, beträgt der interne Umrechnungsfaktor 2,5 cm/Zoll statt der eigentlich korrekten 2,54 cm/Zoll (1,5 cm entsprechen dann 0,6 Zoll, 1 cm 0,4 Zoll).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Double	Absolute Breite des rechten Seitenrandes in Zoll <b>Standardwert:</b> 0

**Code-Beispiel VB.NET**

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5 ' 0.5 inches
```

**Code-Beispiel C#**

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5; // 0.5 inches
```

**AbsoluteTopMarginInCM****Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die absolute Höhe des oberen Seitenrandes in der Maßeinheit cm setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Double	Höhe des oberen Seitenrandes in cm <b>Standardwert:</b> 0

**Code-Beispiel VB.NET**

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5 ' 2 cm
```

**Code-Beispiel C#**

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5; // 1.5 cm
```

## AbsoluteTopMarginInInches

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie die absolute Höhe des oberen Seitenrandes in der Maßeinheit Zoll setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

**Hinweis:** Damit im Dialog **Seite einrichten** glattere Werte erzielt werden, beträgt der interne Umrechnungsfaktor 2,5 cm/Zoll statt der eigentlich korrekten 2,54 cm/Zoll (1,5 cm entsprechen dann 0,6 Zoll, 1 cm 0,4 Zoll).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Double	Absolute Höhe des oberen Seitenrandes in Zoll <b>Standardwert:</b> 0

### Code-Beispiel VB.NET

```
VcNet1.Printer.AbsoluteBottomMarginInInches = 0.5 ' 0.5 inches
```

### Code-Beispiel C#

```
vcNet1.Printer.AbsoluteBottomMarginInInches = 0.5; // 0.5 inches
```

## Alignment

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie die Ausrichtung des Ausdrucks auf einer Seite setzen oder erfragen. Sie hat nur dann eine Auswirkung, wenn die Gesamtgrafik auf einer einzigen Seite dargestellt wird, oder wenn die Eigenschaft **RepeatTitleAndLegend** eingeschaltet ist. In allen anderen Fällen wird die Gesamtgrafik zentriert ausgerichtet.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcPrinterAlignment	Ausrichtung des Ausdrucks auf der Seite <b>Standardwert:</b> vcPCenterCenter
	<b>Mögliche Werte:</b>	
	.vcPBottomCenter 28	vertikale Ausrichtung: unten, horizontale Ausrichtung: mittig
	.vcPBottomLeft 27	vertikale Ausrichtung: unten, horizontale Ausrichtung: links
	.vcPBottomRight 29	vertikale Ausrichtung: unten, horizontale Ausrichtung: rechts
	.vcPCenterCenter 25	vertikale Ausrichtung: mittig, horizontale Ausrichtung: mittig
	.vcPCenterLeft 24	vertikale Ausrichtung: mittig, horizontale Ausrichtung: links
	.vcPCenterRight 26	vertikale Ausrichtung: mittig, horizontale Ausrichtung: rechts

.vcPTopCenter 22	vertikale Ausrichtung: oben, horizontale Ausrichtung: mittig
.vcPTopLeft 21	vertikale Ausrichtung: oben, horizontale Ausrichtung: links
.vcPTopRight 23	vertikale Ausrichtung: oben, horizontale Ausrichtung: rechts

**Code-Beispiel VB.NET**

```
VcNet1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft
```

**Code-Beispiel C#**

```
vcNet1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft;
```

## CurrentHorizontalPagesCount

**Nur-Lese-Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die tatsächliche Anzahl der Seiten des Ausdrucks in der Breite ermitteln. Siehe auch die Eigenschaften **CurrentVerticalPagesCount** und **MaxHorizontalPagesCount**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Tatsächliche Anzahl Seiten in horizontaler Richtung

## CurrentVerticalPagesCount

**Nur-Lese-Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die tatsächliche Anzahl der Seiten des Ausdrucks in der Höhe ermitteln. Siehe auch die Eigenschaften **CurrentHorizontalPagesCount** und **MaxVerticalPagesCount**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Tatsächliche Anzahl Seiten in vertikaler Richtung

## CurrentZoomFactor

**Nur-Lese-Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie den tatsächlichen Zoomfaktor in Prozent für die Einstellung **FitToPage = False** erfragen (Zoomfaktor = 100: Originalgröße, Zoomfaktor > 100: Vergrößerung, Zoomfaktor < 100: Verkleinerung).

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Tatsächlicher Zoomfaktor

## CuttingMarks

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob Schnittmarkierungen auf eine Seite gedruckt werden sollen (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Schnittmarken werden (True) / werden nicht (False) gedruckt <b>Standardwert:</b> False

### Code-Beispiel VB.NET

```
VcNet1.Printer.CuttingMarks = True
```

### Code-Beispiel C#

```
vcNet1.Printer.CuttingMarks = true;
```

## DefaultPrinterName

**Nur-Lese-Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie den Namen des aktuellen Standard-Systemdruckers erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des aktuellen Standard-Systemdruckers

## DocumentName

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie den Namen des Dokumentes bestimmen oder auslesen. Der Dokumentenname wird beim Drucker in der Liste der zu druckenden Dokumente angezeigt und hat bei speziellen Druckertreibern wie z. B. einigen, die PDF-Dateien erzeugen, besondere Funktionen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Dokumentenname <b>Standardwert:</b> " "

## FitToPage

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob die über die Eigenschaften **MaxHorizontalPagesCount** und **MaxVerticalPagesCount** definierte Anzahl von Seiten gedruckt werden soll (True) oder ob das Diagramm in der mit der Eigenschaft **ZoomFactor** eingestellten Größe ausgegeben werden soll (False).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Diagramm wird auf eine definierte Anzahl von Seiten verteilt/wird in der voreingestellten Größe ausgegeben.

### Code-Beispiel VB.NET

```
VcNet1.Printer.FitToPage = True
```

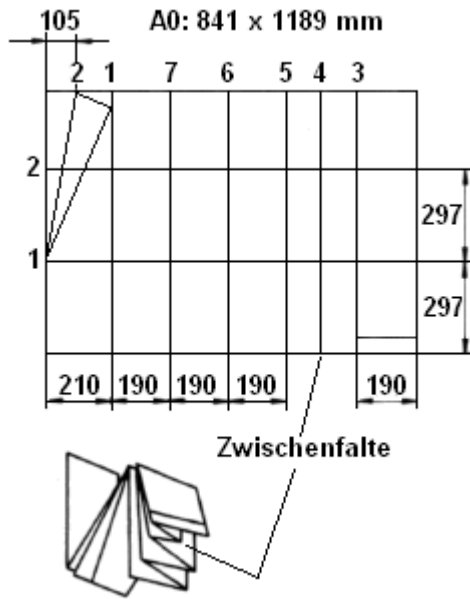
### Code-Beispiel C#

```
vcNet1.Printer.FitToPage = true;
```

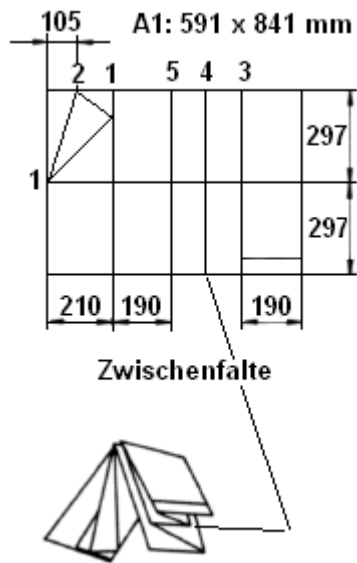
## FoldingMarksType

**Eigenschaft von VcPrinter**

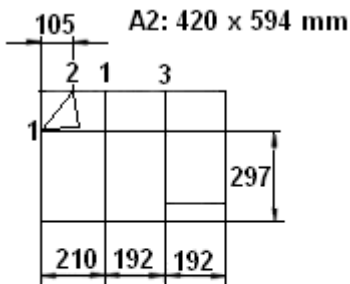
Mit dieser Eigenschaft können Sie folgende Faltmarkierungen nach DIN 824 für den Ausdruck festlegen oder erfragen. Diese ermöglichen das standardisierte Falten für DIN-A-Blattgrößen:



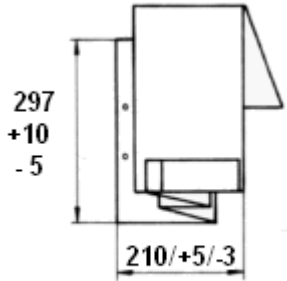
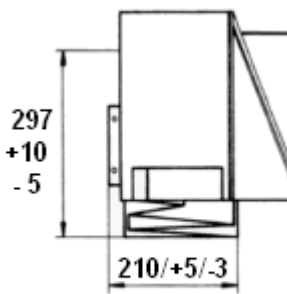
Faltung des DIN-A-0 Formats



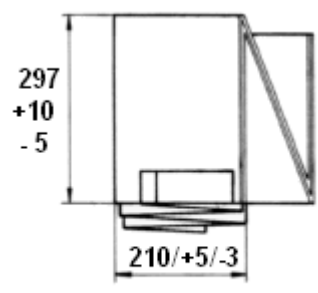
Faltung des DIN-A-1 Formats



### Faltung des DIN-A-2 Formats

	Datentyp	Beschreibung
<p><b>Eigenschaftswert</b></p>	<p>VcFoldingMarksType</p> <p><b>Mögliche Werte:</b>                      .vcFMTDIN824FormA 65                      .vcFMTDIN824FormB 66</p>	<p>Faltmarkierungen</p> <p><b>Standardwert:</b> vcFMTNone</p> <p>Ausgabe von Falmarkierungen nach DIN824-A:                      Die gefaltete Zeichnung kann gelocht und ohne Heftstreifen abgeheftet werden.</p>  <p><b>Faltung nach DIN 824-A</b></p> <p>Ausgabe von Falmarkierungen nach DIN824-B:                      Die gefaltete Zeichnung kann gelocht und mittels Heftstreifens abgeheftet werden.</p>  <p><b>Faltung nach DIN 824-B</b></p>



.vcFMTDIN824FormC 67	<p>Ausgabe von Faltmarkierungen nach DIN824-C: Die gefaltete Zeichnung wird nicht gelocht, sondern in eine Sichthülle gelegt.</p>  <p><b>Faltung nach DIN 824-C</b></p>
.vcFMTNone 0	Keine Ausgabe von Faltmarkierungen

## MarginsShownInInches

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Dialog **Seite einrichten** die Maßeinheit für Seitenränder in Zoll ein- und ausgegeben wird. (Gegenwärtig nur zur Laufzeit möglich).

**Hinweis:** Damit im Dialog **Seite einrichten** glattere Werte erzielt werden, beträgt der interne Umrechnungsfaktor 2,5 cm/Zoll statt der eigentlich korrekten 2,54 cm/Zoll (1,5 cm entsprechen dann 0,6 Zoll, 1 cm 0,4 Zoll).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Maßeinheit der Seitenränder im Dialog <b>Seite einrichten</b> in Zoll (True)/ in cm (False) <b>Standardwert:</b> False

## MaxHorizontalPagesCount

Eigenschaft von VcPrinter

Diese Eigenschaft dient der Festlegung oder Erfragung der horizontalen Seitenzahl beim Drucken und für die Druckvorschau. Die Festlegung ist nur wirksam, wenn Sie in der Eigenschaft **ScalingMode** den Wert **vcFitToPageCount** oder **vcZoomWithHorizontalFit** festgelegt haben. S. auch Eigenschaft **MaxVerticalPagesCount**.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Maximale Anzahl Seiten in horizontaler Richtung <b>Standardwert:</b> 1

**Code-Beispiel VB.NET**

```
VcNet1.Printer.MaxHorizontalPagesCount = 4
```

**Code-Beispiel C#**

```
vcNet1.Printer.MaxHorizontalPagesCount = 4;
```

**MaxVerticalPagesCount****Eigenschaft von VcPrinter**

Diese Eigenschaft dient der Festlegung oder Erfragung der vertikalen Seitenzahl beim Drucken und für die Druckvorschau. Diese Festlegung ist nur wirksam, wenn Sie in der Eigenschaft **ScalingMode** den Wert **vcFitToPageCount** festgelegt haben. S. auch Eigenschaft **MaxHorizontalPagesCount**.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Maximale Anzahl Seiten in vertikaler Richtung <b>Standardwert:</b> 1

**Code-Beispiel VB.NET**

```
VcNet1.Printer.MaxVerticalPagesCount = 4
```

**Code-Beispiel C#**

```
vcNet1.Printer.MaxVerticalPagesCount = 4;
```

**Orientation****Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen, ob die einzelnen Seiten des Ausdrucks im Hoch- oder Querformat verwendet werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcOrientation	Ausrichtung <b>Standardwert:</b> VcPortrait
	<b>Mögliche Werte:</b> .vcLandscape 42 .vcPortrait 41	Querformat Hochformat

**Code-Beispiel VB.NET**

```
VcNet1.Printer.Orientation = VcOrientation.vcLandscape
```

**Code-Beispiel C#**

```
vcNet1.Printer.Orientation = VcOrientation.vcLandscape;
```

## PageDescription

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Seitenbeschriftung für die linke untere Ecke jeder Seite erscheinen soll (True) oder nicht (False). Den Inhalt der Seitenbeschriftung legen Sie über die Eigenschaft **PageDescriptionString** fest.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Seitenbeschriftung wird (True) / wird nicht gedruckt (False) <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.Printer.PageDescription = True
```

**Code-Beispiel C#**

```
vcNet1.Printer.PageDescription = true;
```

## PageDescriptionString

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie eine Seitenbeschriftung für die linke untere Ecke jeder Seite festlegen oder erfragen. Die Ausgabe der Seitenbeschriftung können Sie mit der Eigenschaft **PageDescription** steuern. Für die Seitennummerierung können Sie folgende Platzhalter angeben, die dann beim Ausdruck durch die entsprechenden Inhalte ersetzt werden:

{PAGE} = fortlaufende Seitennummer

{NUMPAGES} = Gesamtanzahl der Seiten

{ROW} = Zeilenposition des Ausschnitts im Gesamtdiagramm

{COLUMN} = Spaltenposition des Ausschnitts im Gesamtdiagramm

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Seitenbeschriftung <b>Standardwert:</b> Empty string ""

**Code-Beispiel VB.NET**

```
VcNet1.Printer.PageDescriptionString = "Net-Graphics"
```

**Code-Beispiel C#**

```
vcNet1.Printer.PageDescriptionString = "Net-Graphics";
```

## PageFrame

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob um den Ausdruck ein Rahmen gezogen werden soll (True) oder nicht (False). Wenn die Eigenschaft **TableTimeScaleOnAllPages** eingeschaltet ist, so wird der Rahmen um jede Einzelseite gezogen, anderenfalls wird er um die gesamte Grafik gezogen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Seitenrahmen wird dargestellt (True) / wird nicht dargestellt (False). <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.Printer.PageFrame = True
```

**Code-Beispiel C#**

```
vcNet1.Printer.PageFrame = true;
```

## PageNumberMode

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen bzw. erfragen, wie die Seitennummerierung ausgegeben werden soll: "Seite N von M Seiten" oder "x.y" (Zeilennummer/Spaltennummer).

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcPageNumberMode	Art der Seitennummerierung <b>Standardwert:</b> vcPRowColumn
	<b>Mögliche Werte:</b> .vcPageNOfM 1597	"Seite N von M Seiten"

.vcPRowColumn 1596	"x,y" (Zeilennummer.Spaltennummer)
--------------------	------------------------------------

**Code-Beispiel VB.NET**

```
Dim printer As VcPrinter
printer.Orientation = VcOrientation.vcLandscape
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM
printer.PageNumbers = True
printer.FitToPage = False
VcNet1.ShowPrintPreviewDialog()
```

**Code-Beispiel C#**

```
VcPrinter printer = vcNet1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM;
printer.PageNumbers = true;
printer.FitToPage = false;
vcNet1.ShowPrintPreviewDialog();
```

## PageNumbers

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Seitenzahl in der linken unteren Ecke einer Seite erscheinen soll (True) oder nicht (False). Die Art der Nummerierung können Sie mit Hilfe der Eigenschaft **PageNumberMode** festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Seitenzahlen werden (True) / werden nicht (False) ausgegeben <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.Printer.PageNumbers = True
```

**Code-Beispiel C#**

```
vcNet1.Printer.PageNumbers = true;
```

## PagePaddingEnabled

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob zwischen dem Diagramm und den Boxen für Titel und Legende so viel Platz gelassen werden soll, dass die Boxen auf jeder Druckseite immer in voller Breite gedruckt werden können und fest am Blattrand positioniert sind. Ist die Eigenschaft auf **False** gesetzt, werden die Boxen ohne Zwischenraum am

Diagramm gedruckt und können dann je nach Diagramm auf den verschiedenen Druckseiten in der Breite variieren.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Zwischenraum zwischen Diagramm und Boxen für Legende/Titel wird (True) / wird nicht (False) ausgegeben <b>Standardwert:</b> True

#### Code-Beispiel VB.NET

```
VcNet1.Printer.PagePaddingEnabled = True
```

#### Code-Beispiel C#

```
vcNet1.Printer.PagePaddingEnabled = true;
```

## PaperSize

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie die zu verwendende Papiergröße festlegen oder erfragen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcPaperSize	Papiergröße
	<b>Mögliche Werte:</b> .vcDIN_A2 66 .vcDIN_A3 8 .vcDIN_A4 9 .vcISO_C 24 .vcISO_D 25 .vcISO_E 26 .vcUS_LEGAL 5 .vcUS_LETTER 1	DIN A2 DIN A3 DIN A4 ISO C ISO D ISO E US LEGAL US LETTER

#### Code-Beispiel VB.NET

```
VcNet1.Printer.PaperSize = VcPaperSize.vcDIN_A3
```

#### Code-Beispiel C#

```
vcNet1.Printer.PaperSize = VcPaperSize.vcDIN_A3;
```

## PrintDate

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das Druckdatum in der linken unteren Ecke jeder Seite erscheinen soll (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Druckdatum wird/wird nicht ausgegeben

### Code-Beispiel VB.NET

```
VcNet1.Printer.PrintDate = True
```

### Code-Beispiel C#

```
vcNet1.Printer.PrintDate = true;
```

## PrinterName

**Nur-Lese-Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie den Namen des aktuell ausgewählten Druckers auslesen bzw. setzen. Dies kann zum Speichern und Wiederherstellen des Zustands des Printer-Objekts verwendet werden.

Wenn man beim Setzen der Eigenschaft einen leeren String übergibt, wird der im System eingestellte Standarddrucker benutzt.

**Hinweis:** Bitte beachten Sie, dass bei Netzwerkdruckern der Druckername in UNC-Notation angegeben werden muss, bspw. "\\server01\printer5".

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Druckername

## PrintPreviewWithFirstPage

**Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen bzw. erfragen, wie die Seitenansicht beim Aufruf aussehen soll: als Gesamtansicht über alle Blätter des Diagramms oder als Darstellung der ersten Seite.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Beim Aufruf der Seitenansicht: Darstellung der ersten Seite (True) / Gesamtansicht über alle Blätter des Diagramms (False)

**Code-Beispiel VB.NET**

```
Dim printer As VcPrinter
printer.Orientation = VcOrientation.vcLandscape
printer.PrintPreviewWithFirstPage = True
printer.FitToPage = False

VcNet1.ShowPrintPreviewDialog()
```

**Code-Beispiel C#**

```
VcPrinter printer = vcNet1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PrintPreviewWithFirstPage = true;
printer.FitToPage = false;

vcNet1.ShowPrintPreviewDialog();
```

**TitleAndLegendOnAllPages****Eigenschaft von VcPrinter**

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob Titel und Legende auf jeder Seite erscheinen sollen (True) oder nicht (False). Außerdem wird hiermit festgelegt, ob die Seitenaufteilung automatisch berechnet wird, so dass die Knoten nicht durchgeschnitten werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Titel und Legende werden auf jeder Seite wiederholt (True) / Titel und Legende werden nur einmal ausgegeben und ggf. beim Seitenumbruch durchtrennt (False) <b>Standardwert:</b> False

**Code-Beispiel VB.NET**

```
VcNet1.Printer.RepeatTitleAndLegend = True
```

**Code-Beispiel C#**

```
vcNet1.Printer.RepeatTitleAndLegend = true;
```



## VcCalendarGrid

### Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie die absolute Höhe des unteren Seitenrandes in der Maßeinheit cm setzen oder erfragen. Die tatsächliche Breite kann größer sein, wenn der verwendete Drucker nicht randlos drucken kann.

	Datentyp	Beschreibung

#### Code-Beispiel VB.NET

```
VcNet1.Printer.AbsoluteTopMarginInCM = 1.5 ' 2 cm
```

#### Code-Beispiel C#

```
vcNet1.Printer.AbsoluteTopMarginInCM = 1.5; // 1.5 cm
```

## ZoomFactorAsDouble

### Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie den Zoomfaktor in Prozent für die Einstellung **FitToPage = False** setzen oder erfragen (Zoomfaktor = 100: Originalgröße, Zoomfaktor > 100: Vergrößerung, Zoomfaktor < 100: Verkleinerung).

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Zoomfaktor für das Diagramm

#### Code-Beispiel VB.NET

```
VcNet1.Printer.ZoomFactor = 150
```

#### Code-Beispiel C#

```
vcNet1.Printer.ZoomFactor = 150
```

## 7.46 VcRect

Rect

Ein Objekt vom Typ **VcRect** bezeichnet ein Rechteck-Objekt und kommt nur in VcInPlaceEditorShowing vor.

### Eigenschaften

- Bottom
- Height
- Left
- Right
- Top
- Width

---

## Eigenschaften

### Bottom

**Eigenschaft von VcRect**

Diese Eigenschaft gibt die untere Position des Rechteckobjekts an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Position des unteren Rands des Rechtecks

### Height

**Nur-Lese-Eigenschaft von VcRect**

Diese Eigenschaft gibt die Höhe des Rechteckobjekts an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Höhe des Rechtecks

## Left

Eigenschaft von VcRect

Diese Eigenschaft gibt die linke Position des Rechteckobjekts an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Position des linken Rands des Rechtecks

### Code-Beispiel VB.NET

```

Private Sub VcNet1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs) Handles
VcNet1.VcInPlaceEditorShowing
    Dim node As VcNode
    node = e.EditObject
    If e.EditObjectType = VcObjectType.vcObjTypeNodeInTable Then
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        Select Case e.FieldIndex
            Case 1 'Name
                TextBox1.Left = e.FldRectVisible.Left + VcNet1.Left
                TextBox1.Top = e.FldRectVisible.Top + VcNet1.Top
                TextBox1.Width = e.FldRectVisible.Width
                TextBox1.Height = e.FldRectVisible.Height
                TextBox1.Text = node.DataField(0)
                TextBox1.Visible = True
                TextBox1.Focus()
            Case 2, 3 'Start or End
                DateTimePicker1.Left = e.FldRectVisible.Left + VcNet1.Left
                DateTimePicker1.Top = e.FldRectVisible.Top + VcNet1.Top
                DateTimePicker1.Value = node.DataField(0)
                DateTimePicker1.Visible = True
                DateTimePicker1.Focus()
            Case 13 'Employee
                ComboBox1.Left = e.FldRectVisible.Left + VcNet1.Left
                ComboBox1.Top = e.FldRectVisible.Top + VcNet1.Top
                ComboBox1.Width = e.FldRectVisible.Width
                ComboBox1.Height = e.FldRectVisible.Height
                ComboBox1.Text = node.DataField(0)
                ComboBox1.Visible = True
                ComboBox1.Focus()
        End Select
    End If
End Sub

```

**Code-Beispiel C#**

```
private void vcNet1_VcInPlaceEditorShowing(object sender,
NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs e)
{
    VcNode node = (VcNode)e.EditObject;
    if (e.EditObjectType == VcObjectType.vcObjTypeNodeInTable)
    {
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
        switch (e.FieldIndex)
        {
            case 1: //Name
                textBox1.Left = e.FldRectVisible.Left + vcNet1.Left;
                textBox1.Top = e.FldRectVisible.Top + vcNet1.Top;
                textBox1.Width = e.FldRectVisible.Width;
                textBox1.Height = e.FldRectVisible.Height;
                textBox1.Text = Convert.ToString(node.get_DataField(0));
                textBox1.Visible = true;
                textBox1.Focus();
                break;
            case 2: //Start or end
                dateTimePicker1.Left = e.FldRectVisible.Left + vcNet1.Left;
                dateTimePicker1.Top = e.FldRectVisible.Top + vcNet1.Top;
                dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
                dateTimePicker1.Visible = true;
                dateTimePicker1.Focus();
                break;
            case 13: //Employee
                comboBox1.Left = e.FldRectVisible.Left + vcNet1.Left;
                comboBox1.Top = e.FldRectVisible.Top + vcNet1.Top;
                comboBox1.Width = e.FldRectVisible.Width;
                comboBox1.Height = e.FldRectVisible.Height;
                comboBox1.Text = Convert.ToString(node.get_DataField(0));
                comboBox1.Visible = true;
                comboBox1.Focus();
                break;
        }
    }
}
```

**Right****Eigenschaft von VcRect**

Diese Eigenschaft gibt die rechte Position des Rechteckobjekts an.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Position des rechten Rands des Rechtecks

**Top****Eigenschaft von VcRect**

Diese Eigenschaft gibt die obere Position des Rechteckobjekts an.

## 764 API Referenz: VcRect

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Position des oberen Rands des Rechtecks

### Code-Beispiel VB.NET

```
DateTimePicker1.Top = e.FldRectVisible.Top + VcNet1.Top
```

### Code-Beispiel C#

```
dateTimePicker1.Top = e.FldRectVisible.Top + vcNet1.Top;
```

## Width

Nur-Lese-Eigenschaft von VcRect

Diese Eigenschaft gibt die Breite des Rechteckobjekts an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Breite des Rechtecks

### Code-Beispiel VB.NET

```
Text1.Width = fldRectVisible.Width
```

### Code-Beispiel C#

```
textBox1.Width = e.FldRectVisible.Width;
```

## 7.47 VcScheduler

### Scheduler

Ein Objekt vom Typ **VcScheduler** bezeichnet ein Rechenmodul, mit dem Sie einfache Projektdaten wie frühestmögliches Ende, frühestmöglicher Anfang (bei Rückwärtsrechnung), freie oder Gesamt-Pufferzeit eines Projektes berechnen können.

### Eigenschaften

- ActualEndDateDataFieldIndex
- ActualStartDateDataFieldIndex
- AutomaticSchedulingEnabled
- DurationDataFieldIndex
- EarlyEndDateDataFieldIndex
- EarlyStartDateDataFieldIndex
- EndDateForAutomaticScheduling
- EndDateNotLaterThanDataFieldIndex
- FreeFloatDataFieldIndex
- LateEndDateDataFieldIndex
- LateStartDateDataFieldIndex
- LinkDurationDataFieldIndex
- ScheduledProjectEndDate
- ScheduledProjectStartDate
- ScheduleSuccessorsOnlyEnabled
- StartDateForAutomaticScheduling
- StartDateNotEarlierThanDataFieldIndex
- TotalFloatDataFieldIndex

### Methoden

- ScheduleProject

## Eigenschaften

### ActualEndDateDataFieldIndex

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das aktuelle Enddatum des Projektes enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das aktuelle Enddatum enthält

### ActualStartDateDataFieldIndex

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das aktuelle Anfangsdatum des Projektes enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das aktuelle Startdatum enthält

### AutomaticSchedulingEnabled

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob die Zeitberechnung automatisch erfolgt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Automatische Zeitberechnung ist an- (true) oder abgeschaltet (false) <b>Standardwert:</b> false

## DurationDataFieldIndex

### Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das die Dauer des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	SystemInt.32	Index des Datenfeldes, das die Vorgangsdauer enthält

## EarlyEndDateDataFieldIndex

### Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das frühestmögliche Enddatum des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	SystemInt.32	Index des Datenfeldes, das das frühestmögliche Enddatum eines Vorgangs enthält

## EarlyStartDateDataFieldIndex

### Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das frühestmögliche Anfangsdatum des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das frühestmögliche Startdatum eines Vorgangs enthält



## EndDateForAutomaticScheduling

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie für den Fall, dass die automatische Zeitberechnung durchgeführt wird, das Enddatum des Projektes setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Gewünschtes Enddatum für automatische Zeitrechnung

## EndDateNotLaterThanDataFieldIndex

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das das gewünschte späteste Enddatum des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int.32	Index des Datenfeldes, das das gewünschte späteste Enddatum enthält

## FreeFloatDataFieldIndex

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das den berechneten freien Puffer des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int.32	Index des Datenfeldes, das den freien Puffer enthält

## LateEndDateDataFieldIndex

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das berechnete spätestmögliche Enddatum des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das spätestmögliche Enddatum eines Vorgangs enthält

## LateStartDateDataFieldIndex

### Eigenschaft von VcScheduler

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das berechnete spätestmögliche Anfangsdatum des Vorgangs enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das spätestmögliche Startdatum eines Vorgangs enthält

## LinkDurationDataFieldIndex

### Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, in dem ein minimaler zeitlicher Abstand zwischen Vorgänger und Nachfolger abgelegt werden kann. Dies ist nur möglich, solange noch keine Daten geladen wurden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das den minimalen zeitlichen Abstand zwischen Vorgänger und Nachfolger enthält

## ScheduledProjectEndDate

### Nur-Lese-Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie, nachdem mit der Methode **VcScheduler.ScheduleProject** die Projektdaten berechnet wurden, das **Früheste Ende** des Projektes erfragen, wenn bei **VcScheduler.ScheduleProject** ein Anfangsdatum vorgegeben wurde.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Index des Datenfeldes, das das berechnete Enddatum des Projektes enthält

## ScheduledProjectStartDate

Nur-Lese-Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie, nachdem mit der Methode **VcScheduler.ScheduleProject** die Projektdaten berechnet wurden, den **Spätesten Anfang** des Projektes erfragen, wenn bei **VcScheduler.ScheduleProject** ein Enddatum vorgegeben wurde.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Index des Datenfeldes, das das berechnete Startdatum des Projektes enthält

## ScheduleSuccessorsOnlyEnabled

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob bei der Zeitrechnung nur die Vorgänge, die Vorgänger besitzen, berechnet werden. Ein "Projektstart" beim Aufruf der Zeitrechnung wird hier also ignoriert.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Berechnung nur der Knoten mit Vorgängern ist an/abgeschaltet

## StartDateForAutomaticScheduling

Eigenschaft von VcScheduler

Mit dieser Eigenschaft können Sie für den Fall, dass die automatische Zeitberechnung durchgeführt wird, das Startdatum des Projektes setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Gewünschtes Startdatum für automatische Zeitrechnung

## StartDateNotEarlierThanDataFieldIndex

**Eigenschaft von VcScheduler**

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das gewünschte früheste Startdatum eines Vorgangs enthält.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int.32	Index des Datenfeldes, das das gewünschte früheste Startdatum enthält

## TotalFloatDataFieldIndex

**Eigenschaft von VcScheduler**

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das den berechneten Gesamtpuffer des Vorgangs enthält.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int.32	Index des Datenfeldes, das den Gesamtpuffer enthält

---

## Methoden

### ScheduleProject

**Methode von VcScheduler**

Mit dieser Methode können Sie die Daten (frühester/spätester Anfang, frühestes/spätestes Ende, Freier Puffer, Gesamtpuffer) eines Projektes berechnen lassen, wobei Sie das gewünschte Anfangs- und Enddatum mit dieser Methode setzen. Bei alleiniger Übergabe des Starttermins wird das Projektende, bei alleiniger Übergabe des Endtermins wird der Projektstart berechnet. Es können auch beide Termine übergeben werden, die Vorgänge erhalten dann entsprechende Pufferzeiten. (Dies geht allerdings nur, wenn die

Termine zueinander passen, d.h. der Endtermin sollte beispielsweise nicht innerhalb der Projektzeit liegen.) Das Fehlen beider Daten führt zu einer Fehlermeldung. Falls ein Zyklus der Knoten und Verbindungen festgestellt wird, werden diese automatisch markiert.

Die Ergebnisse werden in Feldern gespeichert, die Sie mit den Eigenschaften **EarlyStartDateDataFieldIndex**, **LateStartDateDataFieldIndex**, **EarlyEndDateDataFieldIndex**, **LateEndDateDataFieldIndex**, **FreeFloatDataFieldIndex** und **TotalFloatDataFieldIndex** festlegen.

	Datentyp	Beschreibung
<b>Parameter:</b>		
⇒ startDate	System.DateTime	Gewünschtes Anfangsdatum
⇒ endDate	System.DateTime	Gewünschtes Enddatum
<b>Rückgabewert</b>	System.Boolean	Die Projektdaten wurden erfolgreich berechnet (True) / nicht berechnet (False)

#### Code-Beispiel VB.NET

```
' Vorwärtsberechnung (ASAP)
VcScheduler.ScheduleProject(2.5.2017, newDate(0))

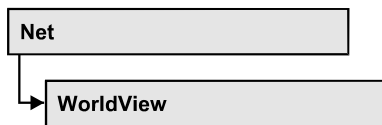
' Rückwärtsberechnung (JIT)
VcScheduler.ScheduleProject(newDate(0), 2.5.2017)
```

#### Code-Beispiel C#

```
// Vorwärtsberechnung (ASAP)
vcScheduler.ScheduleProject(2.5.2017, newDate(0));

// Rückwärtsberechnung (JIT)
vcScheduler.ScheduleProject(newDate(0), 2.5.2017);
```

## 7.48 VcWorldView



Ein Objekt vom Typ **VcWorldView** bezeichnet das Komplettansicht-Fenster.

### Eigenschaften

- Border
- Height
- HeightActualValue
- Left
- LeftActualValue
- MarkingColor
- Mode
- ParentHWnd
- ScrollBarMode
- Top
- TopActualValue
- UpdateBehaviorName
- Visible
- Width
- WidthActualValue

---

## Eigenschaften

### Border

**Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann gesetzt oder erfragt werden, ob die Komplettansicht einen Rahmen besitzt (nicht im Modus **vcPopupWindow**). Die Rahmenfarbe ist **Color.Black**. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Rahmen um die Komplettansicht (True)/kein Rahmen um die Komplettansicht (False) <b>Standardwert:</b> True

**Code-Beispiel VB.NET**

```
VcNet1.WorldView.Mode = VcWorldViewMode.vcNotFixed
VcNet1.WorldView.Border = True
```

**Code-Beispiel C#**

```
vcNet1.WorldView.Mode = VcWorldViewMode.vcNotFixed;
vcNet1.WorldView.Border = true;
```

## Height

**Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann die vertikale Ausdehnung der Komplettansicht erfragt werden. In den Positionen **vcFixedAtTop**, **vcFixedAtBottom**, **vcNotFixed** und **vcPopupWindow** der Eigenschaft **Mode** kann sie außerdem gesetzt werden. Die Koordinaten werden in Pixel, bezogen auf den Bildschirm, angegeben.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Höhe der Komplettansicht <b>Standardwert:</b> 100

**Code-Beispiel VB.NET**

```
VcNet1.WorldView.Height = 100
```

**Code-Beispiel C#**

```
vcNet1.WorldView.Height = 100;
```

## HeightActualValue

**Nur-Lese-Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann die tatsächlich dargestellte vertikale Ausdehnung der Komplettansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**,

**vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32 {0, ...}	Tatsächliche Höhe der Komplettansicht  {0, ...} <b>Standardwert:</b> 100

#### Code-Beispiel VB.NET

```
VcNet1.LegendView.Height = 300
```

#### Code-Beispiel C#

```
vcNet1.LegendView.Height = 100;
```

## Left

### Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die linke Position der Komplettansicht erfragt werden. In den Positionen **vcNotFixed** und **vcPopupWindow** der Eigenschaft **Mode** kann sie außerdem gesetzt werden. Die Koordinaten werden in Pixel, bezogen auf den Bildschirm, angegeben.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Linke Position der Komplettansicht <b>Standardwert:</b> 0

#### Code-Beispiel VB.NET

```
VcNet1.WorldView.Left = 200
```

#### Code-Beispiel C#

```
vcNet1.WorldView.Left = 200;
```

## LeftActualValue

### Nur-Lese-Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die tatsächlich dargestellte linke Position der Komplettansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**,



**vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Tatsächliche linke Position der Komplettansicht  {0, ...} <b>Standardwert:</b> 0

#### Code-Beispiel VB.NET

```
VcNet1.LegendView.LeftActualValue = 150
```

#### Code-Beispiel C#

```
vcNet1.LegendView.LeftActualValue = 150;
```

## MarkingColor

**Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann die Farbe der Linie des Rechtecks erfragt oder gesetzt werden, das in der Komplettansicht den aktuell gewählten Ausschnitt anzeigt. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Drawing.Color	RGB-Farbwerte  {{0...255},{0...255},{0...255}} <b>Standardwert:</b> RGB(0, 0, 255)

#### Code-Beispiel VB.NET

```
VcNet1.WorldView.MarkingColor = Color.Red
```

#### Code-Beispiel C#

```
vcNet1.WorldView.MarkingColor = Color.Red;
```

## Mode

**Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann der Modus der Komplettansicht erfragt oder gesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	VcWorldViewMode	Modus der Gesamtansicht <b>Standardwert:</b> vcPopupWindow
	<b>Mögliche Werte:</b>	
	.vcFixedAtBottom 4	Die Komplettansicht wird unten im Fenster des Steuerelements angezeigt. Das Bezugssystem der Koordinaten ist das Steuerelement. Bei dieser Einstellung kann nur die Höhe festgelegt werden, während Position und Breite vorgegeben sind.
	.vcFixedAtLeft 1	Die Komplettansicht wird links im Fenster des Steuerelements angezeigt. Das Bezugssystem der Koordinaten ist das Steuerelement. Bei dieser Einstellung kann nur die Breite festgelegt werden, während Position und Höhe vorgegeben sind.
	.vcFixedAtRight 2	Die Komplettansicht wird rechts im Fenster des Steuerelements angezeigt. Das Bezugssystem der Koordinaten ist das Steuerelement. Bei dieser Einstellung kann nur die Breite festgelegt werden, während Position und Höhe vorgegeben sind.
	.vcFixedAtTop 3	Die Komplettansicht wird oben im Fenster des Steuerelements angezeigt. Das Bezugssystem der Koordinaten ist das Steuerelement. Bei dieser Einstellung kann nur die Höhe festgelegt werden, während Position und Breite vorgegeben sind.
	.vcNotFixed 5	Die Komplettansicht ist ein untergeordnetes Kindfenster des aktuellen Vaterfensters des Steuerelements und kann an beliebiger Position mit beliebiger Ausdehnung angeordnet werden. Das Bezugssystem der Koordinaten ist das Fenster der Anwendung. Das Kindfenster ist ohne eigenen Fensterrahmen und kann vom Benutzer nicht interaktiv verschoben werden. Das Vaterfenster kann bei Bedarf über die Eigenschaft <b>VcWorldView.ParentHwnd</b> geändert werden.
	.vcPopupWindow 6	Die Komplettansicht ist ein Popup-Fenster, das einen eigenen Rahmen besitzt und vom Benutzer in Position und Größe verändert werden kann. Das Bezugssystem der Koordinaten ist der Bildschirm. Es kann über das Standard-Kontextmenü ein- bzw. ausgeschaltet oder über die <b>Schließen</b> -Schaltfläche in der Titelleiste ausgeschaltet werden.

**Code-Beispiel VB.NET**

```
VcNet1.WorldView.Mode = VcWorldViewMode.vcFixedAtBottom
```

**Code-Beispiel C#**

```
vcNet1.WorldView.Mode = VcWorldViewMode.vcFixedAtBottom;
```

**ParentHwnd****Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann im Modus **vcNotFixed** das Hwnd-Handle des Vaterfensters festgelegt werden, wenn die Komplettansicht beispielsweise in einem selbst implementierten Rahmenfenster erscheinen soll. Standardmäßig

steht dies auf dem HWnd-Handle des Vaterfensters des VARCHART-  
Windows-Forms-Hauptfensters. Diese Eigenschaft kann nur zur Laufzeit  
verwendet werden.

	Datentyp	Beschreibung
Eigenschaftswert	OLE_HANDLE	Zugriffsnummer

## ScrollBarMode

Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann der Scrollbarmodus der Gesamtansicht erfragt  
oder gesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite  
**Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcWorldViewScrollBarMode	Scrollbarmodus <b>Standardwert:</b> NoScrollBar
	<b>Mögliche Werte:</b> .vcAutomaticScrollBar 3 .vcHorizontalScrollBar 1 .vcNoScrollBar 0 .vcVerticalScrollBar 2	Anzeige einer horizontalen oder vertikalen Bildlaufleiste, wenn nötig. Anzeige einer horizontalen Bildlaufleiste, wenn nötig. Es wird immer das vollständige Diagramm ohne Bildlaufleisten angezeigt. Anzeige einer vertikalen Bildlaufleiste, wenn nötig.

### Code-Beispiel VB.NET

```
VcNet1.WorldView.ScrollBarMode = vcAutomaticScrollbar
```

### Code-Beispiel C#

```
vcNet1.WorldView.ScrollBarMode = vcAutomaticScrollBar;
```

## Top

Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die obere Position der Komplettansicht erfragt  
werden. In den Positionen **vcNotFixed** und **vcPopupWindow** der  
Eigenschaft **Mode** kann sie außerdem gesetzt werden. Die Koordinaten  
werden in Pixel, bezogen auf den Bildschirm, angegeben.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche  
Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Obere Position der Komplettansicht <b>Standardwert:</b> 0

**Code-Beispiel VB.NET**

```
VcNet1.WorldView.Top = 20
```

**Code-Beispiel C#**

```
vcNet1.WorldView.Top = 20;
```

**TopActualValue****Nur-Lese-Eigenschaft von VcWorldView**

Mit dieser Eigenschaft kann die tatsächlich dargestellte obere Position der Komplettansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Tatsächliche obere Position der Komplettansicht  {0, ...}

**Code-Beispiel VB.NET**

```
VcNet1.LegendView.TopActualValue = 40
```

**Code-Beispiel C#**

```
vcNet1.LegendView.TopActualValue = 40;
```

**UpdateBehaviorName****Nur-Lese-Eigenschaft von VcWorldView**

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.String	Name des Aktualisierungsverhaltens

## Visible

### Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann festgelegt oder erfragt werden, ob die Komplettansicht sichtbar ist. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Boolean	Komplettansicht sichtbar (True)/unsichtbar (False) <b>Standardwert:</b> False

#### Code-Beispiel VB.NET

```
VcNet1.WorldView.Visible = True
```

#### Code-Beispiel C#

```
vcNet1.WorldView.Visible = true;
```

## Width

### Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die horizontale Ausdehnung der Komplettansicht erfragt werden. In den Positionen **vcFixedAtLeft**, **vcFixedAtRight**, **vcNotFixed** und **vcPopupWindow** der Eigenschaft **Mode** kann diese Eigenschaft außerdem gesetzt werden. Die Koordinaten werden in Pixel, bezogen auf den Bildschirm, angegeben.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Zusätzliche Ansichten** festgelegt werden.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32	Horizontale Ausdehnung der Komplettansicht <b>Standardwert:</b> 100

#### Code-Beispiel VB.NET

```
VcNet1.WorldView.Width = 200
```

#### Code-Beispiel C#

```
vcNet1.WorldView.Width = 200;
```

## WidthActualValue

Nur-Lese-Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die tatsächlich dargestellte horizontale Ausdehnung der Komplettansicht erfragt werden. Dieser tatsächliche Wert kann in den Modi **vcLVFixedAtBottom**, **vcLVFixedAtLeft**, **vcLVFixedAtRight**, **vcLVFixedAtTop** von dem eingestellten Wert abweichen, da in diesen Modi je nach Einstellung die Höhe oder Breite vorgegeben ist.

	Datentyp	Beschreibung
<b>Eigenschaftswert</b>	System.Int32 {0, ...}	Tatsächliche horizontale Ausdehnung der Komplettansicht  {0, ...} <b>Standardwert:</b> 100

### Code-Beispiel VB.NET

```
VcNet1.LegendView.WidthActualValue = 600
```

### Code-Beispiel C#

```
vcNet1.LegendView.WidthActualValue = 600;
```



---



---

## 8 Index

### A

**AbsoluteBottomMarginInInches**

Eigenschaft von  
VcPrinter 743

**AbsoluteLeftMarginInCM**

Eigenschaft von  
VcPrinter 743

**AbsoluteLeftMarginInInches**

Eigenschaft von  
VcPrinter 744

**AbsoluteRightMarginInCM**

Eigenschaft von  
VcPrinter 744

**AbsoluteRightMarginInInches**

Eigenschaft von  
VcPrinter 745

**AbsoluteTopMarginInCM**

Eigenschaft von  
VcPrinter 745

**AbsoluteTopMarginInInches**

Eigenschaft von  
VcPrinter 746

**Active**

Eigenschaft von  
VcCalendarCollection 356

**ActiveNodeFilter**

Eigenschaft von  
VcNet 531

**ActualEndDateDataFieldIndex**

Eigenschaft von  
VcScheduler 766

**ActualStartDateDataFieldIndex**

Eigenschaft von

VcScheduler 766

**Add**

Methode von  
VcBoxCollection 319  
VcBoxFormatCollection 332  
VcCalendarCollection 358  
VcCalendarProfileCollection 367  
VcDataRecordCollection 379  
VcDataTableCollection 391  
VcDataTableFieldCollection 405  
VcFilterCollection 418  
VcIntervalCollection 448  
VcLinkAppearanceCollection 477  
VcLinkFormatCollection 492  
VcMapCollection 510  
VcNodeAppearanceCollection 707  
VcNodeFormatCollection 722

**AddBySpecification**

Methode von  
VcBoxCollection 319  
VcBoxFormatCollection 332  
VcCalendarCollection 358  
VcCalendarProfileCollection 367  
VcFilterCollection 419  
VcIntervalCollection 448  
VcLinkAppearanceCollection 478  
VcLinkFormatCollection 493  
VcMapCollection 511  
VcNodeAppearanceCollection 707  
VcNodeFormatCollection 723

**AddDuration**

Methode von  
VcCalendar 350

**AddSubCondition**



Methode von  
VcFilter 414

### **Alignment**

Eigenschaft von  
VcBoundingBox 297  
VcBoxFormatField 338  
VcLinkFormatField 498  
VcNodeFormatField 729  
VcPrinter 746

### **AllData**

Eigenschaft von  
VcDataRecord 372  
VcLink 460  
VcNode 675

### **Anordnen 272**

### **Anschlussknoten 158**

### **Arrange**

Methode von  
VcNet 580

### **Auslieferung 15**

### **AutomaticSchedulingEnabled**

Eigenschaft von  
VcScheduler 766

## **B**

### **BackgroundColor**

Eigenschaft von  
VcGroup 428  
VcNodeAppearance 684  
VcNodeFormatField 729

### **BackgroundColorDataFieldIndex**

Eigenschaft von  
VcNodeAppearance 685  
VcNodeFormatField 730

### **BackgroundColorMapName**

Eigenschaft von  
VcNodeAppearance 685

VcNodeFormatField 730

### **Benutzerkonten**

Problem mit Control 290

### **Bildschirmausschnitt**

verschieben 160

### **Border**

Eigenschaft von  
VcLegendView 452  
VcWorldView 773

### **BorderArea**

Eigenschaft von  
VcNet 532  
siehe auch  
VcBorderArea 295

### **BorderBox**

Ausrichtung 297  
Methode von  
VcBorderArea 295  
siehe auch  
VcBorderBox 297

### **Bottom**

Eigenschaft von  
VcRect 761

### **BottomMargin**

Eigenschaft von  
VcNodeFormatField 731

### **Box**

markieren 311  
Mehrfachmarkierung zulassen 155  
siehe auch  
VcBox 306  
über Index 320

### **BoxByIndex**

Methode von  
VcBoxCollection 320

### **BoxByName**

Methode von

VcBoxCollection 321

### **BoxCollection**

Eigenschaft von

VcNet 532

siehe auch

VcBoxCollection 318

### **Boxen 75**

Ausdehnung 315

Offset in Pixel umrechnen 315

Pixel in Offset umrechnen 317

### **Boxformat**

über Index 334

### **BoxFormat**

siehe auch

VcBoxFormat 325

### **BoxFormatCollection**

Eigenschaft von

VcNet 532

siehe auch

VcBoxFormatCollection 331

### **Boxformatfeld**

Ausrichtung 338

Hintergrundfarbe 343

Höhe Grafik 340

Index 340

maximale Zeilenzahl 341

Mindestbreite 342

minimale Zeilenzahl 341

Muster 344

Musterfarbe 343

Name des Formats 339

Schriftart 345

Schriftfarbe 345

Typ 346

### **BoxFormatField**

siehe auch

VcBoxFormatField 338

## C

### **CalcDuration**

Methode von

VcCalendar 350

### **Calendar**

siehe auch

VcCalendar 347

### **CalendarByIndex**

Methode von

VcCalendarCollection 359

### **CalendarByName**

Methode von

VcCalendarCollection 359

### **CalendarCollection**

Eigenschaft von

VcNet 533

siehe auch

VcCalendarCollection 356

### **CalendarProfile**

siehe auch

VcCalendarProfile 363

### **CalendarProfileByIndex**

Methode von

VcCalendarProfileCollection 368

### **CalendarProfileByName**

Methode von

VcCalendarProfileCollection 368

### **CalendarProfileCollection**

Eigenschaft von

VcCalendar 348

VcNet 533

siehe auch

VcCalendarProfileCollection 366

### **CalendarProfileName**

Eigenschaft von

VcInterval 441

**Clear**

- Methode von
  - VcCalendar 351
  - VcNet 580

**Clustering 101, 163**

**Color**

- Eigenschaft von
  - VcMapEntry 517

**ComparisonValueAsString**

- Eigenschaft von
  - VcFilterSubCondition 424

**CompleteViewMode**

- Methode von
  - VcNet 581

**ConnectionOperator**

- Eigenschaft von
  - VcFilterSubCondition 425

**ConsiderFilterEntries**

- Eigenschaft von
  - VcMap 502

**ConstantText**

- Eigenschaft von
  - VcLinkFormatField 499
  - VcNodeFormatField 731

**Copy**

- Methode von
  - VcBoxCollection 321
  - VcBoxFormatCollection 333
  - VcCalendarCollection 360
  - VcCalendarProfileCollection 368
  - VcDataTableCollection 392
  - VcDataTableFieldCollection 406
  - VcFilterCollection 419
  - VcIntervalCollection 449
  - VcLinkAppearanceCollection 478
  - VcLinkFormatCollection 493
  - VcMapCollection 511

- VcNodeAppearanceCollection 708
- VcNodeFormatCollection 723

**CopyFormatField**

- Methode von
  - VcBoxFormat 328
  - VcLinkFormat 489
  - VcNodeFormat 719

**CopyNodesIntoClipboard**

- Methode von
  - VcNet 581

**CopySubCondition**

- Methode von
  - VcFilter 414

**Count**

- Eigenschaft von
  - VcBoxCollection 318
  - VcBoxFormatCollection 331
  - VcCalendarCollection 357
  - VcCalendarProfileCollection 367
  - VcDataRecordCollection 378
  - VcDataTableCollection 390
  - VcDataTableFieldCollection 404
  - VcFilterCollection 417
  - VcGroupCollection 435
  - VcIntervalCollection 448
  - VcLinkAppearanceCollection 476
  - VcLinkCollection 483
  - VcLinkFormatCollection 491
  - VcMap 503
  - VcMapCollection 510
  - VcNodeAppearanceCollection 706
  - VcNodeCollection 712
  - VcNodeFormatCollection 721

**CreateEntry**

- Methode von
  - VcMap 505

**CtrlCXVProcessingEnabled**

Eigenschaft von  
VcNet 533

**CurrentHorizontalPagesCount**  
Eigenschaft von  
VcPrinter 747

**CurrentVerticalPagesCount**  
Eigenschaft von  
VcPrinter 747

**CurrentZoomFactor**  
Eigenschaft von  
VcPrinter 747

**CutNodesIntoClipboard**  
Methode von  
VcNet 581

**CuttingMarks**  
Eigenschaft von  
VcPrinter 748

## D

**DataDefinitionTable**  
Eigenschaft von  
VcFilter 411

**DataField**  
Eigenschaft von  
VcDataRecord 373  
VcLink 461  
VcNode 676

**DataFieldIndex**  
Eigenschaft von  
VcFilterSubCondition 426

**DataFieldValue**  
Eigenschaft von  
VcMapEntry 518

**DataRecord**  
Methode von  
VcLink 464  
VcNode 680

siehe auch  
VcDataRecord 372

**DataRecordByID**  
Methode von  
VcDataRecordCollection 380

**DataRecordCollection**  
Eigenschaft von  
VcDataTable 386  
siehe auch  
VcDataRecordCollection 378

**DataTable**  
siehe auch  
VcDataTable 386

**DataTableByIndex**  
Methode von  
VcDataTableCollection 392

**DataTableByName**  
Methode von  
VcDataTableCollection 393

**DataTableCollection**  
Eigenschaft von  
VcNet 534  
siehe auch  
VcDataTableCollection 390

**DataTableField**  
siehe auch  
VcDataTableField 397

**DataTableFieldByIndex**  
Methode von  
VcDataTableFieldCollection 406

**DataTableFieldByName**  
Methode von  
VcDataTableFieldCollection 407

**DataTableFieldCollection**  
Eigenschaft von  
VcDataTable 387  
siehe auch

- VcDataTableFieldCollection 404
- DataTableName**
  - Eigenschaft von
    - VcDataRecord 374
    - VcDataTableField 397
- DateFormat**
  - Eigenschaft von
    - VcDataTableField 398
- Daten**
  - aus Datei einlesen 32
- Datenfeld**
  - für Tooltiptext 166
- Datenfelder**
  - Knoten 257, 258
  - Verbindung 259
- Datensatz**
  - abhängiger Datensatz nicht gefunden 623
  - Aktualisierung 384
  - alle Daten 372
  - Anzahl in Collection 378
  - aus Collection entfernen 383
  - datenbasiertes Objekt 376
  - Datenfeld 373
  - Enumerator-Objekt 382
  - ID 375
  - Iteration, Erstwert 381
  - Iteration, Folgewert 383
  - löschen 375
  - Name der zugehörigen Tabelle 374
  - über ID** 381
  - zu Collection hinzufügen 379
  - zugeordneter Datensatz 376
- Datentabelle**
  - Aktualisierung 395
  - Anzahl in Collection 390
  - Auflistung 534
  - Beschreibung 387
  - Datensatz-Auflistung 386
  - Enumerator-Objekt 394
  - Erweiterte Datentabellen setzen 538
  - innerhalb der Collection kopieren 392
  - Iteration, Erstwert 394
  - Iteration, Folgewert 395
  - Name 583
  - Name 388
  - Tabellendatenfeld-Auflistung 387
  - über Index 392
  - über Name 393
  - zu Collection hinzufügen 391
- Datentabellen 79**
  - erweiterte Nutzung 154
- Datentabellenfeld**
  - angezeigt 399
  - Anzahl in Collection 404
  - Datentyp 403
  - Datumsformat 398
  - editierbar 399
  - Enumerator-Objekt 408
  - Index 584
  - Index 400
  - Index des Bezugfeldes 401
  - Iteration, Erstwert 408
  - Iteration, Folgewert 409
  - kopieren 406
  - Name 583
  - Name 400
  - Primärschlüssel 401
  - über Index 406
  - über Name 407
  - zu Collection hinzufügen 405
  - zugehöriger Tabellename 397
- DateOutputFormat**
  - Eigenschaft von

VcNet 534

**DatesWithHourAndMinute**  
Eigenschaft von  
VcFilter 411

**Datumsausgabeformat 152**

**DayInEndMonth**  
Eigenschaft von  
VcInterval 441

**DayInStartMonth**  
Eigenschaft von  
VcInterval 441

**DefaultPrinterName**  
Eigenschaft von  
VcPrinter 748

**Delete**  
Methode von  
VcDataRecord 375  
VcLink 464  
VcNode 680

**DeleteEntry**  
Methode von  
VcMap 506

**DeleteLinkRecord**  
Methode von  
VcNet 582

**DeleteNodeRecord**  
Methode von  
VcNet 582

**Description**  
Eigenschaft von  
VcDataTable 387

**DetectDataTableFieldName**  
Methode von  
VcNet 583

**DetectDataTableName**  
Methode von  
VcNet 583

**DetectFieldIndex**  
Methode von  
VcNet 584

**Deutsche Version 17**

**DiagramBackgroundColor**  
Eigenschaft von  
VcNet 536

**Diagramm**  
Ausrichtung 265  
exportieren 72, 273  
Hintergrundfarbe 152

**Dialogfeld**  
Boxen bearbeiten 205  
Boxen verwalten 202  
Boxformat bearbeiten 209  
Boxformate verwalten 207  
Datentabellen verwalten 179  
Druckvorschau 268  
Filter bearbeiten 184  
Filter verwalten 182  
In-Flow-Gruppierung bearbeiten 226  
Kalender festlegen 231  
Knotenaussehen bearbeiten 198  
Knotenaussehen verwalten 194  
Knotenformat bearbeiten 212  
Knotenformate verwalten 207  
Linie bearbeiten 229  
Lizenzierung 249  
Lizenzinformationen anfordern 251  
Muster 230  
Seite einrichten 264  
Texte, Grafiken und Legende festlegen 243  
Verbindung bearbeiten 259  
Verbindungsformat bearbeiten 219  
Zuordnung einstellen 192  
Zuordnungstabelle bearbeiten 190

Zuordnungstabellen verwalten 188

### **DialogFont**

Eigenschaft von  
VcNet 536

### **DocumentName**

Eigenschaft von  
VcPrinter 748

### **Doppelklick**

auf Knoten 257

### **Double-Ausgabeformat 153**

### **DoubleFeature**

Eigenschaft von  
VcNodeAppearance 686

### **DoubleOutputFormat**

Eigenschaft von  
VcNet 537

### **Drag & Drop 90**

### **Druckdatum 267**

### **Drucken 71, 272**

Absolute Breite des linken  
Seitenrandes in cm 743  
Absolute Breite des linken  
Seitenrandes in Zoll 744  
Absolute Breite des rechten  
Seitenrandes in cm 744  
Absolute Breite des rechten  
Seitenrandes in Zoll 745  
Absolute Höhe des oberen  
Seitenrandes in cm 745  
Absolute Höhe des oberen  
Seitenrandes in Zoll 746  
Absolute Höhe des unteren  
Seitenrandes in cm 760  
Absolute Höhe des unteren  
Seitenrandes in Zoll 743  
aktueller Drucker 748  
an Seitenzahlvorgabe anpassen 265  
Drucker einrichten 272  
Faltmarkierungen 751

in eine Datei 600  
Zoomfaktor 265, 747, 760

### **Druckvorschau 268, 272**

### **DumpConfiguration**

Methode von  
VcNet 584

### **DurationDataFieldIndex**

Eigenschaft von  
VcScheduler 767

## **E**

### **EarlyEndDateDataFieldIndex**

Eigenschaft von  
VcScheduler 767

### **EarlyStartDateDataFieldIndex**

Eigenschaft von  
VcScheduler 767

### **Editable**

Eigenschaft von  
VcDataTableField 399

### **Eigenschaften**

AbsoluteBottomMarginInInches  
VcPrinter 743  
AbsoluteLeftMarginInCM  
VcPrinter 743  
AbsoluteLeftMarginInInches  
VcPrinter 744  
AbsoluteRightMarginInCM  
VcPrinter 744  
AbsoluteRightMarginInInches  
VcPrinter 745  
AbsoluteTopMarginInCM  
VcPrinter 745  
AbsoluteTopMarginInInches  
VcPrinter 746  
Active  
VcCalendarCollection 356

- ActiveNodeFilter
  - VcNet 531
- ActualEndDateDataFieldIndex
  - VcScheduler 766
- ActualStartDateDataFieldIndex
  - VcScheduler 766
- Alignment
  - VcBoundingBox 297
  - VcBoxFormatField 338
  - VcLinkFormatField 498
  - VcNodeFormatField 729
  - VcPrinter 746
- AllData
  - VcDataRecord 372
  - VcLink 460
  - VcNode 675
- AutomaticSchedulingEnabled
  - VcScheduler 766
- BackgroundColor
  - VcGroup 428
  - VcNodeAppearance 684
  - VcNodeFormatField 729
- BackgroundColorDataFieldIndex
  - VcNodeAppearance 685
  - VcNodeFormatField 730
- BackgroundColorMapName
  - VcNodeAppearance 685
  - VcNodeFormatField 730
- Border
  - VcLegendView 452
  - VcWorldView 773
- BorderArea
  - VcNet 532
- Bottom
  - VcRect 761
- BottomMargin
  - VcNodeFormatField 731
- BoxCollection
  - VcNet 532
- BoxFormatCollection
  - VcNet 532
- CalendarCollection
  - VcNet 533
- CalendarProfileCollection
  - VcCalendar 348
  - VcNet 533
- CalendarProfileName
  - VcInterval 441
- Color
  - VcMapEntry 517
- ComparisonValueAsString
  - VcFilterSubCondition 424
- ConnectionOperator
  - VcFilterSubCondition 425
- ConsiderFilterEntries
  - VcMap 502
- ConstantText
  - VcLinkFormatField 499
  - VcNodeFormatField 731
- Count
  - VcBoxCollection 318
  - VcBoxFormatCollection 331
  - VcCalendarCollection 357
  - VcCalendarProfileCollection 367
  - VcDataRecordCollection 378
  - VcDataTableCollection 390
  - VcDataTableFieldCollection 404
  - VcFilterCollection 417
  - VcGroupCollection 435
  - VcIntervalCollection 448
  - VcLinkAppearanceCollection 476
  - VcLinkCollection 483
  - VcLinkFormatCollection 491
  - VcMap 503



- VcMapCollection 510
- VcNodeAppearanceCollection 706
- VcNodeCollection 712
- VcNodeFormatCollection 721
- CtrlCXVProcessingEnabled
  - VcNet 533
- CurrentHorizontalPagesCount
  - VcPrinter 747
- CurrentVerticalPagesCount
  - VcPrinter 747
- CurrentZoomFactor
  - VcPrinter 747
- CuttingMarks
  - VcPrinter 748
- DataDefinitionTable
  - VcFilter 411
- DataField
  - VcDataRecord 373
  - VcLink 461
  - VcNode 676
- DataFieldIndex
  - VcFilterSubCondition 426
- DataFieldValue
  - VcMapEntry 518
- DataRecordCollection
  - VcDataTable 386
- DataTableCollection
  - VcNet 534
- DataTableFieldCollection
  - VcDataTable 387
- DataTableName
  - VcDataRecord 374
  - VcDataTableField 397
- DateFormat
  - VcDataTableField 398
- DateOutputFormat
  - VcNet 534
- DatesWithHourAndMinute
  - VcFilter 411
- DayInEndMonth
  - VcInterval 441
- DayInStartMonth
  - VcInterval 441
- DefaultPrinterName
  - VcPrinter 748
- Description
  - VcDataTable 387
- DiagramBackgroundColor
  - VcNet 536
- DialogFont
  - VcNet 536
- DocumentName
  - VcPrinter 748
- DoubleFeature
  - VcNodeAppearance 686
- DoubleOutputFormat
  - VcNet 537
- DurationDataFieldIndex
  - VcScheduler 767
- EarlyEndDateDataFieldIndex
  - VcScheduler 767
- EarlyStartDateDataFieldIndex
  - VcScheduler 767
- Editable
  - VcDataTableField 399
- Enabled
  - VcNet 538
- EndDateForAutomaticScheduling
  - VcScheduler 768
- EndDateNotLaterThanDataFieldIndex
  - VcScheduler 768
- EndTime
  - VcInterval 442
- EndMonth

- VcInterval 442
- EndTime
  - VcInterval 442
- EndWeekday
  - VcInterval 443
- ExtendedDataTablesEnabled
  - VcNet 538
- FieldsSeparatedByLines
  - VcBoxFormat 325
  - VcNodeFormat 716
- FieldText
  - VcBox 307
- FilePath
  - VcNet 538
- FilterCollection
  - VcNet 539
- FilterName
  - VcFilterSubCondition 426
  - VcLinkAppearance 467
  - VcNodeAppearance 686
- FitToPage
  - VcPrinter 749
- FoldingMarksType
  - VcPrinter 749
- FontAntiAliasingEnabled
  - VcNet 540
- FontBody
  - VcMapEntry 519
- FontName
  - VcMapEntry 519
- FontSize
  - VcMapEntry 520
- FormatField
  - VcBoxFormat 326
  - VcLinkFormat 487
  - VcNodeFormat 717
- FormatFieldCount
  - VcBoxFormat 326
  - VcLinkFormat 488
  - VcNodeFormat 717
- FormatName
  - VcBox 307
  - VcBoxFormatField 339
  - VcLinkAppearance 468
  - VcLinkFormatField 499
  - VcNodeAppearance 687
  - VcNodeFormatField 731
- FrameAroundFieldsVisible
  - VcNodeAppearance 688
- FrameShape
  - VcNodeAppearance 688
- FreeFloatDataFieldIndex
  - VcScheduler 768
- GraphicsFileName
  - VcBoundingBox 298
  - VcMapEntry 521
  - VcNodeFormatField 731
- GraphicsFileNameDataFieldIndex
  - VcNodeFormatField 732
- GraphicsFileNameMapName
  - VcNodeFormatField 732
- GraphicsHeight
  - VcBoxFormatField 340
  - VcNodeFormatField 732
- GroupCollection
  - VcNet 540
- GroupHorizontalMargin
  - VcNet 541
- GroupingActivated
  - VcNet 541
- GroupingDataFieldIndex
  - VcNet 541
- GroupingTitlesFullyVisible
  - VcNet 542

- GroupingType
  - VcNet 543
- GroupInteractionsAllowed
  - VcNet 543
- GroupSortingDataFieldIndex
  - VcNet 544
- GroupSortMode
  - VcNet 544
- GroupTitleDataFieldIndex
  - VcNet 545
- GroupTitlesFileName
  - VcNet 545
- GroupVerticalMargin
  - VcNet 546
- Height
  - VcLegendView 453
  - VcRect 761
  - VcWorldView 774
- HeightActualValue
  - VcLegendView 453
  - VcWorldView 774
- Hidden
  - VcDataTableField 399
- ID
  - VcDataRecord 375
  - VcLink 462
  - VcNode 677
- InbuiltMouseCursorWhileDraggingEnabled
  - VcNet 547
- IncomingLinks
  - VcNode 677
- Index
  - VcBoxFormatField 340
  - VcDataTableField 400
  - VcFilterSubCondition 426
  - VcLinkFormatField 499
  - VcNodeFormatField 733
- InFlowGroupingActivated
  - VcNet 547
- InFlowGroupingDataFieldIndex
  - VcNet 548
- InFlowGroupSeparationLineColor
  - VcNet 548
- InFlowGroupSeparationLineType
  - VcNet 549
- InFlowGroupTimeInterval
  - VcNet 550
- InFlowGroupTitleDataFieldIndex
  - VcNet 550
- InFlowGroupTitlesBackgroundColor
  - VcNet 551
- InFlowGroupTitlesFileName
  - VcNet 551
- InFlowGroupTitlesFont
  - VcNet 552
- InFlowGroupTitlesVisibleAtBottomOrRight
  - VcNet 552
- InFlowGroupTitlesVisibleAtTopOrLeft
  - VcNet 552
- InFlowGroupTitleTimeFormat
  - VcNet 553
- InFlowGroupVerticalCaptionWidth
  - VcNet 553
- InPlaceEditingAllowed
  - VcNet 553
- InteractionMode
  - VcNet 554
- InterfaceNodesShown
  - VcNet 554
- IntervalCollection
  - VcCalendar 348
  - VcCalendarProfile 363

- LateEndDateDataFieldIndex
  - VcScheduler 768
- LateStartDateDataFieldIndex
  - VcScheduler 769
- Left
  - VcLegendView 454
  - VcRect 762
  - VcWorldView 775
- LeftActualValue
  - VcLegendView 454
  - VcWorldView 775
- LeftMargin
  - VcNodeFormatField 733
- LegendElementsArrangement
  - VcBoundingBox 299
- LegendElementsBottomMargin
  - VcBoundingBox 299
- LegendElementsMaximumColumnCount
  - VcBoundingBox 300
- LegendElementsMaximumRowCount
  - VcBoundingBox 300
- LegendElementsTopMargin
  - VcBoundingBox 300
- LegendFont
  - VcBoundingBox 300
- LegendText
  - VcNodeAppearance 690
- LegendTitle
  - VcBoundingBox 301
- LegendTitleFont
  - VcBoundingBox 301
- LegendTitleVisible
  - VcBoundingBox 302
- LegendView
  - VcNet 555
- LineColor
  - VcBox 308
  - VcGroup 429
  - VcLinkAppearance 468
  - VcNodeAppearance 690
- LineColorDataFieldIndex
  - VcNodeAppearance 691
- LineColorMapName
  - VcNodeAppearance 691
- LineThickness
  - VcBox 308
  - VcGroup 429
  - VcLinkAppearance 469
  - VcNodeAppearance 691
- LineType
  - VcBox 309
  - VcGroup 430
  - VcLinkAppearance 470
  - VcNodeAppearance 692
- LinkAnnotationColumnNumberDataFieldIndex
  - VcNet 555
- LinkAnnotationRowNumberDataFieldIndex
  - VcNet 556
- LinkAppearanceCollection
  - VcNet 556
- LinkCollection
  - VcNet 557
- LinkCreationWithDialog
  - VcNet 557
- LinkDurationDataFieldIndex
  - VcScheduler 769
- LinkFormatCollection
  - VcNet 558
- LinkPredecessorDataFieldIndex
  - VcNet 558
- LinksDataTableName

- VcNet 559
- LinkSuccessorDataFieldIndex
  - VcNet 560
- LinkTypeDataFieldIndex
  - VcNet 561
- MapCollection
  - VcNet 562
- MarginsShownInInches
  - VcPrinter 752
- Marked
  - VcBox 311
  - VcLink 462
  - VcNode 678
- MarkedNodesFilter
  - VcFilterCollection 418
- MarkingColor
  - VcWorldView 776
- MaxHorizontalPagesCount
  - VcPrinter 752
- MaximumTextLineCount
  - VcBoxFormatField 341
  - VcNodeFormatField 733
- MaxVerticalPagesCount
  - VcPrinter 753
- MinimumColumnWidth
  - VcNet 563
- MinimumRowHeight
  - VcNet 563
- MinimumTextLineCount
  - VcBoxFormatField 341
  - VcNodeFormatField 734
- MinimumWidth
  - VcBoxFormatField 342
  - VcLinkFormatField 500
  - VcNodeFormatField 734
- Mode
  - VcWorldView 776
- MouseProcessingEnabled
  - VcNet 564
- Moveable
  - VcBox 311
- MovingCollapsedClustersAllowed
  - VcNet 564
- MultiplePrimaryKeysAllowed
  - VcDataTable 388
- Name
  - VcBox 312
  - VcBoxFormat 327
  - VcCalendar 348
  - VcCalendarProfile 364
  - VcDataTable 388
  - VcDataTableField 400
  - VcFilter 411
  - VcGroup 431
  - VcInterval 443
  - VcLinkAppearance 471
  - VcLinkFormat 488
  - VcMap 503
  - VcNodeAppearance 694
  - VcNodeFormat 718
- NodeAndLinkCreationAllowed
  - VcNet 565
- NodeAppearanceCollection
  - VcNet 565
- NodeCalendarNameDataFieldIndex
  - VcNet 566
- NodeChangeRankToPredecessorRankDataFieldIndex
  - VcNet 566
- NodeCollection
  - VcGroup 432
  - VcNet 566
- NodeColumnNumberDataFieldIndex
  - VcNet 567

- NodeCreationWithDialog
  - VcNet 567
- NodeFormatCollection
  - VcNet 567
- NodeRowNumberDataFieldIndex
  - VcNet 568
- NodesDataTableName
  - VcNet 569
- NodesUseCalendars
  - VcNet 569
- NodeToolTipTextDataFieldIndex
  - VcNet 570
- Number
  - VcMapEntry 522
- ObliqueTracksOnLinks
  - VcNet 570
- Operator
  - VcFilterSubCondition 426
- Orientation
  - VcNet 571
  - VcPrinter 753
- Origin
  - VcBox 312
- OutgoingLinks
  - VcNode 679
- PageDescription
  - VcPrinter 754
- PageDescriptionString
  - VcPrinter 754
- PageFrame
  - VcPrinter 755
- PageNumberMode
  - VcPrinter 755
- PageNumbers
  - VcPrinter 756
- PagePaddingEnabled
  - VcPrinter 756
- PaperSize
  - VcPrinter 757
- ParentHWND
  - VcWorldView 777
- Pattern
  - VcMapEntry 522
  - VcNodeAppearance 694
- PatternBackgroundColor
  - VcBoxFormatField 342
- PatternBackgroundColorAsARGB
  - VcNodeFormatField 734
- PatternBackgroundColorDataFieldIndex
  - VcNodeFormatField 735
- PatternBackgroundColorMapName
  - VcNodeFormatField 735
- PatternColor
  - VcNodeAppearance 698
- PatternColorAsARGB
  - VcBoxFormatField 343
  - VcNodeFormatField 735
- PatternColorDataFieldIndex
  - VcNodeAppearance 698
  - VcNodeFormatField 736
- PatternColorMapName
  - VcNodeAppearance 699
  - VcNodeFormatField 736
- PatternDataFieldIndex
  - VcNodeAppearance 699
- PatternEx
  - VcBoxFormatField 344
  - VcNodeFormatField 737
- PatternExDataFieldIndex
  - VcNodeFormatField 737
- PatternExMapName
  - VcNodeFormatField 738
- PatternMapName

- VcNodeAppearance 699
- PhantomDrawingWhileDraggingEnabled
  - VcNet 571
- PileEffect
  - VcNodeAppearance 700
- PredecessorNode
  - VcLink 462
- PredecessorPortSymbol
  - VcLinkAppearance 472
- PrimaryKey
  - VcDataTableField 401
- PrintDate
  - VcPrinter 758
- Printer
  - VcNet 572
- PrinterName
  - VcPrinter 758
- PrintPreviewWithFirstPage
  - VcPrinter 758
- Priority
  - VcBox 313
- ReferencePoint
  - VcBox 313
- RelationshipFieldIndex
  - VcDataTableField 401
- Right
  - VcRect 763
- RightMargin
  - VcNodeFormatField 738
- RoundedLinkSlantsEnabled
  - VcNet 572
- RoutingType
  - VcLinkAppearance 472
- ScheduledProjectEndDate
  - VcScheduler 769
- ScheduledProjectStartDate
  - VcScheduler 770
- Scheduler
  - VcNet 573
- ScheduleSuccessorsOnlyEnabled
  - VcScheduler 770
- ScrollBarMode
  - VcLegendView 455
  - VcWorldView 778
- SecondsPerWorkday
  - VcCalendar 349
- Shadow
  - VcNodeAppearance 700
- ShadowColor
  - VcNodeAppearance 701
- ShortenedLinks
  - VcNet 573
- Specification
  - VcBoxFormat 327
  - VcCalendar 349
  - VcCalendarProfile 364
  - VcFilter 412
  - VcInterval 443
  - VcLinkAppearance 473
  - VcLinkFormat 488
  - VcMap 504
  - VcNodeAppearance 701
  - VcNodeFormat 718
- StartDateForAutomaticScheduling
  - VcScheduler 770
- StartDateNotEarlierThanDataFieldIndex
  - VcScheduler 771
- StartDateTime
  - VcInterval 444
- StartMonth
  - VcInterval 444
- StartTime

- VcInterval 444
- StartWeekday
  - VcInterval 445
- StraightLinkDrawing
  - VcNet 573
- StrikeThrough
  - VcNodeAppearance 702
- StrikeThroughColor
  - VcNodeAppearance 703
- StringsCaseSensitive
  - VcFilter 412
- SubCondition
  - VcFilter 413
- SubConditionCount
  - VcFilter 413
- SuccessorNode
  - VcLink 463
- SuccessorPortSymbol
  - VcLinkAppearance 473
- Text
  - VcBoundingBox 302
- TextAndGraphicsCombined
  - VcNodeFormatField 738
- TextDataFieldIndex
  - VcLinkFormatField 500
  - VcNodeFormatField 739
- TextEntrySupplyingEventEnabled
  - VcNet 574
- TextFont
  - VcBoundingBox 303
  - VcBoxFormatField 345
  - VcLinkFormatField 500
  - VcNodeFormatField 739
- TextFontColor
  - VcBoxFormatField 345
  - VcLinkFormatField 501
  - VcNodeFormatField 739
- TextFontDataFieldIndex
  - VcNodeFormatField 740
- TextFontMapName
  - VcNodeFormatField 740
- TextLineCount
  - VcLinkFormatField 501
- ThreeDEffect
  - VcNodeAppearance 703
- TimeUnit
  - VcNet 574
- Title
  - VcGroup 432
- TitleAndLegendOnAllPages
  - VcPrinter 759
- TitleLineCount
  - VcGroup 433
- ToolTipChangeDuration
  - VcNet 575
- ToolTipDuration
  - VcNet 575
- ToolTipPointerDuration
  - VcNet 576
- ToolTipShowAfterClick
  - VcNet 576
- ToolTipTextSupplyingEventEnabled
  - VcNet 576
- Top
  - VcLegendView 455
  - VcRect 763
  - VcWorldView 778
- TopActualValue
  - VcLegendView 456
  - VcWorldView 779
- TopMargin
  - VcNodeFormatField 740
- TotalFloatDataFieldIndex
  - VcScheduler 771



- Type
  - VcBoundingBox 304
  - VcBoxFormatField 346
  - VcCalendarProfile 364
  - VcDataTableField 403
  - VcInterval 445
  - VcMap 504
  - VcNodeFormatField 741
- UngroupedNodesAllowed
  - VcNet 577
- UpdateBehaviorName
  - VcBox 314
  - VcWorldView 779
- VcCalendarGrid
  - VcPrinter 760
- ViewXCoordinate
  - VcNet 577
- ViewYCoordinate
  - VcNet 578
- Visible
  - VcBox 314
  - VcLegendView 456
  - VcLinkAppearance 474
  - VcWorldView 780
- VisibleInLegend
  - VcNodeAppearance 704
- WaitCursorEnabled
  - VcNet 578
- Width
  - VcLegendView 457
  - VcRect 764
  - VcWorldView 780
- WidthActualValue
  - VcLegendView 457
  - VcWorldView 781
- WidthOfExteriorSurrounding
  - VcNodeFormat 719
- WindowMode
  - VcLegendView 458
- WorldView
  - VcNet 578
- X
  - VcGroup 433
- Y
  - VcGroup 433
- ZoomFactor
  - VcNet 579
- ZoomFactorAsDouble
  - VcPrinter 760
- ZoomingPerMouseWheelAllowed
  - VcNet 579
- Eigenschaftenseite**
  - Allgemeines 151
  - Außenbereich 161
  - Gruppierung 163
  - Knoten 166
  - Objekte 173
  - Verbindungen 175
  - Zeitrechnung 177
  - Zusätzliche Ansichten 169
- Enabled**
  - Eigenschaft von
    - VcNet 538
- EndDateForAutomaticScheduling**
  - Eigenschaft von
    - VcScheduler 768
- EndDateNotLaterThanDataFieldIndex**
  - Eigenschaft von
    - VcScheduler 768
- EndDateTime**
  - Eigenschaft von
    - VcInterval 442
- EndLoading**
  - Methode von

- VcNet 585
- EndMonth**
  - Eigenschaft von
  - VcInterval 442
- EndTime**
  - Eigenschaft von
  - VcInterval 442
- EndWeekday**
  - Eigenschaft von
  - VcInterval 443
- Ereignisobjekte**
  - VcBoxModifiedEventArgs 614
  - VcBoxModifyingEventArgs 615
  - VcDataRecordCreatedEventArgs 618
  - VcDataRecordCreatingEventArgs 619
  - VcDataRecordDeletedEventArgs 620
  - VcDataRecordDeletingEventArgs 621
  - VcDataRecordModifiedEventArgs 622
  - VcDataRecordModifyingEventArgs 623
  - VcDataRecordNotFoundEventArgs 624
  - VcDragCompletingEventArgs 626
  - VcDragStartingEventArgs 627
  - VcFieldSelectingEventArgs 628
  - VcHelpRequestedEventArgs 635
  - VcLEgendViewClosedEventArgs 638
  - VcNodeModifiedEventArgs 655, 656
  - VcTextEntrySupplyingEventArgs 661
- Ereignisse 92**
  - VcBoxLeftClicking
    - VcNet 613
  - VcBoxLeftDoubleClicking
    - VcNet 613
  - VcBoxModified
    - VcNet 614
  - VcBoxModifying
    - VcNet 615
  - VcBoxRightClicking
    - VcNet 616
  - VcDataModified
    - VcNet 617
  - VcDataRecordCreated
    - VcNet 618
  - VcDataRecordCreating
    - VcNet 619
  - VcDataRecordDeleted
    - VcNet 620
  - VcDataRecordDeleting
    - VcNet 621
  - VcDataRecordModified
    - VcNet 622
  - VcDataRecordModifying
    - VcNet 622
  - VcDataRecordNotFound
    - VcNet 623
  - VcDiagramLeftClicking
    - VcNet 624
  - VcDiagramLeftDoubleClicking
    - VcNet 625
  - VcDiagramRightClicking
    - VcNet 625
  - VcDragCompleting
    - VcNet 626
  - VcDragStarting
    - VcNet 627
  - VcErrorOccurring
    - VcNet 627
  - VcFieldSelecting
    - VcNet 628
  - VcGiveFeedbackOnNodeCreating
    - VcNet 629

VcGroupCreated	VcNet 630	VcNet 645
VcGroupDeleting	VcNet 630	VcLinksMarking
VcGroupLeftClicking	VcNet 631	VcNet 646
VcGroupLeftDoubleClicking	VcNet 632	VcLinksRightClicking
VcGroupModified	VcNet 632	VcNet 647
VcGroupModifying	VcNet 633	VcMouseDoubleClicking
VcGroupRightClicking	VcNet 634	VcNet 648
VcHelpRequested	VcNet 635	VcMouseDown
VcInPlaceEditorShowing	VcNet 636	VcNet 648
VcLegendViewClosed	VcNet 638	VcMouseMove
VcLinkCreated	VcNet 639	VcNet 649
VcLinkCreating	VcNet 640	VcMouseUp
VcLinkDeleted	VcNet 641	VcNet 650
VcLinkDeleting	VcNet 641	VcNodeCreated
VcLinkModified	VcNet 642	VcNet 651
VcLinkModifying	VcNet 642	VcNodeCreating
VcLinksLeftClicking	VcNet 643	VcNet 651
VcLinksLeftDoubleClicking	VcNet 644	VcNodeDeleted
VcLinksMarked		VcNet 652
		VcNodeDeleting
		VcNet 653
		VcNodeLeftClicking
		VcNet 653
		VcNodeLeftDoubleClicking
		VcNet 654
		VcNodeModifiedEx
		VcNet 655
		VcNodeModifying
		VcNet 656
		VcNodeRightClicking
		VcNet 657
		VcNodesMarked
		VcNet 658
		VcNodesMarking
		VcNet 659
		VcStatusLineTextShowing
		VcNet 660

- VcTextEntrySupplying
    - VcNet 661
  - VcToolTipTextSupplying
    - VcNet 671
  - VcWorldViewClosed
    - VcNet 673
  - VcZoomFactorModified
    - VcNet 673
  - Erzeugemodus 271**
  - Evaluate**
    - Methode von
      - VcFilter 415
  - Export 273**
  - ExportGraphicsToFileEx**
    - Methode von
      - VcNet 585
  - ExtendedDataTablesEnabled**
    - Eigenschaft von
      - VcNet 538
- F**
- Fehlermeldungen 289**
  - FieldsSeparatedByLines**
    - Eigenschaft von
      - VcBoxFormat 325
      - VcNodeFormat 716
  - FieldText**
    - Eigenschaft von
      - VcBox 307
  - FilePath**
    - Eigenschaft von
      - VcNet 538
  - Filter 93**
    - bearbeiten 184
    - für Knoten 40
    - siehe auch
      - VcFilter 410
      - über Index 420
    - Vergleichswert 185
    - verwalten 182
  - FilterByIndex**
    - Methode von
      - VcFilterCollection 420
  - FilterByName**
    - Methode von
      - VcFilterCollection 420
  - FilterCollection**
    - Eigenschaft von
      - VcNet 539
    - siehe auch
      - VcFilterCollection 417
  - FilterName**
    - Eigenschaft von
      - VcFilterSubCondition 426
      - VcLinkAppearance 467
      - VcNodeAppearance 686
  - FilterSubCondition**
    - siehe auch
      - VcFilterSubCondition 424
  - FirstBox**
    - Methode von
      - VcBoxCollection 322
  - FirstCalendar**
    - Methode von
      - VcCalendarCollection 360
  - FirstCalendarProfile**
    - Methode von
      - VcCalendarProfileCollection 369
  - FirstDataRecord**
    - Methode von
      - VcDataRecordCollection 381
  - FirstDataTable**
    - Methode von
      - VcDataTableCollection 393

**FirstDataTableField**

Methode von  
VcDataTableFieldCollection 407

**FirstFilter**

Methode von  
VcFilterCollection 421

**FirstFormat**

Methode von  
VcBoxFormatCollection 333  
VcLinkFormatCollection 494  
VcNodeFormatCollection 724

**FirstGroup**

Methode von  
VcGroupCollection 436

**FirstInterval**

Methode von  
VcIntervalCollection 449

**FirstLink**

Methode von  
VcLinkCollection 484

**FirstLinkAppearance**

Methode von  
VcLinkAppearanceCollection 478

**FirstMap**

Methode von  
VcMapCollection 512

**FirstMapEntry**

Methode von  
VcMap 506

**FirstNode**

Methode von  
VcNodeCollection 713

**FirstNodeAppearance**

Methode von  
VcNodeAppearanceCollection 708

**FitToPage**

Eigenschaft von

VcPrinter 749

**Flußrichtung 35**

**Flussrichtung 151**

**FoldingMarksType**

Eigenschaft von  
VcPrinter 749

**FontAntiAliasingEnabled**

Eigenschaft von  
VcNet 540

**FontBody**

Eigenschaft von  
VcMapEntry 519

**FontName**

Eigenschaft von  
VcMapEntry 519

**Fonts 291**

**FontSize**

Eigenschaft von  
VcMapEntry 520

**FormatByIndex**

Methode von  
VcBoxFormatCollection 334  
VcLinkFormatCollection 494  
VcNodeFormatCollection 724

**FormatByName**

Methode von  
VcBoxFormatCollection 334  
VcLinkFormatCollection 495  
VcNodeFormatCollection 725

**FormatField**

Eigenschaft von  
VcBoxFormat 326  
VcLinkFormat 487  
VcNodeFormat 717

**FormatFieldCount**

Eigenschaft von  
VcBoxFormat 326

VcLinkFormat 488

VcNodeFormat 717

### **FormatName**

Eigenschaft von

VcBox 307

VcBoxFormatField 339

VcLinkAppearance 468

VcLinkFormatField 499

VcNodeAppearance 687

VcNodeFormatField 731

### **Formular**

anpassen 23

### **FrameAroundFieldsVisible**

Eigenschaft von

VcNodeAppearance 688

### **FrameShape**

Eigenschaft von

VcNodeAppearance 688

### **FreeFloatDataFieldIndex**

Eigenschaft von

VcScheduler 768

## **G**

### **Gesamtnetz 273, 275**

#### **GetActualExtent**

Methode von

VcBox 315

#### **GetAValueFromARGB**

Methode von

VcNet 588

#### **GetBValueFromARGB**

Methode von

VcNet 588

#### **GetEndOfPreviousWorktime**

Methode von

VcCalendar 351

#### **GetEnumerator**

Methode von

VcBoxCollection 322

VcBoxFormat 329

VcBoxFormatCollection 335

VcCalendarCollection 360

VcDataRecordCollection 382

VcDataTableCollection 394

VcDataTableFieldCollection 408

VcFilter 415

VcFilterCollection 421

VcGroupCollection 436

VcLinkAppearanceCollection 479

VcLinkCollection 484

VcLinkFormat 490

VcLinkFormatCollection 495

VcMapCollection 512

VcNodeAppearanceCollection 709

VcNodeCollection 713

VcNodeFormat 720

VcNodeFormatCollection 725

#### **GetGValueFromARGB**

Methode von

VcNet 589

#### **GetLinkByID**

Methode von

VcNet 590

#### **GetLinkByNodeIDs**

Methode von

VcNet 591

#### **GetMapEntry**

Methode von

VcMap 507

#### **GetNextIntervalBorder**

Methode von

VcCalendar 352

#### **GetNodeByID**

Methode von

- VcNet 591
- GetPreviousIntervalBorder**
  - Methode von
    - VcCalendar 352
- GetRValueFromARGB**
  - Methode von
    - VcNet 592
- GetStartOfInterval**
  - Methode von
    - VcCalendar 353
- GetStartOfNextWorktime**
  - Methode von
    - VcCalendar 354
- GetTopLeftPixel**
  - Methode von
    - VcBox 315
- GetXYOffset**
  - Methode von
    - VcBox 316
- Grafik**
  - exportieren 72
- Grafik exportieren 273**
- Grafikformat 95**
- GraphicsFileName**
  - Eigenschaft von
    - VcBoundingBox 298
    - VcMapEntry 521
    - VcNodeFormatField 731
- GraphicsFileNameDataFieldIndex**
  - Eigenschaft von
    - VcNodeFormatField 732
- GraphicsFileNameMapName**
  - Eigenschaft von
    - VcNodeFormatField 732
- GraphicsHeight**
  - Eigenschaft von
    - VcBoxFormatField 340
    - VcNodeFormatField 732
- Group**
  - siehe auch
    - VcGroup 428
- GroupByName**
  - Methode von
    - VcGroupCollection 437
- GroupCollection**
  - Eigenschaft von
    - VcNet 540
  - siehe auch
    - VcGroupCollection 435
- GroupHorizontalMargin**
  - Eigenschaft von
    - VcNet 541
- GroupingActivated**
  - Eigenschaft von
    - VcNet 541
- GroupingDataFieldIndex**
  - Eigenschaft von
    - VcNet 541
- GroupingTitlesFullyVisible**
  - Eigenschaft von
    - VcNet 542
- GroupingType**
  - Eigenschaft von
    - VcNet 543
- GroupInteractionsAllowed**
  - Eigenschaft von
    - VcNet 543
- GroupSortingDataFieldIndex**
  - Eigenschaft von
    - VcNet 544
- GroupSortMode**
  - Eigenschaft von
    - VcNet 544
- GroupTitleDataFieldIndex**

Eigenschaft von

VcNet 545

### **GroupTitlesFileName**

Eigenschaft von

VcNet 545

### **GroupVerticalMargin**

Eigenschaft von

VcNet 546

### **Gruppe**

Linienstärke 430

### **Gruppen**

Hintergrundfarbe 164

Linienfarbe 429

Ränder 164

Randlinie 164

Titel 165

### **Gruppencode 163**

### **Gruppierung 100, 101, 163**

Linientyp 430

## H

### **Height**

Eigenschaft von

VcLegendView 453

VcRect 761

VcWorldView 774

### **HeightActualValue**

Eigenschaft von

VcLegendView 453

VcWorldView 774

### **Hidden**

Eigenschaft von

VcDataTableField 399

### **Hilfsknoten 109**

übersichtlich anordnen 58

### **Hintergrundfarbe 152**

## ID

Eigenschaft von

VcDataRecord 375

VcLink 462

VcNode 677

### **IdentifyFormatField**

Methode von

VcBox 316

VcNet 592

### **IdentifyObject**

Methode von

VcDataRecord 376

### **IdentifyObjectAt**

Methode von

VcNet 593

### **ImportConfiguration**

Methode von

VcNet 595

### **InbuiltMouseCursorWhileDraggingEnabled**

Eigenschaft von

VcNet 547

### **IncomingLinks**

Eigenschaft von

VcNode 677

### **Index**

Eigenschaft von

VcBoxFormatField 340

VcDataTableField 400

VcFilterSubCondition 426

VcLinkFormatField 499

VcNodeFormatField 733

### **InFlowGroupingActivated**

Eigenschaft von

VcNet 547



**InFlowGroupingDataFieldIndex**

Eigenschaft von

VcNet 548

**InFlowGroupSeparationLineColor**

Eigenschaft von

VcNet 548

**InFlowGroupSeparationLineType**

Eigenschaft von

VcNet 549

**InFlowGroupTimeInterval**

Eigenschaft von

VcNet 550

**InFlowGroupTitleDataFieldIndex**

Eigenschaft von

VcNet 550

**InFlowGroupTitlesBackgroundColor**

Eigenschaft von

VcNet 551

**InFlowGroupTitlesFileName**

Eigenschaft von

VcNet 551

**InFlowGroupTitlesFont**

Eigenschaft von

VcNet 552

**InFlowGroupTitlesVisibleAtBottomOr  
Right**

Eigenschaft von

VcNet 552

**InFlowGroupTitlesVisibleAtTopOrLeft**

Eigenschaft von

VcNet 552

**InFlowGroupTitleTimeFormat**

Eigenschaft von

VcNet 553

**InFlowGroupVerticalCaptionWidth**

Eigenschaft von

VcNet 553

**In-Flow-Gruppierung 103, 168, 226**

**InPlaceEditingAllowed**

Eigenschaft von

VcNet 553

**InsertLinkRecord**

Methode von

VcNet 596

**InsertNodeRecord**

Methode von

VcNet 596

**Installation 12**

**InteractionMode**

Eigenschaft von

VcNet 554

**InterfaceNodesShown**

Eigenschaft von

VcNet 554

**Internet 72, 141, 250, 273**

**Interval**

siehe auch

VcInterval 439

**IntervalByIndex**

Methode von

VcIntervalCollection 450

**IntervalByName**

Methode von

VcIntervalCollection 450

**IntervalCollection**

Eigenschaft von

VcCalendar 348

VcCalendarProfile 363

siehe auch

VcIntervalCollection 447

**Intervall**

Anzahl 448

Enddatum und -zeit 442

Endmonat 442

Endzeit 442  
 erster Wochentag 445  
 erstes Intervall 449  
 hinzufügen 448  
 Kalenderprofil 441  
 kopieren 449  
 letzter Wochentag 443  
 löschen 451  
 nächstes Intervall 450  
 Name 443  
 Reihenfolge 445  
 Startdatum und -zeit 444  
 Startmonat 444  
 Startzeit 444  
 Tag des ersten Monats 441  
 Tag des letzten Monats 441  
 Typ 445  
 über Index 450  
 Zugriff über Intervallnamen 450

### **Intervall-Collection**

aktualisieren 451

### **Intervalle**

bearbeiten 233, 236, 238, 239, 241

### **IsValid**

Methode von  
     VcFilter 416  
     VcFilterSubCondition 427

### **IsWorktime**

Methode von  
     VcCalendar 354

## K

### **Kalender**

Anzahl der Sekunden eines  
     Arbeitstages 349  
 Name 364  
 über Index 359

### **Kalenderprofil**

Anzahl 367  
 Reihenfolge 365  
 Typ 364  
 über Index 368  
 Zugriff über Kalenderprofilnamen 368

### **Kalenderprofil verwalten**

Dialogfeld: 235

### **Knoten 105**

3D-Effekt 199  
 auf dem Rang ihrer Vorgänger  
     anordnen 167  
 Datensatz 464, 680  
 Doppelrahmen 198, 199  
 erzeugen und bearbeiten 37  
 Gestaffelt 201  
 gruppieren 63  
 Hilfsknoten 109  
 Hintergrundfarbe 199  
 ID 677  
 interaktiv erzeugen 284, 286  
 Knotenaussehen 43, 111  
 Knotenaussehen bearbeiten 198  
 Knotenform 198  
 Knotenformat 47, 113  
 markieren 39  
 Markierungstyp 168  
 neue Knoten bearbeiten 156  
 neue zulassen 156  
 nicht durchtrennen 265  
 Positionen 106  
 Positionen mit Datenfeldern  
     synchronisieren 167  
 Rang 107  
 Schatten 200  
 zugeordneter Datensatz 465, 681

### **Knoten und Verbindungen**

- erzeugen 261
- markieren und verschieben 263
- Knotenaussehen**
  - 3D-Effekt 703
  - Doppelte Umrahmung 686
  - Durchstreichmuster 702
  - Durchstreichmuster, Farbe 703
  - Filter 686
  - Format 468, 687
  - Hintergrundfarbe 685
  - Knotenstapel 700
  - Legende 704
  - Legendentext 690
  - Liniendicke 692
  - Linienfarbe 690, 691
  - Linienfarbenzuordnungstabelle 691
  - Linientyp 693
  - Name 694
  - Rahmen um Felder 688
  - Rahmenform 688
  - Schatten 700
  - Sortierung 704
  - Spezifikation 473, 701
- Knotenaussehen-Auflistung**
  - Anzahl 706
  - Enumerator 709
  - erstes Knotenaussehen 708
  - hinzufügen 707
  - hinzufügen über Spezifikation 707
  - kopieren 708
  - löschen 711
  - nächstes Knotenaussehen 710
  - Zugriff über Index 710
  - Zugriff über Name 711
- Knotenformat-Auflistung**
  - Anzahl 721
  - Enumerator 725

- Erstes Format 724
- hinzufügen 723
- hinzufügen über Spezifikation 723
- kopieren 723
- löschen 727
- nächstes Format 726
- Zugriff über Index 724
- zugriff über Name 725
- Knotenformatfeld**
  - Hintergrundfarbe 735
  - Musterfarbe 736
- Knotenpositionen 55**
- Komplettansicht 116, 273**
- Konfiguration 73, 154**
  - speichern 584
- Kontextmenü**
  - abschalten 287
  - für das Diagramm 271
  - für Knoten 275
  - für Verbindungen 276
- Kundendienst 19**

**L**

- LateEndDateDataFieldIndex**
  - Eigenschaft von
    - VcScheduler 768
- LateStartDateDataFieldIndex**
  - Eigenschaft von
    - VcScheduler 769
- Leerseiten unterdrücken 265**
- Left**
  - Eigenschaft von
    - VcLegendView 454
    - VcRect 762
    - VcWorldView 775
- LeftActualValue**
  - Eigenschaft von

- VcLegendView 454
- VcWorldView 775
- LeftMargin**
  - Eigenschaft von
  - VcNodeFormatField 733
- Legende**
  - Anordnung 247
  - Anordnung 248
  - erweiterte Attribute 247
  - Schrift 248
  - Titel 247
- LegendElementsArrangement**
  - Eigenschaft von
  - VcBoundingBox 299
- LegendElementsBottomMargin**
  - Eigenschaft von
  - VcBoundingBox 299
- LegendElementsMaximumColumnCount**
  - Eigenschaft von
  - VcBoundingBox 300
- LegendElementsMaximumRowCount**
  - Eigenschaft von
  - VcBoundingBox 300
- LegendElementsTopMargin**
  - Eigenschaft von
  - VcBoundingBox 300
- Legendenansicht 120, 273**
- LegendFont**
  - Eigenschaft von
  - VcBoundingBox 300
- LegendText**
  - Eigenschaft von
  - VcNodeAppearance 690
- LegendTitle**
  - Eigenschaft von
  - VcBoundingBox 301
- LegendTitleFont**
  - Eigenschaft von
  - VcBoundingBox 301
- LegendTitleVisible**
  - Eigenschaft von
  - VcBoundingBox 302
- Legendview 120**
- LegendView**
  - Eigenschaft von
  - VcNet 555
  - siehe auch
  - VcLegendView 452
- LineColor**
  - Eigenschaft von
  - VcBox 308
  - VcGroup 429
  - VcLinkAppearance 468
  - VcNodeAppearance 690
- LineColorDataFieldIndex**
  - Eigenschaft von
  - VcNodeAppearance 691
- LineColorMapName**
  - Eigenschaft von
  - VcNodeAppearance 691
- LineThickness**
  - Eigenschaft von
  - VcBox 308
  - VcGroup 429
  - VcLinkAppearance 469
  - VcNodeAppearance 691
- LineType**
  - Eigenschaft von
  - VcBox 309
  - VcGroup 430
  - VcLinkAppearance 470
  - VcNodeAppearance 692
- Link**

- siehe auch
  - VcLink 460
- LinkAnnotationColumnNumberDataFieldIndex**
  - Eigenschaft von
    - VcNet 555
- LinkAnnotationRowNumberDataFieldIndex**
  - Eigenschaft von
    - VcNet 556
- LinkAppearance**
  - siehe auch
    - VcLinkAppearance 466
- LinkAppearanceByIndex**
  - Methode von
    - VcLinkAppearanceCollection 479
- LinkAppearanceByName**
  - Methode von
    - VcLinkAppearanceCollection 480
- LinkAppearanceCollection**
  - Eigenschaft von
    - VcNet 556
  - siehe auch
    - VcLinkAppearanceCollection 476
- LinkAppearance-Objekt**
  - Anzahl in Collection 476
  - Enumerator-Objekt 479
  - Iteration, Erstwert 479
  - Iteration, Folgewert 480
  - Nachfolger-Portsymbol 473
  - sichtbar 474
  - Typ der Verbindungsrouten 473
  - über Index 479
  - über Namen 480
  - Vorgänger-Portsymbol 472
- LinkCollection**
  - Eigenschaft von
    - VcNet 557
  - siehe auch
    - VcLinkCollection 483
- LinkCreationWithDialog**
  - Eigenschaft von
    - VcNet 557
- LinkDurationDataFieldIndex**
  - Eigenschaft von
    - VcScheduler 769
- LinkFormat**
  - siehe auch
    - VcLinkFormat 487
- LinkFormatCollection**
  - Eigenschaft von
    - VcNet 558
  - siehe auch
    - VcLinkFormatCollection 491
- Linkformatfeld**
  - Index 499
- LinkFormatField**
  - siehe auch
    - VcLinkFormatField 498
- LinkPredecessorDataFieldIndex**
  - Eigenschaft von
    - VcNet 558
- LinksDataTableName**
  - Eigenschaft von
    - VcNet 559
- LinkSuccessorDataFieldIndex**
  - Eigenschaft von
    - VcNet 560
- LinkTypeDataFieldIndex**
  - Eigenschaft von
    - VcNet 561
- Lizenzierung 14, 160, 283**
- Load**
  - Methode von

VcNet 597

## M

**MakeARGB**

Methode von  
VcNet 597

**Map**

siehe auch  
VcMap 502

**MapByIndex**

Methode von  
VcMapCollection 512

**MapByName**

Methode von  
VcMapCollection 513

**MapCollection**

Eigenschaft von  
VcNet 562  
siehe auch  
VcMapCollection 509

**MapEntry**

siehe auch  
VcMapEntry 517

**MarginsShownInInches**

Eigenschaft von  
VcPrinter 752

**Marked**

Eigenschaft von  
VcBox 311  
VcLink 462  
VcNode 678

**MarkedNodesFilter**

Eigenschaft von  
VcFilterCollection 418

**Markiermodus 271****Markierungstyp**

Knoten 39

Verbindungen 39

**MarkingColor**

Eigenschaft von  
VcWorldView 776

**MaxHorizontalPagesCount**

Eigenschaft von  
VcPrinter 752

**MaximumTextLineCount**

Eigenschaft von  
VcBoxFormatField 341  
VcNodeFormatField 733

**MaxVerticalPagesCount**

Eigenschaft von  
VcPrinter 753

**Methoden**

Add  
VcBoxCollection 319  
VcBoxFormatCollection 332  
VcCalendarCollection 358  
VcCalendarProfileCollection 367  
VcDataRecordCollection 379  
VcDataTableCollection 391  
VcDataTableFieldCollection 405  
VcFilterCollection 418  
VcIntervalCollection 448  
VcLinkAppearanceCollection 477  
VcLinkFormatCollection 492  
VcMapCollection 510  
VcNodeAppearanceCollection 707  
VcNodeFormatCollection 722

AddBySpecification  
VcBoxCollection 319  
VcBoxFormatCollection 332  
VcCalendarCollection 358  
VcCalendarProfileCollection 367  
VcFilterCollection 419  
VcIntervalCollection 448

- VcLinkAppearanceCollection 478
- VcLinkFormatCollection 493
- VcMapCollection 511
- VcNodeAppearanceCollection 707
- VcNodeFormatCollection 723
- AddDuration
  - VcCalendar 350
- AddSubCondition
  - VcFilter 414
- Arrange
  - VcNet 580
- BorderBox
  - VcBorderArea 295
- BoxByIndex
  - VcBoxCollection 320
- BoxByName
  - VcBoxCollection 321
- CalcDuration
  - VcCalendar 350
- CalendarByIndex
  - VcCalendarCollection 359
- CalendarByName
  - VcCalendarCollection 359
- CalendarProfileByIndex
  - VcCalendarProfileCollection 368
- CalendarProfileByName
  - VcCalendarProfileCollection 368
- Clear
  - VcCalendar 351
  - VcNet 580
- CompleteViewMode
  - VcNet 581
- Copy
  - VcBoxCollection 321
  - VcBoxFormatCollection 333
  - VcCalendarCollection 360
  - VcCalendarProfileCollection 368
- VcDataTableCollection 392
- VcDataTableFieldCollection 406
- VcFilterCollection 419
- VcIntervalCollection 449
- VcLinkAppearanceCollection 478
- VcLinkFormatCollection 493
- VcMapCollection 511
- VcNodeAppearanceCollection 708
- VcNodeFormatCollection 723
- CopyFormatField
  - VcBoxFormat 328
  - VcLinkFormat 489
  - VcNodeFormat 719
- CopyNodesIntoClipboard
  - VcNet 581
- CopySubCondition
  - VcFilter 414
- CreateEntry
  - VcMap 505
- CutNodesIntoClipboard
  - VcNet 581
- DataRecord
  - VcLink 464
  - VcNode 680
- DataRecordById
  - VcDataRecordCollection 380
- DataTableByIndex
  - VcDataTableCollection 392
- DataTableByName
  - VcDataTableCollection 393
- DataTableFieldByIndex
  - VcDataTableFieldCollection 406
- DataTableFieldByName
  - VcDataTableFieldCollection 407
- Delete
  - VcDataRecord 375
  - VcLink 464

- VcNode 680
- DeleteEntry
  - VcMap 506
- DeleteLinkRecord
  - VcNet 582
- DeleteNodeRecord
  - VcNet 582
- DetectDataTableFieldName
  - VcNet 583
- DetectDataTableName
  - VcNet 583
- DetectFieldIndex
  - VcNet 584
- DumpConfiguration
  - VcNet 584
- EndLoading
  - VcNet 585
- Evaluate
  - VcFilter 415
- ExportGraphicsToFileEx
  - VcNet 585
- FilterByIndex
  - VcFilterCollection 420
- FilterByName
  - VcFilterCollection 420
- FirstBox
  - VcBoxCollection 322
- FirstCalendar
  - VcCalendarCollection 360
- FirstCalendarProfile
  - VcCalendarProfileCollection 369
- FirstDataRecord
  - VcDataRecordCollection 381
- FirstDataTable
  - VcDataTableCollection 393
- FirstDataTableField
  - VcDataTableFieldCollection 407
- FirstFilter
  - VcFilterCollection 421
- FirstFormat
  - VcBoxFormatCollection 333
  - VcLinkFormatCollection 494
  - VcNodeFormatCollection 724
- FirstGroup
  - VcGroupCollection 436
- FirstInterval
  - VcIntervalCollection 449
- FirstLink
  - VcLinkCollection 484
- FirstLinkAppearance
  - VcLinkAppearanceCollection 478
- FirstMap
  - VcMapCollection 512
- FirstMapEntry
  - VcMap 506
- FirstNode
  - VcNodeCollection 713
- FirstNodeAppearance
  - VcNodeAppearanceCollection 708
- FormatByIndex
  - VcBoxFormatCollection 334
  - VcLinkFormatCollection 494
  - VcNodeFormatCollection 724
- FormatByName
  - VcBoxFormatCollection 334
  - VcLinkFormatCollection 495
  - VcNodeFormatCollection 725
- GetActualExtent
  - VcBox 315
- GetAValueFromARGB
  - VcNet 588
- GetBValueFromARGB
  - VcNet 588
- GetEndOfPreviousWorktime



- VcCalendar 351
- GetEnumerator
  - VcBoxCollection 322
  - VcBoxFormat 329
  - VcBoxFormatCollection 335
  - VcCalendarCollection 360
  - VcDataRecordCollection 382
  - VcDataTableCollection 394
  - VcDataTableFieldCollection 408
  - VcFilter 415
  - VcFilterCollection 421
  - VcGroupCollection 436
  - VcLinkAppearanceCollection 479
  - VcLinkCollection 484
  - VcLinkFormat 490
  - VcLinkFormatCollection 495
  - VcMapCollection 512
  - VcNodeAppearanceCollection 709
  - VcNodeCollection 713
  - VcNodeFormat 720
  - VcNodeFormatCollection 725
- GetGValueFromARGB
  - VcNet 589
- GetLinkByID
  - VcNet 590
- GetLinkByNodeIDs
  - VcNet 591
- GetMapEntry
  - VcMap 507
- GetNextIntervalBorder
  - VcCalendar 352
- GetNodeByID
  - VcNet 591
- GetPreviousIntervalBorder
  - VcCalendar 352
- GetRValueFromARGB
  - VcNet 592
- GetStartOfInterval
  - VcCalendar 353
- GetStartOfNextWorktime
  - VcCalendar 354
- GetTopLeftPixel
  - VcBox 315
- GetXyOffset
  - VcBox 316
- GroupByName
  - VcGroupCollection 437
- IdentifyFormatField
  - VcBox 316
  - VcNet 592
- IdentifyObject
  - VcDataRecord 376
- IdentifyObjectAt
  - VcNet 593
- ImportConfiguration
  - VcNet 595
- InsertLinkRecord
  - VcNet 596
- InsertNodeRecord
  - VcNet 596
- IntervalByIndex
  - VcIntervalCollection 450
- IntervalByName
  - VcIntervalCollection 450
- IsValid
  - VcFilter 416
  - VcFilterSubCondition 427
- IsWorktime
  - VcCalendar 354
- LinkAppearanceByIndex
  - VcLinkAppearanceCollection 479
- LinkAppearanceByName
  - VcLinkAppearanceCollection 480
- Load

- VcNet 597
- MakeARGB
  - VcNet 597
- MapByIndex
  - VcMapCollection 512
- MapByName
  - VcMapCollection 513
- NextBox
  - VcBoxCollection 323
- NextCalendar
  - VcCalendarCollection 361
- NextCalendarProfile
  - VcCalendarProfileCollection 369
- NextDataRecord
  - VcDataRecordCollection 383
- NextDataTable
  - VcDataTableCollection 395
- NextDataTableField
  - VcDataTableFieldCollection 409
- NextFilter
  - VcFilterCollection 422
- NextFormat
  - VcBoxFormatCollection 335
  - VcLinkFormatCollection 496
  - VcNodeFormatCollection 726
- NextGroup
  - VcGroupCollection 437
- NextInterval
  - VcIntervalCollection 450
- NextLink
  - VcLinkCollection 485
- NextLinkAppearance
  - VcLinkAppearanceCollection 480
- NextMap
  - VcMapCollection 513
- NextMapEntry
  - VcMap 507
- NextNode
  - VcNodeCollection 714
- NextNodeAppearance
  - VcNodeAppearanceCollection 710
- NodeAppearanceByIndex
  - VcNodeAppearanceCollection 710
- NodeAppearanceByName
  - VcNodeAppearanceCollection 711
- PasteNodesFromClipboard
  - VcNet 598
- PixelsToRaster
  - VcNet 599
- PrintEx
  - VcNet 599
- PrintToFile
  - VcNet 600
- PutInOrderAfter
  - VcCalendarProfile 365
  - VcInterval 445
  - VcLinkAppearance 475
  - VcNodeAppearance 704
- RelatedDataRecord
  - VcDataRecord 376
  - VcLink 465
  - VcNode 681
- Remove
  - VcBoxCollection 323
  - VcBoxFormatCollection 336
  - VcCalendarCollection 362
  - VcCalendarProfileCollection 370
  - VcDataRecordCollection 383
  - VcFilterCollection 422
  - VcIntervalCollection 451
  - VcLinkAppearanceCollection 481
  - VcLinkFormatCollection 497
  - VcMapCollection 514
  - VcNodeAppearanceCollection 711

- VcNodeFormatCollection 727
- RemoveFormatField
  - VcBoxFormat 329
  - VcLinkFormat 490
  - VcNodeFormat 720
- RemoveSubCondition
  - VcFilter 416
- Reset
  - VcNet 601
- SaveAsEx
  - VcNet 601
- ScheduleProject
  - VcNet 602
  - VcScheduler 771
- ScrollToNode
  - VcNet 603
- SelectCalendarProfiles
  - VcCalendarProfileCollection 370
- SelectLinks
  - VcLinkCollection 485
- SelectMaps
  - VcMapCollection 515
- SelectNodes
  - VcNodeCollection 714
- SetImageResource
  - VcNet 603
- SetXY
  - VcGroup 434
- SetXYOffset
  - VcBox 316
- SetXYOffsetByTopLeftPixel
  - VcBox 317
- ShowAboutDialog
  - VcNet 604
- ShowExportGraphicsDialog
  - VcNet 605
- ShowLinkEditDialog
  - VcNet 606
- ShowNodeEditDialog
  - VcNet 607
- ShowPageSetupDialog
  - VcNet 607
- ShowPrintDialog
  - VcNet 608
- ShowPrinterSetupDialog
  - VcNet 608
- ShowPrintPreviewDialog
  - VcNet 609
- SuspendUpdate
  - VcNet 609
- Update
  - VcBoxCollection 324
  - VcCalendar 355
  - VcCalendarCollection 362
  - VcCalendarProfileCollection 370, 371
  - VcDataRecordCollection 384
  - VcDataTableCollection 395
  - VcIntervalCollection 451
  - VcLegendView 459
  - VcLink 465
  - VcLinkAppearanceCollection 481
  - VcMapCollection 515
  - VcNode 681
- UpdateLinkRecord
  - VcNet 611
- UpdateNodeRecord
  - VcNet 611
- Zoom
  - VcNet 612
- ZoomOnMarkedNodes
  - VcNet 612
- MinimumColumnWidth**
  - Eigenschaft von

VcNet 563

### **MinimumRowHeight**

Eigenschaft von

VcNet 563

### **MinimumTextLineCount**

Eigenschaft von

VcBoxFormatField 341

VcNodeFormatField 734

### **MinimumWidth**

Eigenschaft von

VcBoxFormatField 342

VcLinkFormatField 500

VcNodeFormatField 734

### **Mode**

Eigenschaft von

VcWorldView 776

### **MouseProcessingEnabled**

Eigenschaft von

VcNet 564

### **Moveable**

Eigenschaft von

VcBox 311

### **MovingCollapsedClustersAllowed**

Eigenschaft von

VcNet 564

### **MultiplePrimaryKeysAllowed**

Eigenschaft von

VcDataTable 388

## N

### **Name**

Eigenschaft von

VcBox 312

VcBoxFormat 327

VcCalendar 348

VcCalendarProfile 364

VcDataTable 388

VcDataTableField 400

VcFilter 411

VcGroup 431

VcInterval 443

VcLinkAppearance 471

VcLinkFormat 488

VcMap 503

VcNodeAppearance 694

VcNodeFormat 718

### **Navigation**

Tastatur 254, 260

### **Net**

siehe auch

VcNet 526

### **NextBox**

Methode von

VcBoxCollection 323

### **NextCalendar**

Methode von

VcCalendarCollection 361

### **NextCalendarProfile**

Methode von

VcCalendarProfileCollection 369

### **NextDataRecord**

Methode von

VcDataRecordCollection 383

### **NextDataTable**

Methode von

VcDataTableCollection 395

### **NextDataTableField**

Methode von

VcDataTableFieldCollection 409

### **NextFilter**

Methode von

VcFilterCollection 422

### **NextFormat**

Methode von

- VcBoxFormatCollection 335
- VcLinkFormatCollection 496
- VcNodeFormatCollection 726
- NextGroup**
  - Methode von
    - VcGroupCollection 437
- NextInterval**
  - Methode von
    - VcIntervalCollection 450
- NextLink**
  - Methode von
    - VcLinkCollection 485
- NextLinkAppearance**
  - Methode von
    - VcLinkAppearanceCollection 480
- NextMap**
  - Methode von
    - VcMapCollection 513
- NextMapEntry**
  - Methode von
    - VcMap 507
- NextNode**
  - Methode von
    - VcNodeCollection 714
- NextNodeAppearance**
  - Methode von
    - VcNodeAppearanceCollection 710
- Node**
  - siehe auch
    - VcNode 675
- NodeAndLinkCreationAllowed**
  - Eigenschaft von
    - VcNet 565
- NodeAppearance**
  - siehe auch
    - VcNodeAppearance 683
- NodeAppearanceByName**
  - Methode von
    - VcNodeAppearanceCollection 710
- NodeAppearanceCollection**
  - Eigenschaft von
    - VcNet 565
  - siehe auch
    - VcNodeAppearanceCollection 706
- NodeCalendarNameDataFieldIndex**
  - Eigenschaft von
    - VcNet 566
- NodeChangeRankToPredecessorRankDataFieldIndex**
  - Eigenschaft von
    - VcNet 566
- NodeCollection**
  - Eigenschaft von
    - VcGroup 432
    - VcNet 566
  - siehe auch
    - VcNodeCollection 712
- NodeColumnNumberDataFieldIndex**
  - Eigenschaft von
    - VcNet 567
- NodeCreationWithDialog**
  - Eigenschaft von
    - VcNet 567
- NodeFormat**
  - siehe auch
    - VcNodeFormat 716
- NodeFormatCollection**
  - Eigenschaft von
    - VcNet 567
  - siehe auch
    - VcNodeFormatCollection 721

**NodeFormatField**

siehe auch

VcNodeFormatField 728

**NodeRowNumberDataFieldIndex**

Eigenschaft von

VcNet 568

**NodesDataTableName**

Eigenschaft von

VcNet 569

**NodesUseCalendars**

Eigenschaft von

VcNet 569

**NodeToolTipTextDataFieldIndex**

Eigenschaft von

VcNet 570

**Number**

Eigenschaft von

VcMapEntry 522

VcDataTableFieldCollection 404

VcFilter 410

VcFilterCollection 417

VcFilterSubCondition 424

VcGroup 428

VcGroupCollection 435

VcInterval 439

VcIntervalCollection 447

VcLegendView 452

VcLink 460

VcLinkAppearance 466

VcLinkAppearanceCollection 476

VcLinkCollection 483

VcLinkFormat 487

VcLinkFormatCollection 491

VcLinkFormatField 498

VcMap 502

VcMapCollection 509

VcMapEntry 517

VcNet 526

VcNode 675

VcNodeAppearance 683

VcNodeAppearanceCollection 706

VcNodeCollection 712

VcNodeFormat 716

VcNodeFormatCollection 721

VcNodeFormatField 728

VcPrinter 742

VcRect 761

VcScheduler 765

VcWorldView 773

**ObliqueTracksOnLinks**

Eigenschaft von

VcNet 570

**Operator**

Eigenschaft von

VcFilterSubCondition 426

**Objekte**

VcBorderArea 295

VcBorderBox 297

VcBox 306

VcBoxCollection 318

VcBoxFormat 325

VcBoxFormatCollection 331

VcBoxFormatField 338

VcCalendar 347

VcCalendarCollection 356

VcCalendarProfile 363

VcCalendarProfileCollection 366

VcDataRecord 372

VcDataRecordCollection 378

VcDataTable 386

VcDataTableCollection 390

VcDataTableField 397

**Orientation**

- Eigenschaft von
  - VcNet 571
  - VcPrinter 753

**Origin**

- Eigenschaft von
  - VcBox 312

**OutgoingLinks**

- Eigenschaft von
  - VcNode 679

**P**

**PageDescription**

- Eigenschaft von
  - VcPrinter 754

**PageDescriptionString**

- Eigenschaft von
  - VcPrinter 754

**PageFrame**

- Eigenschaft von
  - VcPrinter 755

**PageNumberMode**

- Eigenschaft von
  - VcPrinter 755

**PageNumbers**

- Eigenschaft von
  - VcPrinter 756

**PagePaddingEnabled**

- Eigenschaft von
  - VcPrinter 756

**PaperSize**

- Eigenschaft von
  - VcPrinter 757

**ParentHWND**

- Eigenschaft von
  - VcWorldView 777

**PasteNodesFromClipboard**

- Methode von
  - VcNet 598

**Pattern**

- Eigenschaft von
  - VcMapEntry 522
  - VcNodeAppearance 694

**PatternBackgroundColor**

- Eigenschaft von
  - VcBoxFormatField 342

**PatternBackgroundColorAsARGB**

- Eigenschaft von
  - VcNodeFormatField 734

**PatternBackgroundColorDataFieldIndex**

- Eigenschaft von
  - VcNodeFormatField 735

**PatternBackgroundColorMapName**

- Eigenschaft von
  - VcNodeFormatField 735

**PatternColor**

- Eigenschaft von
  - VcNodeAppearance 698

**PatternColorAsARGB**

- Eigenschaft von
  - VcBoxFormatField 343
  - VcNodeFormatField 735

**PatternColorDataFieldIndex**

- Eigenschaft von
  - VcNodeAppearance 698
  - VcNodeFormatField 736

**PatternColorMapName**

- Eigenschaft von
  - VcNodeAppearance 699
  - VcNodeFormatField 736

**PatternDataFieldIndex**

- Eigenschaft von
  - VcNodeAppearance 699

**PatternEx**

Eigenschaft von

VcBoxFormatField 344

VcNodeFormatField 737

**PatternExDataFieldIndex**

Eigenschaft von

VcNodeFormatField 737

**PatternExMapName**

Eigenschaft von

VcNodeFormatField 738

**PatternMapName**

Eigenschaft von

VcNodeAppearance 699

**PDF-Dateien**

Export 126

**Performance 288****PhantomDrawingWhileDraggingEnabled**

Eigenschaft von

VcNet 571

**PileEffect**

Eigenschaft von

VcNodeAppearance 700

**PixelsToRaster**

Methode von

VcNet 599

**Plattformen x86 und x64 122****Positionen von Knoten und Verbindungsbeschriftungen**

speichern und laden 55

**PredecessorNode**

Eigenschaft von

VcLink 462

**PredecessorPortSymbol**

Eigenschaft von

VcLinkAppearance 472

**Primärschlüssel**

zusammengesetzt 388

**PrimaryKey**

Eigenschaft von

VcDataTableField 401

**PrintDate**

Eigenschaft von

VcPrinter 758

**Printer**

Eigenschaft von

VcNet 572

siehe auch

VcPrinter 742

**PrinterName**

Eigenschaft von

VcPrinter 758

**PrintEx**

Methode von

VcNet 599

**PrintPreviewWithFirstPage**

Eigenschaft von

VcPrinter 758

**PrintToFile**

Methode von

VcNet 600

**Priorität 195**

Boxen 204

**Priority**

Eigenschaft von

VcBox 313

**Publizieren im Internet 72, 141, 250, 273****PutInOrderAfter**

Methode von

VcCalendarProfile 365

VcInterval 445

VcLinkAppearance 475

VcNodeAppearance 704



## R

**Rahmen**

außen 265

**Rect**

siehe auch

VcRect 761

**ReferencePoint**

Eigenschaft von

VcBox 313

**RelatedDataRecord**

Methode von

VcDataRecord 376

VcLink 465

VcNode 681

**RelationshipFieldIndex**

Eigenschaft von

VcDataTableField 401

**Remove**

Methode von

VcBoxCollection 323

VcBoxFormatCollection 336

VcCalendarCollection 362

VcCalendarProfileCollection 370

VcDataRecordCollection 383

VcFilterCollection 422

VcIntervalCollection 451

VcLinkAppearanceCollection 481

VcLinkFormatCollection 497

VcMapCollection 514

VcNodeAppearanceCollection 711

VcNodeFormatCollection 727

**RemoveFormatField**

Methode von

VcBoxFormat 329

VcLinkFormat 490

VcNodeFormat 720

**RemoveSubCondition**

Methode von

VcFilter 416

**Reset**

Methode von

VcNet 601

**Return Status 92****Right**

Eigenschaft von

VcRect 763

**RightMargin**

Eigenschaft von

VcNodeFormatField 738

**RoundedLinkSlantsEnabled**

Eigenschaft von

VcNet 572

**RoutingType**

Eigenschaft von

VcLinkAppearance 472

**Rückgabewerte 92**

## S

**SaveAsEx**

Methode von

VcNet 601

**ScheduledProjectEndDate**

Eigenschaft von

VcScheduler 769

**ScheduledProjectStartDate**

Eigenschaft von

VcScheduler 770

**ScheduleProject**

Methode von

VcNet 602

VcScheduler 771

**Scheduler**

Eigenschaft von

- VcNet 573
- Kalender benutzen 158
- siehe auch
  - VcScheduler 765
- ScheduleSuccessorsOnlyEnabled**
  - Eigenschaft von
    - VcScheduler 770
- Schnittmarkierungen 266**
- Schnittstelle**
  - einrichten 24
- Schriften**
  - Anti-Aliasing 540
- ScrollBarMode**
  - Eigenschaft von
    - VcLegendView 455
    - VcWorldView 778
- ScrollToNode**
  - Methode von
    - VcNet 603
- SecondsPerWorkday**
  - Eigenschaft von
    - VcCalendar 349
- Seite einrichten 272**
- Seitennummerierung 266**
- Seitenränder 267**
- SelectCalendarProfiles**
  - Methode von
    - VcCalendarProfileCollection 370
- SelectLinks**
  - Methode von
    - VcLinkCollection 485
- SelectMaps**
  - Methode von
    - VcMapCollection 515
- SelectNodes**
  - Methode von
    - VcNodeCollection 714
- SetImageResource**
  - Methode von
    - VcNet 603
- SetXY**
  - Methode von
    - VcGroup 434
- SetXYOffset**
  - Methode von
    - VcBox 316
- SetXYOffsetByTopLeftPixel**
  - Methode von
    - VcBox 317
- Shadow**
  - Eigenschaft von
    - VcNodeAppearance 700
- ShadowColor**
  - Eigenschaft von
    - VcNodeAppearance 701
- ShortenedLinks**
  - Eigenschaft von
    - VcNet 573
- ShowAboutDialog**
  - Methode von
    - VcNet 604
- ShowExportGraphicsDialog**
  - Methode von
    - VcNet 605
- ShowLinkEditDialog**
  - Methode von
    - VcNet 606
- ShowNodeEditDialog**
  - Methode von
    - VcNet 607
- ShowPageSetupDialog**
  - Methode von
    - VcNet 607
- ShowPrintDialog**

- Methode von
  - VcNet 608
- ShowPrinterSetupDialog**
  - Methode von
    - VcNet 608
- ShowPrintPreviewDialog**
  - Methode von
    - VcNet 609
- Sicherheitsrichtlinien**
  - Laufzeit 118
- Spaltenbreite**
  - minimale 151
- Specification**
  - Eigenschaft von
    - VcBoxFormat 327
    - VcCalendar 349
    - VcCalendarProfile 364
    - VcFilter 412
    - VcInterval 443
    - VcLinkAppearance 473
    - VcLinkFormat 488
    - VcMap 504
    - VcNodeAppearance 701
    - VcNodeFormat 718
- Sprachanpassung 129**
- StartDateForAutomaticScheduling**
  - Eigenschaft von
    - VcScheduler 770
- StartDateNotEarlierThanDataFieldIndex**
  - Eigenschaft von
    - VcScheduler 771
- StartDateTime**
  - Eigenschaft von
    - VcInterval 444
- StartMonth**
  - Eigenschaft von
    - VcInterval 444
- StartTime**
  - Eigenschaft von
    - VcInterval 444
- StartWeekday**
  - Eigenschaft von
    - VcInterval 445
- Statuszeilentext 131**
- StraightLinkDrawing**
  - Eigenschaft von
    - VcNet 573
- Strg-C, -X und -V verarbeiten 155**
- StrikeThrough**
  - Eigenschaft von
    - VcNodeAppearance 702
- StrikeThroughColor**
  - Eigenschaft von
    - VcNodeAppearance 703
- StringsCaseSensitive**
  - Eigenschaft von
    - VcFilter 412
- SubCondition**
  - Eigenschaft von
    - VcFilter 413
- SubConditionCount**
  - Eigenschaft von
    - VcFilter 413
- SuccessorNode**
  - Eigenschaft von
    - VcLink 463
- SuccessorPortSymbol**
  - Eigenschaft von
    - VcLinkAppearance 473
- SuspendUpdate**
  - Methode von
    - VcNet 609

## T

**Teilnetz 272, 275****Text**

Eigenschaft von  
VcBoundingBox 302

**TextAndGraphicsCombined**

Eigenschaft von  
VcNodeFormatField 738

**TextDataFieldIndex**

Eigenschaft von  
VcLinkFormatField 500  
VcNodeFormatField 739

**TextEntrySupplyingEventEnabled**

Eigenschaft von  
VcNet 574

**TextFont**

Eigenschaft von  
VcBoundingBox 303  
VcBoxFormatField 345  
VcLinkFormatField 500  
VcNodeFormatField 739

**TextFontColor**

Eigenschaft von  
VcBoxFormatField 345  
VcLinkFormatField 501  
VcNodeFormatField 739

**TextFontDataFieldIndex**

Eigenschaft von  
VcNodeFormatField 740

**TextFontMapName**

Eigenschaft von  
VcNodeFormatField 740

**TextLineCount**

Eigenschaft von  
VcLinkFormatField 501

**ThreeDEffect**

Eigenschaft von  
VcNodeAppearance 703

**TimeUnit**

Eigenschaft von  
VcNet 574

**Titel**

in Darstellung wiederholen 265

**Title**

Eigenschaft von  
VcGroup 432

**TitleAndLegendOnAllPages**

Eigenschaft von  
VcPrinter 759

**TitleLineCount**

Eigenschaft von  
VcGroup 433

**Tooltip**

Datenfeld für Text 166

**ToolTip**

Dauer bis zur Anzeige 576  
Erscheinungsdauer 575  
Verschwinden auf Klick 576  
Wechseldauer 575

**ToolTipChangeDuration**

Eigenschaft von  
VcNet 575

**ToolTipDuration**

Eigenschaft von  
VcNet 575

**ToolTipPointerDuration**

Eigenschaft von  
VcNet 576

**Tooltips 155**

zur Laufzeit 132

**ToolTipShowAfterClick**

Eigenschaft von  
VcNet 576

**ToolTipTextSupplyingEventEnabled**

Eigenschaft von

VcNet 576

**Top**

Eigenschaft von

VcLegendView 455

VcRect 763

VcWorldView 778

**TopActualValue**

Eigenschaft von

VcLegendView 456

VcWorldView 779

**TopMargin**

Eigenschaft von

VcNodeFormatField 740

**TotalFloatDataFieldIndex**

Eigenschaft von

VcScheduler 771

**Type**

Eigenschaft von

VcBoundingBox 304

VcBoxFormatField 346

VcCalendarProfile 364

VcDataTableField 403

VcInterval 445

VcMap 504

VcNodeFormatField 741

**U**

**UngroupedNodesAllowed**

Eigenschaft von

VcNet 577

**Update**

Methode von

VcBoxCollection 324

VcCalendar 355

VcCalendarCollection 362

VcCalendarProfileCollection 370,  
371

VcDataRecordCollection 384

VcDataTableCollection 395

VcIntervalCollection 451

VcLegendView 459

VcLink 465

VcLinkAppearanceCollection 481

VcMapCollection 515

VcNode 681

**UpdateBehaviorName**

Eigenschaft von

VcBox 314

VcWorldView 779

**UpdateLinkRecord**

Methode von

VcNet 611

**UpdateNodeRecord**

Methode von

VcNet 611

**V**

**VARCHARTEXT XNet**

automatisch skalieren 23

im Formular platzieren 22

**VcBorderArea 295**

BorderBox 295

**VcBoundingBox 297**

Alignment 297

GraphicsFileName 298

LegendElementsArrangement 299

LegendElementsBottomMargin 299

LegendElementsMaximumColumnCo  
unt 300

LegendElementsMaximumRowCount  
300

LegendElementsTopMargin 300

- LegendFont 300
- LegendTitle 301
- LegendTitleFont 301
- LegendTitleVisible 302
- Text 302
- TextFont 303
- Type 304
- VcBox 306**
  - FieldText 307
  - FormatName 307
  - GetActualExtent 315
  - GetTopLeftPixel 315
  - GetXYOffset 316
  - IdentifyFormatField 316
  - LineColor 308
  - LineThickness 308
  - LineType 309
  - Marked 311
  - Moveable 311
  - Name 312
  - Origin 312
  - Priority 313
  - ReferencePoint 313
  - SetXYOffset 316
  - SetXYOffsetByTopLeftPixel 317
  - UpdateBehaviorName 314
  - Visible 314
- VcBoxCollection 318**
  - Add 319
  - AddBySpecification 319
  - BoxByIndex 320
  - BoxByName 321
  - Copy 321
  - Count 318
  - FirstBox 322
  - GetEnumerator 322
  - NextBox 323
  - Remove 323
  - Update 324
- VcBoxFormat 325**
  - CopyFormatField 328
  - FieldsSeparatedByLines 325
  - FormatField 326
  - FormatFieldCount 326
  - GetEnumerator 329
  - Name 327
  - RemoveFormatField 329
  - Specification 327
- VcBoxFormatCollection 331**
  - Add 332
  - AddBySpecification 332
  - Copy 333
  - Count 331
  - FirstFormat 333
  - FormatByIndex 334
  - FormatByName 334
  - GetEnumerator 335
  - NextFormat 335
  - Remove 336
- VcBoxFormatField 338**
  - Alignment 338
  - FormatName 339
  - GraphicsHeight 340
  - Index 340
  - MaximumTextLineCount 341
  - MinimumTextLineCount 341
  - MinimumWidth 342
  - PatternBackgroundColor 342
  - PatternColorAsARGB 343
  - PatternEx 344
  - TextFont 345
  - TextFontColor 345
  - Type 346
- VcBoxLeftClicking**

- Ereignis von
  - VcNet 613
- VcBoxLeftDoubleClicking**
  - Ereignis von
    - VcNet 613
- VcBoxModified**
  - Ereignis von
    - VcNet 614
- VcBoxModifiedEventArgs**
  - Ereignisobjekt von
    - VcBoxModified 614
- VcBoxModifying**
  - Ereignis von
    - VcNet 615
- VcBoxModifyingEventArgs**
  - Ereignisobjekt von
    - VcBoxModifying 615
- VcBoxRightClicking**
  - Ereignis von
    - VcNet 616
- VcCalendar 347**
  - AddDuration 350
  - CalcDuration 350
  - CalendarProfileCollection 348
  - Clear 351
  - GetEndOfPreviousWorktime 351
  - GetNextIntervalBorder 352
  - GetPreviousIntervalBorder 352
  - GetStartOfInterval 353
  - GetStartOfNextWorktime 354
  - IntervalCollection 348
  - IsWorktime 354
  - Name 348
  - SecondsPerWorkday 349
  - Specification 349
  - Update 355
- VcCalendarCollection 356**
  - Active 356
  - Add 358
  - AddBySpecification 358
  - CalendarByIndex 359
  - CalendarByName 359
  - Copy 360
  - Count 357
  - FirstCalendar 360
  - GetEnumerator 360
  - NextCalendar 361
  - Remove 362
  - Update 362
- VcCalendarGrid**
  - Eigenschaft von
    - VcPrinter 760
- VcCalendarProfile 363**
  - IntervalCollection 363
  - Name 364
  - PutInOrderAfter 365
  - Specification 364
  - Type 364
- VcCalendarProfileCollection 366**
  - Add 367
  - AddBySpecification 367
  - CalendarProfileByIndex 368
  - CalendarProfileByName 368
  - Copy 368
  - Count 367
  - FirstCalendarProfile 369
  - NextCalendarProfile 369
  - Remove 370
  - SelectCalendarProfiles 370
  - Update 370, 371
- VcDataModified**
  - Ereignis von
    - VcNet 617
- VcDataRecord 372**

- AllData 372
- DataField 373
- DataTableName 374
- Delete 375
- ID 375
- IdentifyObject 376
- RelatedDataRecord 376
- VcDataRecordCollection 378**
  - Add 379
  - Count 378
  - DataRecordById 380
  - FirstDataRecord 381
  - GetEnumerator 382
  - NextDataRecord 383
  - Remove 383
  - Update 384
- VcDataRecordCreated**
  - Ereignis von
    - VcNet 618
- VcDataRecordCreatedEventArgs**
  - Ereignisobjekt von
    - VcDataRecordCreated 618
- VcDataRecordCreating**
  - Ereignis von
    - VcNet 619
- VcDataRecordCreatingEventArgs**
  - Ereignisobjekt von
    - VcDataRecordCreating 619
- VcDataRecordDeleted**
  - Ereignis von
    - VcNet 620
- VcDataRecordDeletedEventArgs**
  - Ereignisobjekt von
    - VcDataRecordDeleted 620
- VcDataRecordDeleting**
  - Ereignis von
    - VcNet 621
- VcDataRecordDeletingEventArgs**
  - Ereignisobjekt von
    - VcDataRecordDeleting 621
- VcDataRecordModified**
  - Ereignis von
    - VcNet 622
- VcDataRecordModifiedEventArgs**
  - Ereignisobjekt von
    - VcDataRecordModified 622
- VcDataRecordModifying**
  - Ereignis von
    - VcNet 622
- VcDataRecordModifyingEventArgs**
  - Ereignisobjekt von
    - VcDataRecordModifying 623
- VcDataRecordNotFound**
  - Ereignis von
    - VcNet 623
- VcDataRecordNotFoundEventArgs**
  - Ereignisobjekt von
    - VcDataRecordNotFound 624
- VcDataTable 386**
  - DataRecordCollection 386
  - DataTableFieldCollection 387
  - Description 387
  - MultiplePrimaryKeysAllowed 388
  - Name 388
- VcDataTableCollection 390**
  - Add 391
  - Copy 392
  - Count 390
  - DataTableByIndex 392
  - DataTableByName 393
  - FirstDataTable 393
  - GetEnumerator 394
  - NextDataTable 395
  - Update 395



**VcDataTableField 397**

DataTableName 397  
DateFormat 398  
Editable 399  
Hidden 399  
Index 400  
Name 400  
PrimaryKey 401  
RelationshipFieldIndex 401  
Type 403

**VcDataTableFieldCollection 404**

Add 405  
Copy 406  
Count 404  
DataTableFieldByIndex 406  
DataTableFieldByName 407  
FirstDataTableField 407  
GetEnumerator 408  
NextDataTableField 409

**VcDiagramLeftClicking**

Ereignis von  
VcNet 624

**VcDiagramLeftDoubleClicking**

Ereignis von  
VcNet 625

**VcDiagramRightClicking**

Ereignis von  
VcNet 625

**VcDragCompleting**

Ereignis von  
VcNet 626

**VcDragCompletingEventArgs**

Ereignisobjekt von  
VcDragCompleting 626

**VcDragStarting**

Ereignis von  
VcNet 627

**VcDragStartingEventArgs**

Ereignisobjekt von  
VcDragStarting 627

**VcErrorOccurring**

Ereignis von  
VcNet 627

**VcFieldSelecting**

Ereignis von  
VcNet 628

**VcFieldSelectingEventArgs**

Ereignisobjekt von  
VcFieldSelecting 628

**VcFilter 410**

AddSubCondition 414  
CopySubCondition 414  
DataDefinitionTable 411  
DatesWithHourAndMinute 411  
Evaluate 415  
GetEnumerator 415  
IsValid 416  
Name 411  
RemoveSubCondition 416  
Specification 412  
StringsCaseSensitive 412  
SubCondition 413  
SubConditionCount 413

**VcFilterCollection 417**

Add 418  
AddBySpecification 419  
Copy 419  
Count 417  
FilterByIndex 420  
FilterByName 420  
FirstFilter 421  
GetEnumerator 421  
MarkedNodesFilter 418  
NextFilter 422

- Remove 422
- VcFilterSubCondition 424**
  - ComparisonValueAsString 424
  - ConnectionOperator 425
  - DataFieldIndex 426
  - FilterName 426
  - Index 426
  - IsValid 427
  - Operator 426
- VcGiveFeedbackOnNodeCreating**
  - Ereignis von
    - VcNet 629
- VcGroup 428**
  - BackgroundColor 428
  - LineColor 429
  - LineThickness 429
  - LineType 430
  - Name 431
  - NodeCollection 432
  - SetXY 434
  - Title 432
  - TitleLineCount 433
  - X 433
  - Y 433
- VcGroupCollection 435**
  - Count 435
  - FirstGroup 436
  - GetEnumerator 436
  - GroupByName 437
  - NextGroup 437
- VcGroupCreated**
  - Ereignis von
    - VcNet 630
- VcGroupDeleting**
  - Ereignis von
    - VcNet 630
- VcGroupLeftClicking**
  - Ereignis von
    - VcNet 631
- VcGroupLeftDoubleClicking**
  - Ereignis von
    - VcNet 632
- VcGroupModified**
  - Ereignis von
    - VcNet 632
- VcGroupModifying**
  - Ereignis von
    - VcNet 633
- VcGroupRightClicking**
  - Ereignis von
    - VcNet 634
- VcHelpRequested**
  - Ereignis von
    - VcNet 635
- VcHelpRequestedEventArgs**
  - Ereignisobjekt von
    - VcHelpRequested 635
- VcInPlaceEditorShowing**
  - Ereignis von
    - VcNet 636
- VcInterval 439**
  - CalendarProfileName 441
  - DayInEndMonth 441
  - DayInStartMonth 441
  - EndDateTime 442
  - EndMonth 442
  - EndTime 442
  - EndWeekday 443
  - Name 443
  - PutInOrderAfter 445
  - Specification 443
  - StartDateTime 444
  - StartMonth 444
  - StartTime 444

- StartWeekday 445
- Type 445
- VcIntervalCollection 447**
  - Add 448
  - AddBySpecification 448
  - Copy 449
  - Count 448
  - FirstInterval 449
  - IntervalByIndex 450
  - IntervalByName 450
  - NextInterval 450
  - Remove 451
  - Update 451
- VcLegendView 452**
  - Border 452
  - Height 453
  - HeightActualValue 453
  - Left 454
  - LeftActualValue 454
  - ScrollBarMode 455
  - Top 455
  - TopActualValue 456
  - Update 459
  - Visible 456
  - Width 457
  - WidthActualValue 457
  - WindowMode 458
- VcLegendViewClosed**
  - Ereignis von
    - VcNet 638
- VcLegendViewClosedEventArgs**
  - Ereignisobjekt von
    - VcLegendViewClosed 638
- VcLink 460**
  - AllData 460
  - DataField 461
  - DataRecord 464
  - Delete 464
  - ID 462
  - Marked 462
  - PredecessorNode 462
  - RelatedDataRecord 465
  - SuccessorNode 463
  - Update 465
- VcLinkAppearance 466**
  - FilterName 467
  - FormatName 468
  - LineColor 468
  - LineThickness 469
  - LineType 470
  - Name 471
  - PredecessorPortSymbol 472
  - PutInOrderAfter 475
  - RoutingType 472
  - Specification 473
  - SuccessorPortSymbol 473
  - Visible 474
- VcLinkAppearanceCollection 476**
  - Add 477
  - AddBySpecification 478
  - Copy 478
  - Count 476
  - FirstLinkAppearance 478
  - GetEnumerator 479
  - LinkAppearanceByIndex 479
  - LinkAppearanceByName 480
  - NextLinkAppearance 480
  - Remove 481
  - Update 481
- VcLinkCollection 483**
  - Count 483
  - FirstLink 484
  - GetEnumerator 484
  - NextLink 485

- SelectLinks 485
- VcLinkCreated**
  - Ereignis von
  - VcNet 639
- VcLinkCreating**
  - Ereignis von
  - VcNet 640
- VcLinkDeleted**
  - Ereignis von
  - VcNet 641
- VcLinkDeleting**
  - Ereignis von
  - VcNet 641
- VcLinkFormat 487**
  - CopyFormatField 489
  - FormatField 487
  - FormatFieldCount 488
  - GetEnumerator 490
  - Name 488
  - RemoveFormatField 490
  - Specification 488
- VcLinkFormatCollection 491**
  - Add 492
  - AddBySpecification 493
  - Copy 493
  - Count 491
  - FirstFormat 494
  - FormatByIndex 494
  - FormatByName 495
  - GetEnumerator 495
  - NextFormat 496
  - Remove 497
- VcLinkFormatField 498**
  - Alignment 498
  - ConstantText 499
  - FormatName 499
  - Index 499
  - MinimumWidth 500
  - TextDataFieldIndex 500
  - TextFont 500
  - TextColor 501
  - TextLineCount 501
- VcLinkModified**
  - Ereignis von
  - VcNet 642
- VcLinkModifying**
  - Ereignis von
  - VcNet 642
- VcLinksLeftClicking**
  - Ereignis von
  - VcNet 643
- VcLinksLeftDoubleClicking**
  - Ereignis von
  - VcNet 644
- VcLinksMarked**
  - Ereignis von
  - VcNet 645
- VcLinksMarking**
  - Ereignis von
  - VcNet 646
- VcLinksRightClicking**
  - Ereignis von
  - VcNet 647
- VcMap 502**
  - ConsiderFilterEntries 502
  - Count 503
  - CreateEntry 505
  - DeleteEntry 506
  - FirstMapEntry 506
  - GetMapEntry 507
  - Name 503
  - NextMapEntry 507
  - Specification 504
  - Type 504

**VcMapCollection 509**

- Add 510
- AddBySpecification 511
- Copy 511
- Count 510
- FirstMap 512
- GetEnumerator 512
- MapByIndex 512
- MapByName 513
- NextMap 513
- Remove 514
- SelectMaps 515
- Update 515

**VcMapEntry 517**

- Color 517
- DataFieldValue 518
- FontBody 519
- FontName 519
- FontSize 520
- GraphicsFileName 521
- Number 522
- Pattern 522

**VcMouseDoubleClicking**

- Ereignis von  
VcNet 648

**VcMouseDown**

- Ereignis von  
VcNet 648

**VcMouseMove**

- Ereignis von  
VcNet 649

**VcMouseUp**

- Ereignis von  
VcNet 650

**VcNet 526**

- ActiveNodeFilter 531
- Arrange 580

- BorderArea 532
- BoxCollection 532
- BoxFormatCollection 532
- CalendarCollection 533
- CalendarProfileCollection 533
- Clear 580
- CompleteViewMode 581
- CopyNodesIntoClipboard 581
- CtrlCXVProcessingEnabled 533
- CutNodesIntoClipboard 581
- DataTableCollection 534
- DateOutputFormat 534
- DeleteLinkRecord 582
- DeleteNodeRecord 582
- DetectDataTableFieldName 583
- DetectDataTableName 583
- DetectFieldIndex 584
- DiagramBackgroundColor 536
- DialogFont 536
- DoubleOutputFormat 537
- DumpConfiguration 584
- Enabled 538
- EndLoading 585
- ExportGraphicsToFileEx 585
- ExtendedDataTablesEnabled 538
- FilePath 538
- FilterCollection 539
- FontAntiAliasingEnabled 540
- GetAValueFromARGB 588
- GetBValueFromARGB 588
- GetGValueFromARGB 589
- GetLinkByID 590
- GetLinkByNodeIDs 591
- GetNodeByID 591
- GetRValueFromARGB 592
- GroupCollection 540
- GroupHorizontalMargin 541

- GroupingActivated 541
- GroupingDataFieldIndex 541
- GroupingTitlesFullyVisible 542
- GroupingType 543
- GroupInteractionsAllowed 543
- GroupSortingDataFieldIndex 544
- GroupSortMode 544
- GroupTitleDataFieldIndex 545
- GroupTitlesFileName 545
- GroupVerticalMargin 546
- IdentifyFormatField 592
- IdentifyObjectAt 593
- ImportConfiguration 595
- InbuiltMouseCursorWhileDraggingEnabled 547
- InFlowGroupingActivated 547
- InFlowGroupingDataFieldIndex 548
- InFlowGroupSeparationLineColor 548
- InFlowGroupSeparationLineType 549
- InFlowGroupTimeInterval 550
- InFlowGroupTitleDataFieldIndex 550
- InFlowGroupTitlesBackgroundColor 551
- InFlowGroupTitlesFileName 551
- InFlowGroupTitlesFont 552
- InFlowGroupTitlesVisibleAtBottomOrRight 552
- InFlowGroupTitlesVisibleAtTopOrLeft 552
- InFlowGroupTitleTimeFormat 553
- InFlowGroupVerticalCaptionWidth 553
- InPlaceEditingAllowed 553
- InsertLinkRecord 596
- InsertNodeRecord 596
- InteractionMode 554
- InterfaceNodesShown 554
- LegendView 555
- LinkAnnotationColumnNumberDataFieldIndex 555
- LinkAnnotationRowNumberDataFieldIndex 556
- LinkAppearanceCollection 556
- LinkCollection 557
- LinkCreationWithDialog 557
- LinkFormatCollection 558
- LinkPredecessorDataFieldIndex 558
- LinksDataTableName 559
- LinkSuccessorDataFieldIndex 560
- LinkTypeDataFieldIndex 561
- Load 597
- MakeARGB 597
- MapCollection 562
- MinimumColumnWidth 563
- MinimumRowHeight 563
- MouseProcessingEnabled 564
- MovingCollapsedClustersAllowed 564
- NodeAndLinkCreationAllowed 565
- NodeAppearanceCollection 565
- NodeCalendarNameDataFieldIndex 566
- NodeChangeRankToPredecessorRankDataFieldIndex 566
- NodeCollection 566
- NodeColumnNumberDataFieldIndex 567
- NodeCreationWithDialog 567
- NodeFormatCollection 567
- NodeRowNumberDataFieldIndex 568
- NodesDataTableName 569
- NodesUseCalendars 569
- NodeToolTipTextDataFieldIndex 570
- ObliqueTracksOnLinks 570
- Orientation 571
- PasteNodesFromClipboard 598

- PhantomDrawingWhileDraggingEnabled 571
- PixelsToRaster 599
- Printer 572
- PrintEx 599
- PrintToFile 600
- Reset 601
- RoundedLinkSlantsEnabled 572
- SaveAsEx 601
- ScheduleProject 602
- Scheduler 573
- ScrollToNode 603
- SetImageResource 603
- ShortenedLinks 573
- ShowAboutDialog 604
- ShowExportGraphicsDialog 605
- ShowLinkEditDialog 606
- ShowNodeEditDialog 607
- ShowPageSetupDialog 607
- ShowPrintDialog 608
- ShowPrinterSetupDialog 608
- ShowPrintPreviewDialog 609
- StraightLinkDrawing 573
- SuspendUpdate 609
- TextEntrySupplyingEventEnabled 574
- TimeUnit 574
- ToolTipChangeDuration 575
- ToolTipDuration 575
- ToolTipPointerDuration 576
- ToolTipShowAfterClick 576
- ToolTipTextSupplyingEventEnabled 576
- UngroupedNodesAllowed 577
- UpdateLinkRecord 611
- UpdateNodeRecord 611
- VcBoxLeftClicking 613
- VcBoxLeftDoubleClicking 613
- VcBoxModified 614
- VcBoxModifying 615
- VcBoxRightClicking 616
- VcDataModified 617
- VcDataRecordCreated 618
- VcDataRecordCreating 619
- VcDataRecordDeleted 620
- VcDataRecordDeleting 621
- VcDataRecordModified 622
- VcDataRecordModifying 622
- VcDataRecordNotFound 623
- VcDiagramLeftClicking 624
- VcDiagramLeftDoubleClicking 625
- VcDiagramRightClicking 625
- VcDragCompleting 626
- VcDragStarting 627
- VcErrorOccurring 627
- VcFieldSelecting 628
- VcGiveFeedbackOnNodeCreating 629
- VcGroupCreated 630
- VcGroupDeleting 630
- VcGroupLeftClicking 631
- VcGroupLeftDoubleClicking 632
- VcGroupModified 632
- VcGroupModifying 633
- VcGroupRightClicking 634
- VcHelpRequested 635
- VcInPlaceEditorShowing 636
- VcLegendViewClosed 638
- VcLinkCreated 639
- VcLinkCreating 640
- VcLinkDeleted 641
- VcLinkDeleting 641
- VcLinkModified 642
- VcLinkModifying 642
- VcLinksLeftClicking 643

- VcLinksLeftDoubleClicking 644
- VcLinksMarked 645
- VcLinksMarking 646
- VcLinksRightClicking 647
- VcMouseDoubleClicking 648
- VcMouseDown 648
- VcMouseMove 649
- VcMouseUp 650
- VcNodeCreated 651
- VcNodeCreating 651
- VcNodeDeleted 652
- VcNodeDeleting 653
- VcNodeLeftClicking 653
- VcNodeLeftDoubleClicking 654
- VcNodeModifiedEx 655
- VcNodeModifying 656
- VcNodeRightClicking 657
- VcNodesMarked 658
- VcNodesMarking 659
- VcStatusLineTextShowing 660
- VcTextEntrySupplying 661
- VcToolTipTextSupplying 671
- VcWorldViewClosed 673
- VcZoomFactorModified 673
- ViewXCoordinate 577
- ViewYCoordinate 578
- WaitCursorEnabled 578
- WorldView 578
- Zoom 612
- ZoomFactor 579
- ZoomingPerMouseWheelAllowed 579
- ZoomOnMarkedNodes 612
- VcNode 675**
  - AllData 675
  - DataField 676
  - DataRecord 680
  - Delete 680
  - ID 677
  - IncomingLinks 677
  - Marked 678
  - OutgoingLinks 679
  - RelatedDataRecord 681
  - Update 681
- VcNodeAppearance 683**
  - BackgroundColor 684
  - BackgroundColorDataFieldIndex 685
  - BackgroundColorMapName 685
  - DoubleFeature 686
  - FilterName 686
  - FormatName 687
  - FrameAroundFieldsVisible 688
  - FrameShape 688
  - LegendText 690
  - LineColor 690
  - LineColorDataFieldIndex 691
  - LineColorMapName 691
  - LineThickness 691
  - LineType 692
  - Name 694
  - Pattern 694
  - PatternColor 698
  - PatternColorDataFieldIndex 698
  - PatternColorMapName 699
  - PatternDataFieldIndex 699
  - PatternMapName 699
  - PileEffect 700
  - PutInOrderAfter 704
  - Shadow 700
  - ShadowColor 701
  - Specification 701
  - StrikeThrough 702
  - StrikeThroughColor 703
  - ThreeDEffect 703



- VisibleInLegend 704
- VcNodeAppearanceCollection 706**
  - Add 707
  - AddBySpecification 707
  - Copy 708
  - Count 706
  - FirstNodeAppearance 708
  - GetEnumerator 709
  - NextNodeAppearance 710
  - NodeAppearanceByIndex 710
  - NodeAppearanceByName 711
  - Remove 711
- VcNodeCollection 712**
  - Count 712
  - FirstNode 713
  - GetEnumerator 713
  - NextNode 714
  - SelectNodes 714
- VcNodeCreated**
  - Ereignis von
    - VcNet 651
- VcNodeCreating**
  - Ereignis von
    - VcNet 651
- VcNodeDeleted**
  - Ereignis von
    - VcNet 652
- VcNodeDeleting**
  - Ereignis von
    - VcNet 653
- VcNodeFormat 716**
  - CopyFormatField 719
  - FieldsSeparatedByLines 716
  - FormatField 717
  - FormatFieldCount 717
  - GetEnumerator 720
  - Name 718
  - RemoveFormatField 720
  - Specification 718
  - WidthOfExteriorSurrounding 719
- VcNodeFormatCollection 721**
  - Add 722
  - AddBySpecification 723
  - Copy 723
  - Count 721
  - FirstFormat 724
  - FormatByIndex 724
  - FormatByName 725
  - GetEnumerator 725
  - NextFormat 726
  - Remove 727
- VcNodeFormatField 728**
  - Alignment 729
  - BackgroundColor 729
  - BackgroundColorDataFieldIndex 730
  - BackgroundColorMapName 730
  - BottomMargin 731
  - ConstantText 731
  - FormatName 731
  - GraphicsFileName 731
  - GraphicsFileNameDataFieldIndex 732
  - GraphicsFileNameMapName 732
  - GraphicsHeight 732
  - Index 733
  - LeftMargin 733
  - MaximumTextLineCount 733
  - MinimumTextLineCount 734
  - MinimumWidth 734
  - PatternBackgroundColorAsARGB 734
  - PatternBackgroundColorDataFieldIndex 735
  - PatternBackgroundColorMapName 735

- PatternColorAsARGB 735
- PatternColorDataFieldIndex 736
- PatternColorMapName 736
- PatternEx 737
- PatternExDataFieldIndex 737
- PatternExMapName 738
- RightMargin 738
- TextAndGraphicsCombined 738
- TextDataFieldIndex 739
- TextFont 739
- TextFontColor 739
- TextFontDataFieldIndex 740
- TextFontMapName 740
- TopMargin 740
- Type 741
- VcNodeLeftClicking**
  - Ereignis von
    - VcNet 653
- VcNodeLeftDoubleClicking**
  - Ereignis von
    - VcNet 654
- VcNodeModifiedEventArgs**
  - Ereignisobjekt von
    - VcNodeModifiedEx 655
    - VcNodeModifying 656
- VcNodeModifiedEx**
  - Ereignis von
    - VcNet 655
- VcNodeModifying**
  - Ereignis von
    - VcNet 656
- VcNodeRightClicking**
  - Ereignis von
    - VcNet 657
- VcNodesMarked**
  - Ereignis von
    - VcNet 658
- VcNodesMarking**
  - Ereignis von
    - VcNet 659
- VcPrinter 742**
  - AbsoluteBottomMarginInInches 743
  - AbsoluteLeftMarginInCM 743
  - AbsoluteLeftMarginInInches 744
  - AbsoluteRightMarginInCM 744
  - AbsoluteRightMarginInInches 745
  - AbsoluteTopMarginInCM 745
  - AbsoluteTopMarginInInches 746
  - Alignment 746
  - CurrentHorizontalPagesCount 747
  - CurrentVerticalPagesCount 747
  - CurrentZoomFactor 747
  - CuttingMarks 748
  - DefaultPrinterName 748
  - DocumentName 748
  - FitToPage 749
  - FoldingMarksType 749
  - MarginsShownInInches 752
  - MaxHorizontalPagesCount 752
  - MaxVerticalPagesCount 753
  - Orientation 753
  - PageDescription 754
  - PageDescriptionString 754
  - PageFrame 755
  - PageNumberMode 755
  - PageNumbers 756
  - PagePaddingEnabled 756
  - PaperSize 757
  - PrintDate 758
  - PrinterName 758
  - PrintPreviewWithFirstPage 758
  - TitleAndLegendOnAllPages 759
  - VcCalendarGrid 760
  - ZoomFactorAsDouble 760

**VcRect 761**

- Bottom 761
- Height 761
- Left 762
- Right 763
- Top 763
- Width 764

**VcScheduler 765**

- ActualEndDateDataFieldIndex 766
- ActualStartDateDataFieldIndex 766
- AutomaticSchedulingEnabled 766
- DurationDataFieldIndex 767
- EarlyEndDateDataFieldIndex 767
- EarlyStartDateDataFieldIndex 767
- EndDateForAutomaticScheduling 768
- EndDateNotLaterThanDataFieldIndex 768
- FreeFloatDataFieldIndex 768
- LateEndDateDataFieldIndex 768
- LateStartDateDataFieldIndex 769
- LinkDurationDataFieldIndex 769
- ScheduledProjectEndDate 769
- ScheduledProjectStartDate 770
- ScheduleProject 771
- ScheduleSuccessorsOnlyEnabled 770
- StartDateForAutomaticScheduling 770
- StartDateNotEarlierThanDataFieldIndex 771
- TotalFloatDataFieldIndex 771

**VcStatusLineTextShowing**

- Ereignis von VcNet 660

**VcTextEntrySupplying**

- Ereignis von VcNet 661

**VcTextEntrySupplying-Ereignisse 156**

**VcTextEntrySupplyingEventArgs**

- Ereignisobjekt von VcTextEntrySupplying 661

**VcToolTipTextSupplying**

- Ereignis von VcNet 671

**VcToolTipTextSupplying-Ereignisse 155**

**VcWorldView 773**

- Border 773
- Height 774
- HeightActualValue 774
- Left 775
- LeftActualValue 775
- MarkingColor 776
- Mode 776
- ParentHWND 777
- ScrollBarMode 778
- Top 778
- TopActualValue 779
- UpdateBehaviorName 779
- Visible 780
- Width 780
- WidthActualValue 781

**VcWorldViewClosed**

- Ereignis von VcNet 673

**VcZoomFactorModified**

- Ereignis von VcNet 673

**Verbindung**

- bearbeiten 259
- ID 462
- markiert/nicht markiert 462

**Verbindungen 133, 175**

- abgerundete Schrägen 158, 159
- anzeigen 222
- bearbeiten 37
- erzeugen 37
- interaktiv erzeugen 285, 286
- kürzen beim Anordnen 156
- markieren 39
- Markierungstyp 39, 176
- Nachfolgerknoten 175
- neue zulassen 156
- orthogonal 136, 157
- Positionen von
  - Verbindungsbeschriftungen 106
- schräg 136, 157
- Verbindungsausssehen 50
- Verbindungsausssehen verwalten 222
- Verbindungsformate verwalten 217
- Verbindungstyp 175
- Vorgängerknoten 175
- Verbindungsausssehen 139**
  - Sortierung 475
- Verbindungsausssehen-Auflistung**
  - hinzufügen 477
  - hinzufügen über Spezifikation 478
  - kopieren 478
  - löschen 481
- Verbindungsbeschriftungspositionen 55**
- Verbindungsformat-Auflistung**
  - Anzahl 491
  - Enumerator 495
  - Erstes Format 494
  - hinzufügen 493
  - hinzufügen über Spezifikation 493
  - kopieren 493
  - löschen 497
  - nächstes Format 496
  - Zugriff über Index 494
  - Zugriff über Name 495
- Viewer Metafile (\*.vmf) 141**
- ViewXCoordinate**
  - Eigenschaft von
    - VcNet 577
- ViewYCoordinate**
  - Eigenschaft von
    - VcNet 578
- Visible**
  - Eigenschaft von
    - VcBox 314
    - VcLegendView 456
    - VcLinkAppearance 474
    - VcWorldView 780
- VisibleInLegend**
  - Eigenschaft von
    - VcNodeAppearance 704
- Visualisierungsmodus 101, 163**

## W

- WaitCursorEnabled**
  - Eigenschaft von
    - VcNet 578
- Width**
  - Eigenschaft von
    - VcLegendView 457
    - VcRect 764
    - VcWorldView 780
- WidthActualValue**
  - Eigenschaft von
    - VcLegendView 457
    - VcWorldView 781
- WidthOfExteriorSurrounding**
  - Eigenschaft von
    - VcNodeFormat 719
- WindowMode**

Eigenschaft von  
VcLegendView 458

**Worldview 116**

**WorldView**

Eigenschaft von  
VcNet 578  
Name UpdateBehavior 779  
siehe auch  
VcWorldView 773

**X**

**X**

Eigenschaft von  
VcGroup 433

**Y**

**Y**

Eigenschaft von  
VcGroup 433

**Z**

**Zeilenhöhe**

minimale 151

**Zeitberechnung**

automatisch 766

**Zeiteinheit 152**

für Dauern 177

**Zeitkritische Operationen**

Wartecursor 159

**Zeitrechnung 65, 142, 177, 573**

Aktuelles Anfangsdatum 766

Aktuelles Enddatum 766

Dauer 767

durchführen 772

Enddatum berechnen 770

freier Puffer 768

frühestmögliches Anfangsdatum 767

frühestmögliches Enddatum 767

geplantes Anfangsdatum 771

geplantes Enddatum 768

Gesamtpuffer 771

nur Knoten mit Vorgängern  
berechnen 770

spätestmögliches Anfangsdatum 769

spätestmögliches Enddatum 768

Startdatum berechnen 768

Verbindungsdauer 769

Zeitrechnungseingabe 177

Zeitrechnungsergebnis 178

**Zeitrechnung: 769, 770**

**Zeitumstellung 88**

**Zoom**

Methode von  
VcNet 612

**Zoomen 255**

per Mausrad 155

**ZoomFactor**

Eigenschaft von  
VcNet 579

**ZoomFactorAsDouble**

Eigenschaft von  
VcPrinter 760

**ZoomingPerMouseWheelAllowed**

Eigenschaft von  
VcNet 579

**ZoomOnMarkedNodes**

Methode von  
VcNet 612

**Zuordnungstabelle**

über Index 513

**Zuordnungstabellen 144**

Angabe von Wertebereichen durch  
Filter 502

**Zusatztext 267**