VARCHART XGantt

.NET Edition 5.2
Benutzer- und Referenzhandbuch

NETRONIC

VARCHART XGantt .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
www.netronic.de

Bearbeitungsstand: 27. April 2020

Inhaltsverzeichnis

1 Einleitung

1.1 VARCHART XGantt auf einen Blick

Gantt-Diagramme sind das geeignete Instrument, um die zeitliche Abfolge von Aufgaben oder die Auslastung von Ressourcen zu planen und zu visualisieren. Durch die grafische Darstellung werden Zusammenhänge und Veränderungen im Zeitablauf für alle Beteiligten leichter und schneller erfassbar. Neben dem Einsatz im Projektmanagementbereich haben sich Gantt-Diagramme vor allem in Leitständen der Fertigung und in Systemen der Ressourcenverwaltung und Disposition etabliert.

VARCHART XGantt ist eine interaktive Grafikkomponente, die sich innerhalb kurzer Zeit sehr leicht in eigene Anwendungen integrieren lässt, da die aufwändige Grafikprogrammierung entfällt. Durch die vielfältigen Gestaltungsmöglichkeiten genügt VARCHART XGantt auch sehr individuellen grafischen Anforderungen. Die Druckausgabe überzeugt durch ihre erstklassige Qualität.

VARCHART XGantt .NET ist ein Windows-Forms-Steuerelement, das vollständig auf das Microsoft .NET Framework abgestimmt wurde.

> **Die Leistungsmerkmale von VARCHART XGantt sind:**

- Anlegen, Löschen und Verschieben von Knoten
- Anlegen und Löschen von Verbindungen (vernetztes Gantt-Diagramm)
- Visualisierung der Termine durch Symbole und Balken
- Datengesteuerte Attributierung der Darstellungselemente
- Gruppierung und Sortierung nach beliebigen Kriterien
- Kollabieren und Expandieren von Vorgangsgruppen
- Variabler Aufbau der Zeitskala
- Flexible Gestaltung des Tabellenteils
- Hinzufügen von Stichtaglinien und Rasterlinien
- Stufenloses Zoomen im Diagrammbereich
- Bildschirmfüllende Darstellung von Diagrammausschnitten
- Integrierte Seitenvorschau und Druckausgabe mit Seitenaufteilung

- Austausch der Anwendungsdaten über Dateien oder die Programmierschnittstelle
- Vielfältige Gestaltungsoptionen für Histogramme
- Bequemes Einstellen der Eigenschaften zur Entwurfszeit über die Eigenschaftenseiten
- Abwandlung des Standardverhaltens durch Eingriff in Ereignisse
- Umfangreiche Programmierschnittstelle

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.

Vor der Installation

- **Wenn Sie von XGantt 4.4 auf XGantt 5.0 umsteigen**

  Führen Sie bitte die im Kapitel 6.1 "Was muss ich tun, wenn ich von VARCHART XGantt 4.4 für .NET nach XGantt 5.0 umsteigen möchte?" beschriebenen Schritte aus

- **Wenn Sie im Rahmen eines Service Releases auf den neuen Build wechseln möchten**
- Bitte führen Sie bitte die im Kapitel 6.2 "Was muss ich tun, wenn ich im Rahmen eines Service Releases auf einen neuen Build von VARCHART XGantt umsteigen möchte?" beschriebenen Schritte aus

Installation

Um das Steuerelement VARCHART XGantt .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)

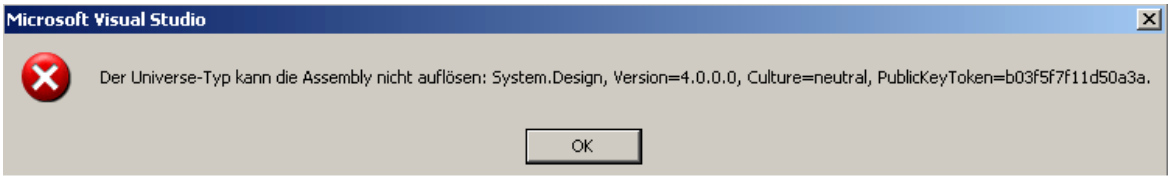
abgelegt.

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.XGantt.dll** aus dem entsprechenden Installationsordner auswählen. Nach dem Bestätigen mit **OK** erscheint das Symbol für VARCHART XGantt .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:



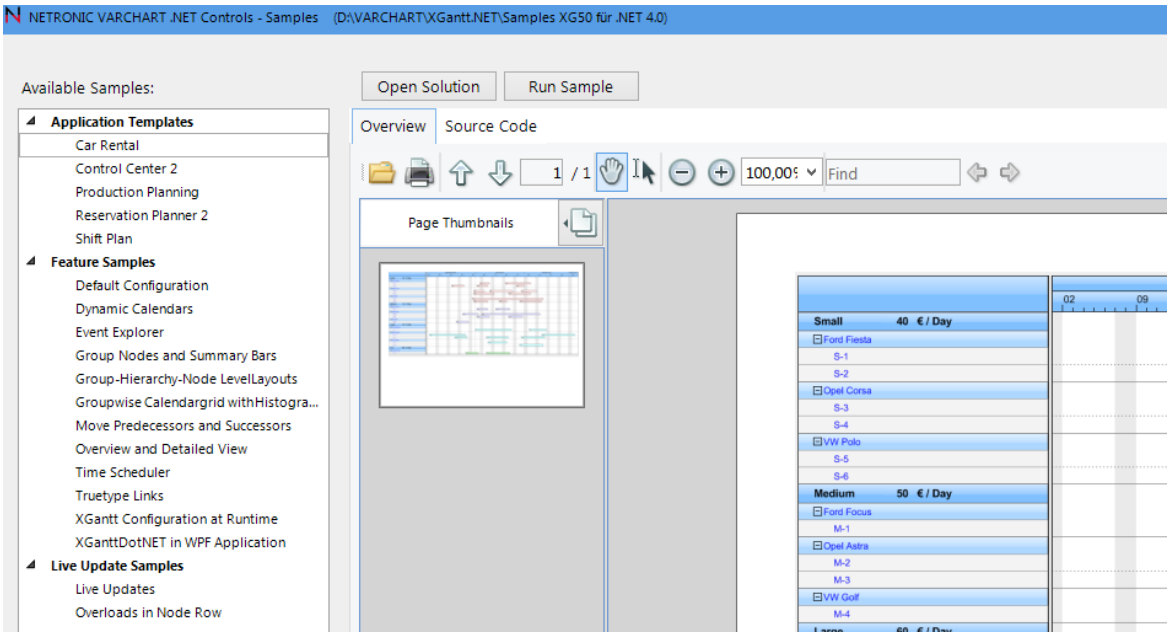
Sie können VARCHART XGantt auch bedienerlos installieren. Dazu geben Sie ein:

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

Damit läuft die Installation ohne jegliche Benutzereingabe und ohne Fortschrittsmeldungen 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. Fortschrittsmeldungen: /qb: Es erscheinen Fortschrittsmeldungen; /qn: Es erscheinen keine Fortschrittsmeldungen, d.h. Sie sehen am Bildschirm nichts.
4. Start/wait sollte man benutzen, wenn die Installation über eine Batch-Datei abläuft; andernfalls liefe die Batch-Datei während der Installation parallel weiter.

Mit dem Setup wird gleichzeitig auch unsere Beispielsammlung installiert und ein entsprechendes Symbol  auf Ihrem Desktop abgelegt. Starten Sie die Beispielsammlung durch Doppelklick auf dieses Symbol.



Zum Starten eines Beispiels markieren Sie es in der Liste und klicken dann auf Run Sample. Mit Klick auf Open Solution wird die Solution des ausgewählten Beispiels in Visual Studio geöffnet, ssodass Sie sich den Quellcode oder die aktuelle Konfiguration von XGantt auf den Eigenschaftenseiten ansehen können. Dies setzt natürlich voraus, daßss Visual Studio auf Ihrem Rechner installiert ist. Der Quellcode kann auf der Registerkarte **Source Code** auch jederzeit im Text-Modus angezeigt werden.

Die Beispiele sind in Gruppen zusammengefasst. In der ersten Gruppe **Application Templates** finden Sie einige praktische Anwendungsbeispiele:

1. Car Rental: Hier werden Mietaufträge für Fahrzeuuge geplant und verwaltet.
2. Control Center 2: Planen Sie Aufträge auf Maschinen ein, die in Gruppen zusammengeffasst sind. Die Aufträge sind in einer Access -Datenbank abgelegt.
3. Production Planning: eine klassische Produktionsplanung
4. Reservation Planner 2: Verwalten Sie Schulungsräume und Referenten für Seminare und lassen Sie sich Konflikte grafisch anzeigen.
5. Shift Plan: Weisen Sie Mitarbeitern verschiedene Arbeitsschichten zu.

Die Gruppe **Feature Samples** stellt verschiedene Features von XGantt exemplarisch vor.

In den Beispielen der Gruppe **Live Update Samples** wird gezeigt, wie Sie die Auswirkungen Ihrer Interaktionen in XGantt bereits während der Interaktion visualisieren können.

1.3 Lizenzierung

1.3.1 Entwickler-Lizenzen

Zum Lizenzieren von VARCHART XGantt .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 "BXnnnn" 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.

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

1.4 Auslieferung

Wenn Sie eine mit VARCHART XGantt .NET entwickelte Anwendung an Ihre Kunden weitergeben möchten, müssen Sie die im folgenden aufgeführten Dateien mit ausliefern. Alle anderen Dateien, die zum Produkt VARCHART XGantt .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.XGantt.dll

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

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

mfc80u.dll

mfcm80u.dll

msvcp80.dll

msvcr80.dll

Die Installation der Bibliotheken *mfc80u.dll*, *msvcp80.dll*, *mfcm80u.dll* und *msvcr80.dll* muss über die im Unterverzeichnis **redist** mitgelieferten Setup-Dateien *vcredist_vs2005sp1_x86.exe* bzw. *vcredist_vs2005sp1_x64.exe* erfolgen.

Nähere Informationen dazu bietet folgende Seite:

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

- **Wenn Sie die Ressourcenplanung nutzen möchten:**
  - für die **x86**-Variante: opsaps.dll
  - für die **x64**-Variante: opsaps64.dll

> Framework .NET 4.0/4.5

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

NETRONIC.XGantt.dll

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

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

mfc100u.dll

mfcm100u.dll

msvcp100.dll

msvcr100.dll

Die Installation der Bibliotheken *mfc100u.dll, msvcp100.dll, mfcm100u.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.

- **Wenn Sie die Ressourcenplanung nutzen möchten:**
  - für die **x86**-Variante: opsaps.dll
  - für die **x64**-Variante: opsaps64.dll

VARCHART XGantt.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)

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 XGantt .NET Edition ist in Deutsch, Englisch und Chinesisch (vereinfachtes Chinesisch) verfügbar. Bei der deutschen bzw. der chinesischen Version wird während der Installation zusätzlich zur Control-Assembly NETRONIC. XGantt.dll die Ressource-Assembly NETRONIC. XGantt**d**.dll/NETRONIC. XGantt**c**. dll ins Installationsverzeichnis kopiert.

Verwendung zur Designzeit

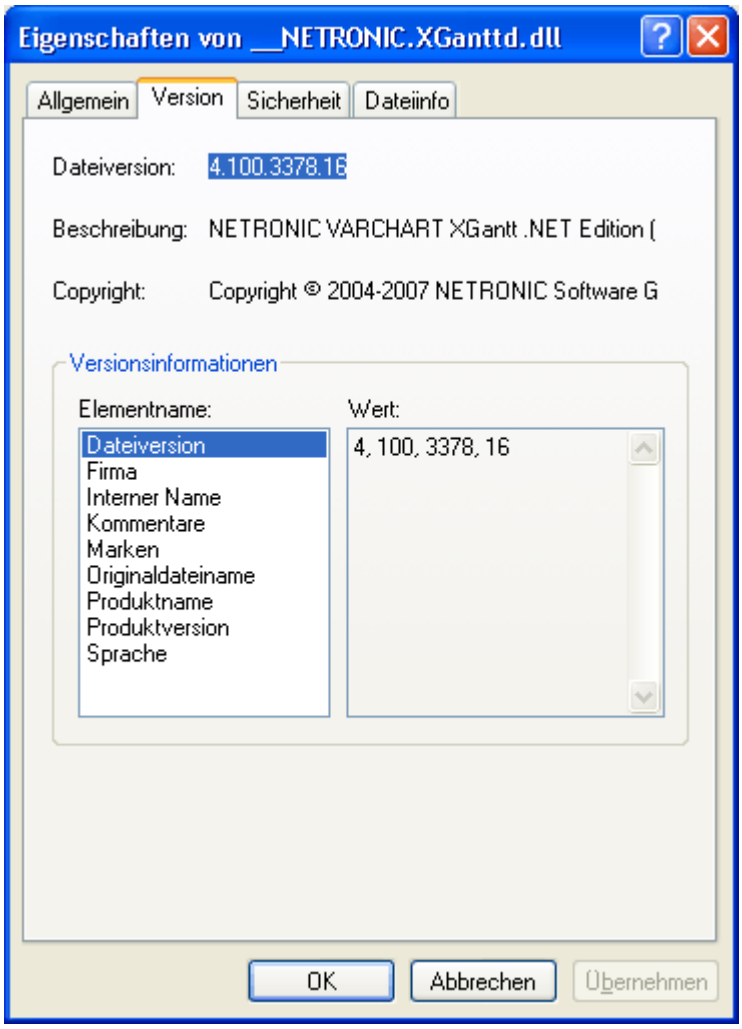
Wenn in den **Regionalen Einstellungen** (Systemsteuerung, Regions- und Sprachoptionen, Regionale Einstellungen) **Deutsch**/**Chinesisch** eingestellt ist, wird die Ressource-Assembly aus dem Installationsverzeichnis geladen und die deutschen/chinesischen Dialoge und Eigenschaftenseiten stehen zur Designzeit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

Verwendung zur Laufzeit

Um sicher zu stellen, dass auch zur Laufzeit auf die Ressource-Assembly zugegriffen wird und deutsche/chinesische 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 Assemblys übereinstimmen (im Windows-Explorer das Kontextmenü der Datei aufrufen, Menüpunkt **Eigenschaften**, Registerkarte **Version**).


Eigenschaften von __NETRONIC.XGanttd.dll
Allgemein
Version
Sicherheit
Dateiinfo
Dateiversion:
4.100.3378.16
Beschreibung:
NETRONIC VARCHART XGantt .NET Edition (
Copyright:
Copyright © 2004-2007 NETRONIC Software G
Versionsinformationen
Elementname:
Wert:
Dateiversion
Firma
Interner Name
Kommentare
Marken
Originaldateiname
Produktname
Produktversion
Sprache
4, 100, 3378, 16
OK
Abbrechen
Übernehmen

1.6 Unterstützung und Beratung

Sie wissen nicht, ob VARCHART XGantt die speziellen Anforderungen Ihres Balkenplans erfüllt?

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

Sie machen gerade den Anfangstest mit VARCHART XGantt und möchten eine ganz spezifische Anforderung Ihres Balkenplans 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

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 XGantt 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 von VARCHART XGantt vertraut, die notwendig sind, um die Balkenplan-Komponente in eigene Anwendungen integrieren zu können.

Wir werden Schritt für Schritt die bedeutsamen Aspekte von VARCHART XGantt für die Anwendungsentwicklung erläutern und auf die vielfältigen Gestaltungsmöglichkeiten näher eingehen. Aus diesem Grund empfiehlt es sich, dieses Kapitel chronologisch 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 XGantt nach Wunsch zu konfigurieren, ohne dass Sie dafür eine einzige Programmzeile 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.

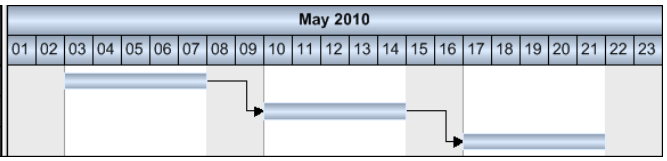
- **API-Referenz**

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

Als Entwicklungsumgebung für die Beispiele verwenden wir Visual Studio .NET 2005.

Unser erstes Programmierbeispiel wird Sie zu folgendem Ergebnis führen:

ID	Name	Start	End	Duration
1	Node 1	03.05.10	08.05.10	5
2	Node 2	10.05.10	15.05.10	5
3	Node 3	17.05.10	22.05.10	5

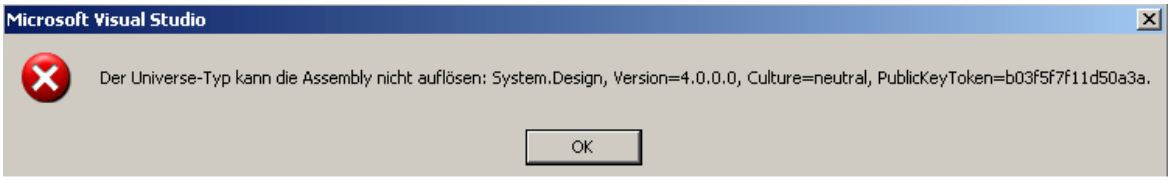


Sie finden dieses Einstiegsbeispiel im Ordner **UserGuideSamples\VB.NET\XGantt_Tutorial01_App** bzw. im Ordner **UserGuideSamples\Csharp\XGantt_Tutorial01_App**.

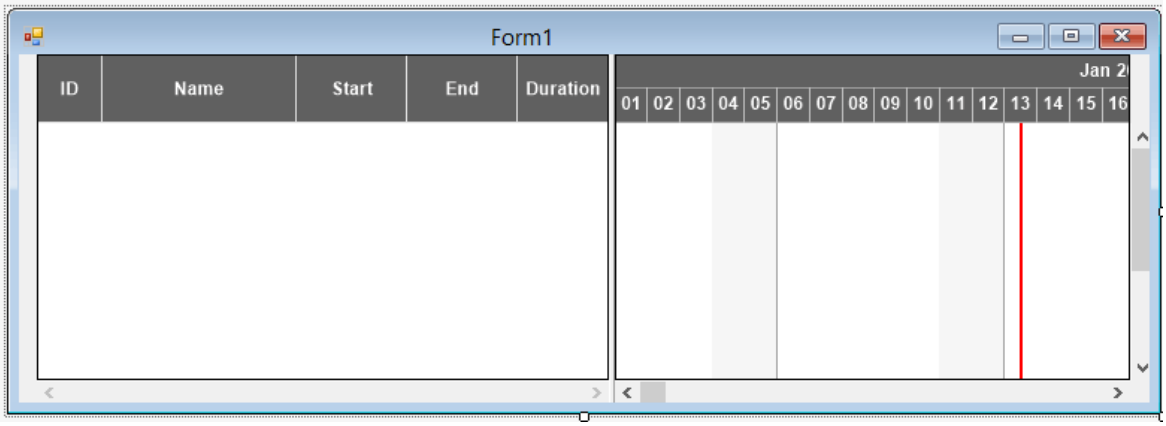
Anhand dieser Anwendung werden Sie die in VARCHART XGantt bereits eingebauten Interaktionen näher kennenlernen.

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 XGantt in einem Formular zu platzieren, wählen Sie das Steuerelement  in der Werkzeugleiste aus und ziehen im Formular an der gewünschten Stelle mit der Maus einen Rahmen auf.



Wenn sich das Steuerelement zur Laufzeit mit seinem rechten und unteren Rand stets an die Fenstergröße anpassen soll, müssen sowohl im Load- als auch im Resize-Ereignis des Formulars die folgenden Programmzeilen enthalten sein:

Code-Beispiel VB.NET

```
Private Sub Form1_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    VcGantt1.Width  = ClientSize.Width - VcGantt1.Left
    VcGantt1.Height = ClientSize.Height - VcGantt1.Top
End Sub

Private Sub Form1_Resize(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Resize
```

```
    VcGantt1.Width  = ClientSize.Width - VcGantt1.Left
    VcGantt1.Height = ClientSize.Height - VcGantt1.Top
End Sub
```

Code-Beispiel C#

```
private void Form1_Load(object sender, System.EventArgs e)
   {
   vcGantt1.Width  = ClientSize.Width - vcGantt1.Left;
   vcGantt1.Height = ClientSize.Height - vcGantt1.Top;
   }

Private void Form1_Resize(object sender, System.EventArgs e)
   {
   vcGantt1.Width  = ClientSize.Width - vcGantt1.Left;
   vcGantt1.Height = ClientSize.Height - vcGantt1.Top;
   }
```

Tipp:

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.XGantt

C#: using NETRONIC.XGantt;

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

2.3 Daten bereitstellen

Damit Vorgänge und Verbindungen dargestellt werden können, muss VARCHART XGantt mit Daten versorgt werden. Die dazu notwendige Kommunikation erfolgt standardmäßig über zwei Tabellen:

1. NodeTable (auch Maindata genannt)
2. LinkTable (auch Relations genannt)

Immer wenn Sie VARCHART XGantt auf ein Formular ziehen, sind die wichtigsten Datenfelder bereits vorgegeben.

Felder der Maindata Datentabelle:

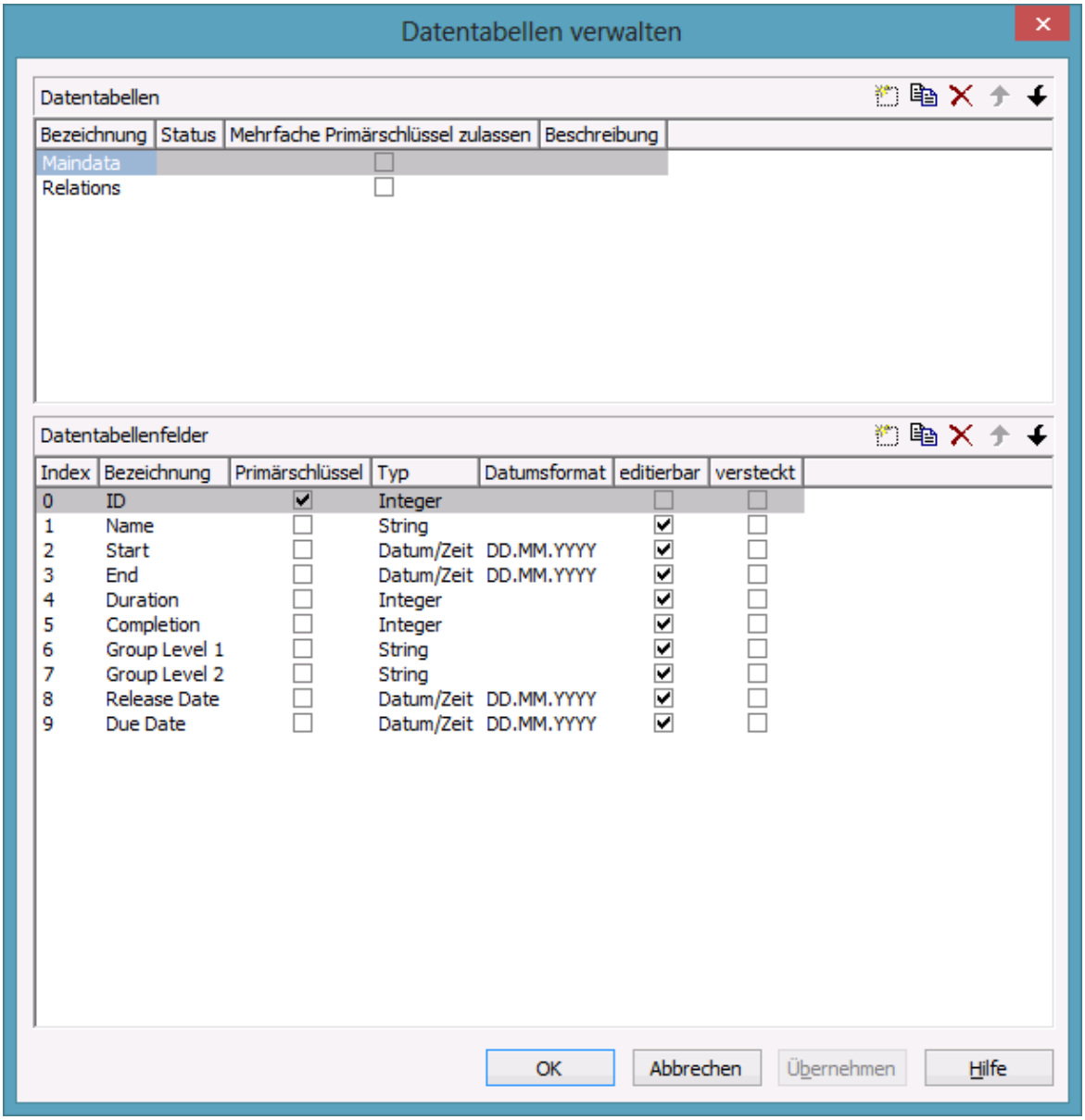
Index	Name	Primary key	Type	DateFormat	Editierbar	Hidden
0	ID	True	Integer		True	False
1	Name	False	String		False	False
2	Start	False	DateTime	DD.MM.YYYY	False	False
3	End	False	DateTime	DD.MM.YYYY	True	False
4	Duration	False	Integer		False	False

Felder der Relations Datentabelle:

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

Alle weiteren Felder müssen Sie selbst definieren. Dies kann zur Designzeit über den Dialog **Datentabellen verwalten** geschehen (unterer Bereich) oder zur Laufzeit über die Methode **Add(...)** des Objekts **VcDataTableFieldCollection** erfolgen.

Sollten Sie mehr Tabellen benötigen als die beiden standardmäßig vorhandenen, können Sie diese im oberen Bereich des Dialogs **Datentabellen verwalten** anlegen, nachdem Sie auf der Eigenschaftenseite **Allgemeines** die Option **Erweiterte Datentabellen zulassen** aktiviert haben.



Mit der Methode **DataRecordByID()** von **VcDataRecordCollection** lässt sich ein gesuchtes Objekt anhand des Primärschlüssels schnell finden.

Damit in unserem Einstiegsbeispiel Vorgänge und Verbindungen im Gantt-Diagramm zu sehen sind, müssen zunächst einige Datensätze in die Datentabellen eingefügt werden.

Dies geschieht mit der Methode **Add(...)** des Objekttyps **VcDataRecordCollection**. Mit **EndLoading** wird die Dateneingabe abgeschlossen und die entsprechende Grafik kann aufgebaut werden. Deshalb ergänzen Sie bitte die nachfolgenden Programmzeilen im **Load**-Ereignis des Formulars.

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
```

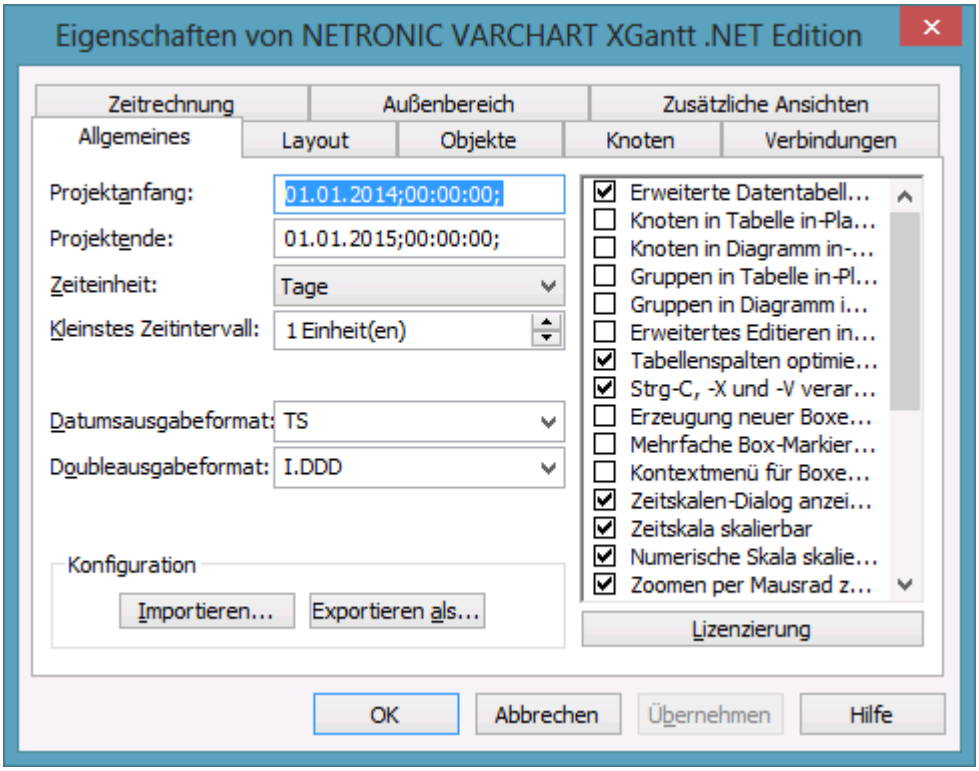
```
   VcGantt1.ExtendedDataTablesEnabled = True
   dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
   dataRecCltn = dataTable.DataRecordCollection
   dataRecCltn.Add("1;Node 1;07.05.2010;;5")
   dataRecCltn.Add("2;Node 2;14.05.2010;;5")
   dataRecCltn.Add("3;Node 3;21.05.2010;;5")

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

   VcGantt1.EndLoading
```

Code-Beispiel C#

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

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

   vcGantt1.EndLoading;
```

Die Werte innerhalb eines Datensatzes werden durch Semikolon getrennt. Die Reihenfolge der Felder muss der Reihenfolge der Felder in der Datendefinition entsprechen. Jeder neue Datensatz erfordert eine eindeutige, nicht leere Identifikation. Die Datumsangabe im Datensatz muss mit der DateFormat-Vereinbarung in der Datendefinitionstabelle übereinstimmen. Die Interpretation der Dauer hängt von der Einstellung **Zeiteinheit** auf der Eigenschaftenseite **Allgemeines** ab und ist auf **Tage** voreingestellt.

Das Format der Datumsausgabe wird einheitlich für die Tabelle und alle Dialoge auf der Eigenschaftenseite **Allgemeines** durch das **Datumsausgabeformat** eingestellt.



Daten aus einer CSV-Datei importieren

Alternativ können Sie die Daten auch aus einer CSV-Datei laden. Der Aufbau der Datei muss folgendem Schema entsprechen:

Code-Beispiel

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

Jeder Datensatz steht in einer eigenen Zeile. Die Zeileninhalte entsprechen dem Übergabeparameter der Methode **Add(...)** des Objekttyps **VcDataRecordCollection**.

Zuerst werden alle Datensätze der Maindata aufgeführt und anschließend die Datensätze der Relations. Die Datensatzgruppen werden mit

**** Tabellenname **** eingeleitet.

Wenn Sie eine solche Datei beispielsweise unter dem Namen **intro.csv** abgespeichert haben, können Sie die Daten mit folgendem Befehl in VARCHART XGantt importieren:

Code-Beispiel VB.NET

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

Code-Beispiel C#

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

Darstellungszeitraum festlegen

Bislang würden Sie noch keine Vorgänge sehen, weil die Zeitskala noch nicht auf den Zeitraum eingestellt ist, in dem die Vorgänge liegen. Der darstellbare Zeitskalenbereich kann entweder durch die Eigenschaften **TimeScaleStart** und **TimeScaleEnd** vorgegeben werden oder durch die Methode **OptimizeTimeScaleStartEnd(...)** des Objekts **VcGantt** aus den Daten ermittelt werden.

Code-Beispiel VB.NET

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

Code-Beispiel C#

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

Hier sind im Überblick noch einmal alle Programmzeilen zusammengefasst, die wir für unser Einstiegsbeispiel benötigen.

Code-Beispiel VB.NET

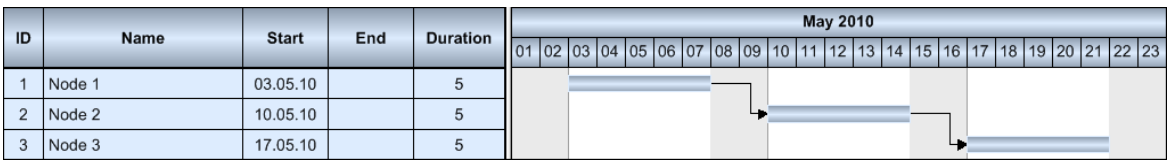
```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

   VcGantt1.Width = ClientSize.Width - VcGantt1.Left
   VcGantt1.Height = ClientSize.Height - VcGantt1.Top

   Dim dataTable As VcDataTable
   Dim dataRecCltn As VcDataRecordCollection

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

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

```
   VcGantt1.EndLoading()

   VcGantt1.OptimizeTimeScaleStartEnd(3)
End Sub

   Private Sub Form1_Resize(ByVal sender As Object, ByVal e As
System.EventArgs) Handles MyBase.Resize
   VcGantt1.Width = ClientSize.Width - VcGantt1.Left
   VcGantt1.Height = ClientSize.Height - VcGantt1.Top
End Sub
```

Code-Beispiel C#

```
private void Form1_Load(object sender, System.EventArgs e)
{

   vcGantt1.Width = ClientSize.Width - vcGantt1.Left;
   vcGantt1.Height = ClientSize.Height - vcGantt1.Top;

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

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

   vcGantt1.EndLoading();

   vcGantt1.OptimizeTimeScaleStartEnd(3);
}

private void Form1_Resize(object sender, System.EventArgs e)
{
   vcGantt1.Width = ClientSize.Width - vcGantt1.Left;
   vcGantt1.Height = ClientSize.Height - vcGantt1.Top;
}
```

Wenn Sie nun das Programm ausführen, sollte das Ergebnis der Abbildung entsprechen.



2.4 Endtermine berechnen

Die Tabellenspalte mit den Endterminen ist noch leer. Das Ende eines Vorgangs kann aus **Start** und **Dauer** mit Hilfe des in VARCHART XGantt enthaltenen Kalenders berechnet werden.

Der Kalender ist so voreingestellt, dass die Wochentage (Montag bis Freitag) als aktive Zeiten, die Wochenenden (Samstag und Sonntag) als nicht aktive Zeiten zählen.

Im Diagramm sind die nicht aktiven Zeiten an den grau hinterlegten Bereichen zu erkennen. Abgeschaltet werden kann der Kalender auf der Eigenschaftenseite **Knoten**, in dem die Option **Knoten verwenden Kalender** deaktiviert wird.

Der Unterschied in der Berechnung mit und ohne Kalender ist folgender:

Ein Vorgang, der am Freitag beginnt und 3 Tage dauert, wird bei aktiviertem Kalender am Dienstag enden. Ohne Kalender wird dieser Vorgang bereits am Sonntag enden. Die Berechnung der Endtermine wird über die Methode **AddDuration(...)** des Objektes **VcCalendar** ausgeführt. Dazu werden von jedem Vorgang **Start** und **Dauer** benötigt. Der Zugriff auf die entsprechenden Datenfelder erfolgt über den Index. Nachdem der Endtermin mit der Methode **set_DataField(..)** gesetzt wurde, muss noch die Methode **Update()** von **VcNode** aufgerufen werden, damit die Datenänderung in der Darstellung sichtbar wird.

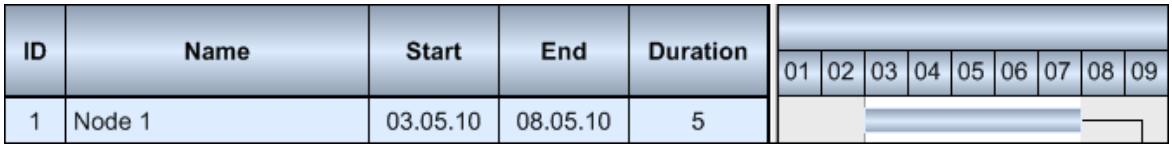
Code-Beispiel VB.NET

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

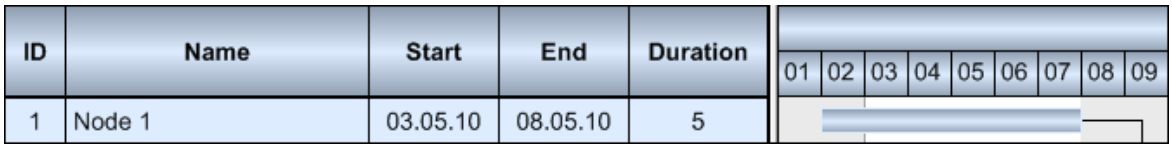
Code-Beispiel C#

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

Durch Mausinteraktionen festgelegte oder veränderte Vorgänge starten und enden immer in einer aktiven Zeit.



Die über die API oder über die Editierdialoge gesetzten Termine können jedoch auch in nicht aktiven Zeiten liegen.



Durch Berechnung ermittelte Termine liegen dagegen immer in einer aktiven Zeit. Wenn auch für die über die API eingefügten Vorgänge sichergestellt werden soll, dass alle Starttermine in einer aktiven Zeit liegen, dann muss der Starttermin aus dem Endtermin und der Dauer berechnet werden.

Code-Beispiel VB.NET

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

Code-Beispiel C#

```
tmpDate = tmpCal.AddDuration((DateTime)node.get_DataField(3),
                (-1) * Convert.ToInt32(node.get_DataField(4)));
node.set_DataField(2, tmpDate);
```

Zur Wahrung der Konsistenz sollten fehlende oder negative Dauern als unzulässige Werte betrachtet werden und auf 0 zurückgesetzt werden. Bei einem fehlendem Starttermin kann keine Berechnung des Endtermins erfolgen. Die notwendigen Codezeilen wurde in eine eigene Methode mit dem Namen **SetNodeEndDate(...)** zusammengefasst.

Code-Beispiel VB.NET

```
Private Sub SetNodeEndDate(ByVal node As VcNode)
   'Avoid empty duration or negative duration
   If node.DataField(4) = "" Or node.DataField(4) < 0 Then
       node.DataField(4) = "0"
   End If
   'Start date empty then end date should also be empty
   If node.DataField(2) = "31.12.1899 00:00:00" Then
       node.DataField(3) = ""
   Else
     'Precondition is property page nodes
     '"Assign calendar to nodes" must be true
     Dim tmpCal As VcCalendar
     tmpCal = VcGantt1.CalendarCollection.Active
     Dim tmpDate As DateTime
     tmpDate = tmpCal.AddDuration(node.DataField(2), node.DataField(4))
     node.DataField(3) = tmpDate
```

```
     'Start date only in active times
     tmpDate = tmpCal.AddDuration(node.DataField(3),
                           (-1) * node.DataField(4))
      node.DataField(2) = tmpDate
      node.Update()
    End If
End Sub
```

Code-Beispiel C#

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

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

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

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

Die Terminberechnung muss in folgenden Situationen ausgeführt werden:

1. Nach dem Laden der Vorgänge
2. Nach Ändern der Termine oder Dauern über den Dateneditierdialog oder Inplace-Editor
3. Nach dem Verändern der Datenwerte eines Vorgangs per API

Die Terminberechnung muss jedoch nicht bei Vorgangsänderungen über Mausinteraktionen angestoßen werden, denn in diesem Fall erfolgt intern eine automatische Berechnung.

Über die Eigenschaft **NodeCollection** von **VcGantt** kann eine Berechnungsschleife über alle Knoten gebildet werden. Dieser Code wird

nach dem Laden der Vorgänge am Ende des Ereignisses **Form1_Load(...)** hinzugefügt.

Code-Beispiel VB.NET

```
'Enddatum aller Knoten berechnen
Dim node As VcNode
For Each node In VcGantt1.NodeCollection
   SetNodeEndDate node
Next
```

Code-Beispiel C#

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

Durch den Benutzer verursachte Änderungen der Daten können über das Ereignis **VcNodeModified** erfasst werden. Der eingefügte Methodenaufruf führt die Berechnung des Endtermins aus.

Code-Beispiel VB.NET

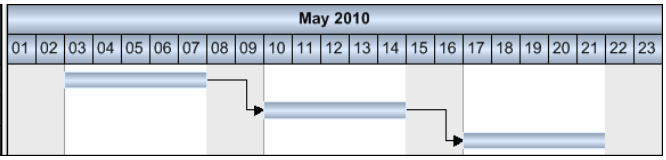
```
Private Sub VcGantt1_VcNodeModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifiedEventArgs) Handles VcGantt1.VcNodeModified
   SetNodeEndDate(e.Node)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeModified(object sender,
NETRONIC.XGantt.VcNodeModifiedEventArgs e)
   {
   SetNodeEndDate(e.Node);
   }
```

Bei Datenwertänderungen über die API muss explizit die Methode **SetNode-EndDate(...)** aufgerufen werden.

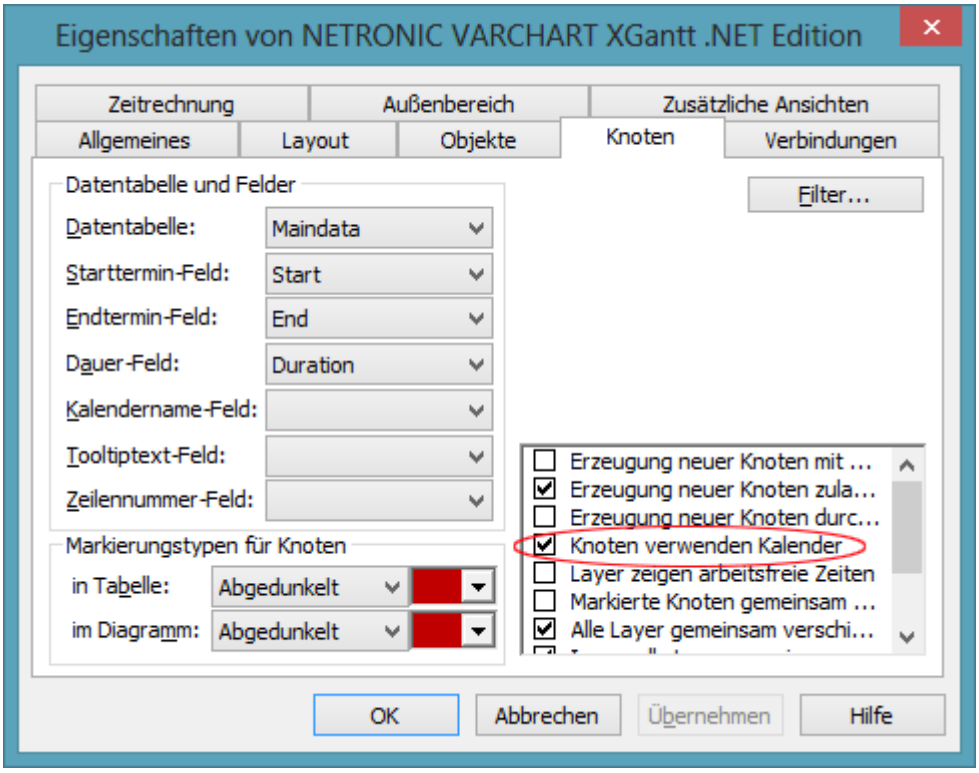
ID	Name	Start	End	Duration
1	Node 1	03.05.10	08.05.10	5
2	Node 2	10.05.10	15.05.10	5
3	Node 3	17.05.10	22.05.10	5



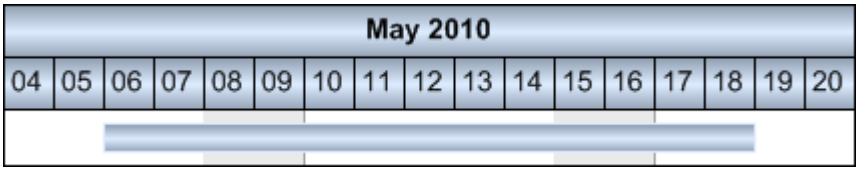
2.5 Arbeitsfreie Zeiten in den Vorgängen kennzeichnen

Die Unterbrechung der Vorgänge durch arbeitsfreie Zeiten kann mit der Darstellungsoption **Layer zeigen arbeitsfreie Zeiten** sichtbar gemacht werden. Die Option wird nur wirksam, wenn die Vorgänge von einem Kalender abhängen. Dies wird durch die Option **Knoten verwenden Kalender** bestimmt.

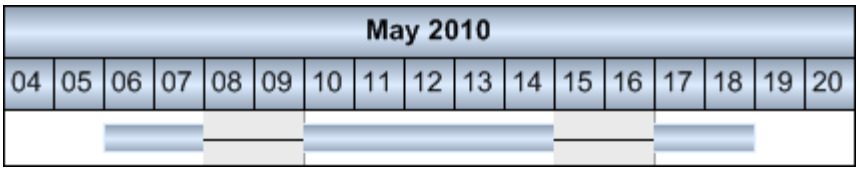
Die Einstellung kann zur Designzeit auf den Eigenschaftenseiten **Knoten** oder zur Laufzeit erfolgen.



Zur Laufzeit erfolgt die Setzung über die Eigenschaft **LayersWithNon-WorkInterval** des Objekts VcGantt.



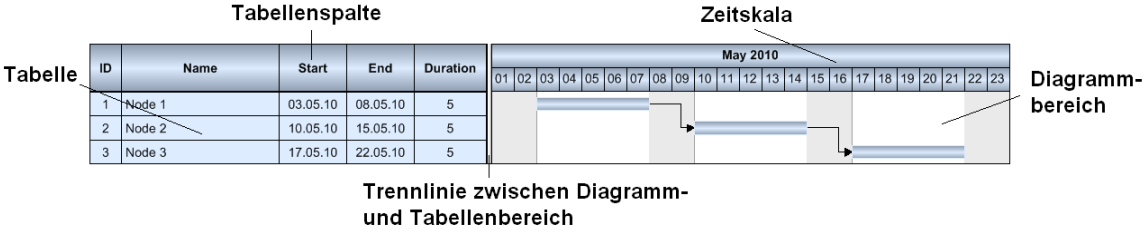
LayersWithNonWorkInterval = false



LayersWithNonWorkInterval = true

2.6 Interaktionen im Tabellen- und Diagrammbereich

In diesem und dem nachfolgenden Unterkapitel werden Sie in einem Kurzüberblick die wichtigsten Interaktionen kennenlernen. Weiterführende Informationen finden Sie in Kapitel 5, **Benutzerschnittstelle**.



> Breitenverhältnis von Tabellen- zu Diagrammbereich verändern

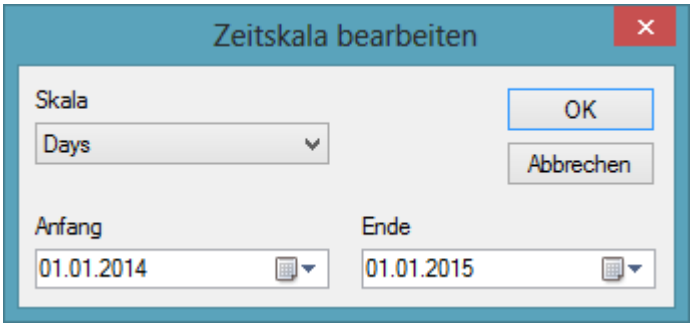
Die Aufteilung zwischen Tabellen- und Diagrammbereich lässt sich durch Verschieben der grauen, vertikalen Trennlinie verändern. Auf den Eigenschaftenseiten unter **Layout** lässt sich das Verhältnis über **Breitenverhältnis Tabelle/Diagramm** für den Programmstart voreinstellen.

> Breite der Tabellenspalten ändern

Durch Ziehen der rechten vertikalen Trennlinie im Bereich der Tabellenüberschriften lässt sich die Breite einer Spalte verändern. Mit einem Doppelklick auf diese Linie wird die Breite der Spalte automatisch an die Länge des darin enthaltenen Textes angepasst. Die automatische Spaltenbreitenanpassung kann auf der Eigenschaftenseite **Allgemeines** unter **Tabellenspalten optimierbar** aktiviert bzw. deaktiviert werden.

> Anfang und Ende der Zeitskala festlegen

Durch einen Doppelklick auf die Zeitskala öffnet sich das Dialogfeld zum Verändern von Anfang und Ende der Zeitskala. Auf der Eigenschaftenseite **Allgemeines** unter **Zeitskalen-Dialog anzeigen** kann dieses Verhalten aktiviert bzw. deaktiviert werden.



> Zeitskala skalieren

Durch Ziehen im Zeitskalenbereich nach rechts oder links lässt sich die Einheitenbreite der jeweiligen Zeitskala vergrößern bzw. verkleinern. Auf der Eigenschaftenseite **Allgemeines** unter **Zeitskala skalierbar** kann dieses Verhalten aktiviert bzw. deaktiviert werden.

2.7 Interaktionen mit Vorgängen

> Neuen Vorgang anlegen

Um einen neuen Vorgang anzulegen, wechseln Sie zunächst über das Kontextmenü des Gantt-Graphen (rechte Maustaste) in den Modus **Knoten erzeugen**. Der Mauszeiger nimmt die Form eines kleinen Kreuzes an. Ziehen Sie nun mit gedrückter linker Maustaste an der gewünschten Position einen neuen Vorgang auf. Schalten Sie nach dem Anlegen des Vorgangs wieder über das Kontextmenü zurück in den **Selektier-Modus**.

In den Erzeugemodus kann vom Programm über das Ereignis **VcNodeCreated()** eingegriffen werden. Dies ist zum Beispiel sinnvoll, wenn Datenwerte vorzubesetzen sind.

Code-Beispiel VB.NET

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

Code-Beispiel C#

```
private void vcGantt1_VcNodeCreated(object sender,
VcNodeCreatedEventArgs e)
   {
   e.Node.set_DataField(1, "Node " + e.Node.get_DataField(0));
   e.Node.Update();
   }
```

Der hier gezeigte Code verändert den Inhalt des Datenfeldes Name; er hängt an den Begriff Node den aktuellen Inhalt des Feldes ID an.

> Dauer eines Vorgangs ändern

Fahren Sie im Selektiermodus mit dem Mauszeiger von innen an den rechten oder linken Rand des Vorgangs heran. Der Mauszeiger ändert seine Form zu einer senkrechten Linie mit Pfeil. Durch Ziehen nach links oder rechts verlängern bzw. verkürzen Sie den Vorgang.

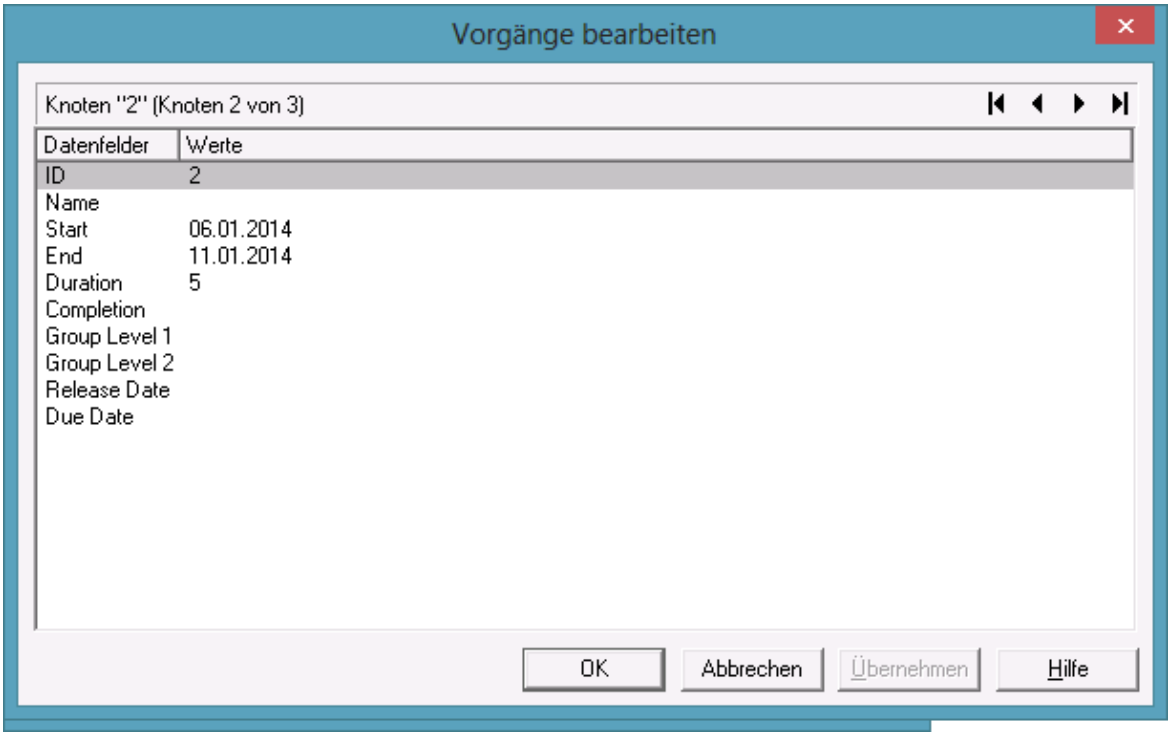
> Vorgang verschieben

Zum Verschieben eines Vorganges fahren Sie im Selektiermodus mit dem Mauszeiger auf den Vorgang. Der Mauszeiger wird zu einem kleinen Quadrat mit vier Pfeilen. Nun lässt sich der Vorgang an eine beliebige Position verschieben. Mehrere Vorgänge können gleichzeitig verschoben werden,

wenn auf der Eigenschaftenseite **Knoten** die Option **Markierte Knoten gemeinsam verschieben** aktiviert ist.

> Daten eines Vorgangs bearbeiten

Durch einen Doppelklick auf einen Vorgang oder auf den zugehörigen Tabelleneintrag gelangen Sie in das Dialogfeld **Vorgänge bearbeiten**.



> Vorgang löschen

Mit der **Entf**-Taste oder über **Knoten löschen** im Kontextmenü des Vorgangs können markierte Vorgänge gelöscht werden.

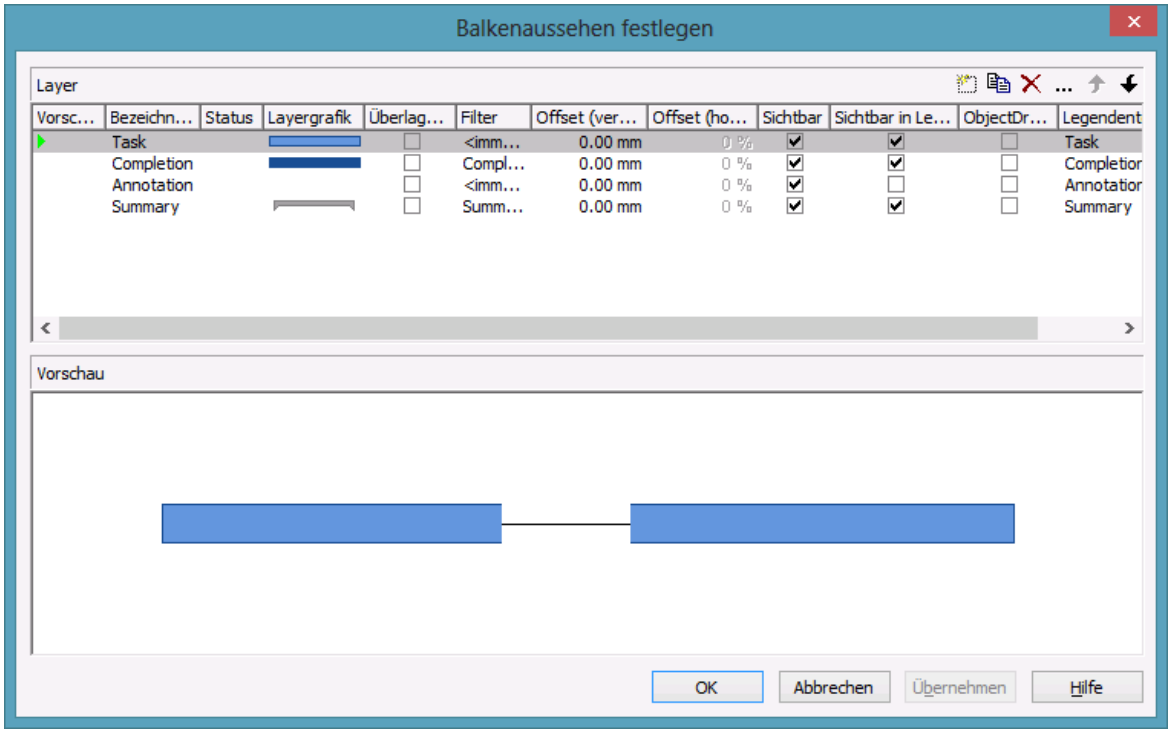
Ein Vorgang wird markiert, in dem Sie ihn direkt oder die entsprechende Tabellenzeile anklicken. Bei gedrückter Strg-Taste wird der Vorgang zu der bestehenden Auswahl hinzugefügt.

2.8 Layer verwenden

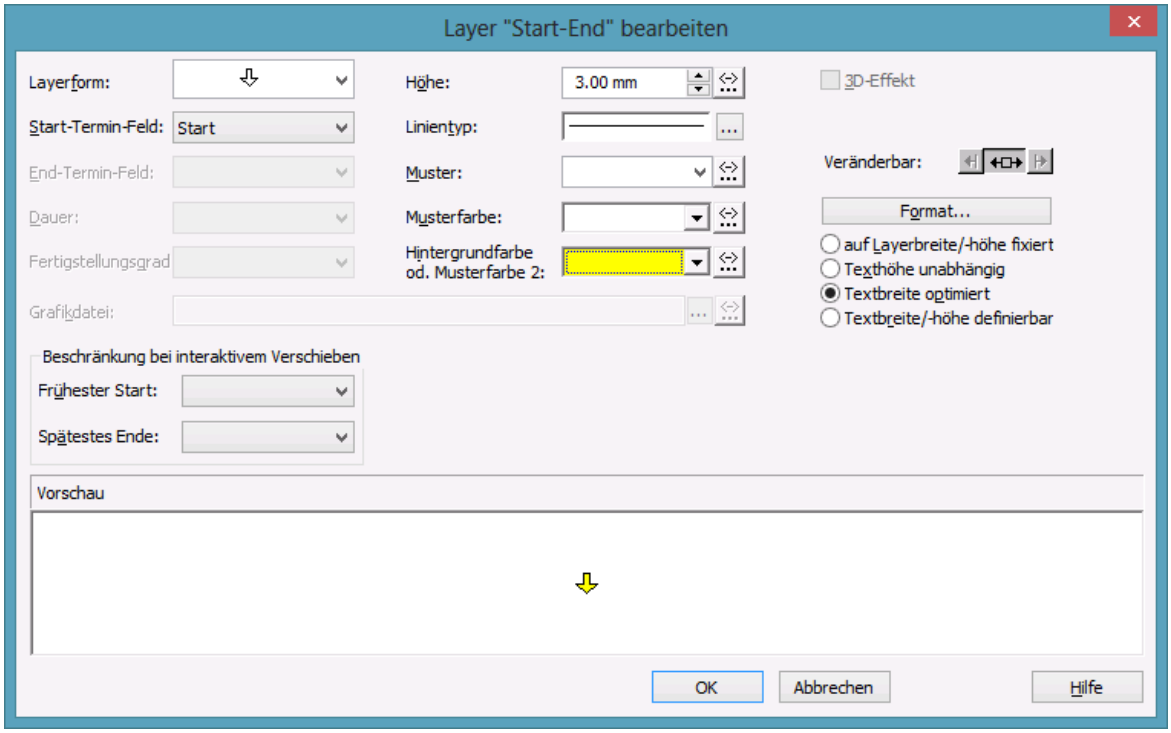
Ein Layer beschreibt die grafische Umsetzung eines Terminpaares. Ein Terminpaar kann zudem auch durch mehrere Layer abgebildet werden, die im logischen Modell schichtweise der Reihe nach übereinander liegen.

Wir möchten in unserem Beispiel einen zweiten Layer mit anderem Aussehen vereinbaren.

1. Wählen Sie **Layer...** auf der Eigenschaftenseite **Objekte**. Sie gelangen in den Dialog **Balkenaussehen festlegen**. Der Layer mit dem Namen **Start-End** ist bereits vordefiniert.

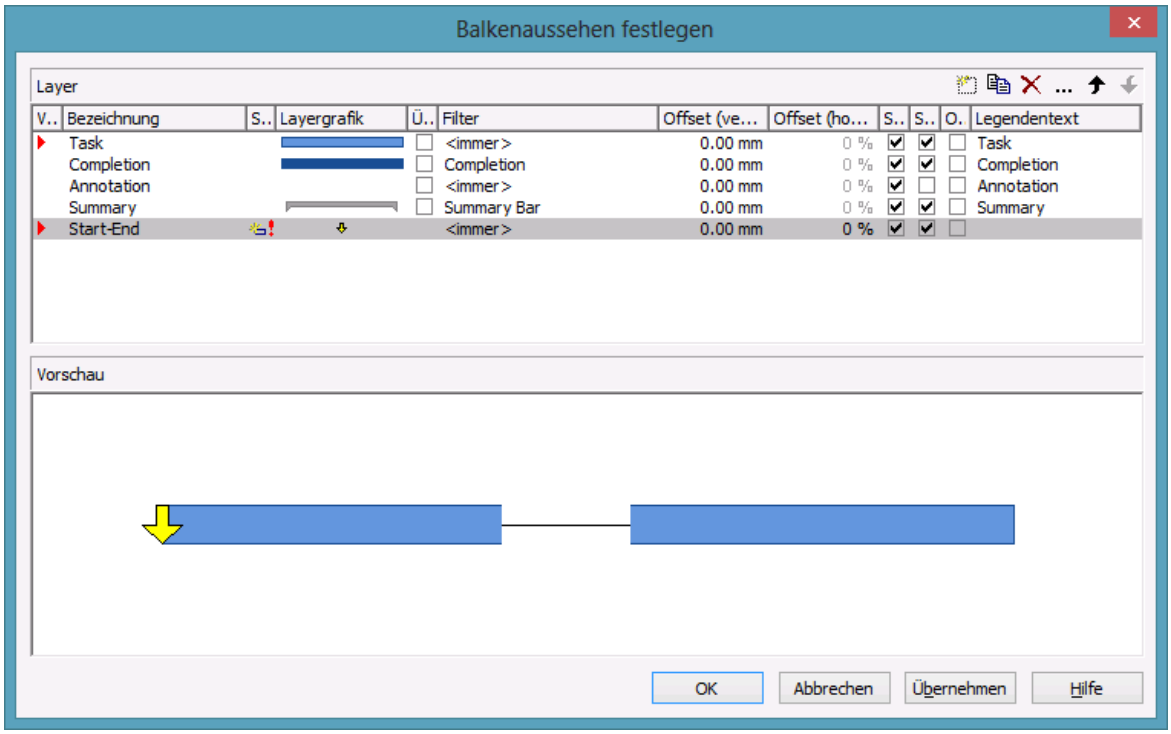


2. Duplizieren Sie die Layerdefinition "Task" mit Hilfe der Schaltfläche .
3. Bearbeiten Sie nun den neuen Layer **NeuerLayer**. Klicken Sie dazu auf die Schaltfläche ...
4. Ändern Sie die die **Form** auf Pfeilspitze nach unten und die **Hintergrundfarbe** auf gelb.



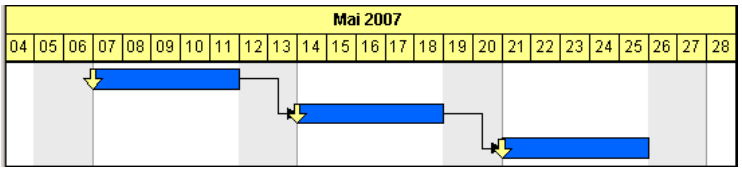
5. Mit der Schaltfläche OK gelangen Sie zurück in den Dialog **Balkenaussehen festlegen**.

6. Jeder Layer eines Knotens wird in der Vorschau unten angezeigt, wenn Sie in die Spalte "Vorschau" des entsprechendes Feldes klicken. Statt des grünen Dreiecks erscheint ein rotes, woran zu erkennen ist, dass der Layer in der Vorschau angezeigt wird..



7. Die geänderte Definition führt im Beispielprogramm zu folgendem Ergebnis:

ID	Name	Start	Ende	Dauer
1	Node 1	07.05.07	12.05.07	5
2	Node 2	14.05.07	19.05.07	5
3	Node 3	21.05.07	26.05.07	5



2.9 Filter verwenden

Als Nächstes möchten wir erreichen, dass der rote Pfeil nur dann erscheint, wenn ein Meilenstein vorliegt, das heist, die Dauer des Vorgangs ist gleich 0.

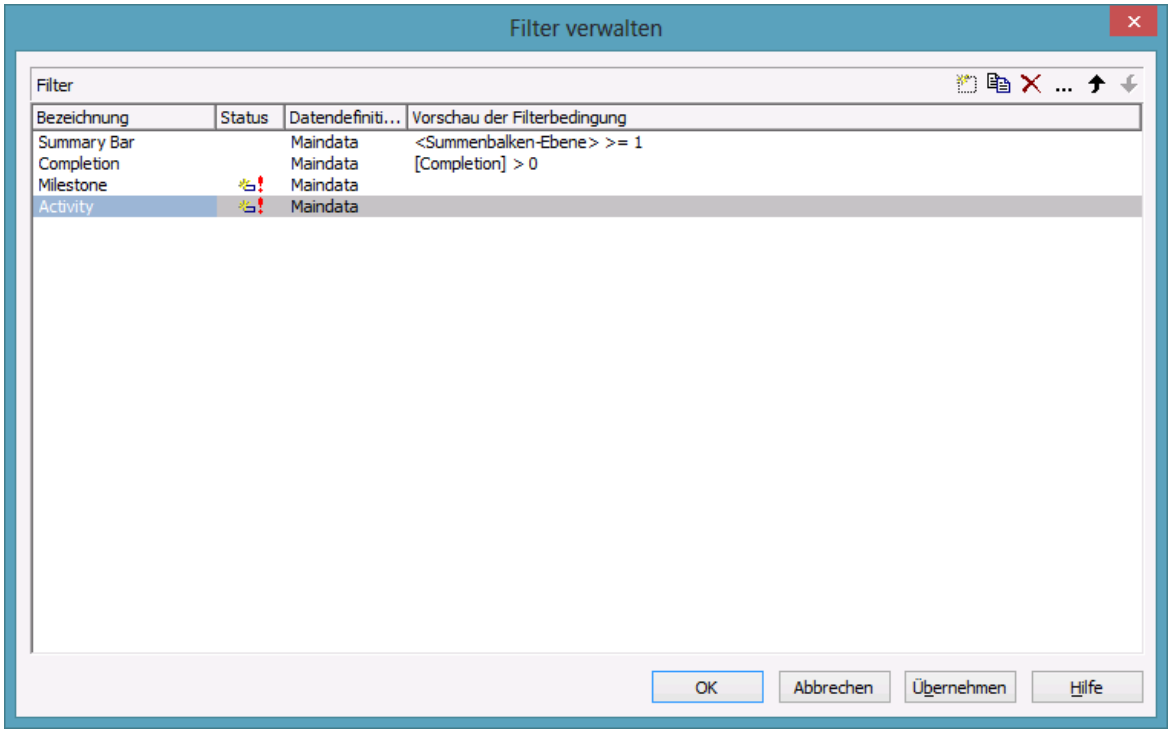
Mit Hilfe von Filtern können wir dieses Problem sehr einfach lösen. Ein Filter besteht aus einer Folge von verknüpften Bedingungen, die logisch zu einer Ja/Nein-Aussage führen.

Layer sind immer an Filter gebunden. Der jeweilige Layer wird nur sichtbar, wenn die Auswertung der Filterbedingung zu einem Ja-Wert führt. Der eingebaute Filter <immer>, der bereits jedem Layer standardmäßig zugewiesen ist, liefert immer einen Ja-Wert zurück.

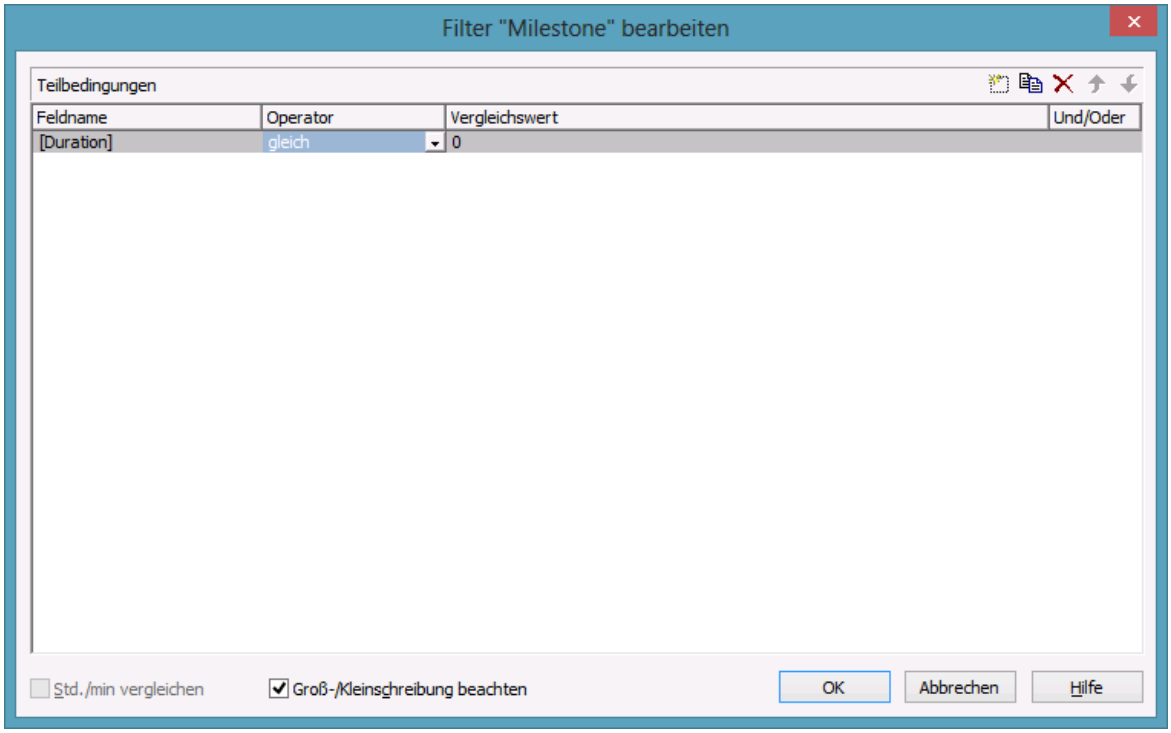
Für unser Beispiel benötigen wir zwei Filter mit je einer Bedingung:

- Der rote Pfeil soll erscheinen, wenn die Dauer = 0 ist.
- Der blaue Balken soll erscheinen, wenn die Dauer > 0 ist.

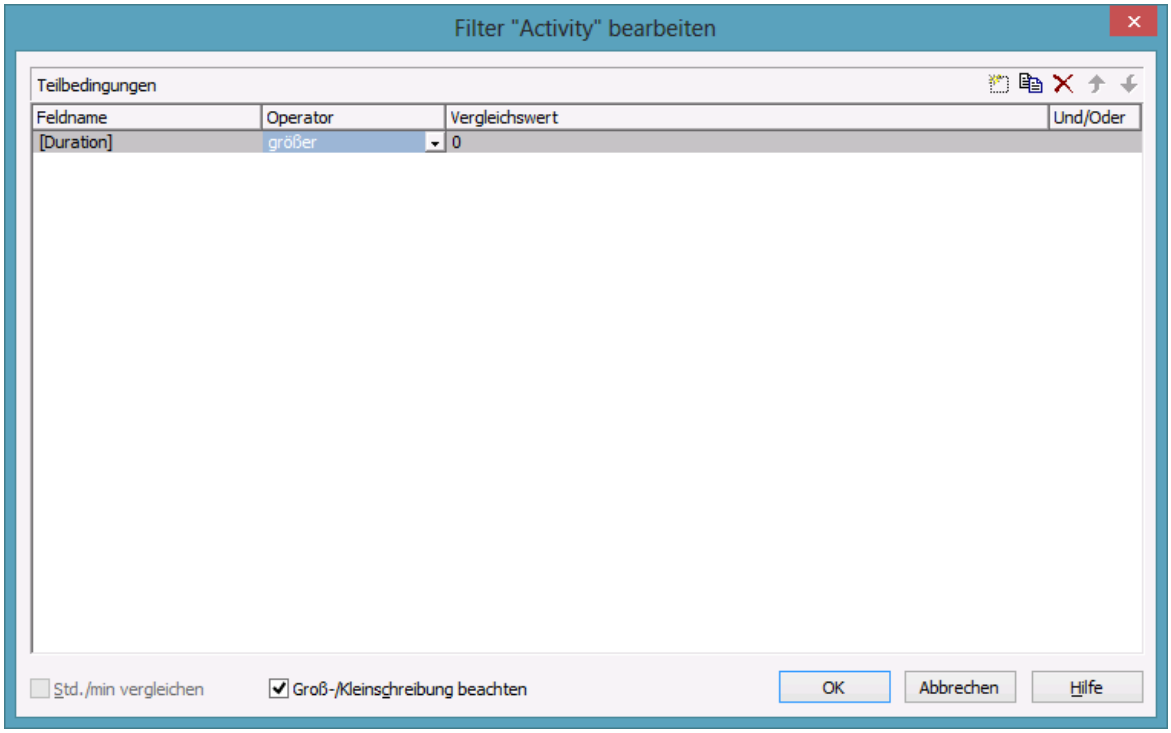
1. Auf der Eigenschaftenseite **Objekte** gelangen Sie über **Filter...** in den Dialog **Filter verwalten**.
2. Legen Sie über die Schaltfläche  zwei neue Filter an.
3. In der Spalte **Name** benennen Sie "NeuerFilter" und "NeuerFilter1" in "Milestone" und "Activity" um.
4. Bestätigen Sie die Änderungen mit **Übernehmen**.



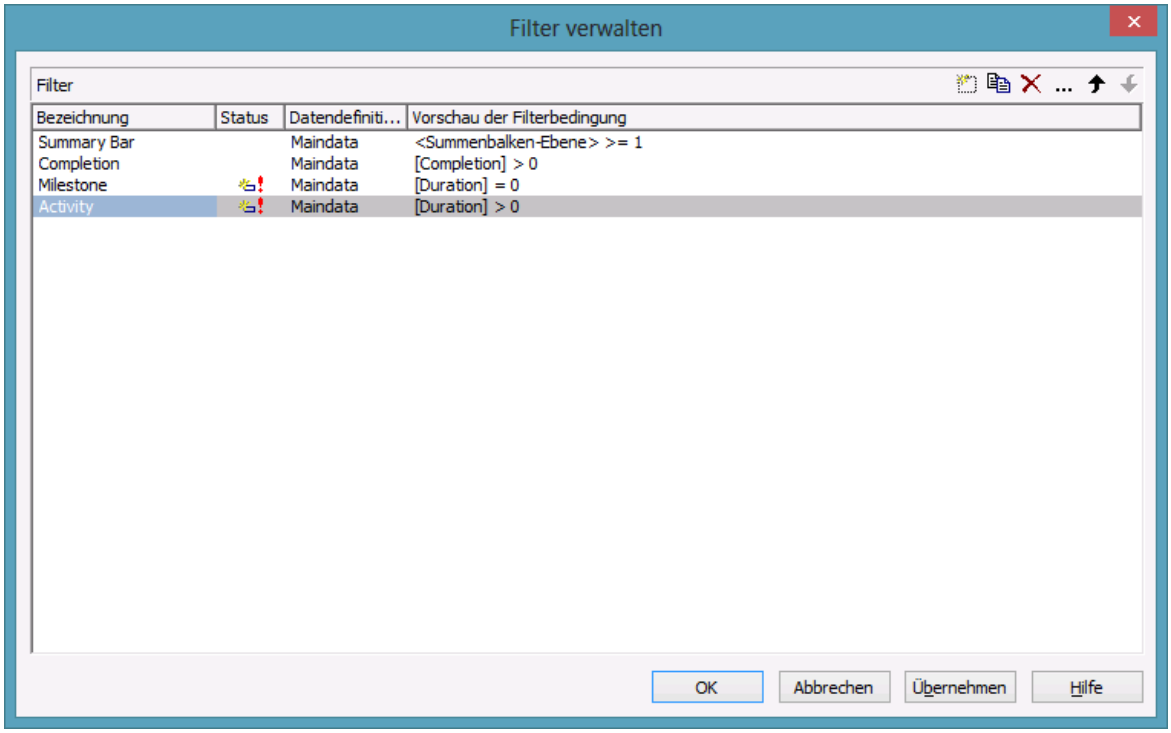
5. Wählen Sie den Filter "Milestone" aus und verzweigen durch Klick auf die Schaltfläche ... in den Dialog **Filter bearbeiten**.
6. Wählen Sie als **Feldname** "Dauer" aus. Für den **Operator** setzen Sie gleich und für **Vergleichswert** 0 ein.



7. Mit **OK** verlassen Sie den Dialog.
8. Wählen Sie Activity aus und verzweigen Sie erneut durch die Schaltfläche ... in den **Filter bearbeiten** Dialog.
9. Wählen Sie als **Feldnamen** "Dauer" aus. Für den **Operator** setzen Sie "grösser als" und für **Vergleichswert** 0 ein.

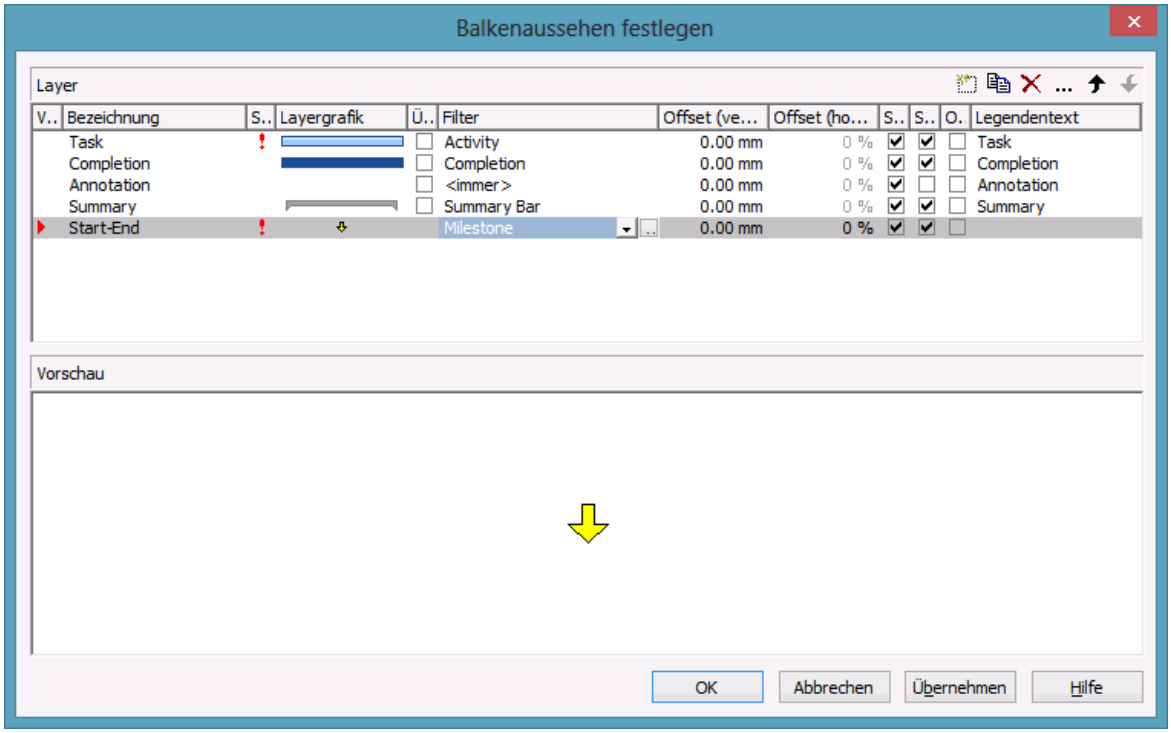


10. Mit **OK** verlassen Sie den Dialog.

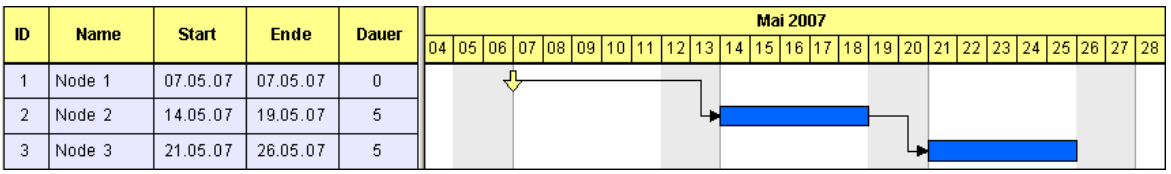


11. Klicken Sie erneut auf **OK** und kehren zu den Eigenschaftenseiten zurück.

12. Damit die Filter wirksam werden, weisen Sie nun bitte die Filter den Layern zu. Bitte rufen Sie über die Schaltfläche **Layer…** den Dialog **Balkenaussehen festlegen** auf.



13. Wenn jetzt das Programm ausgeführt und die Dauer des ersten Vorgangs auf 0 gesetzt wird, dann ergibt sich folgendes Bild:



2.10 Histogramme erstellen

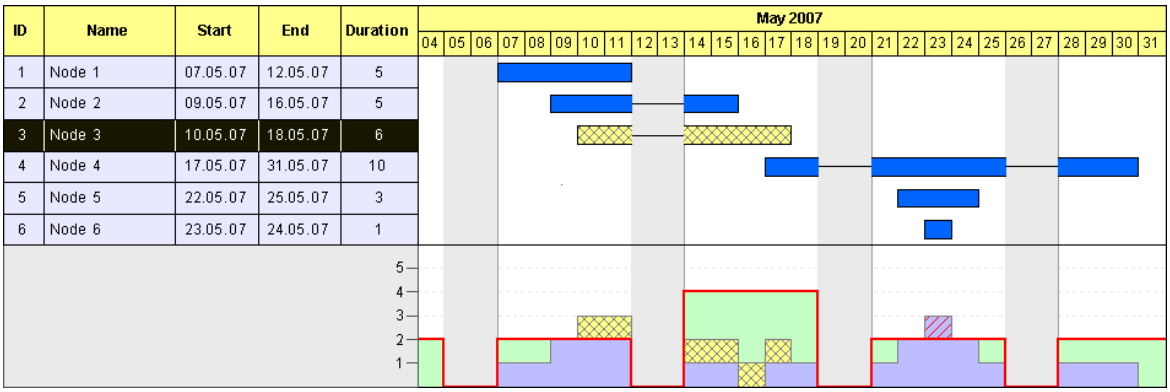
In diesem Beispiel führen wir vor, wie Histogramme programmiert werden können.

Histogramme summieren Vorgänge auf und bilden das summierte Ergebnis in der Form einer Fläche ab. Die Obergrenze dieser Fläche ist als Auslastungskurve definiert (z.B. einer Maschine, einer Fertigungsanlage oder eines Arbeitsteams – wir verwenden stellvertretend den Ausdruck "Produktionssystem").

Die Auslastungskurve kann in Bezug gesetzt werden zur Kapazitätskurve, d.h. einer Kurve, die die maximalen Auslastungswerte des Produktionssystems darstellt. Während sich die Werte der Auslastungskurve aus den Vorgängen ergeben, müssen die Werte der Kapazitätskurve mit vorgegebenen Werten definiert werden.

Es kann beim Abgleich der beiden Kurven zu Unter- und Überschreitungen der Kapazitätskurve durch die Auslastungskurve kommen. Auf diese Weise werden Über- bzw. Unterlasten des Produktionssystems sichtbar. Bereiche, in denen die Kapazitätskurve oberhalb der Auslastungskurve liegt, sind Unterlastbereiche; liegt die Auslastungskurve oberhalb der Kapazitätskurve, ist das System überlastet. Verlaufen beiden Kurven auf gleicher Höhe, ist die Auslastung optimal.

Unter- und Überlastflächen können unterschiedlich dargestellt werden, um Unter- oder Überlasten des Produktionssystems optisch deutlich zu machen.



In der Abbildung oben sehen Sie Folgendes: Das Histogramm bildet summarisch die Vorgänge aus dem Gantt-Graphen in blau-violetten, unschraffierten Flächen ab. Markierte Knoten erscheinen sowohl im Gantt-Graphen als auch im Histogramm gelb und mit Kreuzschraffur (Knoten 3). Die Kapazitätskurve ist als dicke rote Linie dargestellt. Überlasten werden in blau-violett mit einfacher Schraffur dargestellt (Knoten 6). Unterlasten erscheinen als hellgrüne Flächen.

Wir werden dieses Histogramm Schritt für Schritt erstellen. Das fertige Programm finden Sie im Verzeichnis **UserGuideSamples \VB.NET \XGantt_Tutorial02**

und

UserGuideSamples \Csharp \XGantt_Tutorial02.

Grundsätzlich wird in VARCHART XGantt im Histogramm die Darstellung von Flächen mit bestimmten Mustern und Schraffuren durch die Zuweisung einer Bezugskurve zu einer Kurve ermöglicht, wodurch die Fläche entsteht, die dann mit Farben und Mustern gefüllt werden kann.

Wir werden für die Programmierung folgende Schritte vornehmen:

Schritt 1: Die Darstellung eines Histogramms im Gantt-Diagramm wird eingeschaltet.

Schritt 2: Markierte Vorgänge sollen in der Tabelle invertiert erscheinen, im Gantt-Graphen und Histogramm aber schraffiert. Dazu wird als erster Teilschritt die Markierbarkeit der Vorgänge im Gantt-Graphen abgeschaltet.

Schritt 3: Um selektierte und unselektierte Vorgänge unterscheiden zu können, wird für die Vorgänge ein Datenfeld "Selected" eingerichtet, in dem der Selektionszustand festgehalten wird.

Schritt 4: Das Datenfeld wird mit einem Wert gefüllt, der den Markierungszustand wiedergibt.

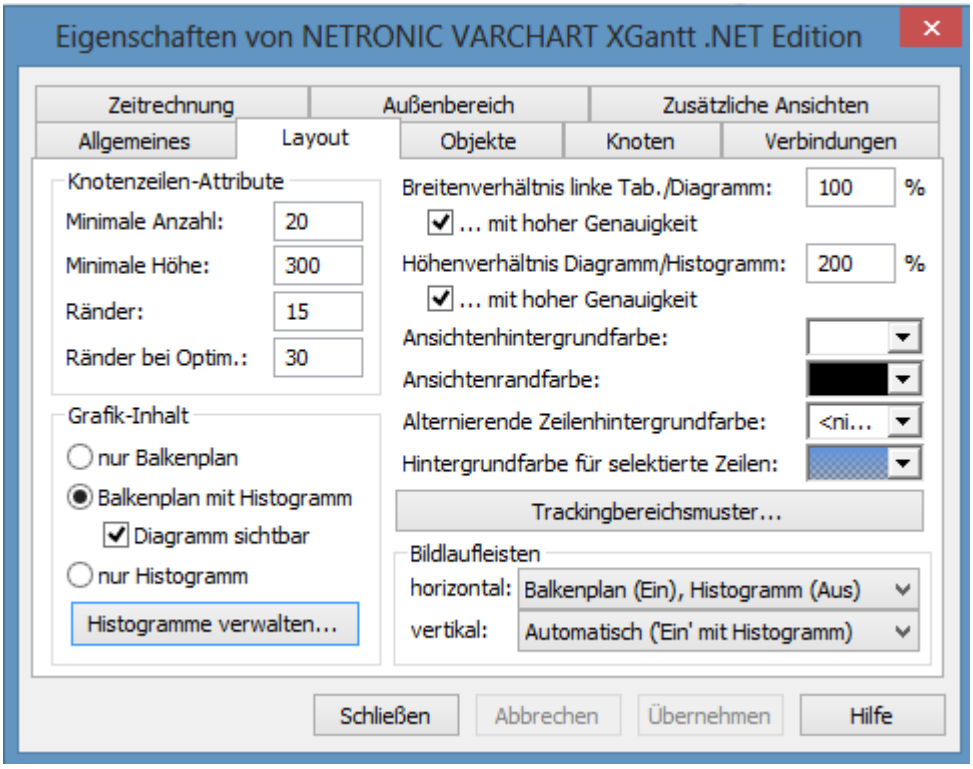
Schritt 5: Es werden zwei Filter eingerichtet, die selektierte bzw. unselektierte Vorgänge auswählen.

Schritt 6: Das Erscheinungsbild selektierter und unselektierter Knoten wird in Abhängigkeit dieser beiden Filter definiert.

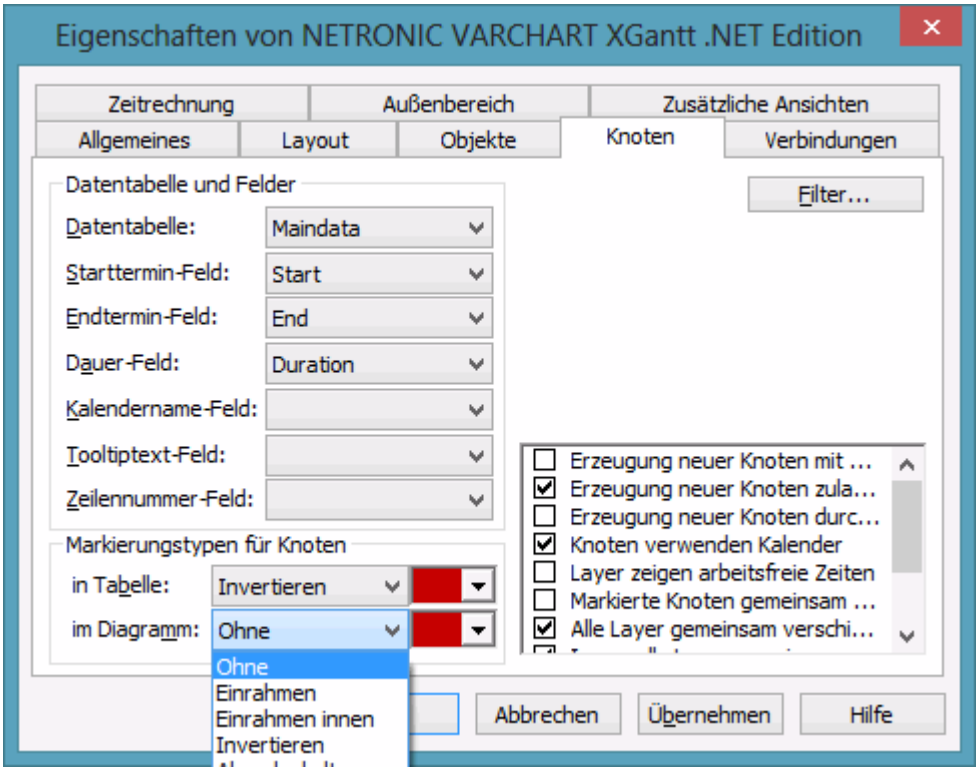
Schritt 7: Vier Kurven werden für das Histogramm angelegt: die Kapazitätskurve, die Kurve aus unmarkierten Vorgängen, die Kurve aus markierten Vorgängen und eine Hilfskurve zur Flächenfüllung. Den dazwischenliegenden Flächen werden Muster und Farben zugewiesen.

Schritt 8: Abschließend werden die Werte der Kapazitätskurve definiert.

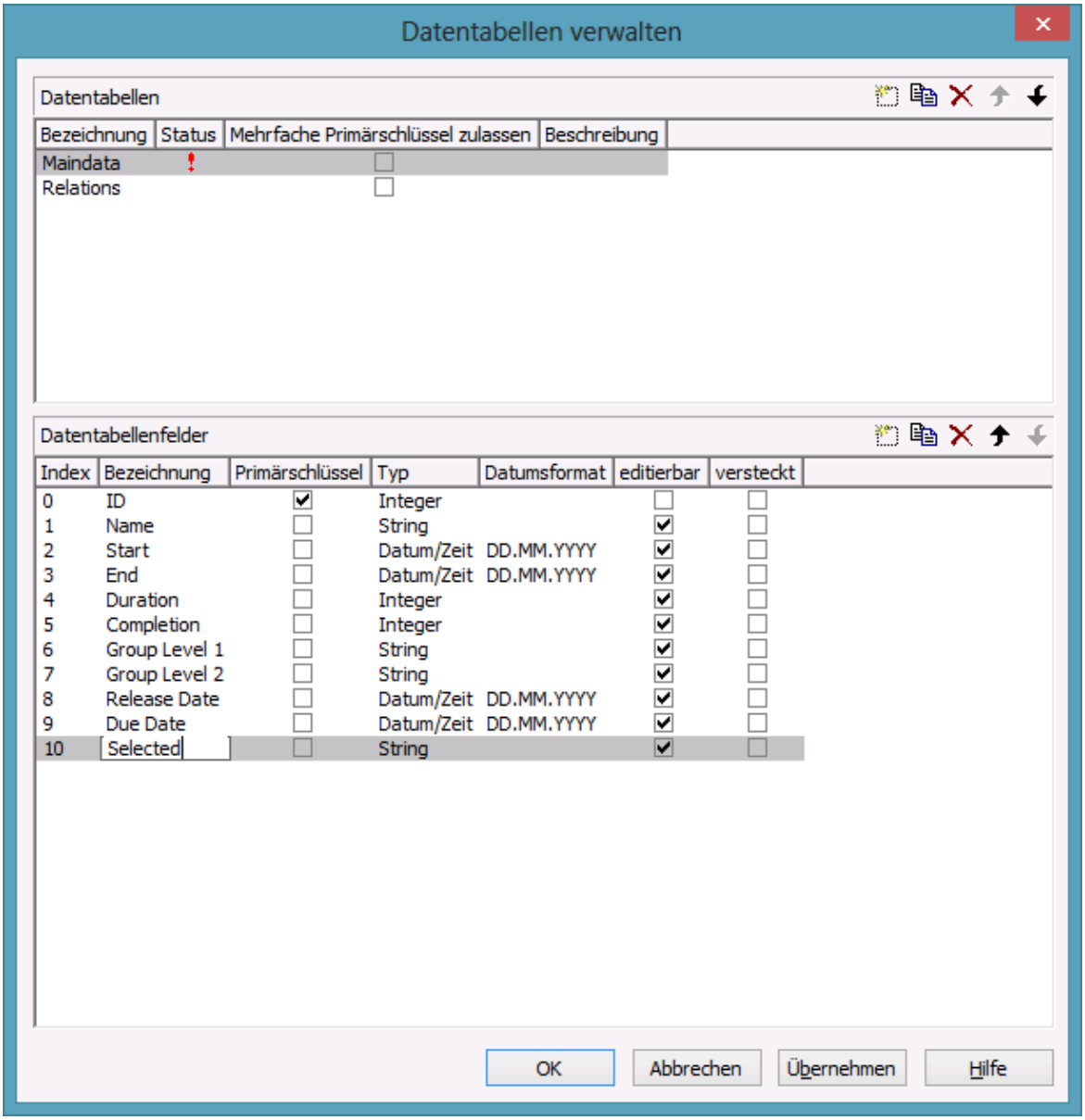
Schritt 1: Zunächst schalten Sie die Darstellung des Histogramms im Gantt Diagramm ein. Dies erfolgt auf der Eigenschaftenseite **Layout** im Bereich **Grafik-Inhalt**, wo Sie die Option **Balkenplan mit Histogramm** einschalten.



Schritt 2: Da markierte Knoten durch eine eigene Kreuzschraffur gekennzeichnet werden sollen, wird die Markierbarkeit von Knoten im Gantt-Graphen abgeschaltet. Dafür rufen Sie bitte die Eigenschaftenseite **Knoten** und setzen im Bereich **Markierungstypen für Knoten** das Feld **im Diagramm** auf **Ohne**.



Schritt 3: Um selektierte und unselektierte Vorgänge unterscheiden zu können, wird für die Vorgänge ein Datenfeld "Selected" eingerichtet, in dem der Markierungszustand festgehalten wird. Bitte wechseln Sie über die Eigenschaftenseite **Datentabellen** in den Dialog **Datentabellen verwalten**, wählen dort die Tabelle **Maindata** zum Bearbeiten aus und fügen dieser ein Feld mit dem Namen "Selected" vom Typ **Integer** hinzu. Dieses Feld wird benötigt, um die Darstellung des Vorgangs davon abhängig zu machen, ob er selektiert ist oder nicht.



Schritt 4: Das Datenfeld "Selected" wird aktualisiert, sobald das Ereignis **VcNodesMarked** ausgelöst wird.

Code-Beispiel VB.NET

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

Code-Beispiel C#

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

Im Ereignis **VcNodeCreated** verhindert der folgende Programmcode, dass ein neu erzeugter Knoten direkt markiert ist. Da gleichzeitig beim Erzeugen eines neuen Knotens die Markierung aller vorher ausgewählten Knoten aufgehoben wird, muss der Feldinhalt von "Selected" aktualisiert werden.

Code-Beispiel VB.NET

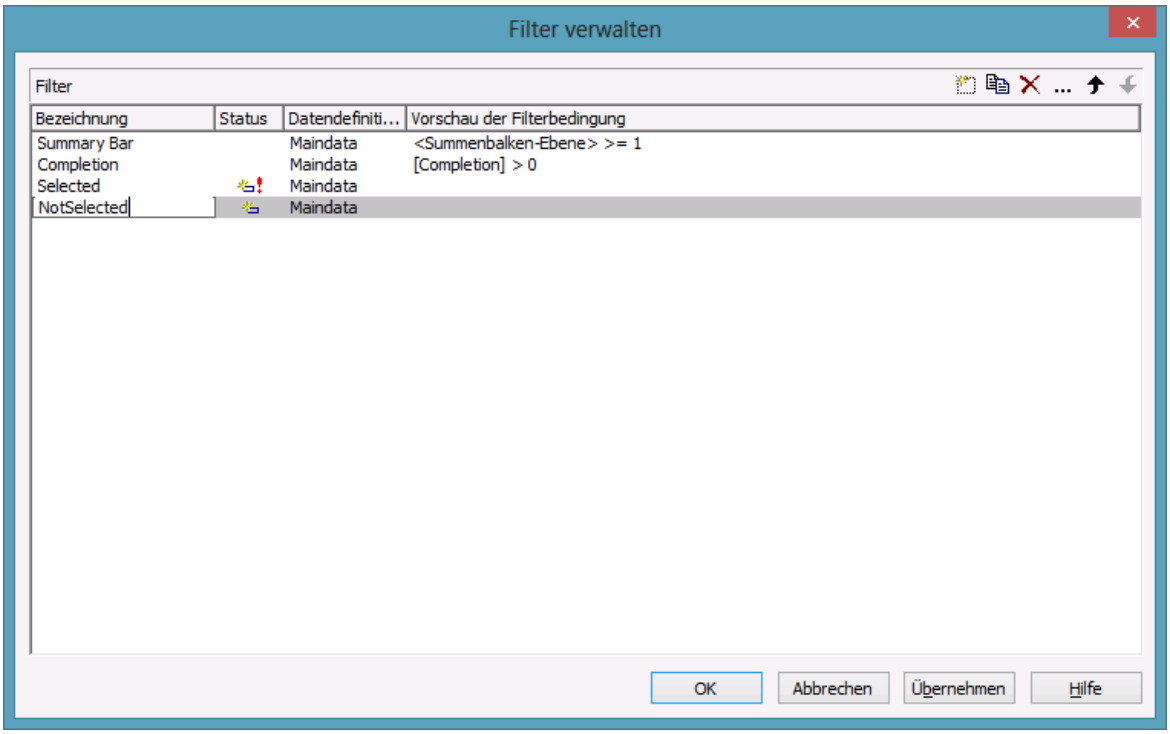
```
Private Sub VcGantt1_VcNodeCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeCreatedEventArgs) Handles VcGantt1.VcNodeCreated
 e.Node.DataField(1) = "Node " + e.Node.DataField(0)
e.Node.Marked = False
e.Node.Update()

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

Code-Beispiel C#

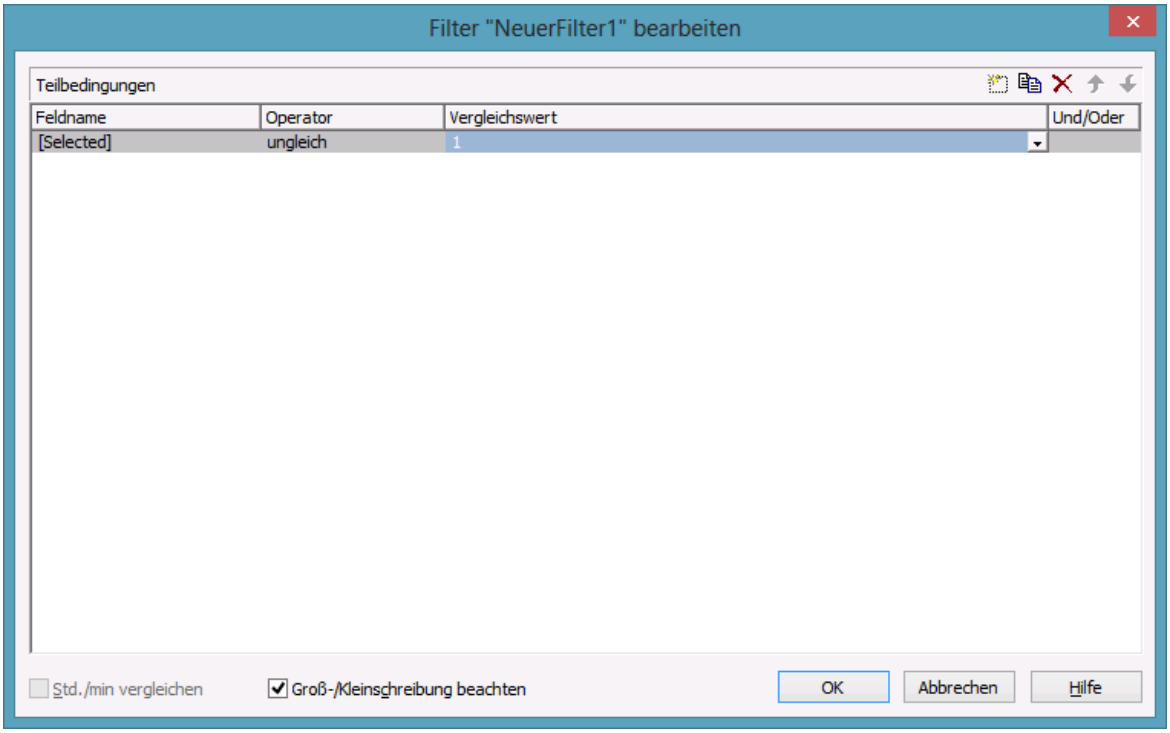
```
private void vcGantt1_VcNodeCreated(object sender, NETRONIC.XGantt.
VcNodeCreatedEventArgs e)
  {
  e.Node.set_DataField(1, "Node " + e.Node.get_DataField(0));
  e.Node.Marked = false;
  e.Node.Update();
```

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

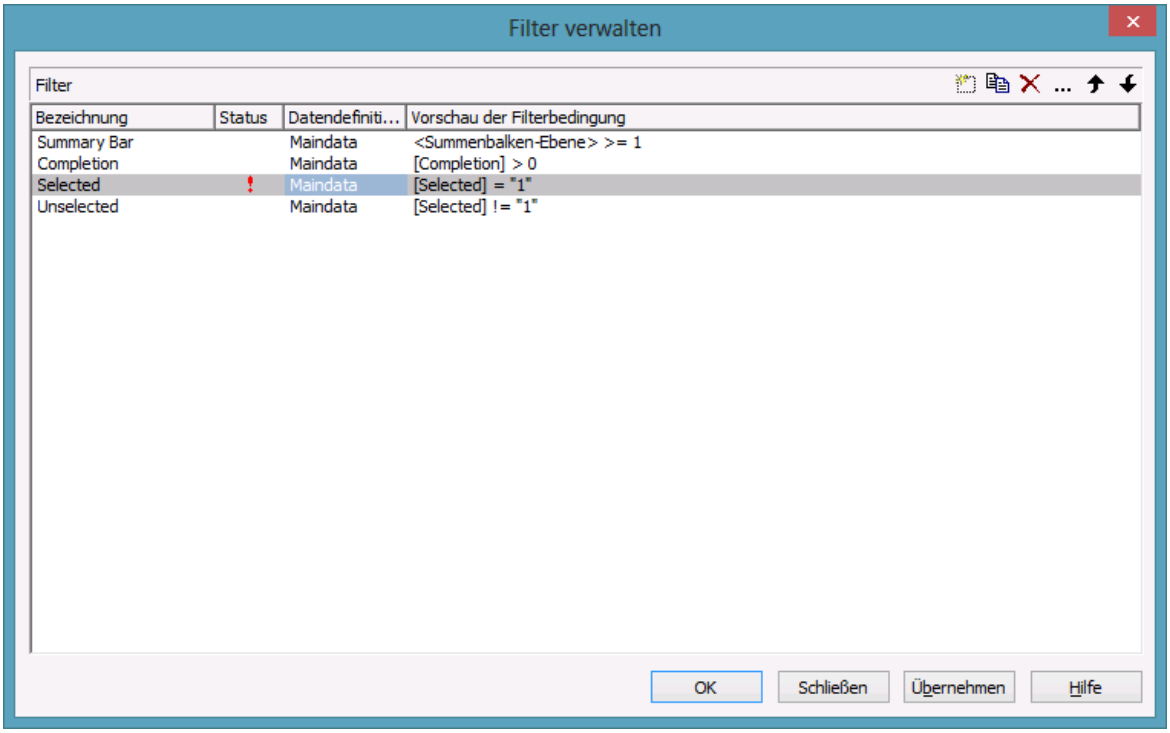
Schritt 5: In diesem Schritt werden zwei Filter eingerichtet, die selektierte bzw. unselektierte Vorgänge auswählen. Auf der Eigenschaftenseite **Objekte** gelangen Sie über die Schaltfläche **Filter...** in den Dialog **Filter verwalten**. Dort legen Sie über die Schaltfläche  zwei neue Filter an und nennen Sie "Selected" und "Not Selected".



Nun definieren Sie die Auswahlkriterien. Dem Filter mit dem Namen "Not Selected" weisen Sie die Bedingung zu, dass das Knotenfeld "Selected" der Maindata-Tabelle ungleich 1 sei. Dadurch werden nur unselektierte Knoten gefiltert. Markieren Sie dazu den Filter **Not Selected** und klicken auf die Schaltfläche  oben rechts. Sie gelangen in den Dialog zum Bearbeiten dieses Filters. Wählen Sie in der Spalte **Feldname** das Feld **Selected** aus, in der Spalte **Operator** den Operator **ungleich** und setzen Sie in der Spalte **Vergleichswert** den Wert 1. Verlassen Sie den Dialog über **OK**.

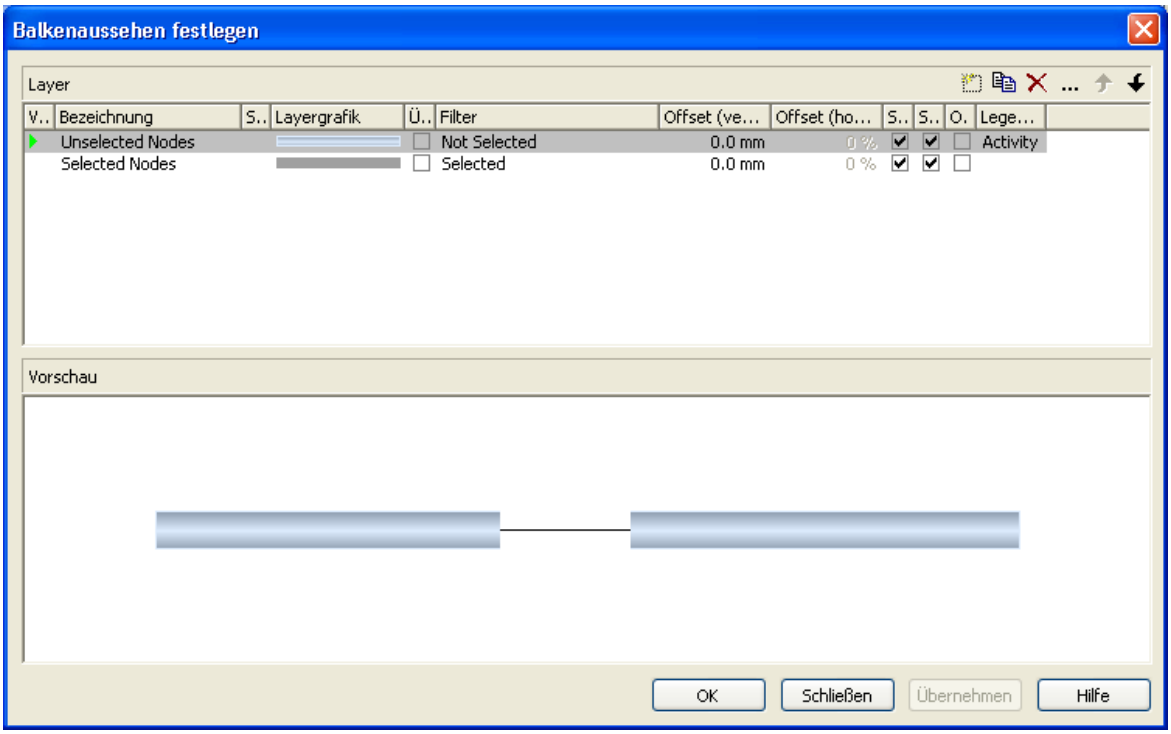


Bitte weisen Sie nun auf die gleiche Weise dem Filter "Selected" die Bedingung zu, dass das Knotenfeld "Selected" der Maindata-Tabelle gleich 1 sei.



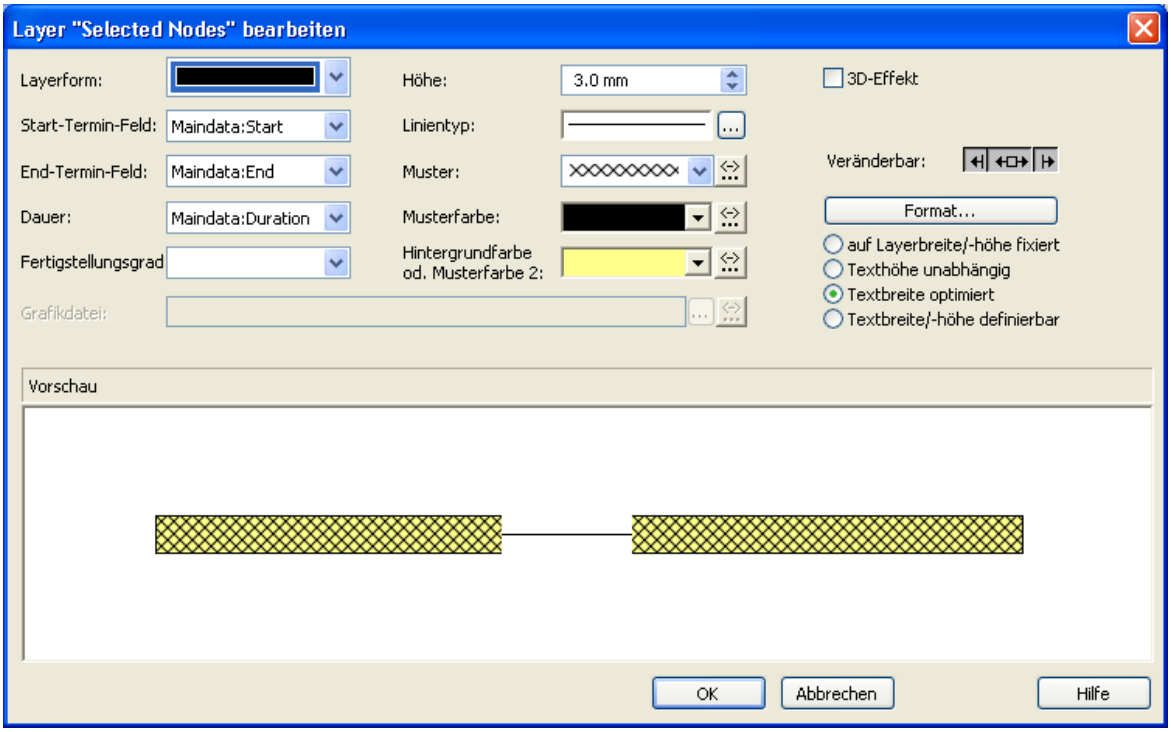
Schritt 6: In diesem Schritt definieren Sie das Erscheinungsbild selektierter und unselektierter Knoten in Abhängigkeit der beiden Filter.

Dazu wählen Sie bitte auf der Eigenschaftenseite **Objekte** die Schaltfläche **Layer...**, die Sie zum Dialog **Balkenaussehen festlegen** führt. Ändern Sie den Namen des vorhandenen Layers "Start-End" um in "Unselected Nodes", indem Sie das Feld in der Spalte **Name** editieren. Weisen Sie nun dem Layer den Filter "Not Selected" zu, indem Sie ihn in der Spalte **Filter** auswählen . Jetzt fertigen Sie mit der Schaltfläche  eine Kopie des Layers an und nennen ihn "Selected Nodes". Weisen Sie dem Layer den Filter "Selected" zu.

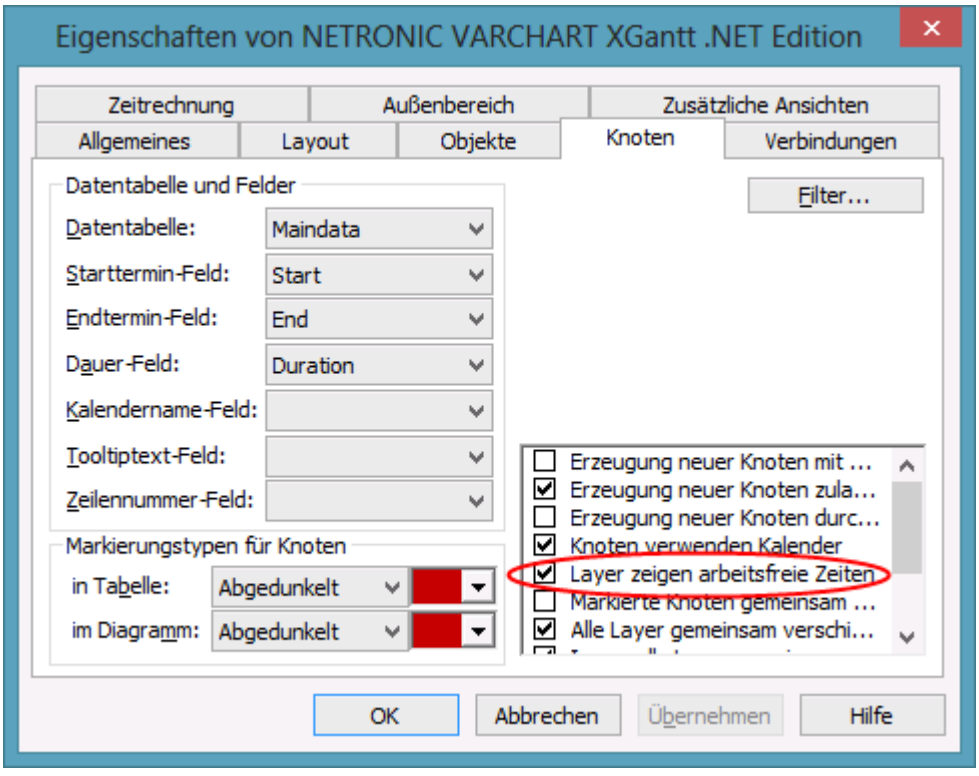


Noch besitzen beide Layer das gleiche Aussehen. Für den Layer "Selected Nodes" ändern Sie das Design, indem Sie durch Doppelklicken des Feldes in der Spalte **Layergrafik** den Dialog **Layer bearbeiten** aufrufen.

Bitte setzen Sie die **Hintergrundfarbe oder Musterfarbe 2** auf gelb, wählen Sie als Schraffurmuster **Muster** die Kreuzschraffur und setzen Sie die Schraffurfarbe **Musterfarbe** auf schwarz.

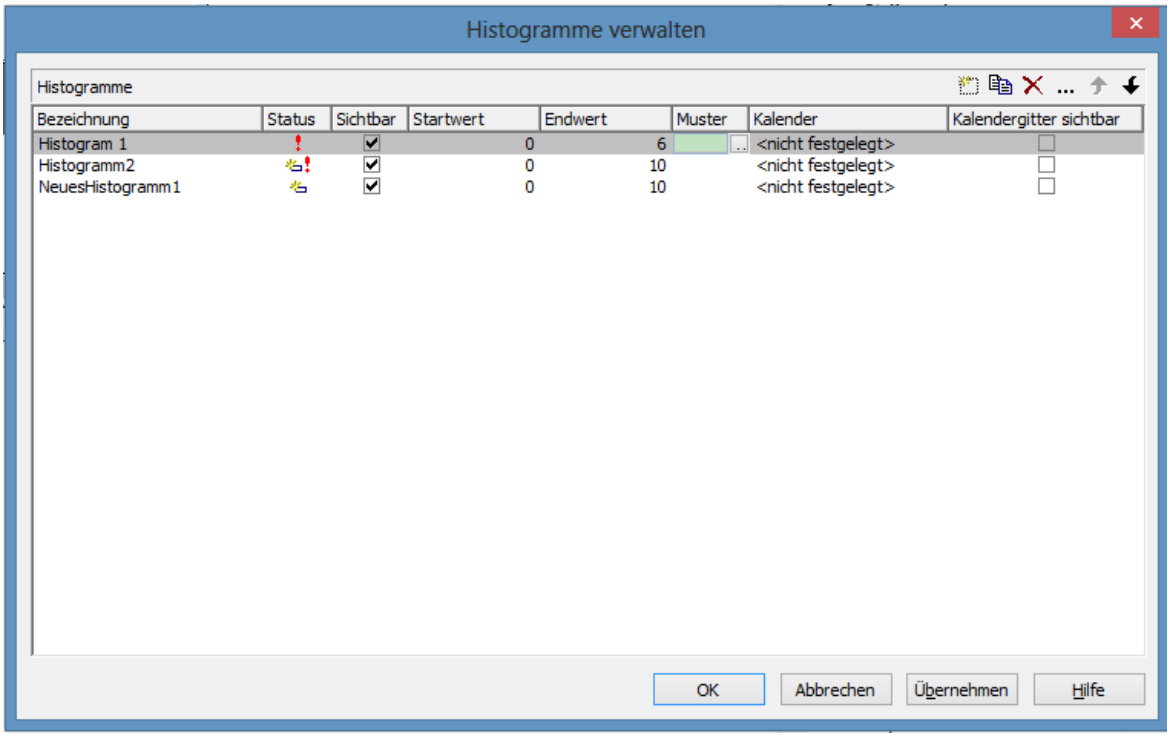


Um Nicht-Arbeitszeiten als dünne Linie darzustellen, nehmen Sie nun noch einen Einstellung auf der Eigenschaftenseite **Knoten** vor. Aktivieren Sie dort bitte die Option **Layer zeigen arbeitsfreie Zeiten**.



Schritt 7: In diesem Schritt werden vier Kurven für das Histogramm angelegt: die Kapazitätskurve, die Kurve aus unmarkierten Vorgängen, die Kurve aus markierten Vorgängen und eine Hilfskurve zur Flächenfüllung.

Auf der Eigenschaftenseite **Layout** gelangen Sie über **Histogramme verwalten...** in den entsprechenden Dialog.



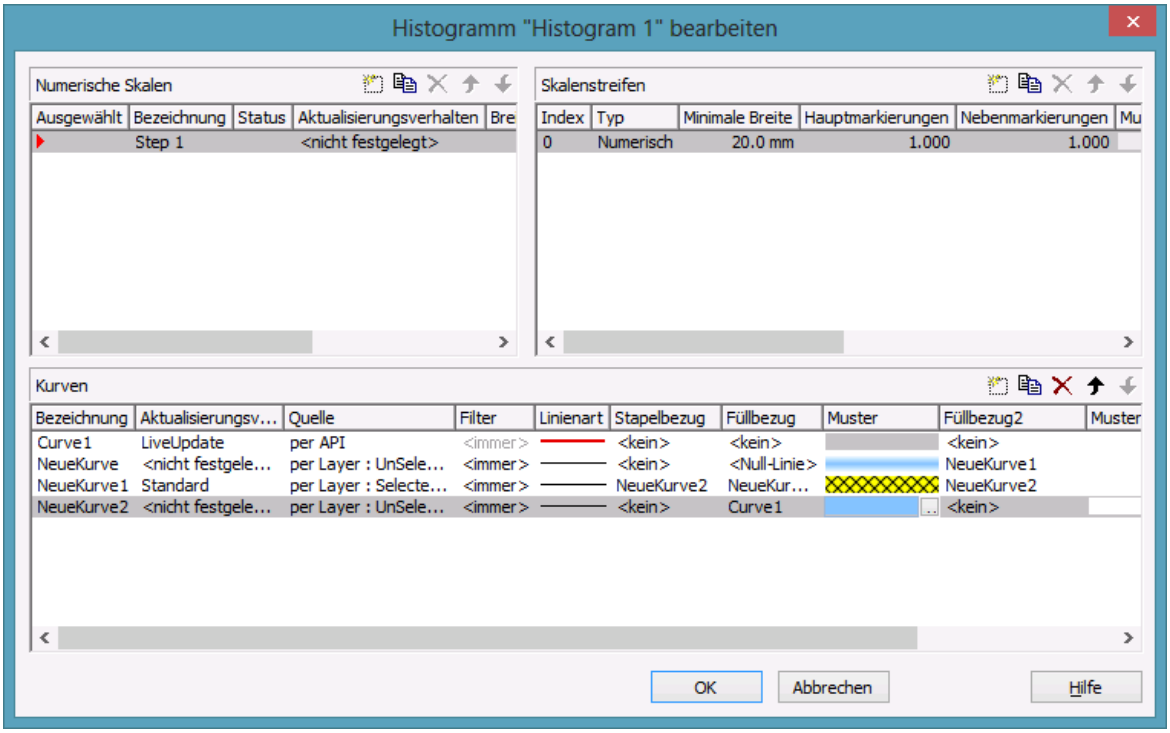
In einem Gantt-Diagramm können gleichzeitig mehrere Histogramme existieren. Jedes Histogramm besteht aus einer numerischen Skala und einer beliebigen Anzahl von Kurven.

Legen Sie nun den Start- und Endwert für die numerische Skala des Histogramms fest. Dazu stellen Sie im **Histogram_1** in der Spalte **Endwert** diesen auf 6.

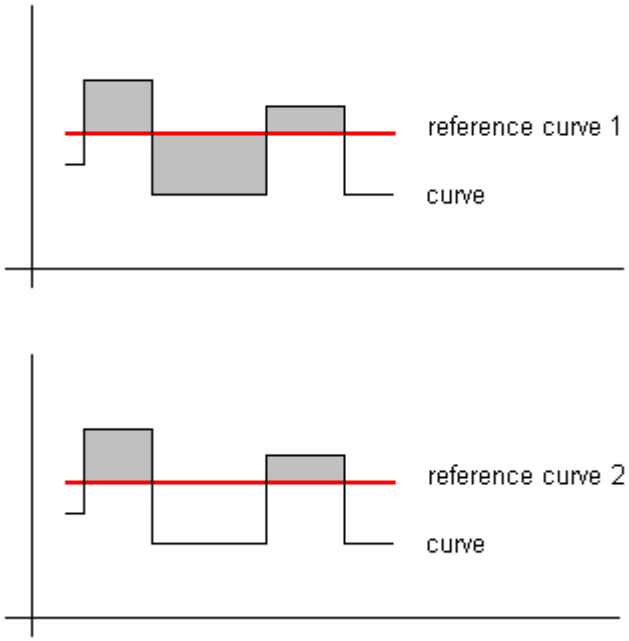
Um nun das bereits vordefinierte Histogramm im Detail weiter zu bearbeiten, klicken Sie oben rechts auf die Schaltfläche **bearbeiten** ···.

"Curve 1" soll die Werte für die rote Kapazitätskurve enthalten. "Curve 2" übernimmt die Darstellung der nicht selektierten Knoten und "Curve 3" die der selektierten Knoten. "Curve 4" sorgt für den grünen Hintergrund zur Darstellung der noch freien Kapazitäten.

Eine Kurve ist bereits vorhanden. Erstellen Sie drei weitere Kurven und legen Sie die Eigenschaften, wie in der Abbildung zu sehen, entsprechend fest.



Einer Kurve können Sie maximal zwei Referenzkurven zuweisen, mit denen die Kurve eine Fläche bildet (Bilder unten). Dabei hat die erste zugewiesene Referenz die Eigenschaft, alle Flächen mit Farben und Mustern auszufüllen, die sich oberhalb und unterhalb unserer Kurve bilden (oberes Bild). Die zweite zugewiesene Referenz hat die Eigenschaft, nur Flächen auszufüllen, die sich unterhalb der Kurve befinden, d.h. deren Y-Werte kleiner sind als die unserer Kurve (unteres Bild). Die Darstellung der Flächen der zweiten Referenzkurve hat Priorität vor der Darstellung der Flächen der ersten. Sie werden an diesem Beispiel sehen, welche Auswirkungen dies auf die Grafik hat.



Curve 1, die Kapazitätskurve, bezieht ihre Werte aus einer Liste, die wir später über Programmierung noch anlegen; daher wird die Datenquelle **Quelle** auf **per API** eingestellt. Ein zusätzlicher Filter erübrigt sich dadurch.

Legen Sie bitte die **Linienart** wie abgebildet als dicke rote Linie fest. Die Werte dieser Kurve werden nicht auf die Werte einer anderen Kurve addiert; aus diesem Grund wird kein **Stapelbezug** angegeben. Weiterhin wird ihr keine Referenzkurve **Füllbezug** zugewiesen, mit der sie eine Fläche bildet, daher bleiben die beiden Felder für die Referenzkurven unbesetzt, wodurch sich die Angabe von Flächenmustern erübrigt. Bitte richten Sie Curve 1 durch Anklicken der entsprechenden Felder wie beschrieben ein.

Curve 2, die die nicht selektierten Knoten darstellt, setzt sich zusammen aus den Werten der Layer, die "Unselected Nodes" heißen. Ein Filter zur weiteren Selektion ist nicht nötig. Bitte stellen Sie die Linienfarbe auf blau-violett. Die Kurvenwerte werden für die Darstellung nicht zu anderen hinzuaddiert; daher bleibt der **Stapelbezug** auch bei dieser Kurve leer. Die Kurve soll eine Fläche mit der X-Achse bilden; daher wählen Sie bitte im Feld **Füllbezug** den Wert **Null-Linie** aus.

Färben Sie die Fläche blau-violett ein. Diese aus nicht ausgewählten Knoten bestehende Kurve sollte auf besondere Weise anzeigen, wo sie die Kapazitätskurve überschreitet, um Kapazitätsengpässe direkt sichtbar zu machen. Sobald also Curve 2 die Kapazitätskurve überschreitet, soll die darunter liegende Fläche schraffiert erscheinen. Setzen Sie daher bitte Curve1 als zweite Referenzkurve und wählen Sie ein Schraffurmuster.

"Curve 3" übernimmt die Darstellung der selektierten Knoten. Aus diesem Grund weisen Sie ihr bitte als Datenquelle alle Layer mit dem Namen "Selected Nodes" ohne weiteren Filter. Legen Sie bitte eine graue Liniefarbe fest. Da die selektierten Knoten auf den nicht-selektierten Knoten aufliegend abgebildet werden sollen, müssen die Kurvenwerte von Curve 3 zu den Werten der nicht-selektierten Knoten, d.h. Curve 2, hinzuaddiert werden. Aus diesem Grund wählen Sie bitte als **Stapelbezug** Curve 2. Dieselbe Kurve dient auch als Referenzkurve, da sich die selektierten Knoten farblich gegen die unselektierten abgrenzen sollen. Wählen Sie daher eine graue Kreuzschraffur auf gelbem Grund als Füllmuster.

Die entstehende Fläche ist unter- wie oberhalb der Curve 2 sichtbar. Sie soll außerdem aber auch oberhalb der Kapazitätskurve erscheinen; aus diesem Grund weisen Sie nun als Referenzkurve 2 die Kapazitätskurve (Curve1) und gleiche Farben und Muster zu. Falls sich nun der markierte Knoten über die Kapaziätskurve erheben sollte, wird er ebenfalls mit grauer Kreuzschraffur auf gelbem Grund angezeigt (Sie könnten die Darstellung hier auch noch weiter differenzieren, indem Sie z.B. den Hintergrund rot wählen, dann zeigten selektierte Knoten oberhalb der Kapazitätsgrenze eine andere Darstellung als die selektierte unterhalb).

Als letzte wird mit Curve 4 eine Fläche definiert, die den Hintergrund zwischen der Kapazitätskurve und den Knoten hellgrün einfärbt, also freie Kapazitäten anzeigt. Sie wird unten begrenzt durch die unselektierten Knoten, daher nehmen wir diese als Datenquelle. Nach oben wird sie begrenzt durch die Kapazitätskurve, die Sie aus diesem Grund als erste Referenzkurve eintragen.

Nun stellt sich die Frage, warum Curve 4 die Darstellung der selektierten Knoten nicht überlagert. Die Antwort: es gibt eine Prioritätenreihenfolge, die durch die Reihenfolge der Knoten in der Liste dieses Dialogs festgelegt wird. Die weiter unten gelisteten Kurven besitzen eine niedrigere Zeichenpriorität als die oberen. Aus diesem Grund werden die Flächen, die Kurve 3 zugeordnet sind, auf den Flächen von Kurve 4 abgebildet. Die Priorität können Sie mit Hilfe der Pfeile oben rechts ändern.

Schritt 8:

Abschließend füllen Sie nun die Kapazitätskurve mit Werten. Dazu ändern Sie den Programmcode im **Load-Ereignis** wie im Beispiel vorgegeben:

Code-Beispiel VB.NET

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
   VcGantt1.Width = ClientSize.Width - VcGantt1.Left
```

```
   VcGantt1.Height = ClientSize.Height - VcGantt1.Top

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

   VcGantt1.EndLoading()

   VcGantt1.OptimizeTimeScaleStartEnd(3)

   'calculate end date
   Dim node As VcNode

   For Each node In VcGantt1.NodeCollection
      setNodeEndDate(node)
   Next
   Dim histogram As VcHistogram
   Dim curve As VcCurve

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

End Sub
```

Code-Beispiel C#

```
private void Form1_Load(object sender, System.EventArgs e)
   {
   vcGantt1.Width  = ClientSize.Width  - vcGantt1.Left;
   vcGantt1.Height = ClientSize.Height - vcGantt1.Top;

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

   vcGantt1.EndLoading();

   vcGantt1.OptimizeTimeScaleStartEnd(3);

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

```
VcHistogram histogram =
vcGantt1.HistogramCollection.FirstHistogram();
VcCurve curve = histogram.CurveCollection.CurveByName("Curve 1");
curve.PointsEquidistant = false;
curve.SetValues(Convert.ToDateTime("01.05.2007"),"2");
curve.SetValues(Convert.ToDateTime("05.05.2007"),"0");
curve.SetValues(Convert.ToDateTime("07.05.2007"),"2");
curve.SetValues(Convert.ToDateTime("12.05.2007"),"0");
curve.SetValues(Convert.ToDateTime("14.05.2007"),"4");
curve.SetValues(Convert.ToDateTime("19.05.2007"),"0");
curve.SetValues(Convert.ToDateTime("21.05.2007"),"2");
curve.SetValues(Convert.ToDateTime("26.05.2007"),"0");
curve.SetValues(Convert.ToDateTime("28.05.2007"),"2");
}
```

Starten Sie nun bitte das Programm und markieren Sie einen Vorgang. Im Histogramm können Sie, wenn Sie einen Knoten markieren, durch die gelb hinterlegte Schraffur sofort erkennen, welchen Anteil der Vorgang an der gesamten Ressourcenbelegung hat.

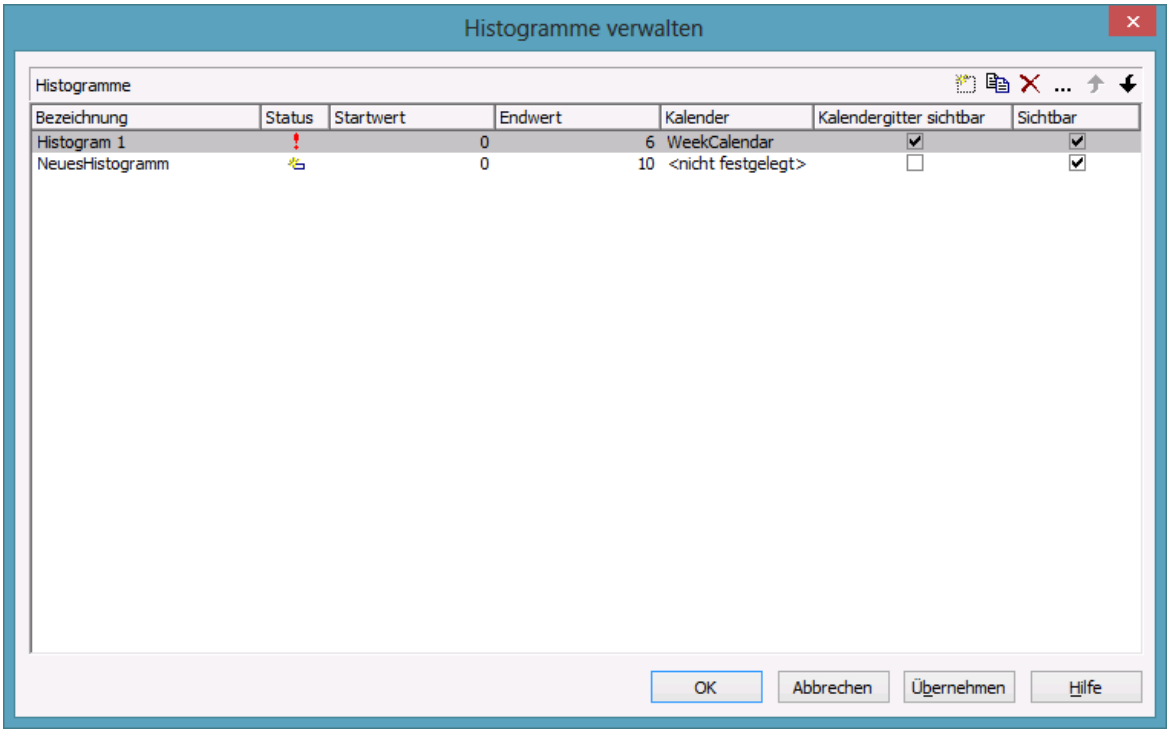
Wenn Sie Vorgänge verschieben, ändert sich das Kapazitätsgebirge und Sie erkennen Über- bzw. Unterlastungen, die sich aus dieser Interaktion ergeben.

Kalendergitter im Histogramm

Sie können einem Histogramm ein oder mehrere Kalendergitter zuweisen, so dass Sie dort z.B. die unterschiedlichen Kalendergitter des Gantt-Graphen auch im Histogramm abbilden können.

Um einem Histogramm ein eigenes Kalendergitter zuweisen zu können, müssen drei Bedingungen erfüllt sein:

1. Dem Histogramm muss ein Kalender zugewiesen sein
2. Das Kalendergitter muss eingeschaltet sein
3. Für die Darstellung des Kalendergitters muss ein Aussehen definiert sein



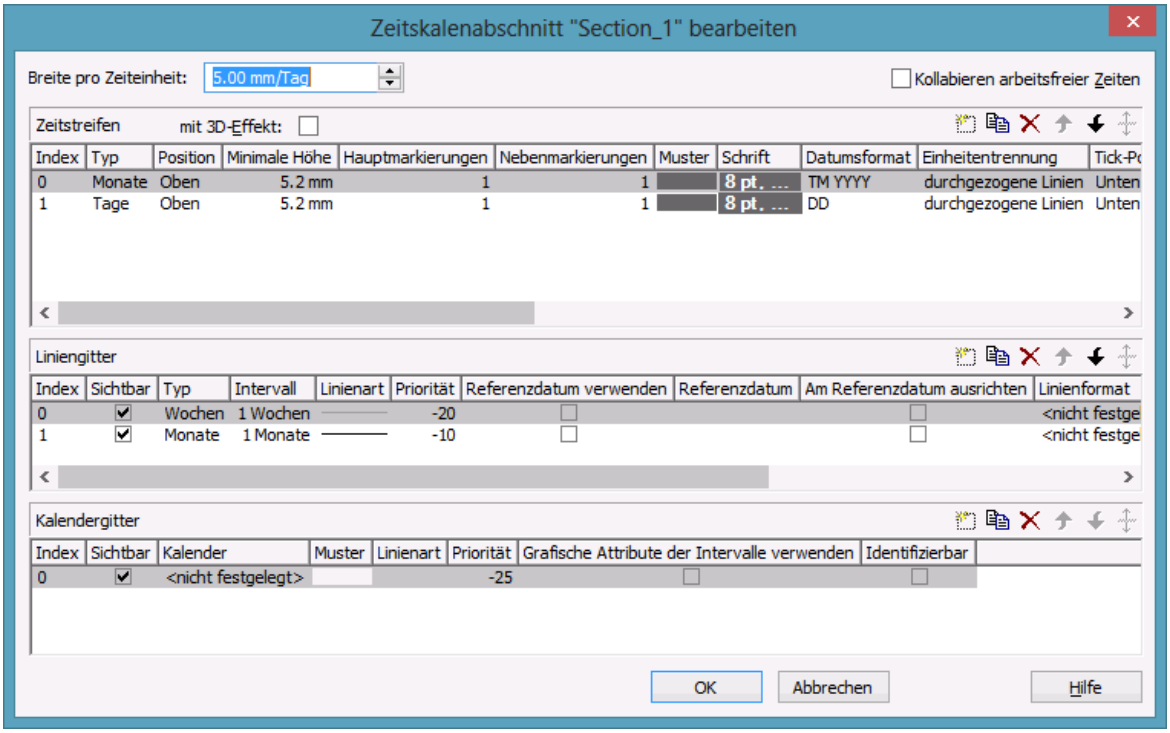
Kalender zugewiesen, Kalendergitter eingeschaltet

Die entsprechenden API-Aufrufe heißen:

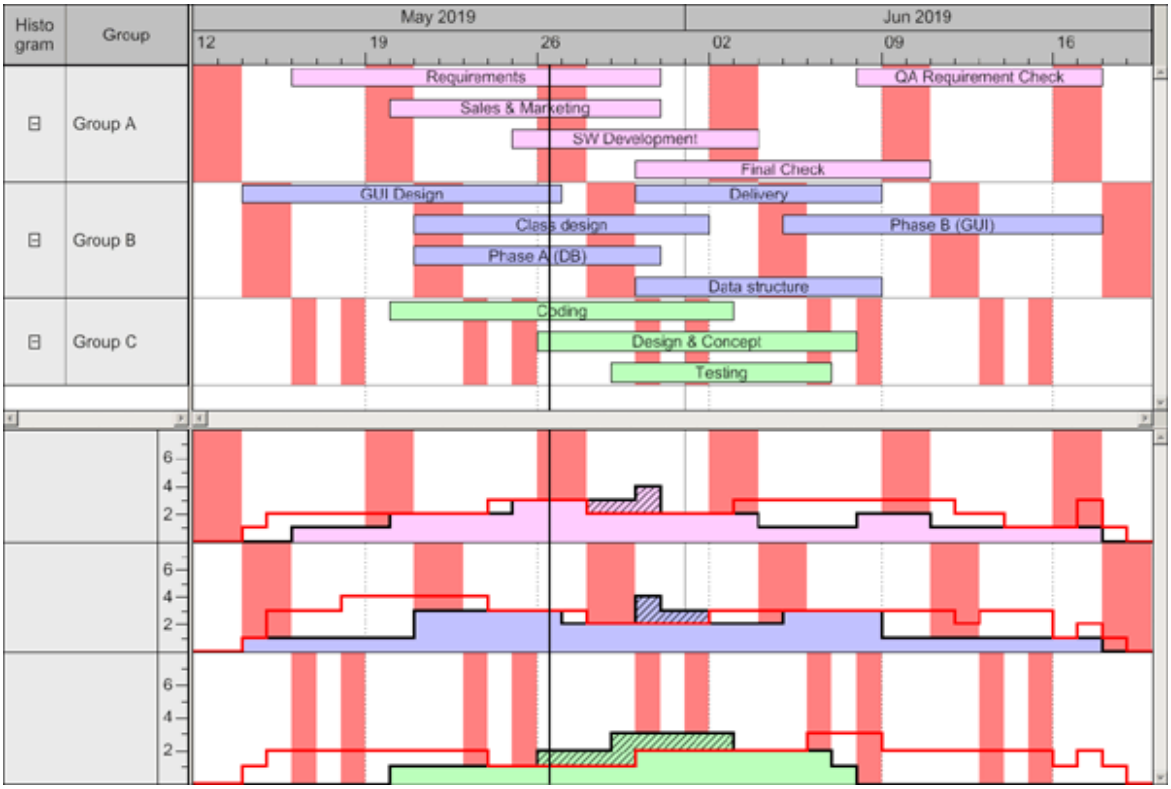
Code-Beispiel VB.NET

```
// Zuweisung des Kalendernamens an das Histogram:
histogram.calendarName = group.DataField(14)
// Einschalten des Kalendergitters
histogram.ShowCalendarGrids = True
// Einschalten des Histogramms
histogram.Visible = True
```

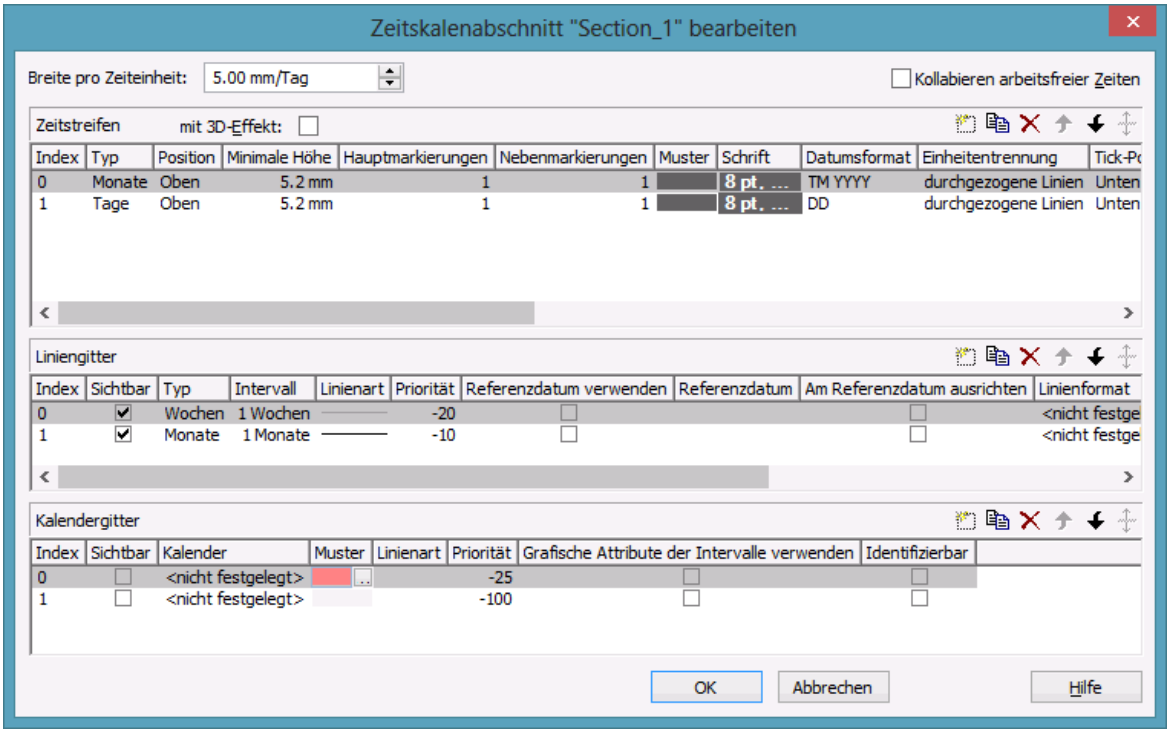
Als Kalendergitter für das Histogramm verwendet VARCHART XGantt das erste nicht sichtbare Kalendergitter des ersten Zeitabschnitts (Section) in der Zeitskala, sofern es dort kein weiteres gibt. Dieses ist auch das Kalendergitter, das vom Gantt-Graphen für die Gruppen verwendet wird (groupwise).



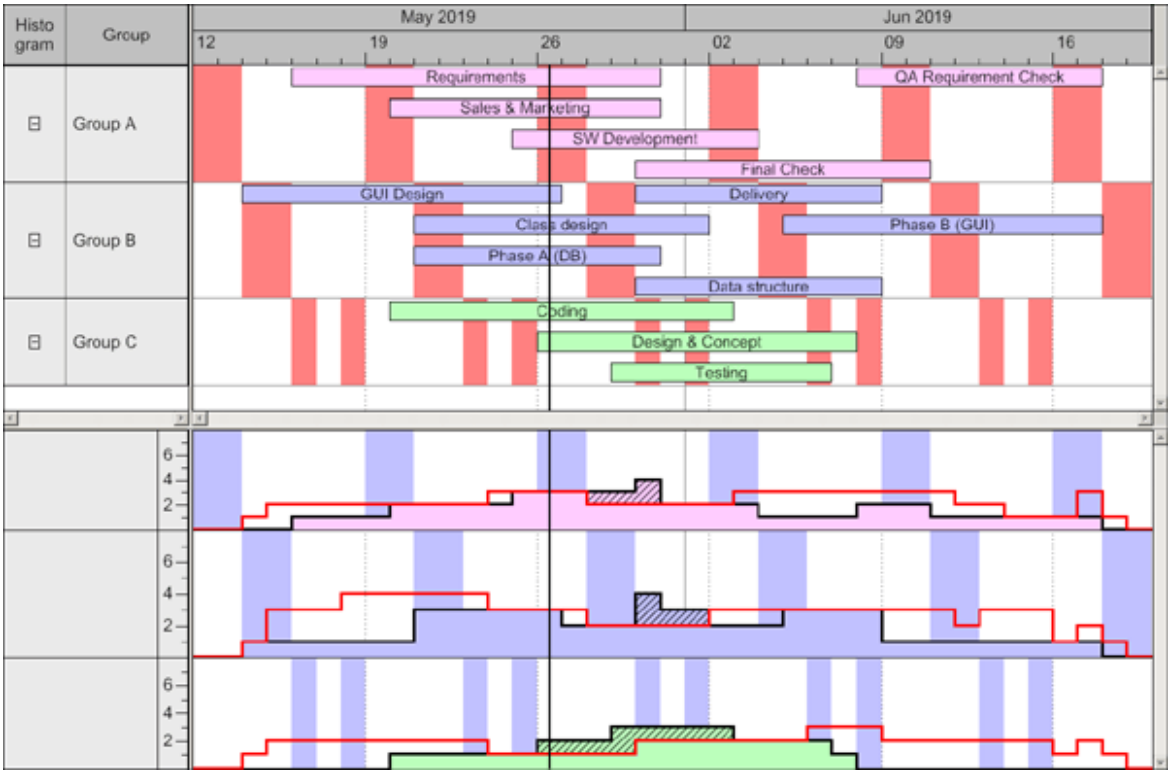
Das Kalendergitter zeigt dadurch dasselbe Aussehen im Gantt-Graphen wie im Histogramm. Im Beispiel unten ist dies ein Kalendergitter, das für jede Gruppe ein unterschiedliches Muster zeigt (groupwise calendar grid):



Setzen Sie dem Zeitskalenabschnitt ein weiteres unsichtbares Kalendergitter, verwendet VARCHART XGantt dieses als Kalendergitter des Histogramms:



Dieses Kalendergitter kann ein anderes Aussehen haben als das Kalendergitter des Gantt-Graphen. In unserem Fall hat es eine andere Farbe:



2.11 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 VcGantt 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 **PrintEx** des Objektes VcGantt 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 VcGantt 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 5.23, "Seite einrichten".

2.12 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**.

2.13 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.

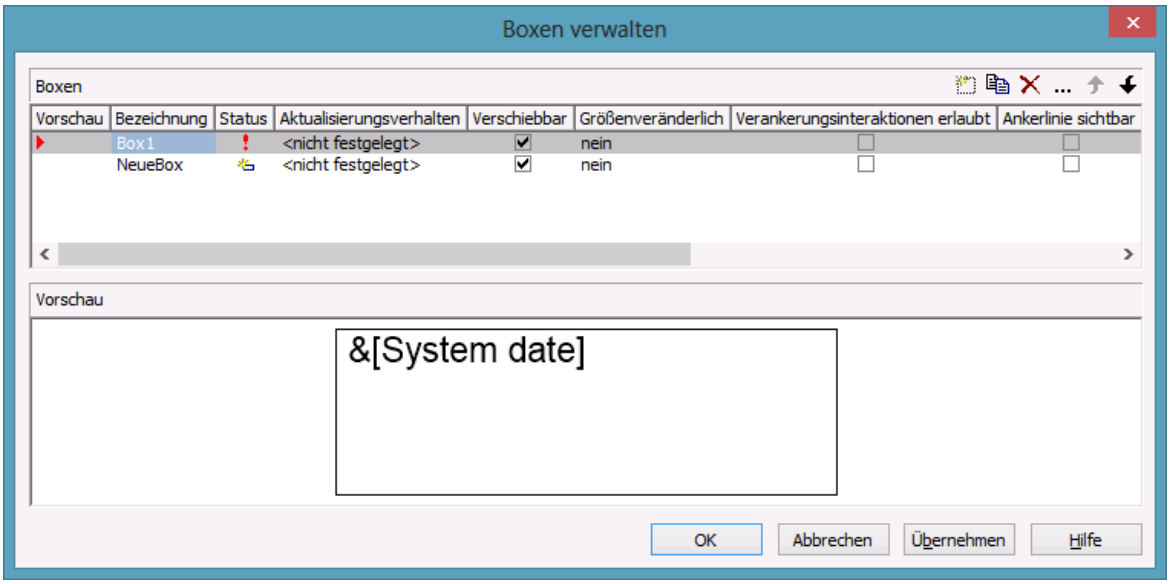
So lesen Sie eine bereits gespeicherte Konfiguration wieder ein:

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:

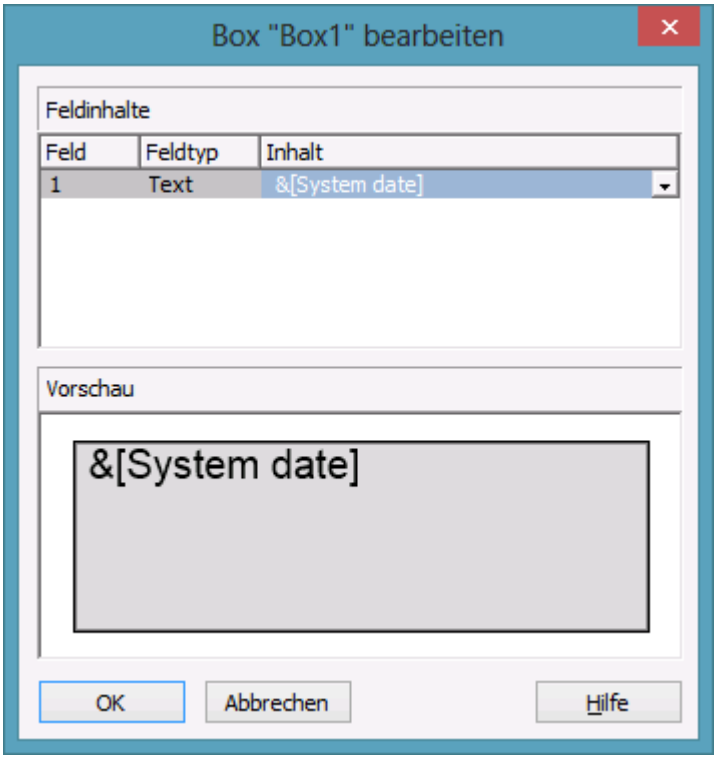
- ihre Bezeichnung
- ob die Box zur Laufzeit frei im Diagrammbereich verschiebbar sein soll
- ob und wie die Größe der Box interaktiv geändert werden kann
- ob Verankerungsinteraktionen per Maus oder Kontextmenü erlaubt sein sollen
- ob bei Verankerung die eingestellten Referenzpunkte beim Knoten und bei der Box durch eine Linie verbunden werden sollen

- eine Knoten-ID, um den Knoten zu identifizieren, an dem die jeweilige Box verankert werden 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)
- 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.)

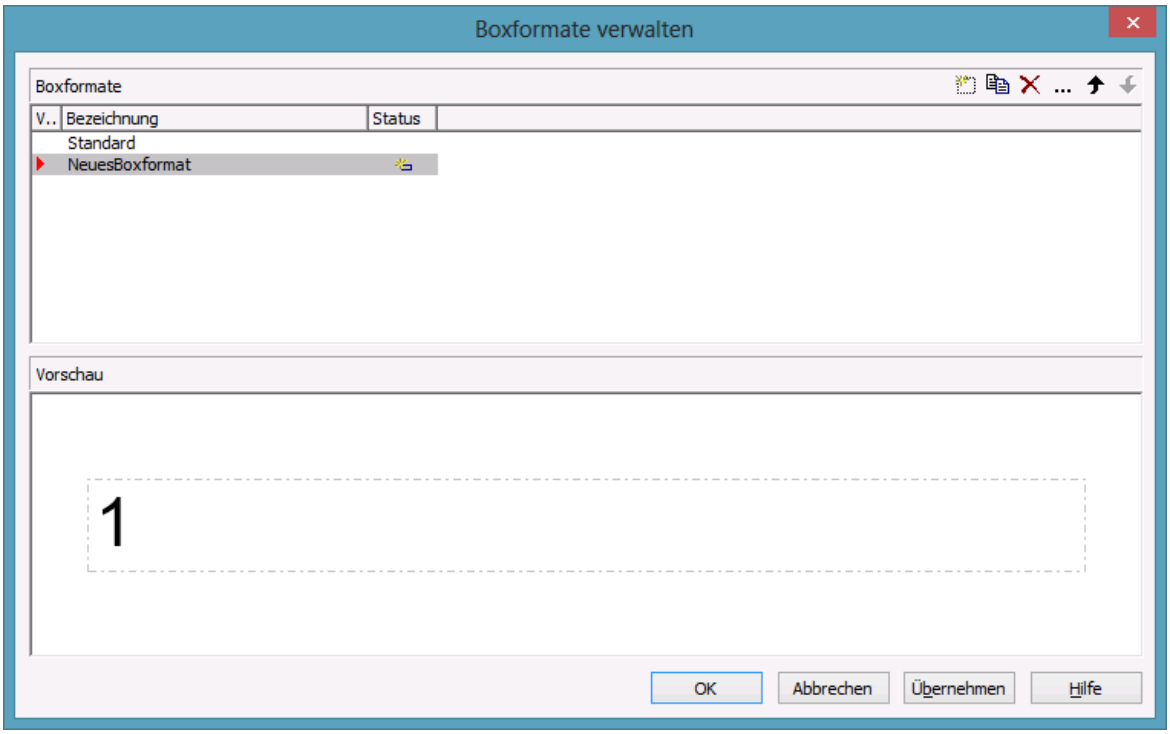
In der Spalte **Typ** wird der Feldtyp jedes Feldes angezeigt (Text oder Grafik).

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.

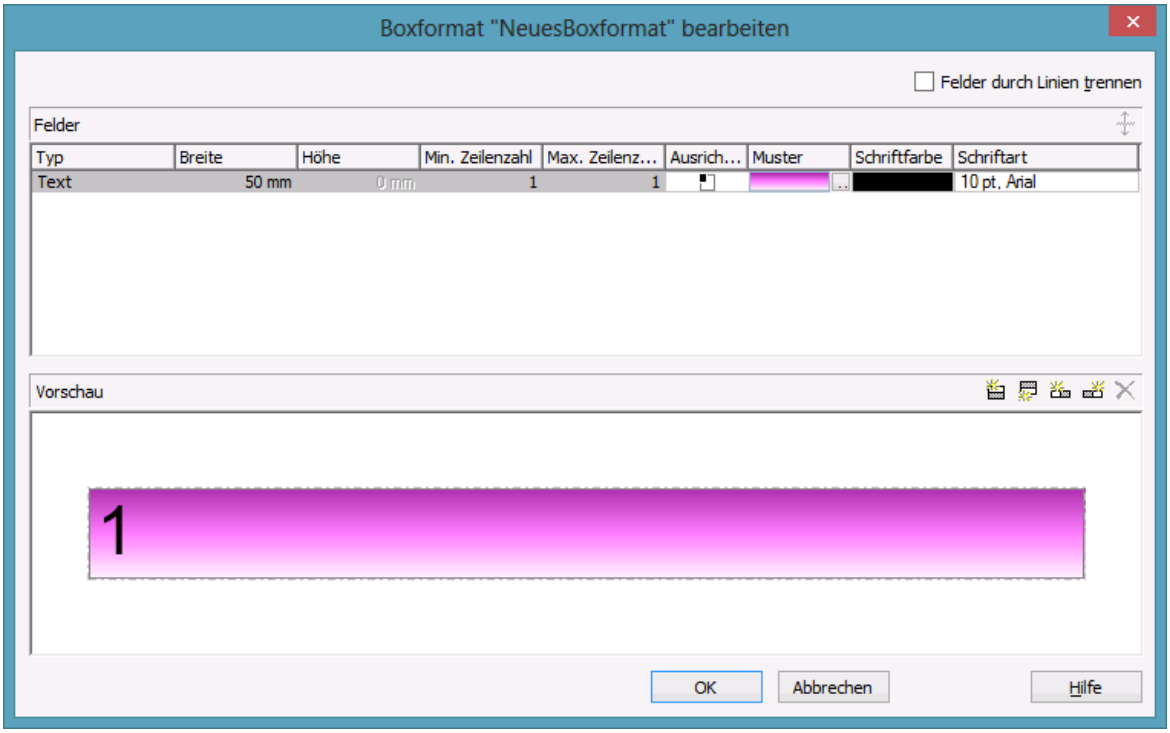
> Boxformate

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.



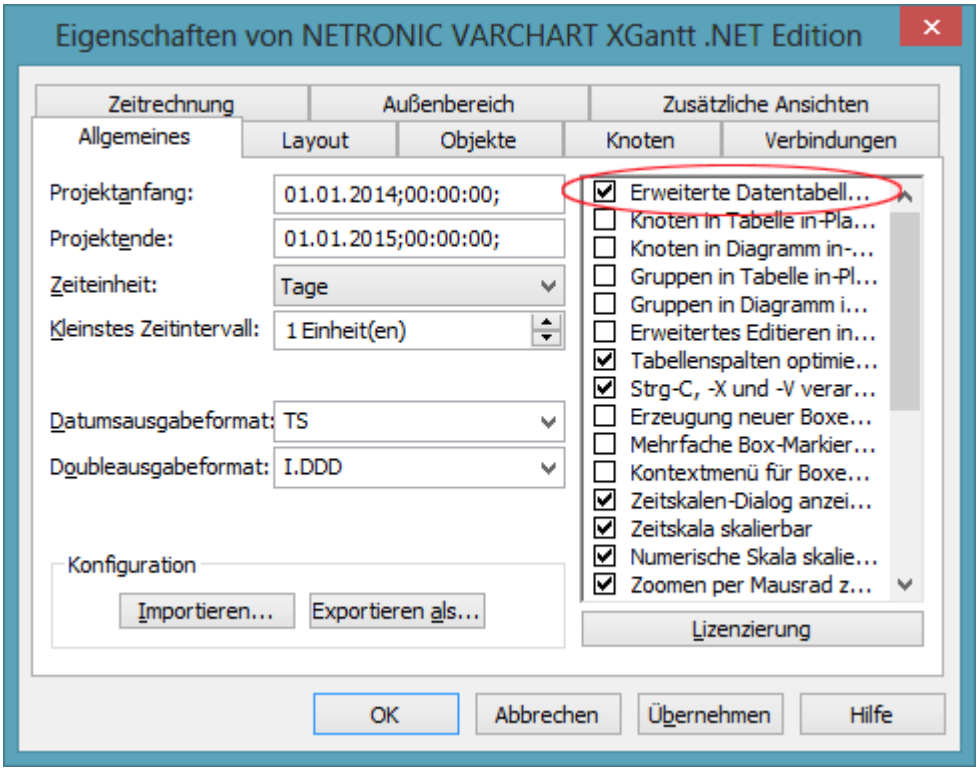
Sie können hier festlegen, ob die Felder der Box durch Linien getrennt werden sollen. Außerdem können Sie für jedes Feld der Box folgendes festlegen:

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

3.2 Datentabellen

Zur Darstellung von Gantt-Diagrammen verwendete VARCHART XGantt ursprünglich zwei Standard-Datentabellen für Knoten und Verbindungen, deren Felder individuell festgelegt werden können. Seit der Version VARCHART XGantt 4.0 gibt es ein erweitertes Konzept:. 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, der Zugriff auf die Felder des Hauptdatensatzes vom Detaildatensatz aus erleichtert und der in VARCHART XGantt integrierte ResourceScheduler2 mit den notwendigen Daten versorgt.

Aus Kompatibilitätsgründen mit bestehenden Anwendungen arbeitet VARCHART XGantt 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**:

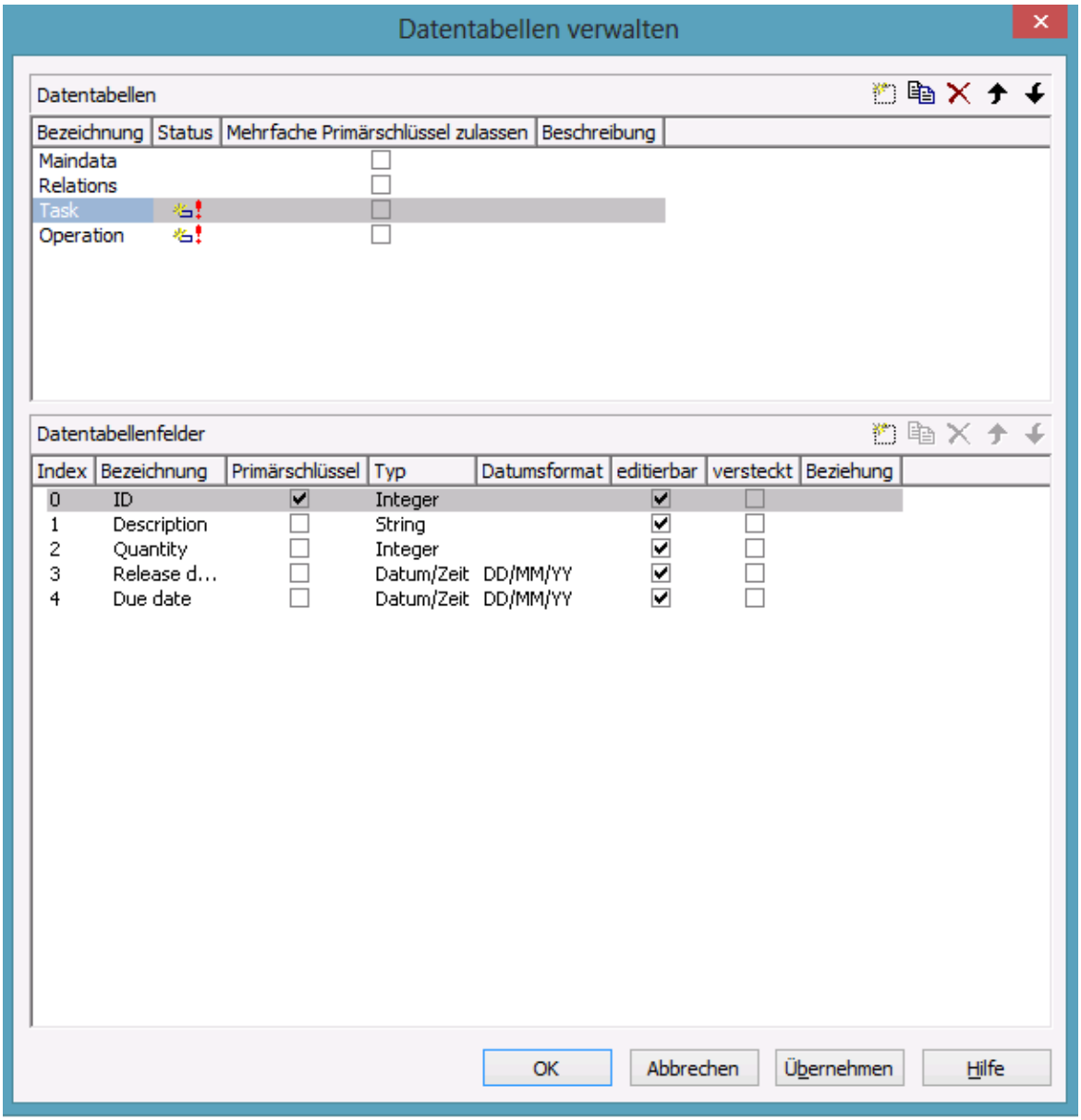


Alternativ können die erweiterten Datentabellen statt zur Designzeit auch zur Laufzeit über die API eingeschaltet werden, indem die Eigenschaft **ExtendendDataTablesEnabled** des Objekts **VcGantt** auf **True** gesetzt wird.

> Datentabellen verwalten

Standardmäßig sind die beiden Datentabellen **Maindata** und **Relations** vorhanden. Auf der Eigenschaftenseite **Objekte** gelangen Sie über die Schaltfläche **Datentabellen…** in den Dialog **Datentabellen verwalten**. Nur im Modus **Erweiterte Datentabellen zulassen** ist es möglich, neue Datentabellen anzulegen. Im Bild unten wurden die Datentabellen **Task** und **Operation** durch Klick auf  im Bereich **Datentabellen** neu angelegt.

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 zusammengesetzen 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 dem Beispiel wird eine Verknüpfung zwischen den Tabellen **Operation** und **Task** durch die Angabe von **Task:ID** in der Spalte **Beziehung** vereinbart.

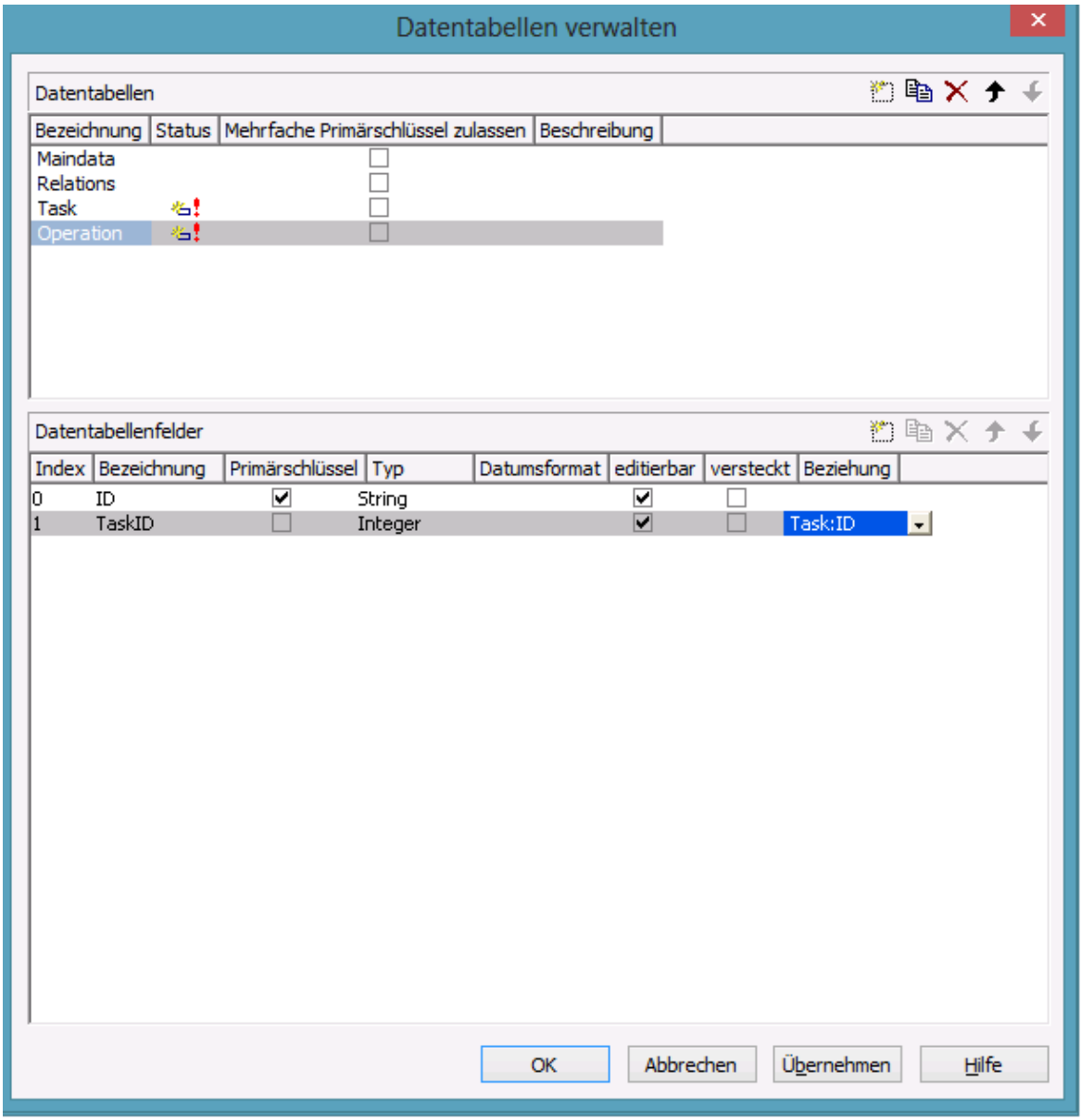


Tabelle Task:

ID	Description	Quantity	Release date	Due date
1	Task 1	10	12.05.13	20.05.13
2	Task 2	20	01.06.13	15.06.13

Tabelle Operation:

ID	TaskID	Description	Start	End
1	1	Operation 1	12.05.13	14.05.13
2	1	Operation 2	15.05.13	19.05.13

ID	TaskID	Description	Start	End
3	2	Operation 3	01.06.13	05.06.13
4	2	Operation 4	05.06.13	11.06.13
5	2	Operation 5	11.06.13	15.06.13

Code-Beispiel VB.NET

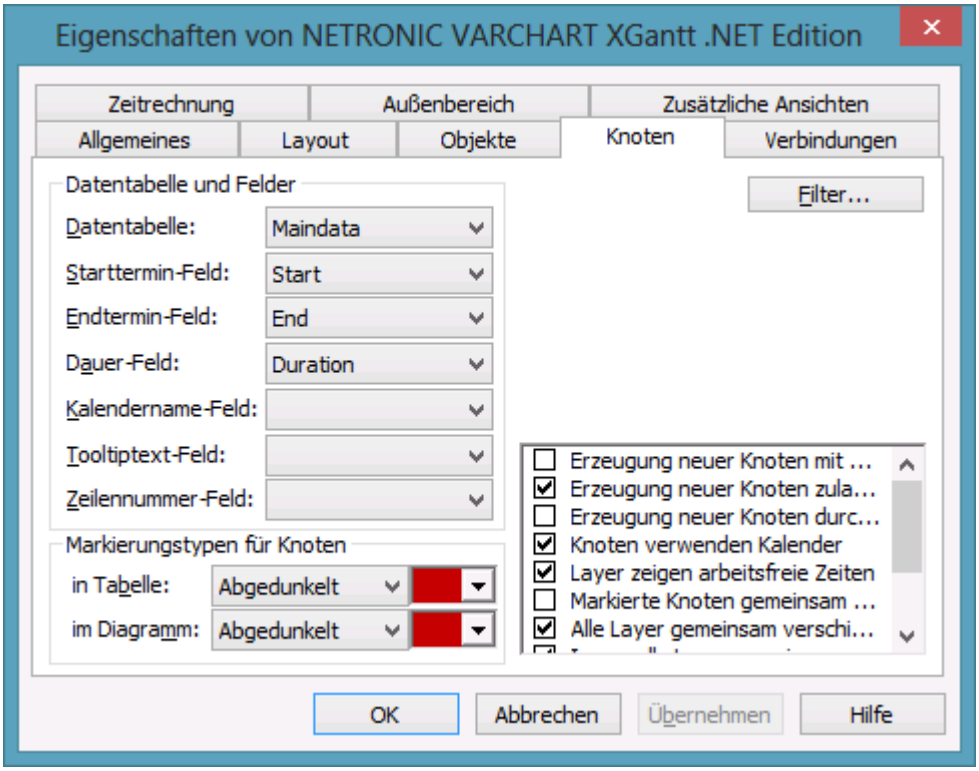
```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
dataTable = dataTableCltn.DataTableByName("Task")

dataTable.DataRecordCollection.Add("1;Task 1;10;12.05.2013;20.05.2013")
dataTable.DataRecordCollection.Add("2;Task 2;10;01.06.2013;15.06.2013")

dataTable = dataTableCltn.DataTableByName("Operation")
dataTable.DataRecordCollection.Add("1;1;Operation
1;12.05.2013;14.05.2013")
dataTable.DataRecordCollection.Add("2;1;Operation
2;15.05.2013;19.05.2013")
dataTable.DataRecordCollection.Add("3;2;Operation
3;01.06.2013;05.06.2013")
dataTable.DataRecordCollection.Add("4;2;Operation
4;05.06.2013;11.06.2013")
dataTable.DataRecordCollection.Add("5;2;Operation
5;11.06.2013;15.06.2013")

VcGantt1.EndLoading()
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.DataTableByName("Task");

dataTable.DataRecordCollection.Add("1;Task 1;10;12.05.2013;20.05.2013");
dataTable.DataRecordCollection.Add("2;Task 2;10;01.06.2013;15.06.2013");

dataTable = dataTableCltn.DataTableByName("Operation");
dataTable.DataRecordCollection.Add("1;1;Operation
1;12.05.2013;14.05.2013");
dataTable.DataRecordCollection.Add("2;1;Operation
2;15.05.2013;19.05.2013");
dataTable.DataRecordCollection.Add("3;2;Operation
3;01.06.2013;05.06.2013");
dataTable.DataRecordCollection.Add("4;2;Operation
4;05.06.2013;11.06.2013");
dataTable.DataRecordCollection.Add("5;2;Operation
5;11.06.2013;15.06.2013");

vcGantt1.EndLoading();
```

In Abhängigkeit davon, welche Tabelle unter **Datentabelle** bei der Eigenschaftenseite **Knoten** angegeben wird, ergibt sich eine andere Basis, aus der

die grafische Darstellung der Knoten hergeleitet wird. Beim interaktiven Anlegen von Knoten ist es diejenige Tabelle, der automatisch neue Datensätze hinzugefügt werden. Die in der Visualisierung sichtbaren Zeilen werden von dem aktiven KnotenFilter, der Gruppierung und von Darstellungsoptionen beeinflusst.



Dies ist das Ergebnis im Tabellenteil des Gantt-Diagramms, wenn die Tabelle **Operation** als Basis genommen wird. Die Einträge für Description, Quantity, Due date stammen aus der Haupttabelle **Task**.

Description	Quantity	Due date	Operation
Task1	10	20.05.13	Operation1
Task1	10	20.05.13	Operation2
Task2	20	15.06.13	Operation3
Task2	20	15.06.13	Operation4
Task2	20	15.06.13	Operation5

Die sichtbare Tabelle in XGantt besteht nur aus zwei Einträgen, wenn statt der Tabelle **Operation** die Tabelle **Task** herangezogen wird.

ID	Description	Quantity	Due date	Operation
1	Task 1	10	20.05.13	
2	Task 2	20	15.06.13	

In der VARCHART XGantt 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 Elementeanzahl 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 = VcGantt1.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 =
vcGantt1.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
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 Indexzahlen.

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

VcGantt1.TimeScaleEnd = DateSerial(2014, 1, 1)
VcGantt1.TimeScaleStart = DateSerial(2013, 1, 1)

VcGantt1.ExtendedDataTablesEnabled = True
dataTable = VcGantt1.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")

VcGantt1.EndLoading()

'...

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

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

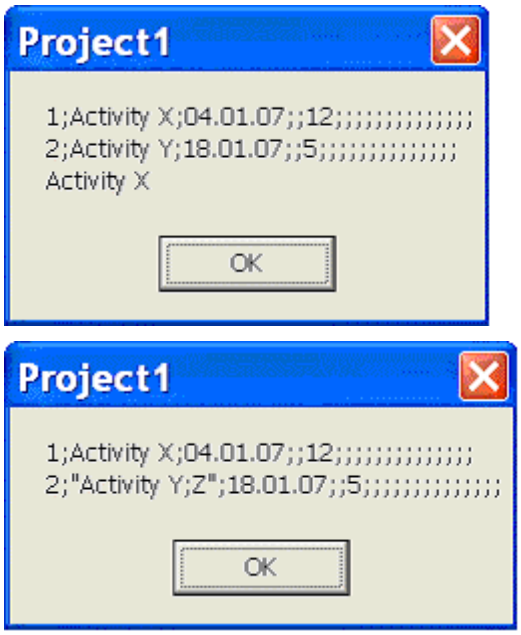
```
dataRec2.AllData = "2;Activity Y;18.01.13;;5"
dataRec2.Update()

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

'...


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

Code-Beispiel C#

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

//...

VcDataRecord dataRec1;
VcDataRecord dataRec2;

string content;

vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.01.2014");
vcGantt1.TimeScaleStart = Convert.ToDateTime("01.01.2013");

vcGantt1.ExtendedDataTablesEnabled = true;
VcDataTable dataTable =
vcGantt1.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");

vcGantt1.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.

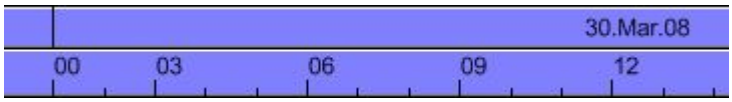
Eine VARCHART-Komponente nimmt bei der Umschaltung den Startzeitpunkt, den Endzeitpunkt inkl. Stunde, Monat und Tag, den das System als Regel mitteilt. Das heißt aber auch, dass die Umschaltzeiten für die Jahre vor und nach dem aktuellen Jahr extrapoliert werden, sodass etwaige Abweichungen, die für vorherige Jahre galten bzw. für kommende Jahre gelten werden, nicht berücksicktigt 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.

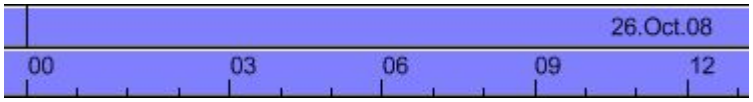
Augenblicklich können VARCHART Komponenten bei der Sommerzeit nur eine Zeitdifferenz zur Winterzeit von exakt einer Stunde berücksichtigen. Außerdem darf die Umschaltung nur zur vollen Stunde erfolgen.

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 derzeitig bei Übergabe, bei Rückgabe und bei Ausgabe nicht unterschieden werden kann.

In der Zeitskala wird die Umschaltzeit sichtbar, wenn diese stundengenau anzeigt.



Umschaltzeit im Bereich zwischen 0 und 3 Uhr im Frühling (1 Stunde fehlt)



Umschaltzeit im Bereich zwischen 0 und 3 Uhr im Herbst (1 Stunde ist doppelt)

> Neues Standarddatum ab Version 4.3

Wenn in einer VARCHART-Komponente ein Datum erfragt wird, das nicht existiert, wurde bis zur Version 4.3 das Datum **31.12.1899 00:00:00** zurückgegeben. Ab Version 4.3 wird das Datum **01.01.0001 00:00:00** zurückgeliefert.

Das kann unter bestimmten Konstellationen die Ausnahme **Parameter außerhalb des zulässigen Bereichs** (ArgumentOutOfRange exception) zur Folge haben, was durch eine Ausnahmebehandlung abgefangen werden kann.

Wenn Sie ein Datum beispielsweise mit DateTimePicker-Controls von .NET innerhalb Ihrer Anwendung verwalten und wenn Sie versuchen ein "leeres" Datum darin anzuzeigen, wurde vor Version 4.3 der Zeitpunkt **31.12.1899 00:00:00** angezeigt. Der Default ab Version 4.3, also das Datum **01.01.0001 00:00:00**, kann aber mit den Standard-Einstellungen des DateTimePickers nicht angezeigt werden und es wird die Ausnahme **ArgumentOutOfRange** ausgelöst.

Ihr Programm sollte also in jedem Fall darauf reagieren, d.h. Ausnahmebehandlungen müssen programmiert werden, andernfalls kann es

zu einer unbehandelten Ausnahme und damit zu einem unerwarteten Programmende kommen.

3.4 Der Einsatz des Kalenders

Ein Kalender besteht aus der lückenlosen Abfolge von Arbeits- und Nichtarbeitszeiten. Bei einem Kalender mit variablem Profil (Schichtkalender) reihen sich unterschiedliche Arbeitszeitabschnitte wie zum Beispiel Früh-, Spät- und Nachtschichten wiederholt aneinander. Der Kalender selbst besitzt keine visuelle Darstellung, er besteht lediglich aus der logischen Unterscheidung zwischen Arbeits- und Nichtarbeitszeiten. Sichtbar werden kann ein Kalender nur mithilfe des Objekts **CalendarGrid**.

In VARCHART XGantt wird ein Kalender dazu benutzt, um über die Dauer die Anfangs- oder Endtermine der Vorgänge zu berechnen. Der vordefinierte Basiskalender mit dem Namen **BaseCalendar** wird für alle Berechnungen herangezogen, solange keine anderweitigen Festlegungen getroffen werden. Im Basiskalender sind die Wochentage von Montag bis Freitag als Arbeitszeit und die Wochenenden Samstag und Sonntag als arbeitsfrei vereinbart. Bei Bedarf kann der Basiskalender verändert werden.

Einen Kalender definieren

Ein Kalender kann zur Entwurfszeit über die Eigenschaftenseiten oder zur Laufzeit über die Programmierschnittstelle spezifiziert werden. In diesem Kapitel werden die grundlegenden Konzepte im Umgang mit Kalendern aus Programmiersicht erläutert und anhand von Programmbeispielen in C# verdeutlicht. Die Festlegung von Kalendern über die Eigenschaftenseiten wird im Kapitel **Eigenschaftenseiten und Dialogfelder** näher beschrieben.

Das Steuerelement **VcGantt** besitzt ein Objekt vom Typ **VcCalendarCollection**, das für die Verwaltung der Kalender verantwortlich ist. Es besitzt die gleichen Verwaltungsfunktionen wie alle Auflistungen (Collections) in VARCHART XGantt. Der vordefinierte Kalender mit dem Namen **BaseCalendar** sowie alle anderen zur Entwurfszeit erstellten Kalender werden automatisch in die Auflistung aufgenommen.

Ein neuer Kalender wird über die Methode **Add** des Objekts **CalendarCollection** angelegt. Als Parameter benötigt die Methode einen eindeutigen Namen zur späteren Identifizierung des Objekts. Der neue Kalender besteht zunächst nur aus Arbeitszeit.

Hinweis: Ein Kalender muss zu jedem Zeitpunkt mindestens eine Arbeitszeitspanne umfassen, da ein Kalender, der nur aus Nichtarbeitszeit besteht, nicht möglich ist.

Damit die Ergebnisse in den Abbildungen der Gantt-Diagramme nachvollziehbar sind, wird für die Zeitskala in den Beispielen ein fester Darstellungs-

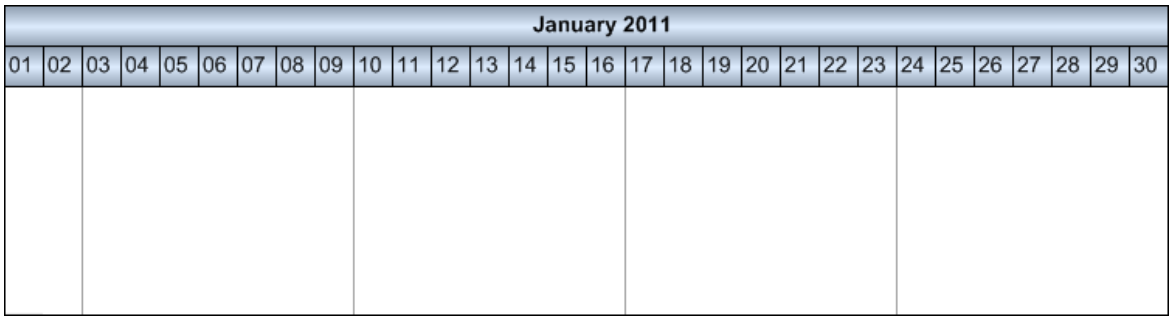
zeitraum vom 1.1.2011 bis zum 31.12.2011 vereinbart. Ein Kalender wird im Hintergrund des Gantt-Diagramms erst sichtbar, wenn er zum aktiven Kalender der Kollektion gemacht wird:

Code-Beispiel C#

```
// Darstellung des neuen Kalenders
vcGantt1.TimeScaleEnd  = new DateTime(2012, 1, 1);
vcGantt1.TimeScaleStart = new DateTime(2011, 1, 1);
VcCalendar calendar =
vcGantt1.CalendarCollection.Add("CompanyCalendar1");
vcGantt1.CalendarCollection.Active = calendar;
```

Code-Beispiel VB.NET

```
'Darstellung des neuen Kalenders
vcGantt1.TimeScaleEnd  = New DateTime(2012, 1, 1)
vcGantt1.TimeScaleStart = New DateTime(2011, 1, 1)
Dim calendar As VcCalendar =
vcGantt1.CalendarCollection.Add("CompanyCalendar1")
vcGantt1.CalendarCollection.Active = calendar
```



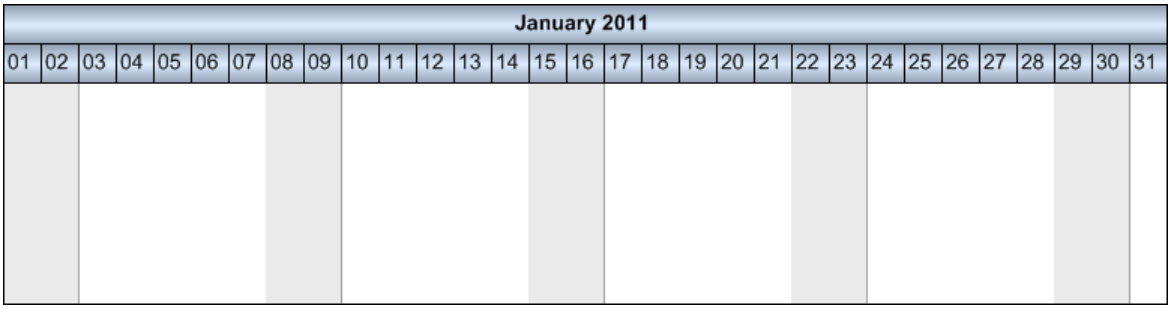
Falls der bereits standardmäßig vorhandene Kalender wieder wirksam werden soll, geschieht dies durch folgende Zuweisung:

Code-Beispiel C#

```
// Darstellung des Standardkalenders
VcCalendar calendar =
vcGantt1.CalendarCollection.CalendarByName("BaseCalendar");
vcGantt1.CalendarCollection.Active = calendar;
```

Code-Beispiel VB.NET

```
' Darstellung des Standardkalenders
Dim calendar As VcCalendar =
vcGantt1.CalendarCollection.CalendarByName("BaseCalendar")
vcGantt1.CalendarCollection.Active = calendar
```



Im nächsten Beispiel wird gezeigt, wie für einen neuen Kalender das Arbeitszeitprofil durch Angabe von **Intervallen** festgelegt wird. Der 1. Januar 2011 sowie die Tage im Zeitraum vom 6.01.2011 bis 20.01.2011 sollen mit Ausnahme der beiden Tage 10.01.2011 und 11.01.2011 als Nichtarbeitszeit definiert werden:

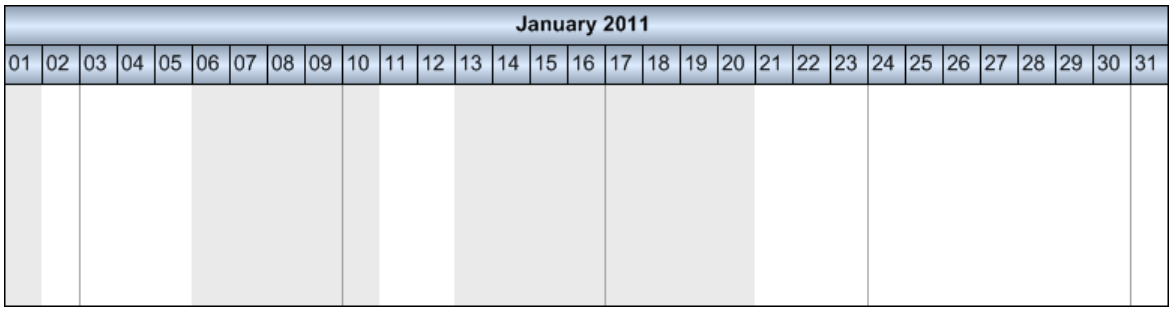
Code-Beispiel C#

```
// Nichtarbeitszeiten definieren
vcGantt1.TimeScaleEnd  = new DateTime(2012, 1, 1);
vcGantt1.TimeScaleStart = new DateTime(2011, 1, 1);
VcCalendar calendar =
vcGantt1.CalendarCollection.Add("CompanyCalendar1");
vcGantt1.CalendarCollection.Active = calendar;

VcInterval interval = calendar.IntervalCollection.Add("NewYear");
interval.CalendarProfileName = "<NONWORK>";
interval.StartDateTime = new DateTime(2011, 1, 1);
interval.EndDateTime = new DateTime(2011, 1, 2);

interval = calendar.IntervalCollection.Add("NonworkPeriod");
interval.CalendarProfileName = "<NONWORK>";
interval.StartDateTime = new DateTime(2011, 1, 6);
interval.EndDateTime = new DateTime(2011, 1, 21);

interval = calendar.IntervalCollection.Add("WorkPeriod");
interval.CalendarProfileName = "<WORK>";
interval.StartDateTime = new DateTime(2011, 1, 11);
interval.EndDateTime = new DateTime(2011, 1, 13);
vcGantt1.CalendarCollection.Update();
```

Code-Beispiel VB.NET

```
' Nichtarbeitszeiten definieren
vcGantt1.TimeScaleEnd  = New DateTime(2012, 1, 1)
vcGantt1.TimeScaleStart = New DateTime(2011, 1, 1)
Dim calendar As VcCalendar =
vcGantt1.CalendarCollection.Add("CompanyCalendar1")
vcGantt1.CalendarCollection.Active = calendar

Dim interval As VcInterval = calendar.IntervalCollection.Add("NewYear")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = New DateTime(2011, 1, 1)
interval.EndDateTime = New DateTime(2011, 1, 2)

interval = calendar.IntervalCollection.Add("NonworkPeriod")
interval.CalendarProfileName = "<NONWORK>"
```

```
interval.StartDateTime = New DateTime(2011, 1, 6)
interval.EndDateTime = New DateTime(2011, 1, 21)

interval = calendar.IntervalCollection.Add("WorkPeriod")
interval.CalendarProfileName = "<work>"
interval.StartDateTime = New DateTime(2011, 1, 11)
interval.EndDateTime = New DateTime(2011, 1, 13)
vcGantt1.CalendarCollection.Update()
```



Optisch sind die Nichtarbeitszeiten an den Flächen mit der hellgrauen Farbe zu erkennen. Da Arbeitszeiten keine standardmäßige Einfärbung besitzen, bleibt der weiße Hintergrund des Diagramms sichtbar. In nächsten Schritt sollen die Arbeitszeiten hellgelb und die arbeitsfreien Zeiten hellblau eingefärbt werden. Dies bewirken grafische Attribute, die bei den Intervallen angegeben werden können.

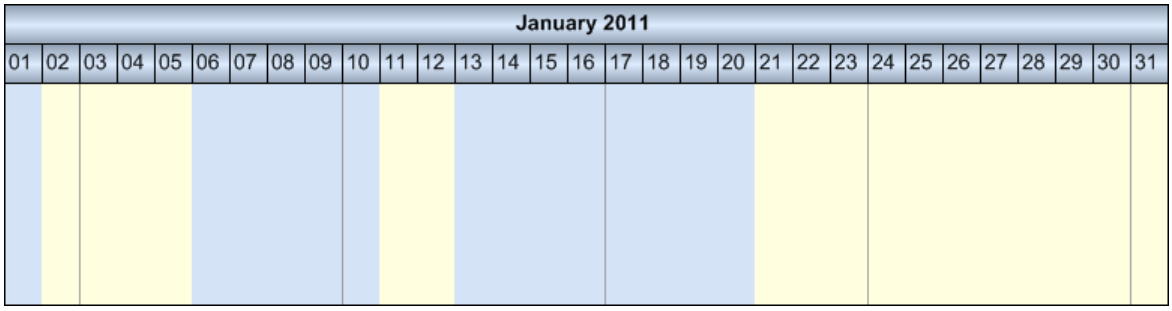
Code-Beispiel C#

```
// Einfärben von Intervallen
vcGantt1.TimeScaleEnd  = new DateTime(2012, 1, 1);
vcGantt1.TimeScaleStart = new DateTime(2011, 1, 1);
VcCalendar calendar =
vcGantt1.CalendarCollection.Add("CompanyCalendar1");
vcGantt1.CalendarCollection.Active = calendar;
vcGantt1.TimeScaleCollection.FirstTimeScale().get_Section(0).
get_CalendarGrid(0).UseGraphicalAttributesOfIntervals = true;

VcInterval interval = calendar.IntervalCollection.Add("Work");
interval.CalendarProfileName = "<WORK>";
interval.BackgroundColor = Color.LightYellow;
interval.UseGraphicalAttributes = true;

VcInterval interval = calendar.IntervalCollection.Add("NewYear");
interval.CalendarProfileName = "<NONWORK>";
interval.StartDateTime = new DateTime(2011, 1, 1);
interval.EndDateTime = new DateTime(2011, 1, 2);
interval.BackgroundColor = Color.FromArgb(212,227,245);
interval.UseGraphicalAttributes = true;

interval = calendar.IntervalCollection.Add("NonworkPeriod");
interval.CalendarProfileName = "<NONWORK>";
interval.StartDateTime = new DateTime(2011, 1, 6);
interval.EndDateTime = new DateTime(2011, 1, 21);
interval.BackgroundColor = Color.FromArgb(212,227,245);
interval.UseGraphicalAttributes = true;

interval = calendar.IntervalCollection.Add("WorkPeriod");
```

```
interval.CalendarProfileName = "<WORK>";
interval.StartDateTime = new DateTime(2011, 1, 11);
interval.EndDateTime = new DateTime(2011, 1, 13);
interval.BackgroundColor = Color.LightYellow;
interval.UseGraphicalAttributes = true;

vcGantt1.CalendarCollection.Update();
```

Code-Beispiel VB.NET

```
' Einfärben von Intervallen
vcGantt1.TimeScaleEnd  = New DateTime(2012, 1, 1)
vcGantt1.TimeScaleStart = New DateTime(2011, 1, 1)
Dim calendar As VcCalendar =
vcGantt1.CalendarCollection.Add("CompanyCalendar1")
vcGantt1.CalendarCollection.Active = calendar
Dim get_CalendarGrid(0).UseGraphicalAttributesOfIntervals As
vcGantt1.TimeScaleCollection.FirstTimeScale().get_Section(0). = True

Dim interval As VcInterval = calendar.IntervalCollection.Add("Work")
interval.CalendarProfileName = "<WORK>"
interval.BackgroundColor = Color.LightYellow
interval.UseGraphicalAttributes = True

interval = calendar.IntervalCollection.Add("NewYear")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = New DateTime(2011, 1, 1)
interval.EndDateTime = New DateTime(2011, 1, 2)
interval.BackgroundColor = Color.FromArgb(212,227,245)
interval.UseGraphicalAttributes = True

interval = calendar.IntervalCollection.Add("NonworkPeriod")
interval.CalendarProfileName = "<NONWORK>"
interval.StartDateTime = New DateTime(2011, 1, 6)
interval.EndDateTime = New DateTime(2011, 1, 21)
interval.BackgroundColor = Color.FromArgb(212,227,245)
interval.UseGraphicalAttributes = True

interval = calendar.IntervalCollection.Add("WorkPeriod")
interval.CalendarProfileName = "<WORK>"
interval.StartDateTime = New DateTime(2011, 1, 11)
interval.EndDateTime = New DateTime(2011, 1, 13)
interval.BackgroundColor = Color.LightYellow
interval.UseGraphicalAttributes = True

vcGantt1.CalendarCollection.Update()
```



Das folgende Beispiel zeigt, wie eine Woche mit Arbeitszeit von Montag bis Freitag und freien Wochenenden vereinbart wird. Die bisher vorgestellten

Möglichkeiten reichen dazu noch nicht aus; es muss ein Objekt vom Typ **VcCalendarProfile** zur Hilfe genommen werden.

Hinweis: Objekte vom Typ VcCalendarProfile können in VARCHART XGantt auf globaler oder lokaler Ebene angelegt werden. Lokale CalendarProfile-Objekte sind nur in dem Kalender verwendbar, in dem sie definiert worden sind, während die globalen Objekte gleichzeitig in mehreren Kalendern eingesetzt werden können. In den Beispielen werden ausschließlich lokale CalendarProfile-Objekte verwendet. Funktionell bestehen keine Unterschiede zu den globalen Objekten. Wenn unter dem gleichen Namen sowohl ein globales als auch lokales Profil angelegt wurde, kann in dem betreffenden Kalender nur auf das lokale Profile zugegriffen werden; das globale Profil ist unzugänglich.

Mit einem **CalendarProfile** vom **Typ vcWeekProfile** werden die Arbeitszeiten und Nichtarbeitszeiten der Wochentage beschrieben. Das Wochenprofil wird erst wirksam, wenn es der Intervallauflistung (IntervalCollection) des Kalenders hinzugefügt wird. Die Angabe von **StartDateTime** und **EndDateTime** beim Kalenderintervall kann entfallen, weil die Setzung ohne zeitliche Einschränkung für den gesamten Kalenderzeitraum gelten soll. Die Kalenderprofile mit den vordefinierten Namen **<WORK>** und **<NONWORK>** haben eine feste Bedeutung. Sie werden dazu benutzt, um Arbeitszeit und Nichtarbeitszeit zuordnen zu können.

Code-Beispiel C#

```
// Definition eines Wochenprofils
VcCalendarProfile calendarProfile =
calendar.CalendarProfileCollection.Add("WeekProfile");
calendarProfile.Type = VcCalendarProfileType.vcWeekProfile;

VcInterval interval = calendarProfile.IntervalCollection.Add("Mo-Fr");
interval.CalendarProfileName = "<WORK>";
interval.StartWeekday = VcWeekday.vcMonday;
interval.EndWeekday = VcWeekday.vcFriday;

interval = calendarProfile.IntervalCollection.Add("Sa");
interval.CalendarProfileName = "<NONWORK>";
interval.BackgroundColor = Color.FromArgb(255, 246, 159);
interval.StartWeekday = VcWeekday.vcSaturday;
interval.EndWeekday = VcWeekday.vcSaturday;

interval = calendarProfile.IntervalCollection.Add("So");
interval.CalendarProfileName = "<NONWORK>";

interval.BackgroundColor = Color.FromArgb(251, 211, 170);
interval.StartWeekday = VcWeekday.vcSunday;
interval.EndWeekday = VcWeekday.vcSunday;
interval = calendar.IntervalCollection.Add("StandardWeek");
interval.CalendarProfileName = "WeekProfile";
```

Code-Beispiel VB.NET

```
' Definition eines Wochenprofils
dim calendarProfile as VcCalendarProfile
Set calendar.Profile =
VcGantt1.CalendarProfileCollection.Add("WeekProfile")
calendarProfile.Type = VcCalendarProfileType.vcWeekProfile

VcInterval interval = calendarProfile.IntervalCollection.Add("Mo-Fr")
interval.CalendarProfileName = "<WORK>"
interval.StartWeekday = VcWeekday.vcMonday
interval.EndWeekday = VcWeekday.vcFriday

interval = calendarProfile.IntervalCollection.Add("Sa")
interval.CalendarProfileName = "<NONWORK>"
interval.BackgroundColor = Color.FromArgb(255, 246, 159)
interval.StartWeekday = VcWeekday.vcSaturday
interval.EndWeekday = VcWeekday.vcSaturday

interval = calendarProfile.IntervalCollection.Add("So")
interval.CalendarProfileName = "<NONWORK>"
interval.BackgroundColor = Color.FromArgb(251, 211, 170)
interval.StartWeekday = VcWeekday.vcSunday
interval.EndWeekday = VcWeekday.vcSunday

interval = calendar.IntervalCollection.Add("StandardWeek")
interval.CalendarProfileName = "WeekProfile"
```

Wenn Arbeitszeiten von Nichtarbeitszeiten innerhalb eines Tages unterschieden werden sollen, dann wird zusätzlich ein Tagesprofil benötigt, in dem die exakten Arbeitszeitintervalle angegeben werden: z.B. von 8.00 Uhr bis 12.00 Uhr und von 13.00 Uhr bis 17.00 Uhr. Da ein neu angelegtes Tagesprofil nur aus Arbeitszeit besteht, müssen die Abweichungen davon als Nichtarbeitszeitintervalle angegeben werden.

Code-Beispiel C#

```
// Tagesprofil vereinbaren
VcCalenderProfile calendarProfile =
calendar.CalendarProfileCollection.Add("DayProfile");
calendarProfile.Type = VcCalendarProfileType.vcDayProfile;

VcInterval interval =
calendarProfile.IntervalCollection.Add("Interval_1");
// 00:00-8:00
interval.CalendarProfileName = "<NONWORK>";
interval.StartTime = new DateTime(2011, 1, 1, 0, 0, 0);
interval.EndTime = new DateTime(2011, 1, 1, 8, 0, 0);

interval = calendarProfile.IntervalCollection.Add("Interval_2");
// 12:00-13:00
interval.CalendarProfileName = "<NONWORK>";
interval.StartTime = new DateTime(2011, 1, 1, 12, 0, 0);
interval.EndTime = new DateTime(2011, 1, 1, 13, 0, 0);

interval = calendarProfile.IntervalCollection.Add("Interval_3");
// 17:00-24:00
interval.CalendarProfileName = "<NONWORK>";
```

```
interval.StartTime = new DateTime(2011, 1, 1, 17, 0, 0);
interval.EndTime = new DateTime(2011, 1, 1, 0, 0, 0);
```

Code-Beispiel VB.NET

```
' Tagesprofil vereinbaren
Dim calendarProfile As VcCalenderProfile =
calendar.CalendarProfileCollection.Add("DayProfile")
calendarProfile.Type = VcCalendarProfileType.vcDayProfile

Dim interval As VcInterval =
calendarProfile.IntervalCollection.Add("Interval_1")
' 00:00-8:00
interval.CalendarProfileName = "<NONWORK>"
interval.StartTime = New DateTime(2011, 1, 1, 0, 0, 0)
interval.EndTime = New DateTime(2011, 1, 1, 8, 0, 0)

interval = calendarProfile.IntervalCollection.Add("Interval_2")
' 12:00-13:00
interval.CalendarProfileName = "<NONWORK>"
interval.StartTime = New DateTime(2011, 1, 1, 12, 0, 0)
interval.EndTime = New DateTime(2011, 1, 1, 13, 0, 0)

interval = calendarProfile.IntervalCollection.Add("Interval_3")
' 17:00-24:00
interval.CalendarProfileName = "<NONWORK>"
interval.StartTime = New DateTime(2011, 1, 1, 17, 0, 0)
interval.EndTime = New DateTime(2011, 1, 1, 0, 0, 0)
```

Die Angabe der Uhrzeit erfolgt durch das Objekt **DateTime**, bei dem der Datumsanteil ignoriert wird, weil er in diesem Zusammenhang ohne Bedeutung ist. Das Datum muss nur mit angegeben werden, damit alle notwendigen Parameter im Aufruf des Konstruktors belegt sind. Im **Interval_3** ist zu beachten, dass statt 24 Uhr, was im Objekt **DateTime** als Stundenangabe nicht akzeptiert wird, 0 Uhr zu spezifizieren ist.

Jährlich wiederkehrende Tage wie zum Beispiel **Neujahr** am 1.1 oder **Weihnachten** am 25.12. und 26.12. werden über ein Jahreskalenderprofil festgelegt.

Code-Beispiel C#

```
// Profil der festen Feiertage vereinbaren
VcCalenderProfile calendarProfile =
calendar.CalendarProfileCollection.Add("YearProfile");
calendarProfile.Type = VcCalendarProfileType.vcYearProfile;

VcInterval interval = calendarProfile.IntervalCollection.Add("New
Year");
interval.CalendarProfileName = "<NONWORK>";
interval.DayInStartMonth = 1 ;
interval.StartMonth = VcMonth.vcJanuary;
interval.DayInEndMonth = 1;
interval.EndMonth = VcMonth.vcJanuary;
SetAppearanceForHolidays(interval);

interval = calendarProfile.IntervalCollection.Add("Chrismas");
```

```
interval.CalendarProfileName = "<NONWORK>";
interval.DayInStartMonth = 25 ;
interval.StartMonth = VcMonth.vcDecember;
interval.DayInEndMonth = 26;
interval.EndMonth = VcMonth.vcDecember;
SetAppearanceForHolidays(interval);
```

Code-Beispiel VB.NET

```
' Profil der festen Feiertage vereinbaren
Dim calendarProfile As VcCalenderProfile =
calendar.CalendarProfileCollection.Add("YearProfile")
calendarProfile.Type = VcCalendarProfileType.vcYearProfile

Dim interval As VcInterval = calendarProfile.IntervalCollection.Add("New
Year")
interval.CalendarProfileName = "<NONWORK>"
interval.DayInStartMonth = 1
interval.StartMonth = VcMonth.vcJanuary
interval.DayInEndMonth = 1
interval.EndMonth = VcMonth.vcJanuary
SetAppearanceForHolidays(interval)

interval = calendarProfile.IntervalCollection.Add("Christmas")
interval.CalendarProfileName = "<NONWORK>"
interval.DayInStartMonth = 25
interval.StartMonth = VcMonth.vcDecember
interval.DayInEndMonth = 26
interval.EndMonth = VcMonth.vcDecember
SetAppearanceForHolidays(interval)
```

Um zu vermeiden, dass die für das gleichartige Aussehen der Feiertage notwendigen Setzungen wiederholt im Programmcode aufgeführt werden, wurden die Anweisungen in einer Methode mit dem Namen **SetAppearance-ForHolidays** gebündelt:

Code-Beispiel C#

```
// Methode zur Festlegung der visuellen Darstellung der Feiertage
void SetAppearanceForHolidays(VcInterval interval)
{
   interval.BackgroundColor = Color.FromArgb(255, 255, 164, 164);
   interval.Pattern = VcFillPattern.vcWeavePattern;
   interval.PatternColor = Color.FromArgb(255, 64, 64, 64);
   interval.LineColor = Color.FromArgb(255, 128, 128, 128);
   interval.LineThickness = 1;
   interval.LineType = VcLineType.vcSolid;
   interval.UseGraphicalAttributes = true;
}
```

Code-Beispiel VB.NET

```
' Methode zur Festlegung der visuellen Darstellung der Feiertage
Private  Sub SetAppearanceForHolidays(ByVal interval As VcInterval)
   interval.BackgroundColor = Color.FromArgb(255, 255, 164, 164)
   interval.Pattern = VcFillPattern.vcWeavePattern
   interval.PatternColor = Color.FromArgb(255, 64, 64, 64)
   interval.LineColor = Color.FromArgb(255, 128, 128, 128)
```

```
   interval.LineThickness = 1
   interval.LineType = VcLineType.vcSolid
   interval.UseGraphicalAttributes = True
End Sub
}
```

Hinweis: Die Eigenschaften zur Farbgestaltung werden nur bei den Intervallen wirksam, deren CalendarProfileName auf eine der beiden vordefinierten Begriffe **<WORK>** und **<NONWORK>** gesetzt ist. Zudem muss beim Intervall die Eigenschaft **UseGraphicalAttribute** auf **true** und beim Objekt **CalendarGrid** die Eigenschaft **UseGraphicalAttributesOfIntervals auf true** gesetzt sein.

Bewegliche Feiertage wie beispielsweise Ostern und davon abhängige Feiertage müssen für jedes Jahr einzeln berechnet werden und dem Kalender als feste Termine zugewiesen werden. Dazu eignet sich folgende Unterstützungsmethode:

Code-Beispiel C#

```
// Methode zur Berechnung der beweglichen Feiertage
 public enum Anniversary
 {
    AshWednesday,
    GoodFriday,
    EasterSunday,
    EasterMonday,
    FeastOfCorpusChristi,
    AscensionOfChrist,
    WhitSunday,
    WhitMonday,
    CentralEuropeanSummerTimeStart,
    CentralEuropeanSummerTimeEnd
 }

private DateTime calculateAnniversaryForYear(int year, Anniversary
specialDay)
{
   int g = year % 19;
   int c = year / 100;
   int h = (c - c / 4 - (8 * c + 13) / 25 + 19 * g + 15) % 30;
   int i = h - (h / 28) * (1 - (29 / (h + 1)) * ((21 - g) / 11));
   int j = (year + year / 4 + i + 2 - c + c / 4) % 7;
   int month = 3 + (i - j + 40) / 44;
   int day = i - j + 28 - 31 * (month / 4);

   int dayOffset = 0;
   switch (specialDay)
   {
      case Anniversary.AshWednesday:
         dayOffset = -40;
         break;
      case Anniversary.GoodFriday:
         dayOffset = -2;
         break;
      case Anniversary.EasterSunday:
```

```
         break;
      case Anniversary.EasterMonday:
         dayOffset = 1;
         break;
      case Anniversary.AscensionOfChrist:
         dayOffset = 39;
         break;
      case Anniversary.WhitSunday:
         dayOffset = 49;
         break;
      case Anniversary.WhitMonday:
         dayOffset = 50;
         break;
      case Anniversary.FeastOfCorpusChristi:
         dayOffset = 60;
         break;
      case Anniversary.CentralEuropeanSummerTimeStart:
         month = 3;
         day = 31 - Convert.ToInt32(new DateTime(year, 3,
31).DayOfWeek);
         break;
      case Anniversary.CentralEuropeanSummerTimeEnd:
         month = 10;
         day = 31 - Convert.ToInt32(new DateTime(year, 10,
31).DayOfWeek);
         break;
         }
      return new DateTime(year, month, day).AddDays(dayOffset);
  }
```

Code-Beispiel VB.NET

```
' Methode zur Berechnung der beweglichen Feiertage
 Public Enum Anniversary
    AshWednesday
    GoodFriday
    EasterSunday
    EasterMonday
    FeastOfCorpusChristi
    AscensionOfChrist
    WhitSunday
    WhitMonday
    CentralEuropeanSummerTimeStart
    CentralEuropeanSummerTimeEnd
End Enum

Private Function calculateAnniversaryForYear(ByVal year As Integer,
ByVal specialDay As Anniversary) As DateTime
   Dim g As Integer = Decimal.Remainder(  year ,  19 )
   Dim c As Integer =  year / 100
   Dim h As Integer = (c - c / 4 -(8 * c + 13) / 25 + 19 * g + 15) % 30
   Dim i As Integer =  h -(h / 28) *(1 -(29 /(h + 1)) *((21 - g) / 11))
   Dim j As Integer = (year + year / 4 + i + 2 - c + c / 4) % 7
   Dim month As Integer =  3 +(i - j + 40) / 44
   Dim day As Integer =  i - j + 28 - 31 *(month / 4)

   Dim dayOffset As Integer =  0
   Select Case specialDay
      Case Anniversary.AshWednesday
         dayOffset = -40
```

```
        Case Anniversary.GoodFriday
          dayOffset = -2
         Case Anniversary.EasterSunday
          Exit Function
       Case Anniversary.EasterMonday
          dayOffset = 1
       Case Anniversary.AscensionOfChrist
          dayOffset = 39
       Case Anniversary.WhitSunday
          dayOffset = 49
       Case Anniversary.WhitMonday
          dayOffset = 50
       Case Anniversary.FeastOfCorpusChristi
          dayOffset = 60
       Case Anniversary.CentralEuropeanSummerTimeStart
          month = 3
          day = 31 - Convert.ToInt32(New DateTime(year, 3, 31).DayOfWeek)
       Case Anniversary.CentralEuropeanSummerTimeEnd
          month = 10
          day = 31 - Convert.ToInt32(New DateTime(year, 10,
31).DayOfWeek)
   End Select
      Return New DateTime(year, month, day).AddDays(dayOffset)
End Function
```

In nächsten Schritt werden das Wochenprofil und das Jahrestageprofil dem Kalender als Intervalle hinzugefügt. Danach werden die beweglichen Feiertage berechnet und ebenso dem Kalender als Intervalle zugewiesen:

Code-Beispiel C#

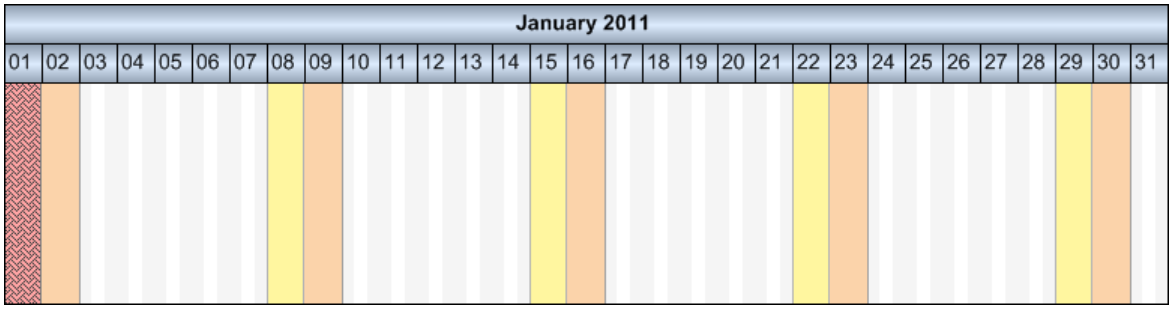
```
// Wochenprofil, Jahrestageprofil und bewegliche Feiertage
zusammenführen
interval = calendar.IntervalCollection.Add("Weekly_Pattern");
interval.CalendarProfileName = "WeekProfile";

interval = calendar.IntervalCollection.Add("Yearly_Pattern");
interval.CalendarProfileName = "YearProfile";

int startYear = vcGantt1.TimeScaleStart.Year;
int endYear   = vcGantt1.TimeScaleEnd.Year;

for (int i=startYear; i<=endYear; i++)
{
   interval = calendar.IntervalCollection.Add("GoodFriday_" +
i.ToString());
   interval.CalendarProfileName = "<NONWORK>";
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.GoodFriday);
   interval.EndDateTime = interval.StartDateTime;
   SetAppearanceForHolidays(interval);

   interval = calendar.IntervalCollection.Add("EasterMonday_" +
i.ToString());
   interval.CalendarProfileName = "<NONWORK>";
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.EasterMonday);
   interval.EndDateTime = interval.StartDateTime;
   SetAppearanceForHolidays(interval);
```

```
   interval = calendar.IntervalCollection.Add("FeastOfCorpusChristi_" +
i.ToString());
   interval.CalendarProfileName = "<NONWORK>";
   interval.StartDateTime = calculateAnniversaryForYear
(i, Anniversary.FeastOfCorpusChristi);
   interval.EndDateTime = interval.StartDateTime;
   SetAppearanceForHolidays(interval);

   interval = calendar.IntervalCollection.Add("AscensionOfChrist_" +
i.ToString());
   interval.CalendarProfileName = "<NONWORK>";
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.AscensionOfChrist);
   interval.EndDateTime = interval.StartDateTime;
   SetAppearanceForHolidays(interval);

   interval = calendar.IntervalCollection.Add("WhitMonday_" +
i.ToString());
   interval.CalendarProfileName = "<NONWORK>";
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.WhitMonday);
   interval.EndDateTime = interval.StartDateTime;
   SetAppearanceForHolidays(interval);
}

vcGantt1.CalendarCollection.Update();
```

Code-Beispiel VB.NET

```
' Wochenprofil, Jahrestageprofil und bewegliche Feiertage zusammenführen
interval = calendar.IntervalCollection.Add("Weekly_Pattern")
interval.CalendarProfileName = "WeekProfile"

interval = calendar.IntervalCollection.Add("Yearly_Pattern")
interval.CalendarProfileName = "YearProfile"

Dim startYear As Integer =  vcGantt1.TimeScaleStart.Year
Dim endYear As Integer =  vcGantt1.TimeScaleEnd.Year

Dim i As Integer
For  i = startYear To endYear Step  i + 1
   interval = calendar.IntervalCollection.Add("GoodFriday_" +
i.ToString())
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.GoodFriday)
   interval.EndDateTime = interval.StartDateTime
   SetAppearanceForHolidays(interval)

   interval = calendar.IntervalCollection.Add("EasterMonday_" +
i.ToString())
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.EasterMonday)
   interval.EndDateTime = interval.StartDateTime
   SetAppearanceForHolidays(interval)

   interval = calendar.IntervalCollection.Add("FeastOfCorpusChristi_" +
i.ToString())
```

```
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear
(i, Anniversary.FeastOfCorpusChristi)
   interval.EndDateTime = interval.StartDateTime
   SetAppearanceForHolidays(interval)

   interval = calendar.IntervalCollection.Add("AscensionOfChrist_" +
i.ToString())
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.AscensionOfChrist)
   interval.EndDateTime = interval.StartDateTime
   SetAppearanceForHolidays(interval)

   interval = calendar.IntervalCollection.Add("WhitMonday_" +
i.ToString())
   interval.CalendarProfileName = "<NONWORK>"
   interval.StartDateTime = calculateAnniversaryForYear(i,
Anniversary.WhitMonday)
   interval.EndDateTime = interval.StartDateTime
   SetAppearanceForHolidays(interval)
Next

vcGantt1.CalendarCollection.Update()
```



Zusammenfassend noch einmal ein Überblick über die Schrittfolge, die nötig ist, um einen Kalender aufzubauen. Je nach Situation können gegebenenfalls einzelne Schritte entfallen:

1. Tagesprofile für unterschiedliche Arbeitstage anlegen
2. Wochenprofil unter Verwendung der Tagesprofile aufbauen
3. Jahrestageprofil festlegen
4. Wochenprofil und Jahrestageprofil der IntervalCollection des Kalenders zuordnen
5. Feste Termine der IntervalCollection des Kalenders zuweisen

Das Intervallobjekt ermöglicht es, Zeitintervalle zu definieren, die als Arbeitszeit oder Nichtarbeitszeit 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 Profile beziehen und dessen Einstellungen überneh-

men. Bei der Angabe der Eigenschaft ist zu beachten, dass abhängig vom Intervalltyp nur bestimmte Profiltypen zugeordnet werden können. Der Intervalltyp ist stets implizit durch den gewählten Profiltyp gegeben. Der vorgegebene Standardwert **vcDayProfile** für **VcCalendarProfile** kann zu Beginn, das heißt, bevor einzelne Intervalle definiert werden, durch eine entsprechende Zuweisung geändert werden.

Objekt	Gewählter Profiltyp	Zugeordneter Intervalltyp
VcCalendar		vcCalendarInterval
VcCalendarProfile	vcYearProfile	vcYearProfileInterval
	vcWeekProfile	vcWeekProfileInterval
	vcDayProfile	vcDayProfileInterval
	vcVariableProfile	vcVariableProfileInterval

Der Intervalltyp gibt vor, welche Art von Beschreibungsintervallen möglich sind. Zum Beispiel besitzt ein Tagesprofil immer Intervalle vom Typ **vcDayProfileInterval**.

Intervall-Typ	<WORK>	<NONWORK>	vcDayProfile	vcWeekProfile	vcYearProfile	vcVariableProfile
vcCalendarInterval	■	■	■	■	■	■
vcVariableProfileInterval	■	■	■	■	■	
vcYearProfileInterval	■	■	■	■		
vcWeekProfileInterval	■	■	■			
vcDayProfileInterval	■	■				

Bei den Kalenderprofilen wird unterschieden zwischen **Tagesprofilen, Wochenprofilen, Jahresprofilen** und **variablen Profilen**. In einem Tagesprofil können ausschließlich Zeitintervalle durch Uhrzeiten definiert werden, die innerhalb der Grenzen eines Tages liegen. In einem Wochenprofil wird bestimmt, welche Tagesprofile an welchen Wochentagen zum Einsatz kommen. Ein Jahresprofil ordnet einzelnen oder mehreren zusammenhängenden Tagen ausgewählte Tagesprofile zu, die jährlich wiederkehrend zum gleichen Zeitpunkt (z.B. Neujahr) angewendet werden. Das variable Profil kann eine Abfolge von unterschiedlich gekennzeichneten, hintereinander liegenden Arbeitszeiten aufnehmen. In Abhängigkeit vom vorliegenden Intervalltyp

vcCalendarInterval, vcDayProfileInterval, vcWeekProfileInterval, vcYearProfileInterval, vcVariableProfileInterval sind jeweils nur bestimmte Eigenschaften des Objektes relevant. Die nachfolgende Tabelle gibt Aufschluss über den Zusammenhang:

vcCalendar-Interval	vcYearProfile-Interval	vcWeekProfile-Interval	vcDayProfile-Interval	vcVariable-Interval
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 **DayProfileInterval** bezieht sich auf die Angabe von Zeiten innerhalb eines Tages. Beispiel: 8.00 bis 17.00 Uhr.

Ein **VariableProfile** beschreibt eine Zeitspanne, ohne sich dabei auf einen bestimmten Termin zu beziehen. Die Einheit der Dauer, ob Tage, Stunden, Minuten oder Sekunden wird in der Eigenschaft TimeUnit des Intervall-Objekts angegeben. Beispiel: 4 Stunden.

Kalenderberechnungen durchführen

Die Berechnungen im Kalender sind nicht an den sichtbaren Bereich der Zeitskala gebunden. Die Methode **AddDuration** des Objekts **Calendar** berechnet aus einem Starttermin und der angegebenen Anzahl von Arbeitszeiteinheiten den Endtermin, der sich aus dem Überspringen der arbeitsfreien Zeiten ergibt. Negativ übergebene Zeiteinheiten bewirken, dass zu einem spezifizierten Endtermin der passende Anfangstermin berechnet wird. Die zu AddDuration komplementäre Methode **CalcDuration** kann wiederum aus einer Zeitspanne, die durch Start- und Endtermin gegeben ist, die Dauer der dazwischen liegende Arbeitszeiteinheiten ermitteln.

> Wirkung der Berechnungsmethoden

Hinweis: Beachten Sie bitte, dass die Angabe der Arbeitszeiteinheiten in der Einheit Tag, Stunde, Minute oder Sekunde erfolgen muss, die aktuell in der Eigenschaft TimeUnit des Objekts VcGantt eingestellt ist.

Die Methode **AddDuration** gewährleistet, dass die berechneten Termine stets innerhalb eines Arbeitszeitintervalls liegen. Dies bedeutet gleichzeitig jedoch auch, dass eine Berechnung nicht eindeutig umkehrbar ist, wenn der Ausgangstermin sich ursprünglich innerhalb der arbeitsfreien Zeit befunden hat.

> Begrenzte Umkehrbarkeit der Berechnung

Beim interaktiven Anlegen und Ändern von Vorgängen sorgt VARCHART XGantt automatisch dafür, dass Vorgänge nicht innerhalb arbeitsfreier Zeitspannen beginnen oder enden können. Wenn Sie erreichen möchten, dass das Verhalten beim Erzeugen oder Ändern von Vorgängen über die API zur interaktiven Arbeitsweise konsistent ist, müssen Sie dies durch die Korrektur des Start- bzw. Enddatums selbst sicherstellen. Sie verlegen den Starttermin, der sich innerhalb einer arbeitsfreien Zeit befindet, auf den Beginn des nächsten Arbeitszeitintervalls und den Endtermin entsprechend auf das Ende des vorherigen Arbeitszeitintervalls. Die Methoden zur Bestimmung der Zeitintervallgrenzen ermöglichen dies auf einfache Weise und werden im nächsten Abschnitt im Detail besprochen.

Code-Beispiel C#

```
if (calendar.IsWorkTime(startDate) == false)
   startDate = calendar.GetNextIntervalBorder(startDate);

if (calendar.IsWorkTime(endDate) == false)
   endDate = calendar.GetStartOfInterval(endDate);
```

Code-Beispiel VB.NET

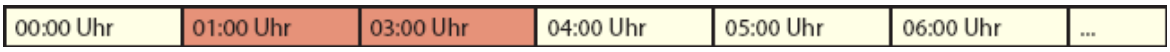
```
If calendar.IsWorkTime(startDate) = False Then
   startDate = calendar.GetNextIntervalBorder(startDate)
End If

If calendar.IsWorkTime(endDate) = False Then
   endDate = calendar.GetStartOfInterval(endDate)
End If
```

> Sommerzeit

VARCHART XGantt unterstützt automatisch die Sommerzeit. Die Sommerzeit beginnt in Mitteleuropa am letzten Sonntag im März und endet am letzten Sonntag im Oktober. Zu Beginn der Sommerzeit werden die Uhren

von 2.00 Uhr auf 3.00 Uhr vorgestellt und am Ende der Sommerzeit von 3.00 Uhr auf 2.00 Uhr zurückgestellt.

Sommerzeitbeginn:



Sommerzeitende:

Die Methode **calcDuration** ermittelt für den Tag des Sommerzeitbeginns eine Zeitspanne von 23 Stunden und für den Tag des Sommerzeitendes eine Zeitspanne von 25 Stunden zurück, wenn Stunden als Einheit verwendet werden. Wenn **TimeUnit** auf Tage eingestellt ist, beträgt die Zeitspanne in beiden Fällen exakt einen Tag.

Zeitintervallgrenzen ermitteln

Die Methoden des Objekts **Calendar** zum Bestimmen der Zeitintervallgrenzen **GetStartOfInterval, GetNextIntervalBorder** und **GetPreviousIntervalBorder** ermöglichen es, über Arbeitszeitintervalle und Nichtarbeitszeitintervalle zu iterieren. Die Ergebnisse der Methoden werden relativ auf ein Referenzdatum bezogen, welches den Methoden als Übergabeparameter mitgegeben wird.

Mit der Methode **IsWorkTime** des Objekts **Calendar** lässt sich für ein beliebiges Datum feststellen, ob es innerhalb eines Arbeitszeitintervalls oder Nichtarbeitszeitintervalls liegt. Obwohl der Beginn eines neuen Intervalls zugleich das Ende des vorherigen Intervalls darstellt, gehört das Startdatum selbst zum neuen Intervall (rechtsoffenes Intervall).

Die beiden Komfortmethoden **GetEndOfPreviousWorkTime** und **GetStartOfNextWorkTime** stellen keine neuen Möglichkeiten bereit, sondern vereinfachen nur den Umgang mit Arbeitszeitintervallen.

Im folgenden Programmbeispiel werden die Zeitintervalle des Kalenders ausgelesen und in eine Datei geschrieben. Zudem wird die im Betrachtungszeitraum verfügbare Arbeitszeit ermittelt:

Code-Beispiel C#

```
void writeCalendarIntervalsToFile (string filename, VcCalendar calendar,
DateTime startDate, DateTime endDate,
bool listWorkIntervals, bool listNonWorkIntervals)
{
   TextWriter tw = new
StreamWriter(Path.GetDirectoryName(Application.ExecutablePath) +
filename);
```

```
   tw.WriteLine("Time Intervals of {0} between \r\n{1} - {2}\r\n",
calendar.Name, startDate, endDate);
   DateTime tmpStartDate = startDate;
   while (tmpStartDate < endDate)
   {
      DateTime nextStartDate =
calendar.GetNextIntervalBorder(tmpStartDate);
      if (tmpStartDate == nextStartDate)
         nextStartDate = endDate;
      if (nextStartDate > endDate)
         nextStartDate = endDate;
      if (calendar.IsWorktime(tmpStartDate))
         if (listWorkIntervals)
            tw.WriteLine("{0} - {1} WorkInterval", tmpStartDate,
nextStartDate);
      else
         if (listNonWorkIntervals)
            tw.WriteLine("{0} - {1} NonWorkInterval", tmpStartDate,
nextStartDate);
      tmpStartDate = nextStartDate;
   }
   int totalWorkTime = calendar.CalcDuration(startDate, endDate);
   tw.WriteLine("Total work time: {0} Units", totalWorkTime);
   tw.Close();
}
```

Code-Beispiel VB.NET

```
Private  Sub writeCalendarIntervalsToFile(ByVal filename As String,
ByVal calendar As VcCalendar, ByVal startDate As DateTime, ByVal endDate
As DateTime, ByVal listWorkIntervals As Boolean, ByVal
listNonWorkIntervals As Boolean)
   Dim tw As TextWriter =  New
StreamWriter(Path.GetDirectoryName(Application.ExecutablePath) +
filename)
   tw.WriteLine("Time Intervals of {0} between \r\n{1} - {2}\r\n",
calendar.Name, startDate, endDate)
   Dim tmpStartDate As DateTime =  startDate
   While tmpStartDate < endDate
      Dim nextStartDate As DateTime =
calendar.GetNextIntervalBorder(tmpStartDate)
      if (tmpStartDate = nextStartDate)
         nextStartDate = endDate
      if (nextStartDate > endDate)
         nextStartDate = endDate
      if (calendar.IsWorktime(tmpStartDate))
         if (listWorkIntervals)
            tw.WriteLine("{0} - {1} WorkInterval", tmpStartDate,
nextStartDate)
      else
         if (listNonWorkIntervals)
            tw.WriteLine("{0} - {1} NonWorkInterval", tmpStartDate,
nextStartDate)
      tmpStartDate = nextStartDate
   End While
   Dim totalWorkTime As Integer =
calendar.CalcDuration(startDate,endDate)
   tw.WriteLine("Total work time: {0} Units", totalWorkTime)
   tw.Close()
End Sub
```

Hinweis: Die Intervalle im Kalender können sekundengenau spezifiziert kann und einen Zeitbereich von maximal 137 Jahren (ulong in Sekunden) umfassen.

> Anweisung zum Schreiben der Intervalle in eine Datei

Code-Beispiel C#

```
writeCalendarIntervalsToFile("CalenderIntervals.txt", calendar,
vcGantt1.TimeScaleStart, vcGantt1.TimeScaleEnd, true, true);

Time Intervals of CompanyCalendar_1 between
01.01.2011 00:00:00 - 01.01.2012 00:00:00

01.01.2011 00:00:00 - 02.01.2011 00:00:00 non-work time
02.01.2011 00:00:00 - 03.01.2011 00:00:00 non-work time
03.01.2011 00:00:00 - 03.01.2011 08:00:00 non-work time
03.01.2011 08:00:00 - 03.01.2011 12:00:00 work time
03.01.2011 12:00:00 - 03.01.2011 13:00:00 non-work time
03.01.2011 13:00:00 - 03.01.2011 17:00:00 work time
03.01.2011 17:00:00 - 04.01.2011 00:00:00 non-work time
04.01.2011 00:00:00 - 04.01.2011 08:00:00 non-work time
04.01.2011 08:00:00 - 04.01.2011 12:00:00 work time
04.01.2011 12:00:00 - 04.01.2011 13:00:00 non-work time
04.01.2011 13:00:00 - 04.01.2011 17:00:00 work time
04.01.2011 17:00:00 - 05.01.2011 00:00:00 non-work time
...
30.12.2011 00:00:00 - 30.12.2011 08:00:00 non-work time
30.12.2011 08:00:00 - 30.12.2011 12:00:00 work time
30.12.2011 12:00:00 - 30.12.2011 13:00:00 non-work time
30.12.2011 13:00:00 - 30.12.2011 17:00:00 work time
30.12.2011 17:00:00 - 31.12.2011 00:00:00 non-work time
31.12.2011 00:00:00 - 01.01.2012 00:00:00 non-work time

Total work time: 2064 Units
```

> Anweisung zum Schreiben der Intervalle in eine Datei

Code-Beispiel VB.NET

```
writeCalendarIntervalsToFile("CalenderIntervals.txt", calendar,
vcGantt1.TimeScaleStart, vcGantt1.TimeScaleEnd, True, True)

Time Intervals of CompanyCalendar_1 between
01.01.2011 00:00:00 - 01.01.2012 00:00:00

01.01.2011 00:00:00 - 02.01.2011 00:00:00 non-work time
02.01.2011 00:00:00 - 03.01.2011 00:00:00 non-work time
03.01.2011 00:00:00 - 03.01.2011 08:00:00 non-work time
03.01.2011 08:00:00 - 03.01.2011 12:00:00 work time
03.01.2011 12:00:00 - 03.01.2011 13:00:00 non-work time
03.01.2011 13:00:00 - 03.01.2011 17:00:00 work time
03.01.2011 17:00:00 - 04.01.2011 00:00:00 non-work time
04.01.2011 00:00:00 - 04.01.2011 08:00:00 non-work time
04.01.2011 08:00:00 - 04.01.2011 12:00:00 work time
04.01.2011 12:00:00 - 04.01.2011 13:00:00 non-work time
04.01.2011 13:00:00 - 04.01.2011 17:00:00 work time
```

```
04.01.2011 17:00:00 - 05.01.2011 00:00:00 non-work time
...
30.12.2011 00:00:00 - 30.12.2011 08:00:00 non-work time
30.12.2011 08:00:00 - 30.12.2011 12:00:00 work time
30.12.2011 12:00:00 - 30.12.2011 13:00:00 non-work time
30.12.2011 13:00:00 - 30.12.2011 17:00:00 work time
30.12.2011 17:00:00 - 31.12.2011 00:00:00 non-work time
31.12.2011 00:00:00 - 01.01.2012 00:00:00 non-work time

Total work time: 2064 Units
```

3.5 Drag & Drop

Neben dem Verschieben oder Kopieren von Vorgängen innerhalb einer Instanz der VARCHART XGantt 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.

XGantt 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).

> VcGantt.LeavingControlWhileDraggingAllowed

Mit dieser Boole'schen Eigenschaft des VcGantt-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.

> VcGantt.NodeCreationAtDroppingEnabled

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

> VcGantt.PhantomDrawingWhileDraggingEnabled

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

> VcGantt.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. Verbotszeichen), oder gar anwendungseigene.

3.6 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

3.7 Filter

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

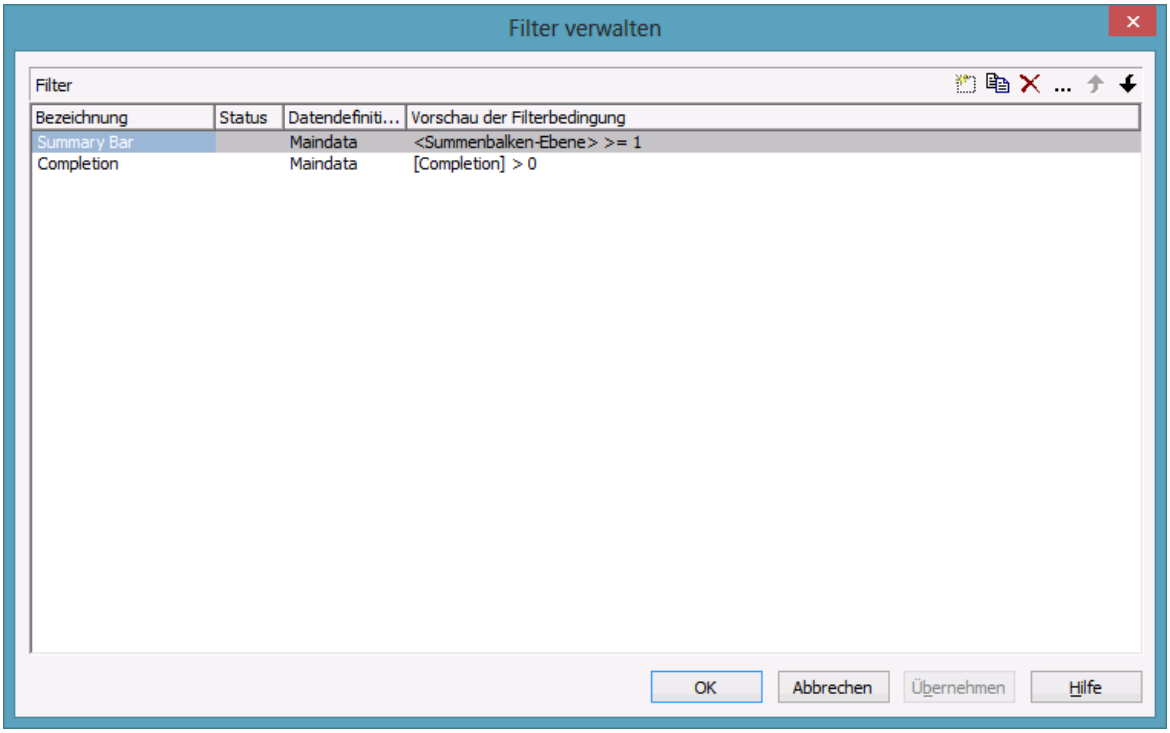
Wenn Sie einen Filter anwenden, werden die Informationen des Datensatzes eines Vorgangs, einer Histogrammkurve, einer Verbindung bzw. eines Tabellenformats mit den Filterkriterien verglichen und die Objekte ausgewählt, die die Filterkriterien erfüllen. Beispielsweise können Sie den Filter "Alle Vorgänge mit Beginn ab Januar 2012" definieren.

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

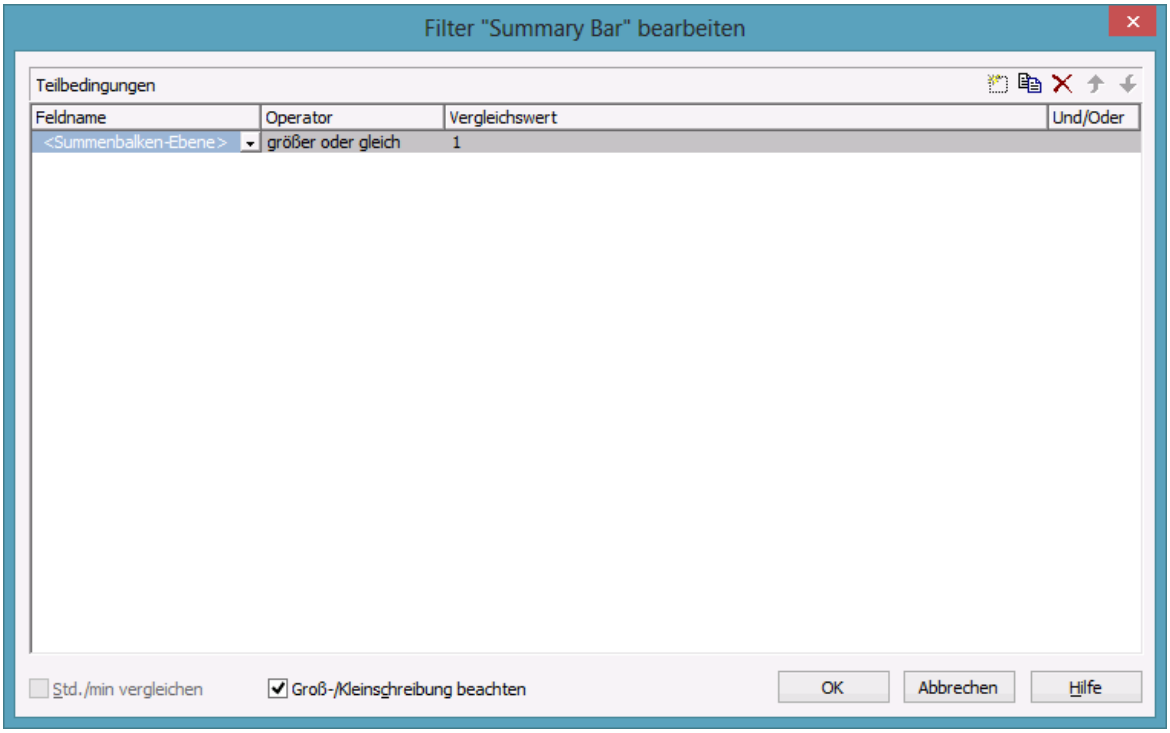
Sie erreichen das Dialogfeld **Filter verwalten**

- von der Eigenschaftenseite **Objekte**
- für Layer: vom Dialogfeld **Balkenaussehen festlegen**
- für Tabellenformate: vom Dialog **Tabelle bearbeiten**
- für Verbindungen: von der Eigenschaftenseite **Verbindungen** über die **Filter**-Schaltfläche
- für Histogrammkurven: vom Dialog **Histogramm bearbeiten** über die **Filter**-Kombobox
- für Knoten: von der Eigenschaftenseite **Knoten** über die **Filter**-Schaltfläche

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



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



3.8 Grafikformate

VARCHART unterstützt die folgenden Grafikformate, was für den Export von Grafiken für Bedeutung ist, vor allem für die Aufrufe **VcGantt1.Show-GraphicsExportDialog** und **VcGantt1.ExportGraphics**.

Das XGantt-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 Blitten 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.

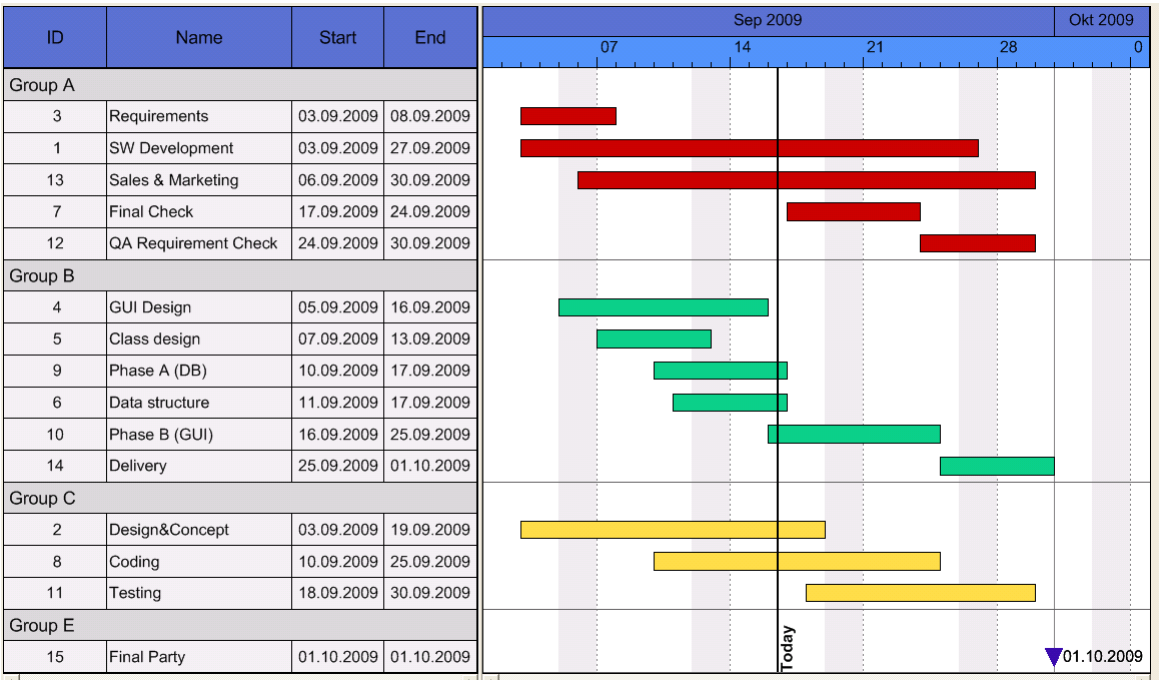
Seit der Version 4.2 wird für den Export die frei verfügbare Bibliothek **libpng** verwendet, um durch die Vorgabe der Auflösung beliebig große Bitmaps speichern zu können. Hierbei muss aber darauf geachtet werden, dass sehr große PNG-Dateien zu Schwierigkeiten beim Einlesen führen können, denn üblicherweise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

3.9 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.).

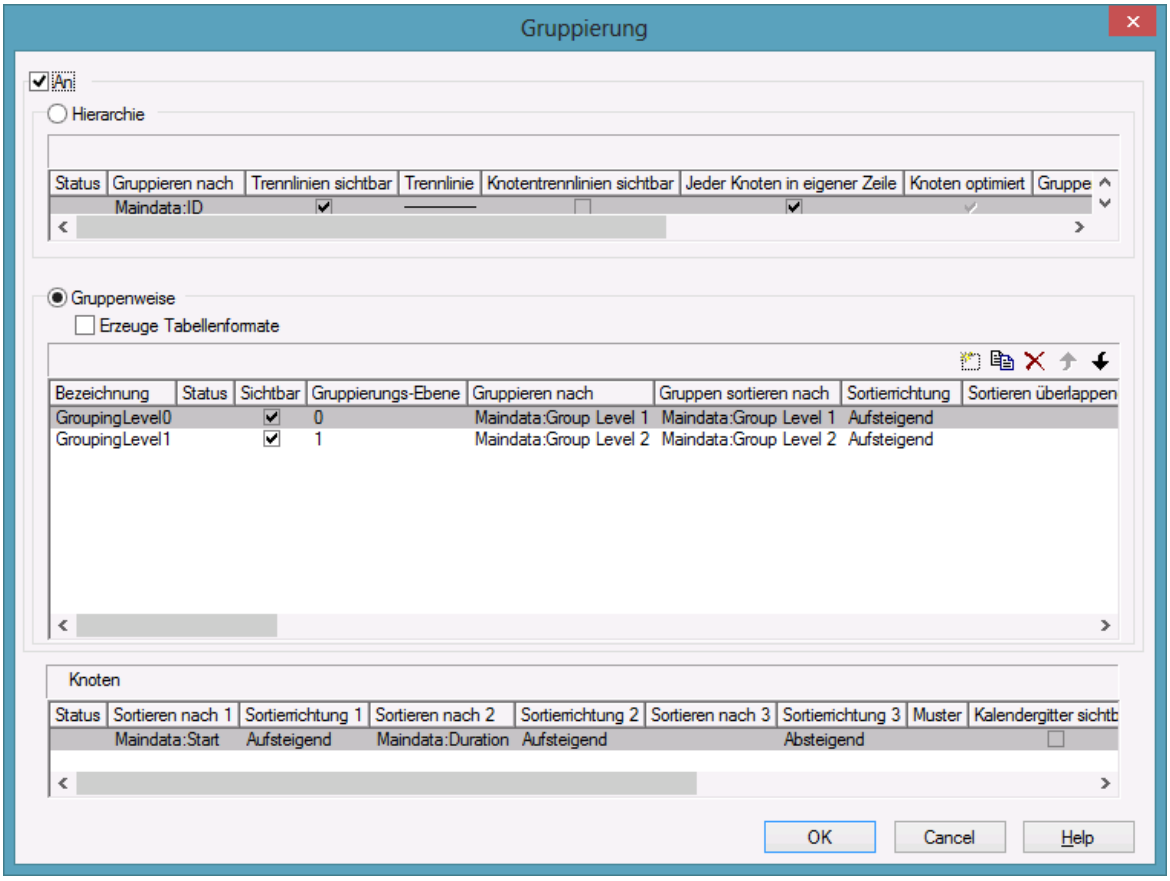
Ein gruppierter Plan sieht beispielsweise folgendermaßen aus:



ID	Name	Start	End
Group A			
3	Requirements	03.09.2009	08.09.2009
1	SW Development	03.09.2009	27.09.2009
13	Sales & Marketing	06.09.2009	30.09.2009
7	Final Check	17.09.2009	24.09.2009
12	QA Requirement Check	24.09.2009	30.09.2009
Group B			
4	GUI Design	05.09.2009	16.09.2009
5	Class design	07.09.2009	13.09.2009
9	Phase A (DB)	10.09.2009	17.09.2009
6	Data structure	11.09.2009	17.09.2009
10	Phase B (GUI)	16.09.2009	25.09.2009
14	Delivery	25.09.2009	01.10.2009
Group C			
2	Design&Concept	03.09.2009	19.09.2009
8	Coding	10.09.2009	25.09.2009
11	Testing	18.09.2009	30.09.2009
Group E			
15	Final Party	01.10.2009	01.10.2009

Gruppierung bedeutet, dass alle Vorgänge ein gemeinsames Merkmal besitzen. Daher bilden alle Knoten, die in einem bestimmten Datenfeld denselben Wert haben, eine Gruppe.

Sämtliche Kriterien für die Gruppierung können Sie im gleichnamigen Dialog vornehmen. Sie erreichen diesen Dialog über die Eigenschaftenseite **Objekte**.



Alle Vorgänge, die denselben Wert für das unter **Gruppieren nach** gewählte Datenfeld besitzen, bilden eine Gruppe.

In der Darstellung wird für jede Gruppe eine zusätzliche Zeile ausgegeben, die nur den Gruppentitel enthält. Dessen Aussehen im Tabellenteil kann im Dialog **Tabellenformat bearbeiten** separat für den expandierten und für den kollabierten Zustand der Gruppe festgelegt werden (Tabellenformate **Subtitle** und **Collapsed**), beispielsweise mit verschiedenen Farben oder Datenfeldern.

Neben jedem Gruppentitel steht ein kleines Plus- bzw. Minus-Symbol, das den kollabierten bzw. expandierten Zustand der Gruppe anzeigt. Indem Sie auf dieses Symbol klicken, verkehrt sich der Zustand in sein Gegenteil. Voraussetzung ist, dass im Dialog **Gruppierung** die Option **Änderungen erlaubt** ausgewählt ist.

Um die Reihenfolge der Gruppen zu kontrollieren, können Sie die Felder **Gruppen sortieren nach** sowie **Sortierrichtung** verwenden.

Für die Gruppierung können Sie weiterhin folgende Kriterien festlegen:

- ob **Tabellenformate** angelegt werden sollen
- ein **Muster** für die Überschriftenzeile der Gruppe (nur im Diagramm)
- Anzeige, Art und Aussehen von **Kalender- und Liniengittern**

- ob alle Vorgänge einer Gruppe in einer eigenen Zeile ausgegeben werden sollen oder nicht (Aktivieren/Deaktivieren der Option **Jeder Knoten in eigener Zeile**) und wenn ja, ob das Layout automatisch optimiert werden soll (**Knoten optimiert**)
- ob die Gruppen beim Start kollabiert sein sollen (**Gruppen kollabiert**)
- Anzeige, Position und Aussehen von **Trennlinien** zwischen den Gruppen
- ob dem Anwender die Kollabieren/Expandieren-Funktionalität angeboten wird (**Änderungen erlaubt**)
- die Anzeige von **Summenbalken**
- ob **Gruppenknoten** sichtbar sein sollen
- ob Gruppen in Tabelle und Diagramm mit der Maus verschoben werden können um damit ihre **Reihenfolge** zu ändern
- ob nach jeder Gruppe **Seitenumbrüche** ausgeführt werden sollen

> Interaktives Neuanlegen von Gruppen

Wenn Sie in einem leeren Diagramm interaktiv den ersten Knoten anlegen, wird dafür automatisch eine neue Gruppe angelegt. Sie können im Dialog **Vorgänge bearbeiten** in das Datenfeld, das im Dialog **Gruppierung** unter **Gruppieren nach** gewählt wurde, einen Namen für die Gruppe des Knotens eintragen.

Falls Sie eine neue Gruppe anlegen möchten, gehen Sie folgendermaßen vor: Erzeugen Sie zunächst einen Knoten innerhalb einer bestehenden Gruppe. Klicken Sie dann doppelt auf diesen Knoten, um den Dialog **Vorgänge bearbeiten** aufzurufen. Tragen Sie nun in das Datenfeld, das im Dialog **Gruppierung** unter **Gruppieren nach** gewählt wurde, einen Namen für die Gruppe des Knotens ein. Die neue Gruppe wird dann automatisch erzeugt.

> Interaktives Umgruppieren von Knoten

Wenn der Anwender mit Hilfe der Maus einen Vorgang von einer Gruppe in eine andere schiebt, wird der Wert in dem Gruppierfeld automatisch angepasst.

> Leere Gruppen

Wenn Sie alle Knoten einer Gruppe löschen, bleibt die Gruppenüberschrift dieser Gruppe in der Tabelle zunächst stehen. Wenn Sie die Gruppierung aus- und dann wieder einschalten oder wenn Sie das Programm beenden und neu starten, werden die Überschriften aller leeren Gruppen nicht mehr angezeigt.

> Interaktives Verschieben von Untergruppen

Sie können interaktiv die Reihenfolge der Untergruppen einer Gruppe verändern. Markieren Sie dazu den Summenbalken der zu verschiebenden Untergruppe. Schieben Sie nun bei gedrückter linker Maustaste das Phantom des Summenbalkens nach oben bzw. unten. Sobald das Phantom auf einem anderen Summenbalken derselben Gruppierungsebene liegt, zeigt ein Pfeil am Cursor-Symbol, ob Sie den Summenbalken ober- bzw. unterhalb davon einfügen können. Wenn Sie die Maustaste loslassen, wird die Gruppe mit allen untergeordneten Knoten an der gewählten Stelle eingefügt.

> Alle Vorgänge aller Gruppen in einer Zeile/in getrennten Zeilen/expandieren/kollabieren

Sie können mit Hilfe weniger Code-Zeilen die Darstellung aller Vorgänge aller Gruppen festlegen. Im folgenden Beispiel werden über einen Menübefehl die Vorgänge aller Gruppen (hier zwei Gruppierungsebenen) in einer Zeile dargestellt.

Code-Beispiel VB.NET

```
Private Sub mnuAllNodesOneRow_Click()

   Dim groupCltn As VcGroupCollection
   Dim group As VcGroup
   Dim subGroupCltn As VcGroupCollection
   Dim subGroup As VcGroup

   groupCltn = VcGantt1.GroupCollection

   For Each group In groupCltn
      subGroupCltn = group.SubGroups
      group.NodesArrangedInOneRow = True
      For Each subGroup In subGroupCltn
         subGroup.NodesArrangedInOneRow = True
      Next
   Next

End Sub
```

Code-Beispiel C#

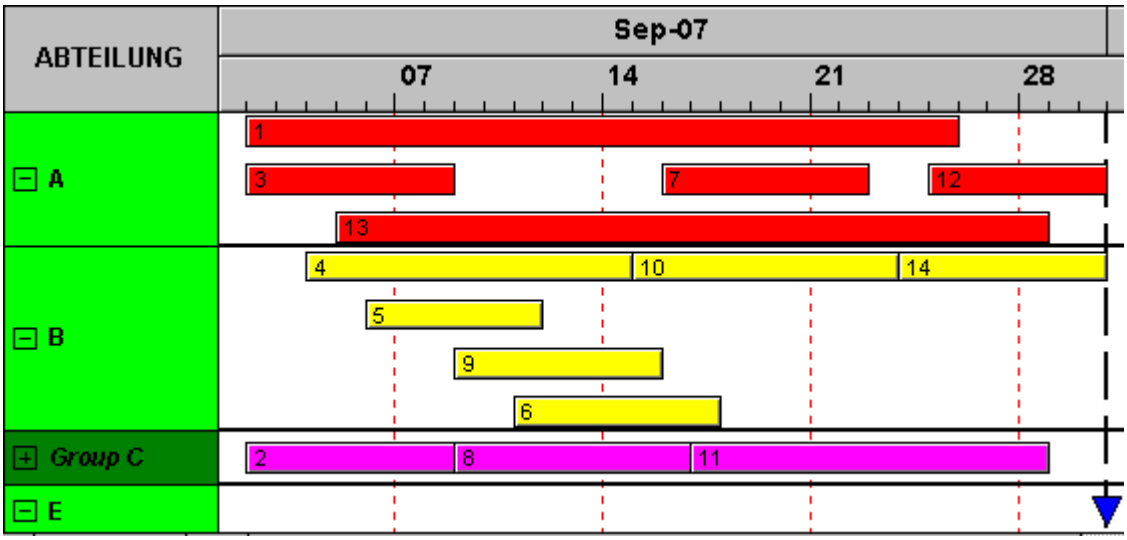
```
private void mnuAllNodesOneRow _Click(object sender, System.EventArgs e)
   {
   VcGroupCollection groupCltn = VcGantt1.GroupCollection;
   VcGroupCollection subGroupCltn;

   foreach (VcGroup group in groupCltn)
      {
      subGroupCltn = group.SubGroups;
      group.NodesArrangedInOneRow = true;
      foreach (VcGroup subGroup in subGroupCltn)
         {
            subGroup.NodesArrangedInOneRow = true;
```

```
            }
        }
    }
```

Analog können Sie alle Knoten aller Gruppen in getrennten Reihen darstellen (group.NodesArrangedInOneRow = False), expandieren (group.Collapsed = False) oder kollabieren (group.Collapsed = True).

> Darstellung jeder Gruppe in einer Zeile

In diesem Abschnitt wird kurz der Sonderfall **Jeder Knoten in eigener Zeile** für die gruppierte Anordnung der Vorgänge besprochen. Ein solcher Plan sieht etwa folgendermaßen aus:



Der Gruppierungsvorgang entspricht dem oben Beschriebenen. Dabei gab es für jeden Vorgang eine eigene Zeile. Wenn die Option **Jeder Knoten in eigener Zeile** im Dialog **Gruppierung** nicht aktiviert wurde, bildet jede Gruppe eine Zeile. Innerhalb dieser Zeile kann es natürlich zu Überlappungen kommen. Um diese sichtbar zu machen, kann die Gruppe expandiert werden, wodurch die Anordnung genau genommen "In möglichst wenigen Zeilen" heißen müsste. Durch entsprechendes Verschieben der Vorgänge kann man erreichen, dass keine Überlappungen mehr vorkommen. So hat man mit der expandierten Darstellung die Gewähr, dass eine Überlappung sofort ins Auge fällt, auch wenn sie nur eine Sekunde dauert.

Wird eine Gruppe kollabiert (wie im Beispiel Group C), so kann man zwar erkennen, dass mehrere Vorgänge in dieser Gruppe liegen, über zeitliche Überschneidungen kann man jedoch keine Aussage machen.

Bei dieser Art der Darstellung hat ein Tabellenformat für die Vorgänge natürlich keinen Sinn, und daher gibt es ein solches auch nicht. Es empfiehlt sich deshalb, den Layer zu beschriften oder einen Tooltip zur Identifizierung der Layer zu verwenden.

> Darstellung übereinander liegender Knoten

Wenn der Modus **Jeder Knoten in eigener Zeile** nicht gewählt wurde, können Sie über die Sortierreihenfolge der Knoten bestimmen, welche Knoten oben liegen. Die Knoten werden gemäß der Sortierreihenfolge gezeichnet, d. h. der letzte Knoten der Sortierung liegt oben und ist vollständig sichtbar.

> Summenbalken

In den Gruppenzeilen können Summenbalken ausgegeben werden. Sie können festlegen, ob und ggf. auf welchen Gruppierungsebenen Summenbalken ausgegeben werden sollen.

Damit Summenbalken auf der unter **Ebene** eingestellten Gruppierungsebene dargestellt werden, müssen Sie im Dialog **Gruppierung** das Kontrollkästchen **Summenbalken anzeigen** für diese Ebene aktivieren.

Um Summenbalken zur Laufzeit an- bzw. abzuschalten, können Sie die VcGantt-Eigenschaft **SummaryBarsVisible** verwenden. Bei nicht-hierarchischer Gruppierung können Sie mit Hilfe des Parameters **GroupingLevel** ebenenweise die Summenbalken ein- bzw. ausschalten.

Das Aussehen von Summenbalken legen Sie auf der Eigenschaftenseite **Layer** fest, indem Sie für jede gewünschte Darstellung einen Layer erstellen. Dabei können Sie sowohl einen Layer für alle bzw. mehrere Ebenen als auch unterschiedliche Layer für jede einzelne Ebene definieren. Möglich wäre z.B. ein Layer "Summenbalken 1" für die 1. Ebene, ein Layer "Summenbalken 2" für die 2. Ebene etc.

Damit die Summenbalken dann auch tatsächlich in der gewünschten Form dargestellt werden, müssen ihnen entsprechende Filter zugewiesen werden, die Sie im Dialog **Filter verwalten** erstellen, z.B."Summenbalken1" für die 1. Ebene. Um die passende Ebene für Ihren Filter einzustellen, wählen Sie im Dialog **Filter bearbeiten** unter **Feldname** den Eintrag "<Summenbalken-Ebene>", wählen den geeigneten **Operator** (gleich, ungleich größer als etc.) und tragen unter **Vergleichswert** die gewünschte Ebene ein.

Wenn Sie nun das Programm starten, werden die so definierten Summenbalken dargestellt.

3.10 Hierarchische Anordnung

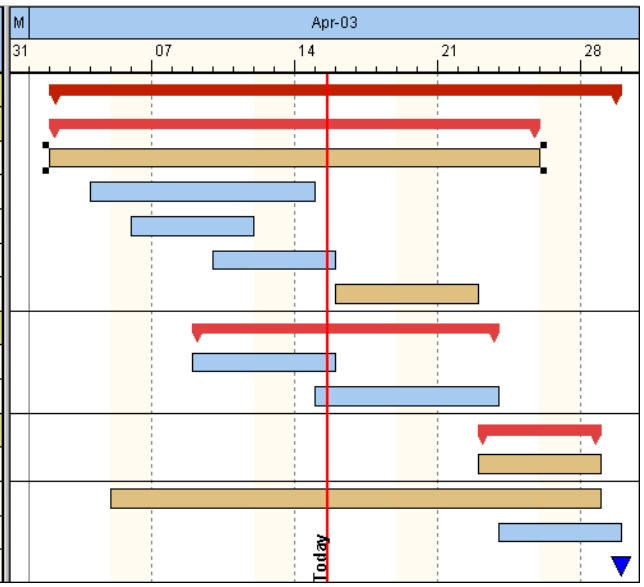
Neben der Gruppierung gibt es noch eine weitere Möglichkeit, Vorgänge anzuordnen: die Hierarchie. Dafür müssen die Projektdatensätze einen hierarchischen Code enthalten, der in folgender Weise aufgebaut sein muss:

1, 1.1, 1.1.1, 1.2, 1.2.1, ...

Der Knoten der obersten Ebene enthält die Ziffer 1, den Knoten auf der zweiten Ebene die Ziffer 1.1, der Knoten daneben die Ziffer 1.2, der Knoten unter letzterem die Ziffer 1.2.1 usw.

Eine hierarchische Anordnung sieht beispielsweise folgendermaßen aus:

Structure	Name	Start	End	ID
⊟ 1	*SW Development*	*02.04.03*	*30.04.03*	*1*
⊟ 1.2	*Design&Concept*	*02.04.03*	*26.04.03*	*2*
1.2.1	Requirements	02.04.03	26.04.03	3
1.2.2	GUI Design	04.04.03	15.04.03	4
1.2.3	Class design	06.04.03	12.04.03	5
1.2.4	Data structure	10.04.03	16.04.03	6
1.2.5	Final Check	16.04.03	23.04.03	7
⊟ 1.3	*Coding*	*09.04.03*	*24.04.03*	*8*
1.3.1	Phase A (DB)	09.04.03	16.04.03	9
1.3.2	Phase B (GUI)	15.04.03	24.04.03	10
⊟ 1.4	*Testing*	*23.04.03*	*29.04.03*	*11*
1.4.1	QA Requirement C	23.04.03	29.04.03	12
1.5	Sales & Marketing	05.04.03	29.04.03	13
1.6	Delivery	24.04.03	30.04.03	14
1.7	Final Party	30.04.03	30.04.03	15

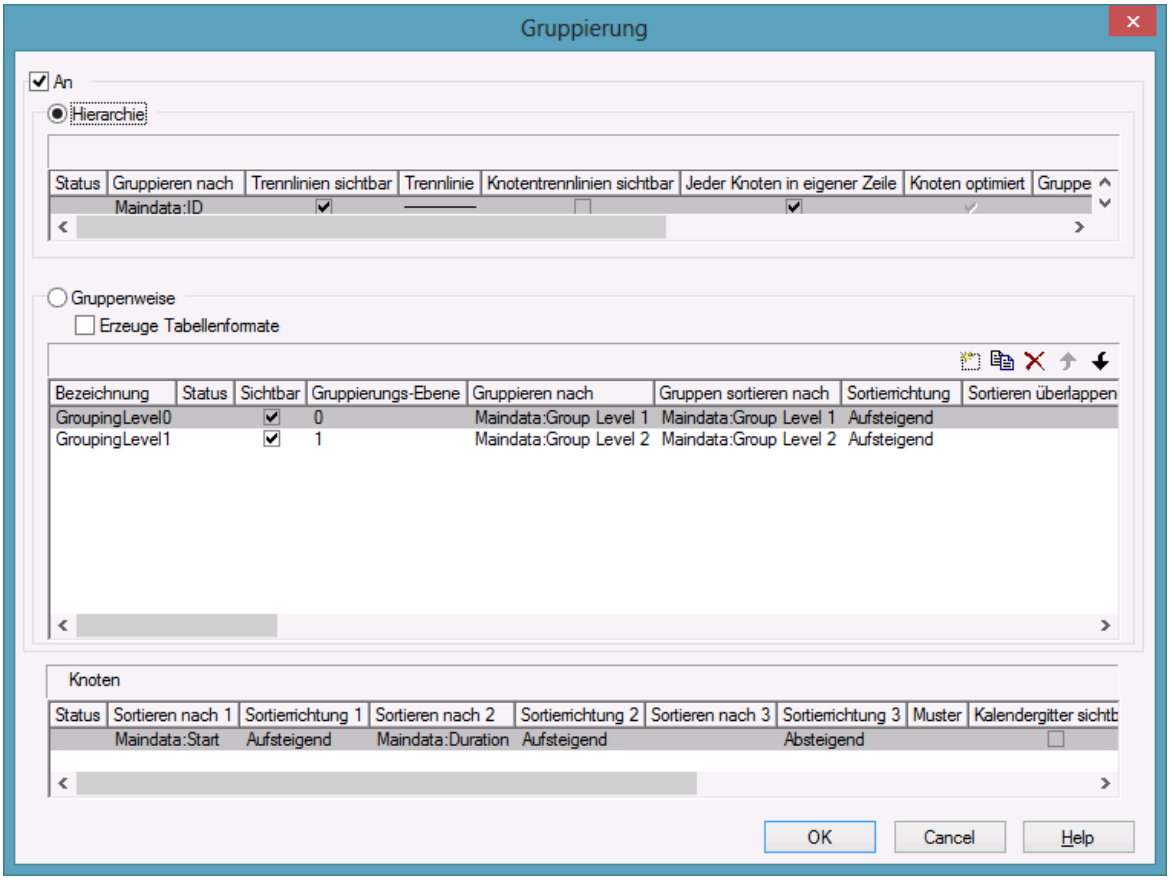


Die Symbole + und – werden automatisch den **Summenvorgängen** vorangestellt. Durch einen Klick auf das --Symbol werden jeweils die untergeordneten Vorgänge kollabiert (eingeklappt), durch einen Klick auf das +-Symbol expandiert (ausgeklappt).

Untergeordnete Ebenen werden automatisch eingerückt. Es wird nicht automatisch überprüft, ob die übergeordneten Vorgänge mit ihren Terminen die untergeordneten wirklich überdecken. Es wird also keine Prüfung oder Setzung der Dauer vorgenommen.

Bei der hierarchischen Anordnung ist keinerlei zusätzliche Gruppierung oder Sortierung möglich.

Die hierarchische Anordnung wird im Dialog **Gruppierung** eingestellt:



Hier muss das Kontrollkästchen **Hierarchie** aktiviert sein. Aus der Kombobox **Gruppieren nach** ist dann das Datenfeld auszuwählen, das den Code für die hierarchische Anordnung enthält. Die Anordnung der Vorgänge wird dadurch eindeutig bestimmt.

Außerdem lassen sich noch folgende Kriterien für die Hierarchie festlegen:

- Anzeige und Aussehen von **Trennlinien**
- ob die Vorgänge beim Start kollabiert sein sollen (**Gruppen kollabiert**)
- ob **Summenbalken** angezeigt werden sollen
- ob nach jeder Gruppe und bis zu welcher Ebene höchstens **Seitenumbrüche** ausgeführt werden sollen

Für die Ausgabe der Summenvorgänge werden die beiden Tabellenformate **Hierarchy** und **HierarchyCollapsed** verwendet. Sie können sie im Dialog **Tabellenformat bearbeiten** festlegen.

> Vorgänge interaktiv verschieben

Sie können Vorgänge interaktiv verschieben. Der verschobene Knoten wird dabei immer vor bzw. hinter dem Bezugsknoten eingefügt, auch bei kollabierten Gruppen.

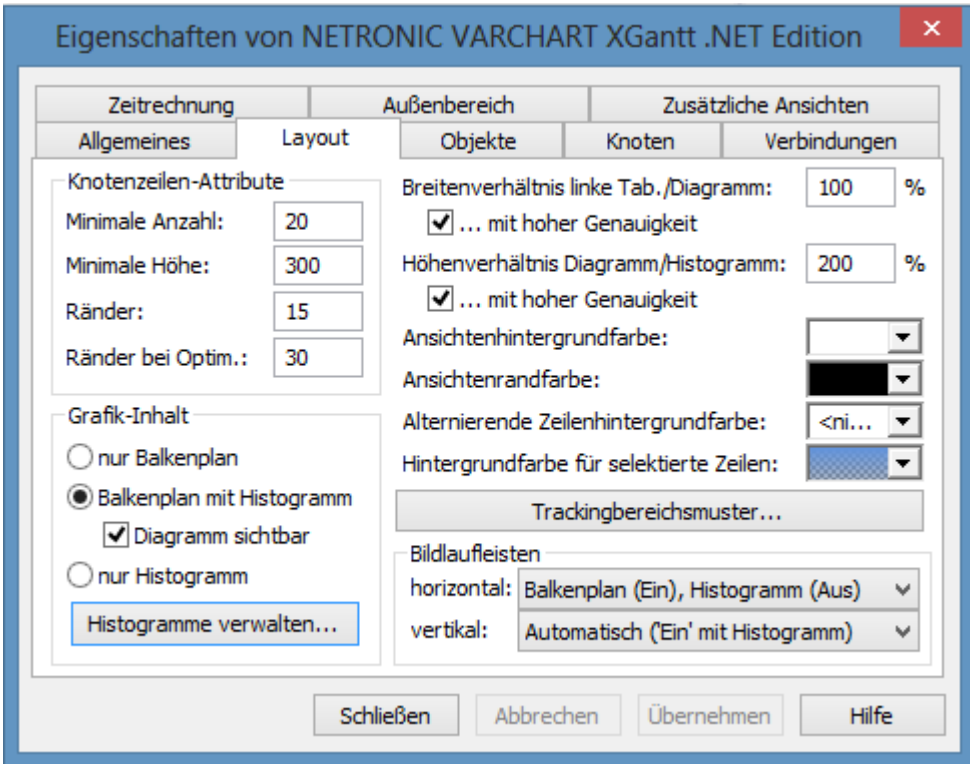
> Summenbalken interaktiv verschieben

Summenbalken können Sie genauso wie Knoten interaktiv verschieben. Dabei werden die untergeordneten Knoten mitverschoben.

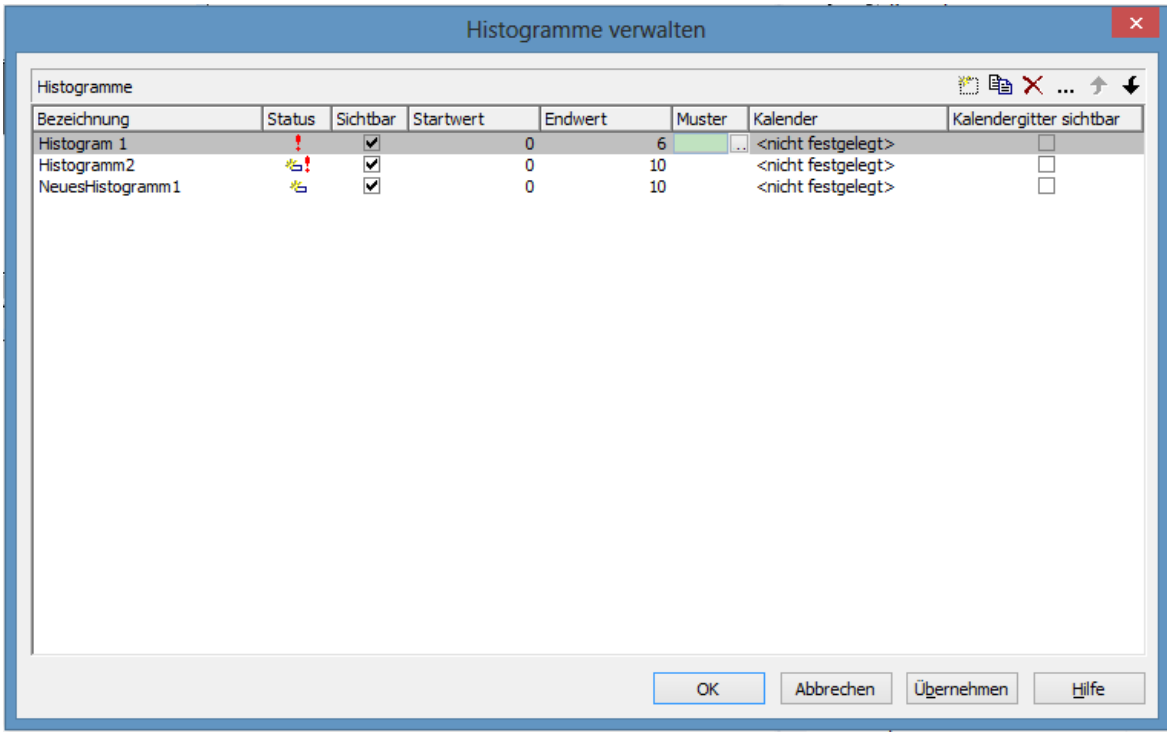
3.11 Histogramme

In den Histogrammen von VARCHART XGantt werden Vorgänge nach Kriterien, die Sie festlegen, über der Zeitachse zu Kurven aufsummiert.

Auf der Eigenschaftenseite **Layout** können Sie unter **Grafik-Inhalt** festlegen, ob nur der Balkenplan, nur das Histogramm oder beides ausgegeben werden soll.



Um auszuwählen, welche Histogramme dargestellt werden sollen, und um Histogramme zu bearbeiten, klicken Sie auf die Schaltfläche **Histogramme verwalten**.

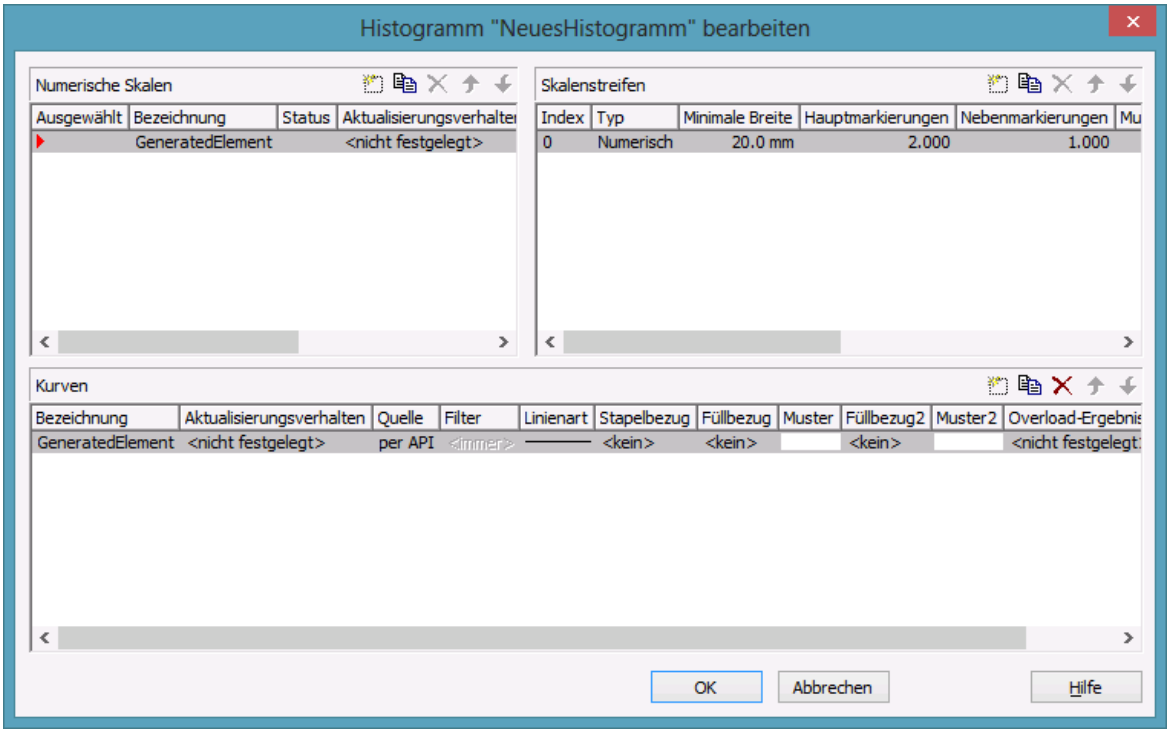


Hier können Sie ein oder mehrere Histogramme auswählen, die dargestellt werden sollen.

Jedes Histogramm besteht aus einer numerischen Skala (y-Achse) und Kurven. Für die x-Richtung wird die Zeitskala des Balkendiagramms verwendet.

Sie können hier für jedes Histogramm den Start- und den Endwert der numerischen Skala festlegen.

Um das Histogramm weiter zu bearbeiten, klicken Sie auf die Schaltfläche **Histogramm bearbeiten** (···).

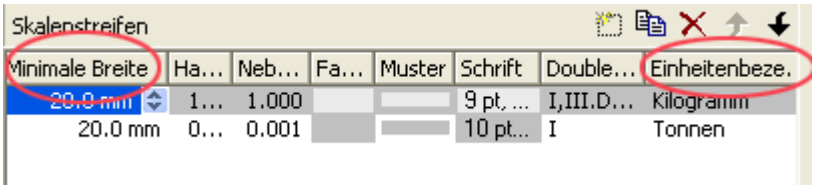


> Numerische Skalen

Sie können hier verschiedene numerische Skalen definieren und auswählen, welche davon für das zu bearbeitende Histogramm verwendet werden soll.

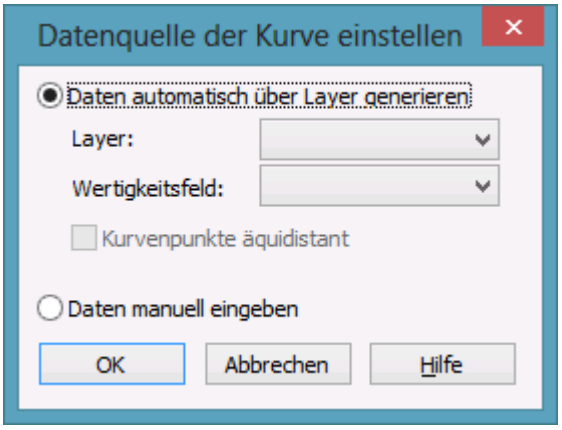
Für jede numerische Skala können Sie hier unter **Breite/Einheit** die Skalierung des Histogramms in y-Richtung festlegen. Außerdem können Sie hier entscheiden, ob ein Liniengitter dargestellt werden soll und ggf. dieses festlegen.

Im Bereich **Skalenstreifen** können Sie für die jeweils markierte numerische Skala einen oder mehrere Skalenstreifen festlegen und für jeden einen **Typ**, die **Minimale Breite**, die Anzahl der Einheiten, nach denen eine **Haupt-** bzw. eine **Nebenmarkierung** erscheinen soll, eine Hintergrund-**Farbe**, verschiedene **Schrift**attribute als auch ein **Doubleformat**. Darüberhinaus können Sie festlegen, ob Sie den Inhalt des Skalenstreifens über die **ObjectDraw-Ereignisse** selber gestalten möchten sowie eine **Bezeichnung** für die im Ribbon verwendeten Einheiten angeben. Für die Darstellung der Bezeichnung beachten Sie bitte, dass Sie über die Minimalbreite des Skalenstreifens genügend Platz dafür einrichten; andernfalls kann die Bezeichnung nicht angezeigt werden und bleibt unsichtbar.



> Histogrammkurven

Ein Histogramm kann mehrere Kapazitätskurven enthalten, für die Sie individuell eine Reihe von Parametern setzen können. Die einfachsten sind **Name** und **Linienart**. Zudem muss für jede Kurve die Daten-**Quelle** angegeben werden, die die Werte für die Kurvenpunkte liefert. Dazu klicken Sie bitte auf das Feld **Quelle** und dann auf die **Bearbeiten**-Schaltfläche (…). Es öffnet sich der folgende Dialog:



Es gibt zwei grundsätzliche Alternativen für die Datenquelle einer Kurve:

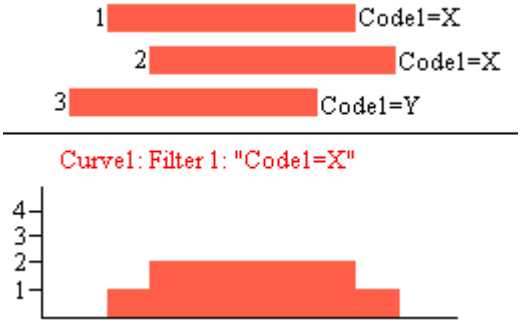
> 1. Daten automatisch über Layer generieren

Die Kurven werden aus den Vorgängen generiert. Beim Aufsummieren der Vorgänge zu einer Kurve werden für jeden Vorgang die Anfangs- und Enddaten des gewählten Layers (z.B. des Layers "Start-Ende") übernommen.

Sie können bei der Datengenerierung aus Layern, falls gewünscht, noch einmal bestimmte Layer herausfiltern, die zu der Kurve beitragen sollen. Dazu können Sie in dem vorgeschalteten Dialog **Histogramm bearbeiten** für jede Kurve einen **Filter** zur Selektion von Vorgängen auswählen.

Beispiel:

Zu Curve1 tragen nur die Vorgänge bei, die die Bedingungen von Filter 1 erfüllen. Filter 1 enthält die Bedingung "Code1 = X". Die Kurve wird also nur aus den Vorgängen 1 und 2 generiert, für die "Code1 = X" gilt.



Für Kurven, die per Layer generiert worden sind, können Sie unter **Wertigkeitsfeld** das Datenfeld auswählen, aus dem der Anteil der Skaleneinheiten entnommen wird, um die die Kurve auf der numerischen Skala steigen soll, wenn ein Vorgang hinzuaddiert wird (z.B. um 5 Einheiten).

> 2. Daten per API generieren

Die Werte werden bei dieser Wahl über die Programmierschnittstelle (API) gesetzt. Darüber können Sie mit der VcCurve-Methode **SetValues** die Werte einer Histogrammkurve beliebig setzen.

Für per API generierte Kurven können Sie im Dialog **Datenquelle der Kurve einstellen** festlegen, ob die **Kurvenpunkte äquidistant**, also in gleichem Abstand in X-Richtung erzeugt werden sollen. Andernfalls werden Kurvenpunkte nur an den Stellen erzeugt, an denen sich der y-Wert ändert.

Kurvenpunkte äquidistant: Hier müssen ein Anfangswert (**startDate**) und die y-Werte der Histogrammkurve gesetzt werden. Aus dem Anfangswert werden zusammen mit den Werten für **Zeiteinheit** und **Kleinstes Zeitintervall** (Eigenschaftenseite **Allgemeines**) die Stützstellen der Histogrammkurve ermittelt.

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

So gesetzte Kurven können nicht interaktiv verändert werden.

Kurvenpunkte nicht äquidistant: Hier müssen Paare von x- und y-Werten gesetzt werden:

Set Values X1, Y1

Set Values X2, Y2

Set Values X3, Y3...

Die **Zeiteinheit** und das **Kleinste Zeitintervall** sind hier irrelevant. Die Kurve kann hier interaktiv verändert werden.

> Referenzkurve

Ene typische Form der Verwendung der mittels **SetValues** über die API definierten Kurven ist die Kapazitätskurve. Sie dient in der Regel als Referenzkurve, die mit anderen Kurven Flächen bildet, die eingefärbt und gemustert werden können.

Im Feld **Füllbezug** legen Sie die Kurve fest, die eine Fläche, ausgehend von der in Bearbeitung befindlichen Kurve, am anderen Ende begrenzen soll. Wählen Sie für eine Kurve hier den Eintrag <Null-Linie>, so reicht die Füllung unter dieser Kurve bis zur x-Achse. Eine in dieser Ebene liegende weitere Kurve würde, abhängig von der Zeichenreihenfolge, ggf. verdeckt.

Die Kurvenlinie und die Füllung unterhalb einer Kurvenlinie legen Sie in den Feldern **Linienart** und **Muster** fest. Wenn Sie auf den Eintrag des Feldes **Linienart** klicken, öffnet sich das Dialogfeld **Linie bearbeiten**, in dem Sie Farbe, Strichstärke und Linienart jeder Kurvenlinie festlegen können. Wenn Sie auf das Feld **Muster** klicken, öffnet sich das Dialogfeld **Muster**, in dem Sie ein Muster sowie die Vorder- und Hintergrundfarbe für die Füllung unter einer Kurve festlegen können.

Sie können außerdem eine zweite Bezugskurve auswählen.

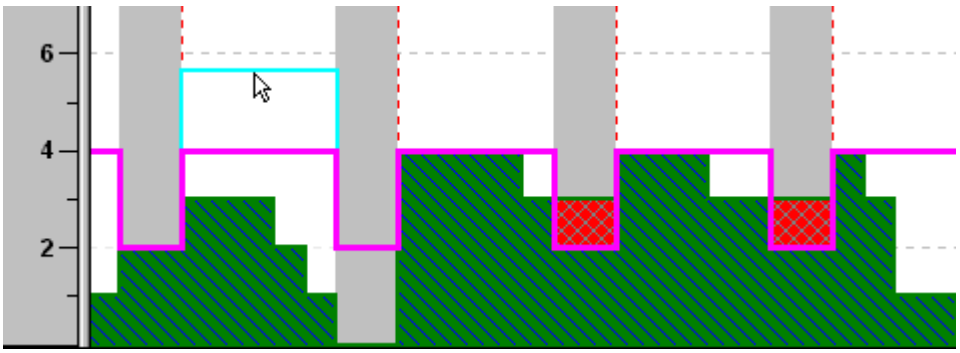
Wählen Sie unter **Füllbezug 2** eine zweite Bezugskurve aus. Die Fläche zwischen dne beiden Kurven wird nur dargestellt, wenn die y-Werte der in Bearbeitung befindlichen Kurve größer sind als die der zweiten Bezugskurve, d.h. wenn sich die Fläche unterhalb der in Bearbeitung befindlichen Kurve ausbreitet.

In dem dazugehörigen **Muster**-Feld können Sie das Muster und die Füllfarbe der Fläche festlegen.

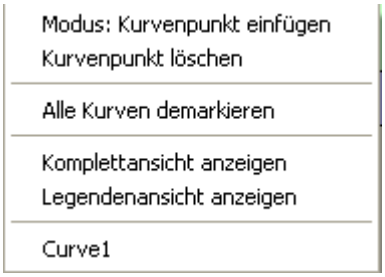
Beispiele zur Verwendung von Histogrammen finden Sie unter "Tutorium: Histogramme erstellen" sowie "Tutorium: Kapazitätsengpässe darstellen".

> Kapazitätskurven interaktiv verändern

Als Benutzer können Sie die per API definierten Kurven (in der Regel die Kapazitätskurve) interaktiv verändern, z.B. wenn sich Kapazitäten im Produktionssystem geändert haben. Dazu ziehen Sie mit der Maus ein horizontales Teilstück der Kurve nach oben oder unten ziehen. Dies ist nur möglich bei Kurven, die nicht aus äquidistanten Punkten erzeugt wurdne. Ein Phantom zeigt Ihnen dabei die neue Position der Kurve an.



Sie können außerdem einzelne Kurvenpunkte hinzufügen oder löschen. Klicken Sie dazu mit der rechten Maustaste in den Histogrammbereich. Das folgende Kontextmenü erscheint:



Wenn mehrere Kapazitätskurven definiert sind, werden deren Namen im Kontextmenü angezeigt. Wenn Sie auf den Namen einer Kurve klicken, wird diese markiert.

Im **Modus: Kurvenpunkt einfügen** können Sie mit jedem Linksklick auf die Verfügbarkeitskurve einen neuen Stützpunkt für diese definieren.

Um einen Stützpunkt zu löschen, klicken Sie diesen mit der rechten Maustaste an und wählen Sie im Kontextmenü den Befehl **Kurvenpunkt löschen**.

> Kurvenpunkte anzeigen

Sobald Sie auf eine Kurve klicken, die nicht aus äquidistanten Punkten besteht, werden die per API definierten Punkte durch kleine schwarze Quadrate angezeigt. Wenn Sie erneut auf die Kurve klicken, werden diese nicht mehr angezeigt.

> Kurven stapeln

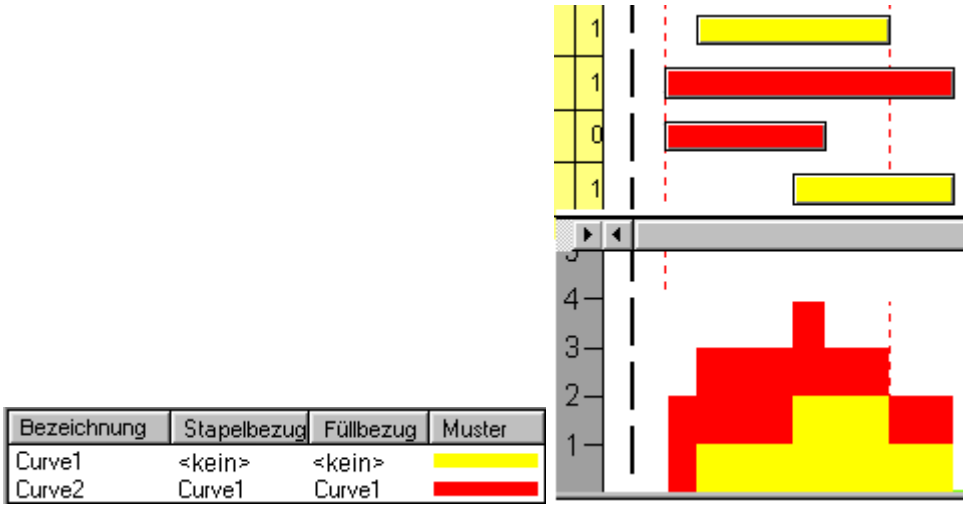
Das Stapeln von Kurven macht beispielsweise Sinn, wenn Sie ein Histogramm aus Kurven bilden, die die Auslastung verschiedner Ressourcen zeigen, gleichzeitig aber auch die Gesamtauslastung dargestellt werden soll.

Im Beispiel unten gibt es rote und gelbe Vorgänge, deren Farbe die Zuordnung zu unterschiedlichen Ressourcen signalisiert. Die Vorgänge werden über entsprechende Filter zu jeweils einer einer Kurve zusammengefasst.

Aufeinandergestapelt zeigen die beiden Kurven die Gesamtauslastung des Systems an.

Für die Darstellung der roten Kurve oberhalb der gelben geben sie die gelbe Kurve (Curve 1) im **Stapelbezug** der roten an.

(Wählen Sie für alle Kurven eines Histogramms den Eintrag <kein>, werden die Kurven nicht aufeinander gestapelt, sondern überlagern einander optisch).



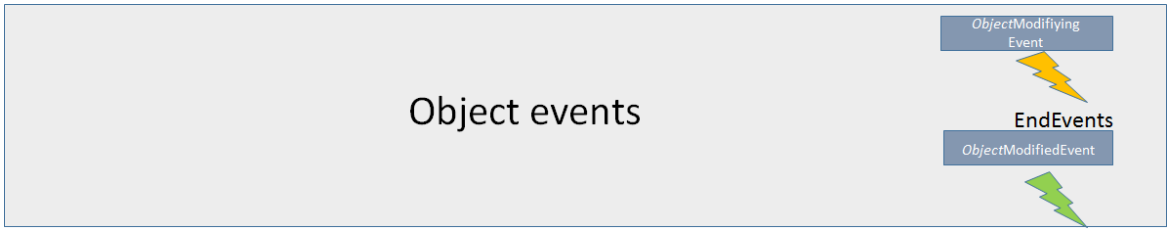
Bezeichnung	Stapelbezug	Füllbezug	Muster
Curve1	<kein>	<kein>	
Curve2	Curve1	Curve1	

Curve2 wird auf Curve1 gestapelt.

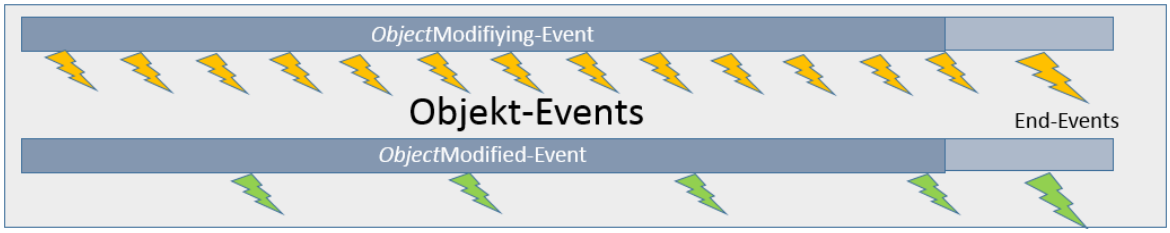
3.12 Interaction Events

Während einer Maus-Interaktion innerhalb des Live-Updates, d.h., während des Verschiebens eines Knotens mit der Maus, ist es hilfreich, Informationen über das betreffende Objekt zu erhalten und zu verarbeiten.

Im Standardverhalten von VARCHART XGantt wird keine Information über den Zustand des betroffenen Objekts geliefert. Erst beim Loslassen der Maustaste liefert ein **ObjectModifying Event** Informationen über den alten (bevor die Maustaste gedrückt wurde) und neuen (nachdem die Maustaste losgelassen wurde) Zustand. Zusätzlich zeigt ein **ObjectModified Event** an, dass der Vorgang intern abgeschlossen ist:



Um bereits während der Mausaktion - und nicht erst nach deren Abschluss - Informationen über das sich verändernde Objekt zu erhalten, kann man einige die Interaktion begleitende und beschreibende Events, die Interaction Events, nutzen. Darüberhinaus wurden mit XGantt-Version 5.0 Änderungen bei Aufrufzeitpunkt und Häufigkeit der Objekt-Events vorgenommen.



Betroffene Interaktionen

Es werden Events erläutert, die den Vorgang einer Interaktion in VARCHART XGantt und der daran beteiligten Objekte näher beschreiben, also "Drag(Drop)" Events bei Interaktionen, die in der Regel:

- mit Drücken der linken Maustaste auf einem Objekt beginnen
- mit gehaltener Maustaste Bewegungen durchführen
- mit einem Loslassen der linken Maustaste enden
- im Rahmen des Aktualisierungsverhaltens "Live Update" behandelt werden.

Terminologie

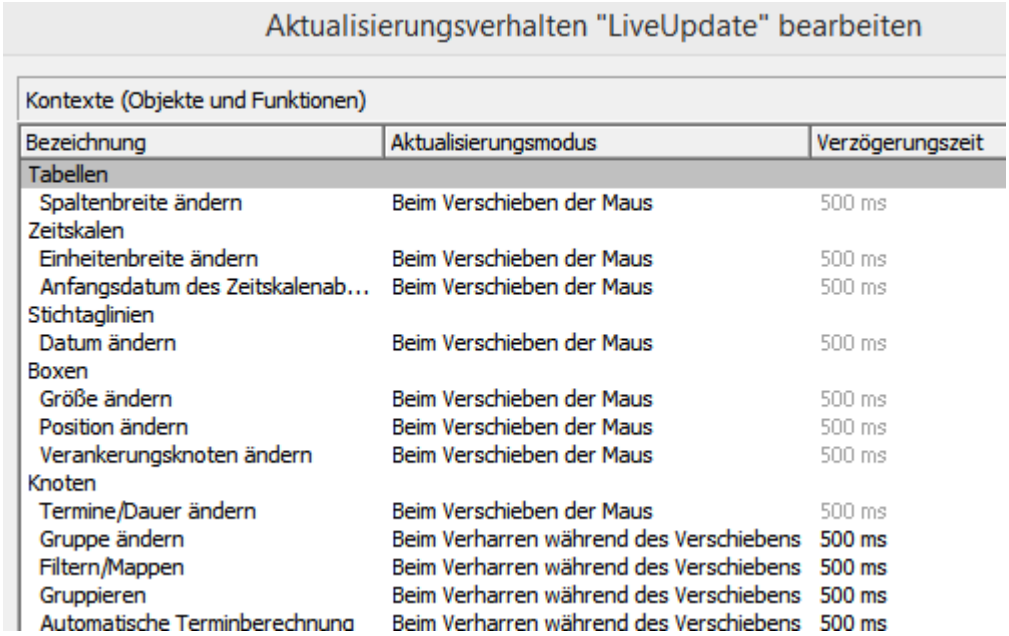
Zum besseren Verständnis werden hier noch einmal kurz einige der nachfolgend verwendeten Begriffe erläutert.

> Objekt-Events

Objekt-Events sind alle Events, die auch schon bisher bei den angesprochenen Interaktionen am Ende einer Aktion geworfen wurden, z.B. **VcDateLineModifying, VcDateLineModified, VcNodeModifying, VcNodeModified** usw.

> Live Update

Beim Live Update wird während einer Drag/Drop-Aktion permanent ein "Was wäre, wenn hier das Objekt aktualisiert würde?"-Szenario gezeigt. Es werden also verschiedene Kontexte, z.B. direkte oder abhängige Funktionalitäten bei einer Interaktion, zu verschiedenen Zeitpunkten abgearbeitet. Verschiebt man z.B. einen Knoten, so werden verschiedene Daten und seine Position verändert und abhängig davon verändern sich z.B. Kurven des Histogramms oder Summenbalken. Durch entsprechende Konfigurierung des Live Updates können diese Veränderungen sofort durchgeführt werden, oder erst, wenn man während der Bewegung eine festzulegende Zeit mit der Maus verharrt oder aber auch erst am Ende der Aktion beim Loslassen der Maus.



Beispiel: Wie erfolgen die Aktualisierungen beim Verschieben eines Knotens wenn als Aktualisierungsverhalten "Beim Verschieben der Maus" gewählt ist?

Zeitlich direkte Anpassungen am Knoten:

- alle Datumswerte des Knotens
- Filter werden ausgewertet, wodurch z.B. andere Farben im Tabellenbereich erscheinen können
- Summenbalken
- Histogrammkurven

Anpassungen nach einer Wartezeit(500ms)

- Positionierung des Knotens z.B. in einer Gruppe
- Optimierung bei entsprechendem Layout der Knotenanordnung

Hierbei sollte darauf geachtet werden, nur diejenigen Aktualisierungen durchzuführen, die im Gesamtkontext der Aktion notwendig und sinnvoll sind, da ansonsten während der Aktion ein sehr unruhiges Bild entstehen könnte.

InInteraction Events

Seit VARCHART XGantt 5.0 SR3 ist es möglich, Objekt-Events auch während der Interaktion zu verarbeiten. Diese Objekt-Events, die während einer Interaktion erscheinen, werden als InInteraction-Events bezeichnet.

Wichtig: Die Verwendung der InInteraction Events muss dazu explizit erlaubt sein. Dies erfolgt entweder über die Eigenschaft **VcGantt.InInteractionEventsEnabled = true** oder auf der Eigenschaften-seite **Allgemeines**.

(Im Folgenden wird bei Interaktionen mit Knoten im Realmodus das Anzeigeobjekt **Real(knoten)** und der echte Knoten, also das Datenelement im Chart, **Chartknoten** genannt. Der Chartknoten ist während der Liveinteraktion im Diagrammbereich nicht sichtbar, weil er dort temporär durch den Realknoten ersetzt wird, sein Vorhandensein wirkt sich aber im Diagramm aus (Bandhöhe, Optimierung, Farbgebung im Tabellenteil usw.)).

Auf diese Art werden auch während der Interaktion passend zum angezeigten Phantom oder Realknoten entsprechende Informationen über die normalen Objekt-Events geliefert.

Beim Verschieben eines Knotens wird bei jedem Einrasten des Knotens (abhängig von der gewählten Zeiteinheit und der Schrittweite) ein **VcNodeModifying Event** gefeuert (gelbe Blitze). Der Realknoten zeigt hierbei die mögliche Position und das mögliche Aussehen und beschreibt diesen Zustand über den **VcNodeModifying** Event.

Der in den Event-Args übergebene Knoten (e.Node) stellt dabei den Zustand des Realknotens dar.

Achtung: Aus diesem Grund sind Abfragen auf Eigenschaften des Chartknotens nicht sinnvoll oder möglich sondern es können nur die Eigenschaften get/setDataField, AllData, ID ausgefragt/gesetzt werden!

Wird nun, abhängig vom gewählten Aktualisierungskontext, z.B. beim Verharren während des Verschiebens, das echte Objekt aktualisiert, wird dies über den **Modified** Event (grüner Blitz) angezeigt. Das kann, muss aber nicht zeitgleich mit den **Modifying** Events geschehen.

Wenn ein Knoten verschoben wird und als Aktualisierungsverhalten für das Verschieben "Beim Verschieben der Maus" gewählt ist, werden beide Events immer gleichzeitig kommen.

Fazit bisher:

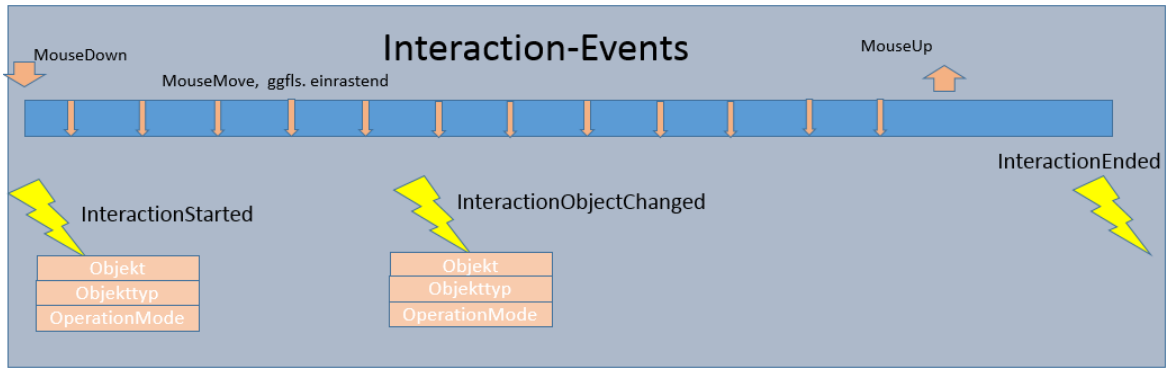
- Verschiebt man einen Knoten, wird seine Veränderung, angezeigt durch den Realknoten, permanent über den VcNodeModifying Event beschrieben.
- Wird der echte Knoten, also der Chartknoten, modifiziert, wird dies durch den VcNodeModified Event angezeigt.
- Am Ende der Interaktion, beim Loslassen der Maustaste, wird wie üblich das abschließende Eventpaar, bestehend aus VcNodeModifying und VcNodeModified Event, zur Verfügung gestellt.

Bei Events, die Realknoten verwenden, sind hierbei nun die betroffenen Objekte die echten Objekte.

Im letzten **VcNodeModifying** Event wird der Chartknoten (im Gegensatz zu den vorhergehenden **VcNodeModifying** Events) mit den im Verlauf der Interaktion zuletzt gesetzten Werten zur Verfügung gestellt, also der Zustand, der beim letzten kleinen grünen Blitz vorlag. **e.OldNode1> der EventArgs** beschreibt den Zustand zu Beginn der Aktion. Es ist also ein Vergleich zwischen Start- und Endzustand der Interaktion möglich.

Im letzten **VcNodeModified**-Event liegt nun wie immer der Chartknoten vor; alle internen Vorgänge sind abgeschlossen.

Interaction Events



Die Objekt-Events werden, wie beschrieben, nun während und am Ende einer Interaktion geworfen. Die Signatur eines Eventhandlers, z.B. die des **VcNodeModifying** Events, unterscheidet sich dabei nicht. Wie kann man nun unterscheiden, ob der Event während oder am Ende der Interaktion geworfen wurde?

Dies könnte wichtig sein, da ja z.B. nicht bei jeder Mausbewegung die Änderung eines Vorganges zeitaufwändig in eine Datenbank geschrieben werden soll; das soll selbstverständlich erst nach Abschluss der Aktion geschehen.

Hierfür gibt es einige Events, die die Interaktion begleiten und beschreiben und die während der Interaktion in den Objekt-Events ausgewertet werden können.

Sobald die linke Maustaste gedrückt wird, erfährt man über den **VcInteractionStarted** Event, auf welchem Objekt der Mauszeiger steht (Objekt und Objekttyp) und was mit dem Objekt passiert. Hier kann man alles vorbereiten, was für die Interaktion notwendig ist.

Tipp: Hier kann auch noch objekt- und kontextspezifisch das Updateverhalten umgeschaltet werden! Man könnte also im Extremfall einen Knoten komplett dynamisch, einen anderen mit blauem Phantomrahmen reagieren lassen. Darüber hinaus kann noch durch entsprechende Setzung (InInteractionEventsEnabled) individuell entschieden werden, ob die ObjectEvents auch während der Interaktion kommen sollen oder nicht.

Beispiel Knoten:

Durch

- Objekt: KnotenObjekt
- Typ: vcObjTypeNodeInDiagram
- OperationMode: vcIIMMoveNode

wird beim Drücken der linken Maustaste im **VcInteractionStarted** Event angezeigt, dass damit begonnen wurde, einen Knoten im Diagramm zu verschieben.

Informationen oder Elemente, die die Interaktion begleiten sollen, können hier initialisiert werden.

> Erzeugen von Objekten

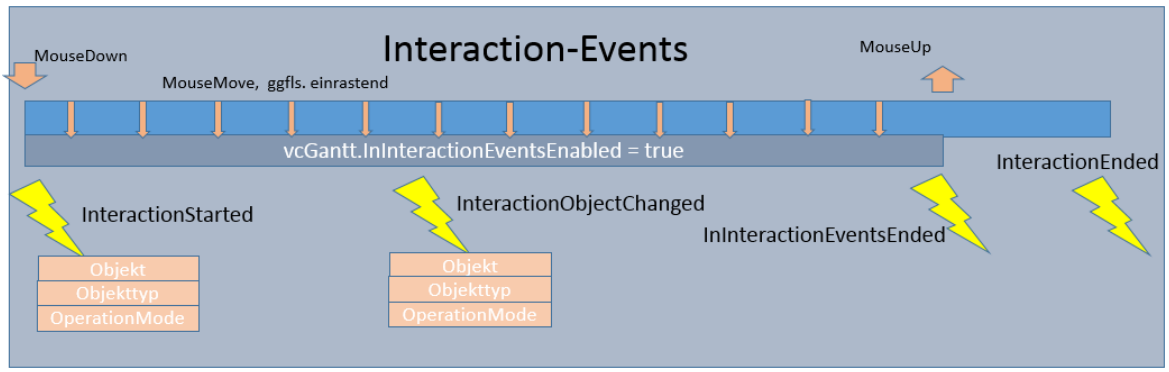
Bei manchen Interaktionen steht initial noch kein Objekt zur Verfügung, z.B. beim Erzeugen eines Knotens oder einer Box. In diesem Fall erscheint der Event **VcInteractionObjectChanged**, sobald intern das Objekt erzeugt wurde. Bei Knoten ist dies der echte Chartknoten.

Ist die Aktion abgeschlossen, wird dies durch den Event **VcInteractionEnded** angezeigt. Hier kann man alles, was man während der Interaktion an zusätzlichen Elementen verwendet hat, abbauen.

Bei Erzeugung von neuen Objekten mit Interaction-Events ist der Ablauf wie folgt:

- VcInteractionStarted
- VcInteractionObjectChanged
- Modifying/Modified Events, die Veränderungen bei der Erzeugung eines Elementes anzeigen
- Creating und Created Events
- VcInteractionEnded

> InInteraction Events während der Interaktion freigeschaltet



Wenn zusätzlich die Interaction Events während der Interaktion erlaubt sind (**vcGantt.InInteractionEventsEnabled = true**), dann gibt es einen weiteren Event, der beim Loslassen der Maustaste das Ende dieser Events anzeigt: **VcInInteractionEventsEnded**.

So können nun leicht die Objekt-Events während der Interaktion von denen am Ende der Interaktion unterschieden werden: Wenn dieser Event geworfen worden ist, so wird der nächste Objekt-Event der abschließende Event sein.

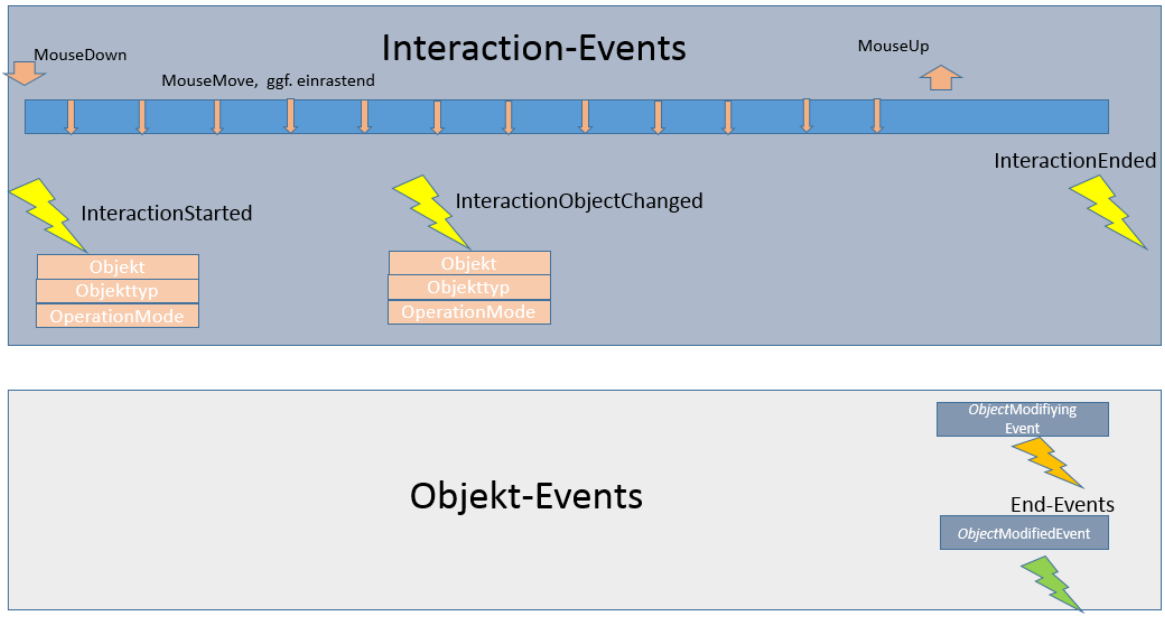
> Mögliche Szenarien

Bei der Verwendung der Interaction Events gibt es also zwei mögliche Zustände:

Steuerung einer Interaktion bei

- deaktivierten InInteraction Events
- aktivierten InInteraction Events

> Zusammenspiel mit den Events der beteiligten Objekte bei deaktivierten InInteraction-Events



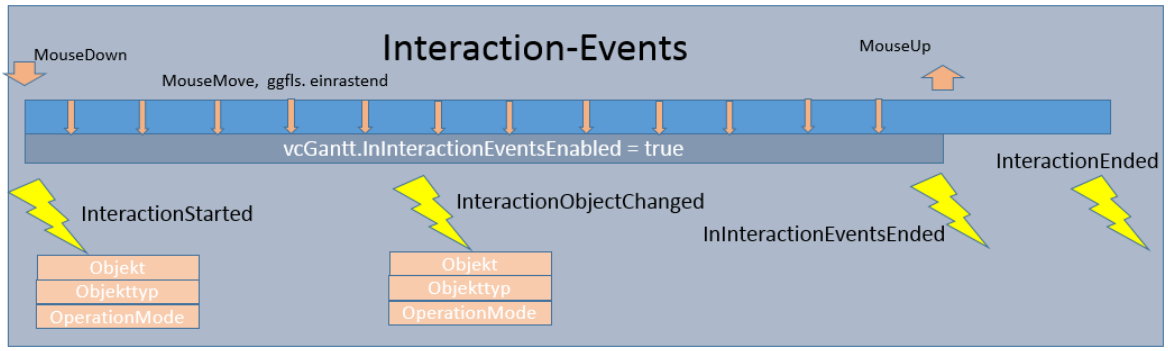
Hier sieht man das Zusammenspiel der Interaction- (gelbe Blitze) und Objekt-Events (ockerfarbene und grüne Blitze) bei deaktivierten InInteraction Events (**vcGantt.InInteractionEventsEnabled = false**):

Die Interaktion wird gestartet, angezeigt durch den Event **InteractionStarted**.

Beim Loslassen der Maustaste kommen zunächst die Objekt-Events, z.B. beim Knoten **VcNodeModifying** und **VcNodeModified1> - hinsichtlich der ObjectEvents also das alte Verhalten, sodass der vorhandene Code in den ObjektEvents nicht geändert werden muss, wenn die InInteractionEvents nicht verwendet werden.**

Danach kommt der Event VcInteractionEnded, der das Ende der Interaktion anzeigt.

> Zusammenspiel mit den Events der beteiligten Objekte bei aktivierten InInteraction-Events



Sind die InInteraction Events in Verwendung, erscheinen folgende Events:

- **VcInteractionStarted** beim Drücken der linken Maustaste
- Modifying- und Modified Events bei der Mausbewegung
- **VcInInteractionEventsEnded** und dann die abschließenden Objekt-Events beim Loslassen der linken Maustaste
- **VcInteractionEnded** um das Ende der Interaktion anzuzeigen

Beispiel1: Verschieben eines Knotens:

Die Interaktion wird gestartet durch Drücken der linken Maustaste, wenn der Mauszeiger auf einem Knoten steht. Es erscheint der Event **VcInteractionStarted**

Die beim Bewegen des Mauszeigers auftretenden Events zeigen den Zustand des Realknotens (**VcNodeModifying**) und bei Updates (**VcNodeModified**) des Chartknotens an.

Wird die Maustaste losgelassen, so erscheint der Event **VcInInteractionEventsEnded1>**

Die danach auftretenden Objekt-Events VcNodeModifying und **VcNodeModified** zeigen den Zustand des Chartknotens am Ende der Interaktion an.

Als letztes erscheint der Event **VcInteractionEnded**.

> Beispiel: Verhalten der Objekt-Events beim Knoten-Aktualisierungsverhalten "Verschieben mit der Maus"

Da der **VcNodeModifying Event** die Modifizierung des EventReturnstatus (e.ReturnStatus) erlaubt, gilt dies nun auch während der Interaktion.

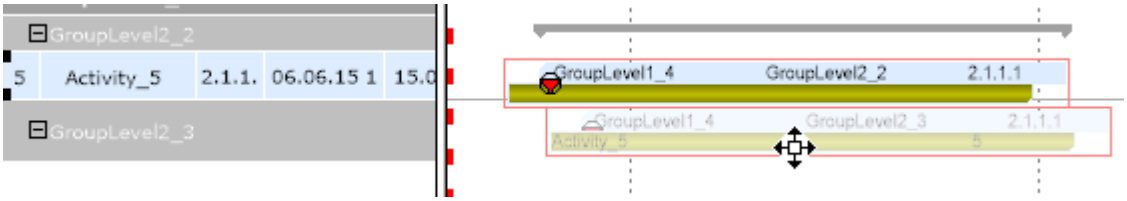
Wird also über e.ReturnStatus = ReturnStatusFalse angezeigt, dass die gelieferten Daten nicht „valide“ sind, so wird im Chart die Objektaktualisierung beim nächsten möglichen Update nicht durchgeführt, ein entsprechender VcNodeModified Event wird nicht geworfen.

Dies wird visualisiert, indem das das Objekt "stehen bleibt" und die aktuelle Position weiter durch das Phantom angezeigt wird.

Bei Objekten mit Realdarstellung (momentan nur Knoten und Knotenboxen) wird der Zustand folgendermaßen dargestellt:

Die aktuelle Position wird durch ein aufgehelltes Real dargestellt; dessen Werte sind auch die, die in den Events weiterhin geliefert werden.

Der letzte valide Zustand, also der, bei dem zum letzten Mal e.ReturnStatus nicht ReturnStatusFalse war, wird durch ein weiteres Real dargestellt, das quasi dort "hängenbleibt". So sind beide notwendigen Informationen visualisiert.



Beim Knoten entsprechen die Werte des letzten validen Zustandes, also des hängengebliebenen Reals, dann dem **e.OldNode** im **VcNodeModifying-Event.**

Falls der letzte **VcNodeModifying**-Event vor dem **VcInInteractionEventsEnded1> Event mit ReturnStatusFalse abgeschlossen wurde, wird in den End-Events der letzte valide Zustand geliefert.**

Dann kann dort noch mal entschieden werden, ob dieser übernommen werden soll oder nicht. Setzt man im End-Event ReturnStatusFalse, wird der originale Startzustand wiederhergestellt.

Tipp: In der Praxis empfiehlt es sich, ein "begleitendes InteractionInfo"-Objekt zu erstellen, das in den Events die notwendigen Informationen zur Interaktion liefert und entsprechend ausgewertet werden kann

3.13 Knoten (Vorgänge)

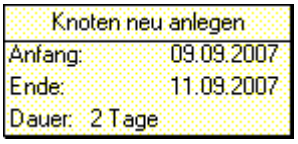
Ein Knoten (Vorgang) entspricht einem Datensatz aus der Maindata-Tabelle. Knoten können über die API geladen oder interaktiv vom Anwender erzeugt werden.

> Knoten erzeugen

Auf der Eigenschaftenseite **Knoten** können Sie festlegen, ob der Anwender

- neue Knoten durch Ziehen mit der Maus erzeugen kann (im **Modus: Knoten erzeugen**) (**Erzeugung neuer Knoten zulassen**)
- neue Knoten durch Doppelklick erzeugen kann (**Erzeugung neuer Knoten durch Doppelklick**)
- neue Knoten direkt im Dialogfeld **Vorgänge bearbeiten** bearbeiten kann (**Erzeugung neuer Knoten mit Dialog**).

Wenn zur Laufzeit ein neuer Knoten durch Ziehen mit der Maus erzeugt wird, erscheint dabei die Infobox **Knoten neu anlegen**, die Ihnen Anfangs- und Endtermin sowie die Dauer des neuen Knotens anzeigt.



Wenn auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Erzeugung neuer Knoten mit Dialog** aktiviert ist, öffnet sich das Dialogfeld **Vorgänge bearbeiten**, sobald Sie einen neuen Knoten erzeugt haben. Hier werden die Daten des interaktiv erzeugten neuen Knotens angezeigt, und Sie können sie nun bearbeiten.

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

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

> Knoten löschen

Zur Laufzeit können Sie Knoten löschen, indem Sie den Cursor auf dem betreffenden Knoten positionieren und die rechte Maustaste drücken. Es erscheint folgendes Kontextmenü:

Wählen Sie die Option **Knoten löschen**.

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

Sie können Knoten auch über die API mit der VcGantt Methode **DeleteNodeRecord** löschen.

> Weitere Festlegungen für Knoten

Auf der Eigenschaftenseite **Knoten** können Sie außerdem Folgendes festlegen:

- die Datenfelder, in die die Daten für Start, Ende und Dauer von interaktiv angelegten Vorgängen geschrieben werden sollen.
- ob arbeitsfreie Zeiten hervorgehoben werden sollen (Bei Rechtecklayern werden arbeitsfreie Zeiten dann durch eine durchgezogene Linie angezeigt.)
- ob den Knoten Kalender zugewiesen werden sollen (Die Kalenderzuweisung wirkt sich zum einen beim Verschieben von Vorgängen aus: Anfang und Ende der Vorgänge werden nicht auf arbeitsfreie Tage gelegt. Zum anderen werden beim Berechnen der Dauer von Vorgängen die arbeitsfreien Zeiten berücksichtigt. Derzeit ist standardmäßig ein Fünf-Tage-Kalender definiert ("WeekCalendar").)
- das Datenfeld, das den Namen des für einen Knoten zu verwendenden Kalenders enthalten soll, wenn ein individueller Kalender gefordert wird.
- ob der Anwender mehrere markierte Knoten gemeinsam verschieben können soll
- ob ein markierter Knoten als Ganzes (d. h. mit all seinen Layern) verschoben werden können soll.

> Ereignisse

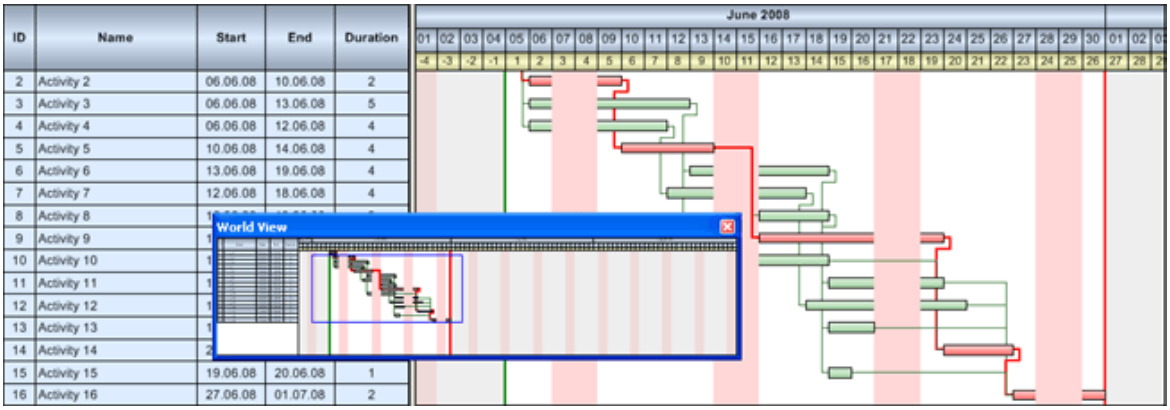
Sie können auf die folgenden Ereignisse reagieren:

- VcNodeCreating
- VcNodeCreated

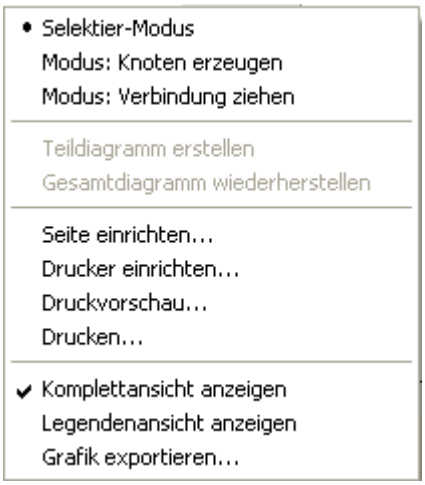
- VcNodeDeleting
- VcNodeLeftClicking
- VcNodeLeftDoubleClicking
- VcNodeModified
- VcNodeRightClicking
- VcNodesMarked
- VcNodesMarking

3.14 Komplettansicht (World View)

Die Komplettansicht ist ein zusätzliches Fenster, in dem das komplette Diagramm und ggf. das Histogramm angezeigt wird. Jeweils ein Rahmen zeigt an, welchen Diagramm- bzw. Histogrammausschnitt das Hauptfenster gerade anzeigt. Wenn Sie mit der Maus einen dieser 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 entsprechenden Rahmens, 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.

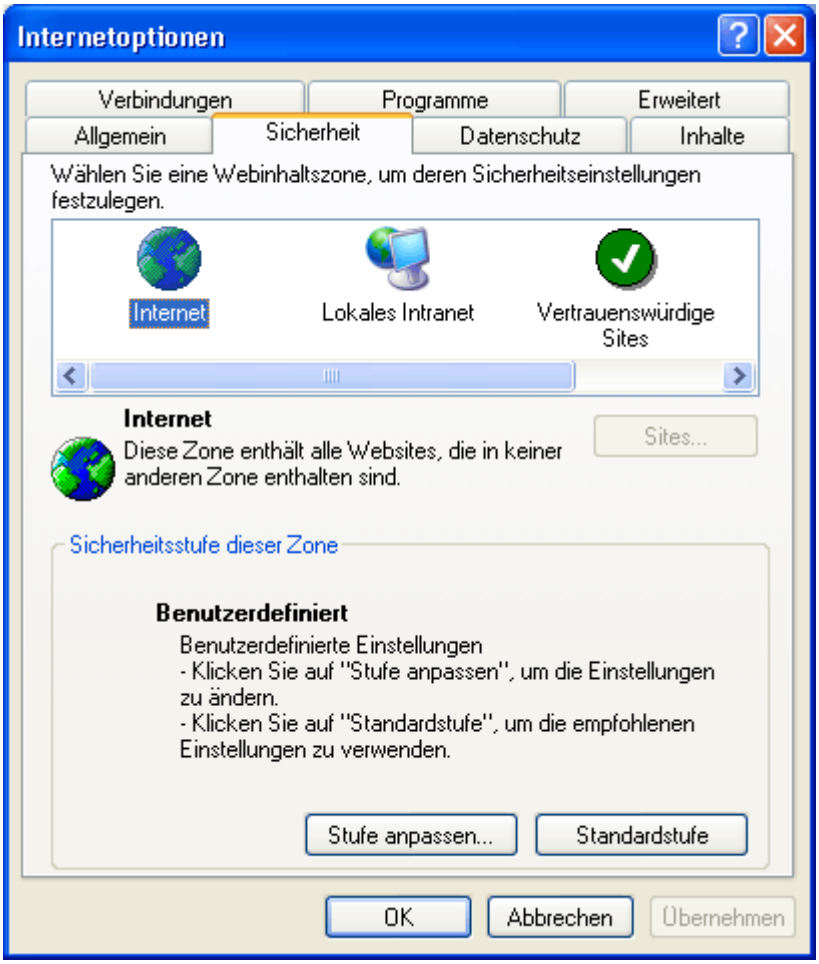
3.15 Laufzeitsicherheitsrichtlinien für den Einsatz im Internet Explorer

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

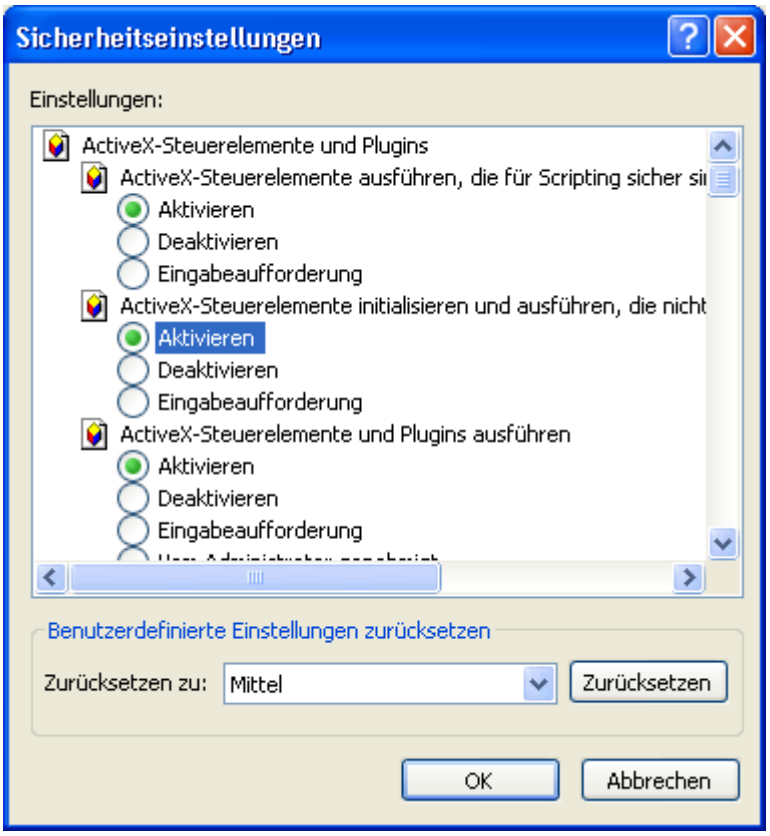
Sobald der Browser das Steuerelement von einem Webserver im Internet lädt, werden die **Laufzeitsicherheitsrichtlinien** der Internet_Zone wirksam. Die Standardeinstellungen verhindern die Ausführung des Steuerelements. Grundsätzlich muss der Internet Explorer das Ausführen von .NET Komponenten zulassen, 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-Steuerelemente 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 Steuerelement mit der erforderlichen Mindestmenge an Berechtigungssätzen auf einer HTML-Seite im Internet Explorer ausgeführt werden.

3.16 Layer

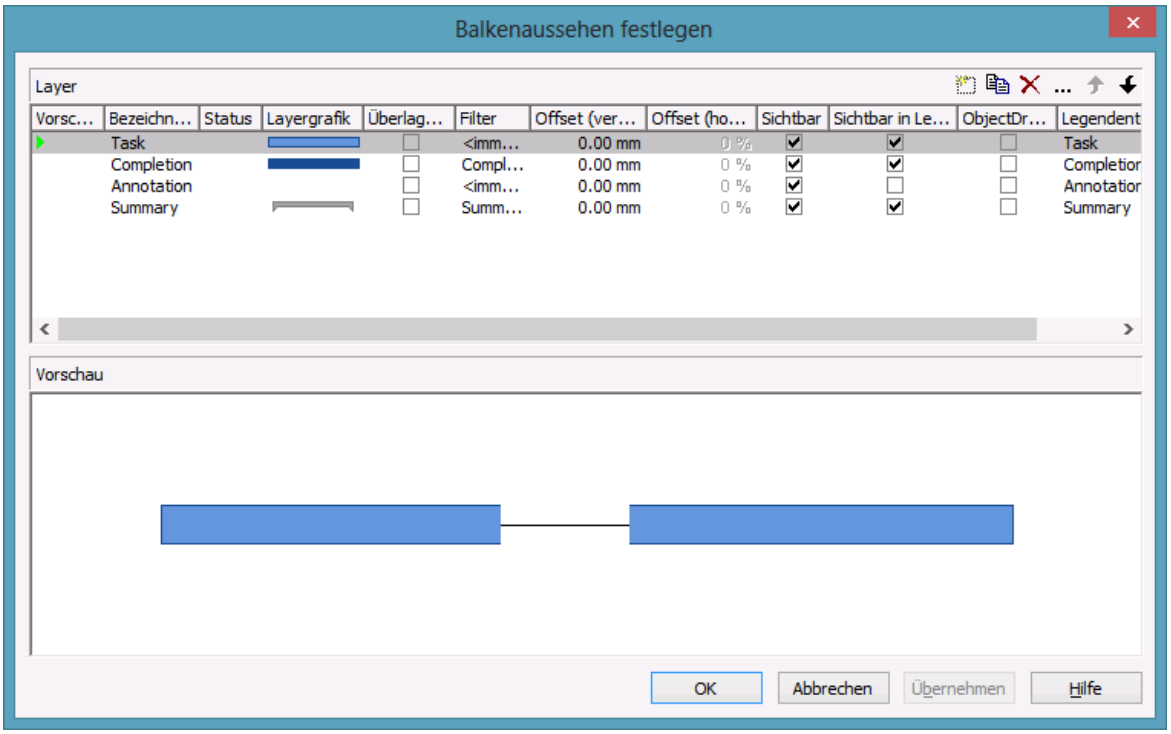
Ein Layer stellt einen Termin (Symbollayer oder Bitmaplayer) oder ein Terminpaar (Rechtecklayer, keilförmige Layer oder Linienlayer) grafisch dar.

Vorgänge (Knoten) werden durch einen oder mehrere Layer grafisch festgelegt. Wenn ein Vorgang aus mehreren Layern besteht, werden die einzelnen Layer dieses Vorgangs grafisch schichtweise übereinander gelegt. Dabei wird mit dem Layer der niedrigsten Priorität angefangen. Der Layer mit der höchsten Priorität liegt zuoberst.

Jedem Layer wird ein Filter zugewiesen, der angibt, unter welchen Bedingungen Vorgänge durch diesen Layer dargestellt werden.

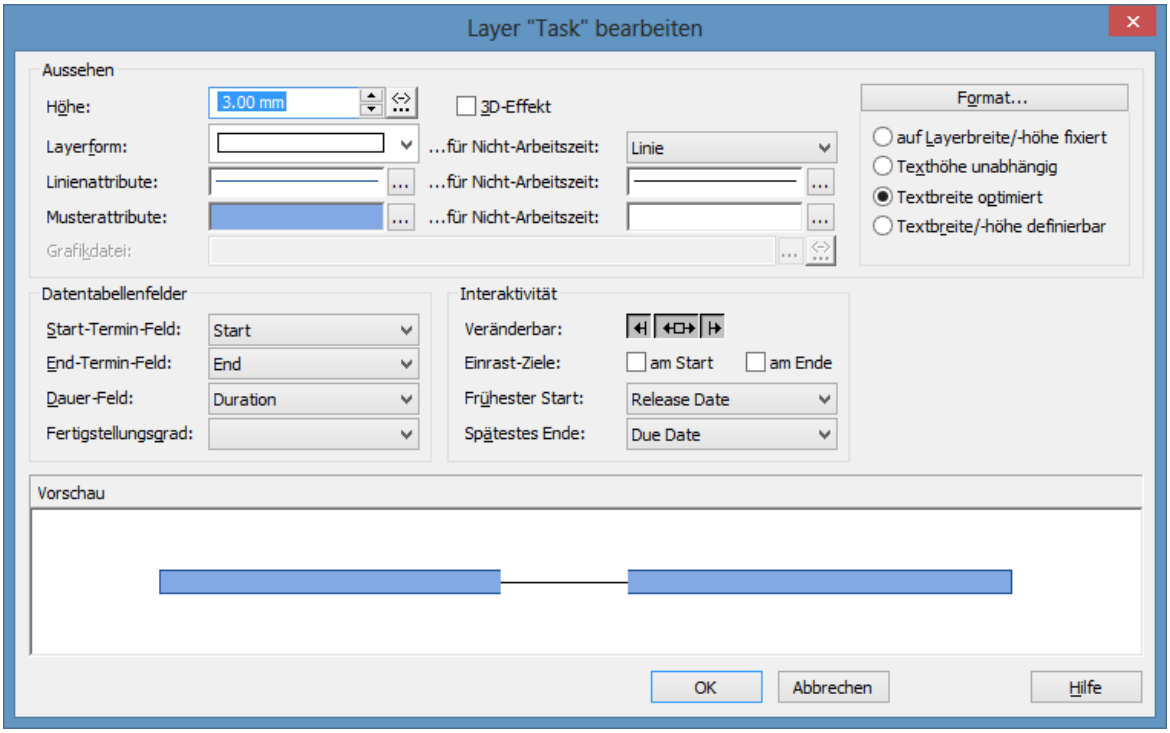
Die Layer können sich hinsichtlich ihres Musters, ihrer Hintergrund- und Musterfarbe und/oder ihrer Beschriftung unterscheiden. Außerdem können sie unterschiedliche Höhen und einen horizontalen und/oder vertikalen Offset haben. Dadurch können Sie sicherstellen, dass alle Layer, die einem Knoten zugeordnet sind, sich unterschieden und sichtbar bleiben.

Sie können im Dialog **Balkenaussehen festlegen** beliebig viele Layer definieren. Alle vorhandenen Layer werden hier, nach Zeichnungsreihenfolge sortiert, angezeigt.



Mit Hilfe der Schaltflächen oberhalb der Layer-Tabelle können Sie Layer hinzufügen ( ), kopieren ( ), löschen ( ) oder bearbeiten ( ).

Um einen Layer zu bearbeiten, wählen Sie einen Layer aus der Liste aus und klicken Sie auf die Schaltfläche **Layer bearbeiten** (...) oder klicken doppelt auf die **Layergrafik**. Das Dialogfeld **Layer bearbeiten** öffnet sich. Hier können Sie die grafischen Attribute für den gewählten Layer einstellen.

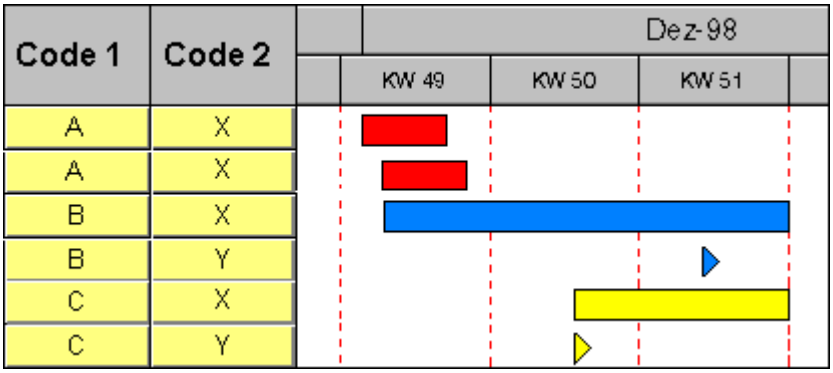


> Filter für Layer

Durch die Verwendung von Filtern erreicht man, dass ein Layer nur einer ausgewählten Menge von Knoten zugewiesen wird, in Abhängigkeit von den Daten.

Um einen Filter zu bearbeiten, klicken Sie im Dialog **Balkenaussehen festlegen** auf das **Filter**-Feld. Es erscheinen zwei Schaltflächen. Klicken Sie auf die **Bearbeiten**-Schaltfläche, um den **Filter verwalten**-Dialog zu öffnen. Von hier können Sie das Dialogfeld **Filter bearbeiten** öffnen, wo Sie die Filterkriterien bearbeiten können. (Siehe hierzu: "Wichtige Begriffe: Filter").

Im folgenden Beispiel ist für den Wert Y im Feld "Code2" ein Rechtecklayer und für "Y" ein Symbollayer definiert. Die Farbgebung erfolgt durch Mapping über das Feld "Code1".



> Layerformen

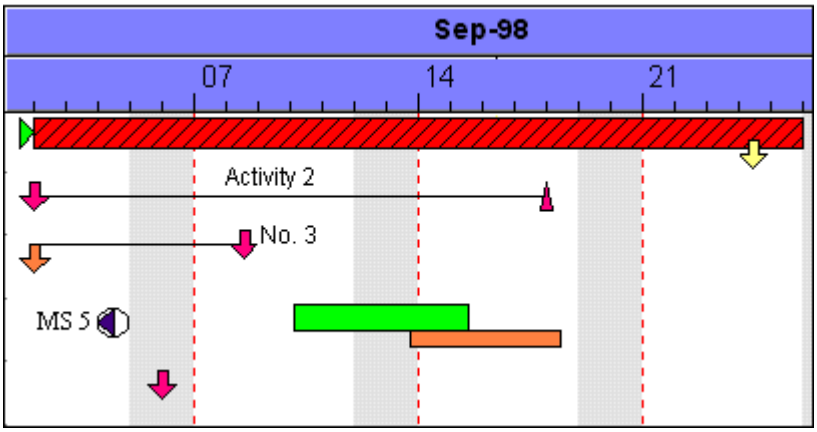
Es gibt verschiedene Layerformen: Rechteck-Layer, keilförmige Layer, Linienlayer, Symbol-Layer, Bitmap-Layer und den "unsichtbares Symbol"-Layer. Die Layer-Form wird im Dialogfeld **Layer bearbeiten** unter **Layer-Form** ausgewählt.

Symbol-Layer werden zur Darstellung spezieller Termine verwendet. Eine Reihe von Symbol-Layern ist vordefiniert. Zusätzlich können Sie Bitmap-Layer selbst definieren, um Ihre eigenen Grafiken (z. B. Firmenlogos) als Layer zu verwenden. Unter **Grafikdatei** können Sie die zu verwendende Bitmap-Datei auswählen.

Terminpaare werden durch Rechteck-Layer, keilförmige Layer oder Linien-Layer grafisch dargestellt. Keilförmige Layer können verwendet werden, um ansteigende bzw. ausklingende Aktivitäten darzustellen, beispielsweise am Anfang bzw. am Ende von Vorgängen.

Der Layer **unsichtbares Symbol** ist mit Ausnahme seiner Beschriftung unsichtbar und wird nicht in der Legende dargestellt. Er kann zur zusätzlichen Beschriftung von Vorgängen verwendet werden.

Die Kombination von Layer-Formen, Hintergrundfarben, Mustern, Beschriftungen und Filtern ermöglicht eine Vielfalt möglicher Layer. Einige Beispiele zeigt die folgende Abbildung:



> Fertigstellungsgrad

Sie können sich den prozentualen Fertigstellungsgrad jedes Vorgangs auf einen Blick zeigen lassen.

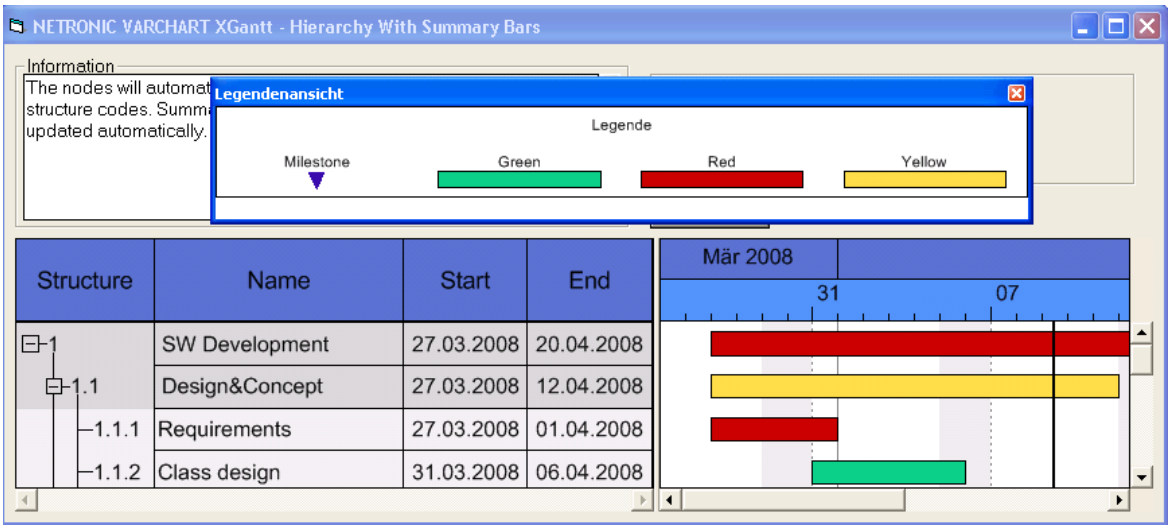
Gehen Sie dazu folgendermaßen vor: Erstellen Sie einen Layer "Fertiggestellt" und bearbeiten Sie ihn im Dialog **Layer bearbeiten**. Hier können Sie für Rechtecklayer und keilförmige Layer das Datenfeld auswählen, das den prozentualen Fertigstellungsgrad eines Vorgangs enthält.

Wählen Sie für den Layer "Fertiggestellt" unter **Fertigstellungsgrad** beispielsweise das Datenfeld "abgeschlossen %" aus, das einen numerischen Wert zwischen 0 und 100 enthalten muss. Aus den Terminen und dem Fertigstellungsgrad wird nun ein neuer Endtermin berechnet. Vereinbaren Sie nun die grafischen Attribute des Layers so, dass der Layer "Fertiggestellt" deutlich von dem Standard-Layer zu unterscheiden ist, beispielsweise durch ein Schraffurmuster oder eine andere Farbe.

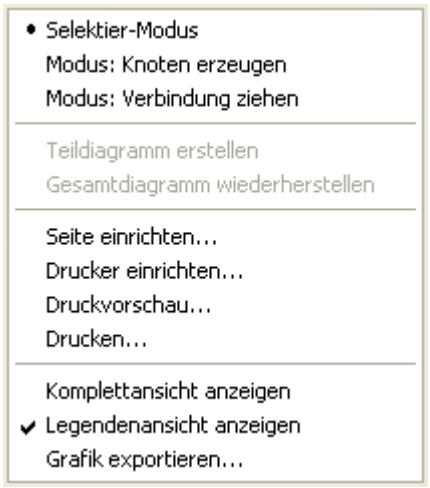
Vorgang, der zu 90 % fertiggestellt ist

3.17 Legendenansicht (Legend View)

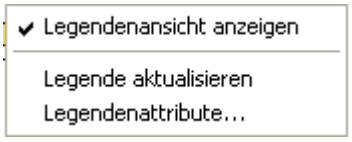
Die Legendenansicht ist ein zusätzliches Fenster zur Darstellung einer Legende auf dem Bildschirm. Das Aussehen der Legende wird festgelegt im Dialog **Legendenattribute**, der über die Eigenschaftenseite **Außenbereich** zu erreichen ist, oder über die Legendenattribute des Objektes **VcBorderBox**.



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 **VcGantt.VcLegendView** festlegen.

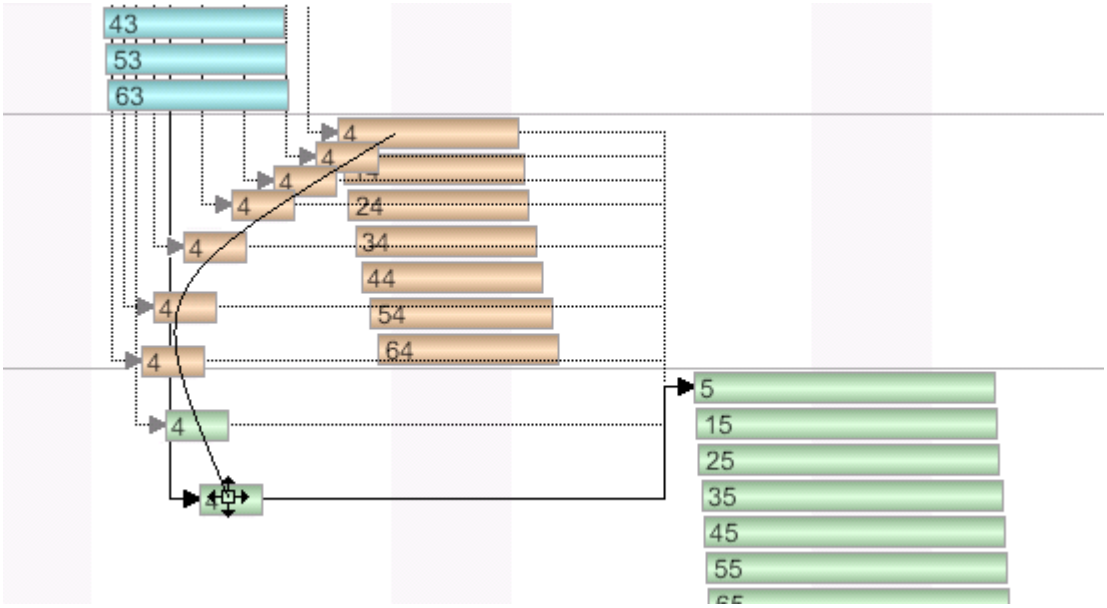
3.18 Live Update

Was ist Live Update?

Mit der Live Update Funktionalität, neu ab der Version XGantt 5, werden Auswirkungen von Mausinteraktionen bereits während der Interaktion visualisiert und nicht erst nachdem sie beendet sind.

Bislang wurde bei Interaktionen mit Phantomen gearbeitet und der Gantt-Graph gab Rückmeldung über die Konsequenzen der Aktion für die Gesamtplanung, sobald sie durch Loslassen der Maustaste an der gewünschten Stelle beendet war. Durch Live Update kann man jetzt schon während der Interaktion erkennen, welche Änderungen aus einer Mausaktion resultieren, da mit jeder Mausaktion die Ansicht aktualisiert wird, d.h. die Änderungen werden stetig wiederholt direkt auf dem Objekt ausgeführt und führen damit zu einem Live Update des Objektes und der Grafik.

Während der Verschiebeinteraktion sieht man also zu jedem Zeitpunkt eine zur jeweiligen Mausposition passende Visualisierung des Knotens mit den angehängten Links.



Arten von Daten-Änderungen

Bei der Änderung von Daten und ihrer anschließende Auswertung unterscheidet man zwischen zwei Arten:

- Änderungen, deren Auswirkungen sich nur auf das einzelne Objekt beziehen, z.B. einfache Datumsänderung,nachfolgend **individuelle**

Änderungen genannt. Individuelle Änderungen treten bei jeder Interaktion auf.

- Änderungen, deren Auswirkungen sich nicht nur auf das einzelne Objekt beziehen, sondern bei denen ganze Strukturen verändert werden können, z.B. Gruppierung oder Optimierung, nachfolgend **strukturelle Änderungen** genannt.1a>
- Strukturelle Änderungen können z.Zt. nur beim Verschieben von Knoten und Gruppen auftreten, denn nur Knoten und Gruppen können in Strukturen zusammengefasst und angeordnet werden.
- Strukturelle Änderungen werden timer-gesteuert vorgenommen (s. weiter unten **Timer-gesteuertes Live-Update**. Ein **OldNode** bzw. **PreviewNode** ist nicht vorgesehen.
- Nach Durchführung einer strukturellen Änderung wird der verschobene Knoten automatisch wieder unter den Cursor gescrollt (Node-Tracking).

Betroffene Aktionen

Vom Live Update betroffen sind Verschiebe-Aktionen von Knoten und Gruppen sowie interaktives Anlegen von Knoten und Links.

> Verschieben von Knoten und Links im Diagramm

Knoten und Links können optisch frei verschoben werden. Dabei wird sowohl die horizontale als auch vertikale Lage des Knotens immer an der Mausposition ausgerichtet, sodass der verschobene Knoten zu jedem Zeitpunkt unter dem Mauscursor steht. Angehängte Links, die mit Linkrouting <orthogonal> oder <straight> gezeichnet werden, werden entsprechend mitgeführt. Das Linkrouting <distinguish> funktioniert hier nicht, es wird dann <orthogonal> verwendet. Bei der Positionsänderung wird auch die Visualisierung des Knotens und der Links ständig aktuell gehalten, d.h. Filterung (=>Layer) und Mapping werden auf das gesamte Konstrukt angewendet. An der bisherigen Knotenposition bleibt ein leerer Bereich, was den Verschiebeeffekt noch verstärkt. Man zieht optisch den Knoten von seinem bisherigen Platz weg. Dazu wird der Knoten mit seinen Links auf **VC_VISIBILITY=VC_NO** gesetzt. Kopien vom Knoten und den Links werden angelegt und beim Verschiebevorgang aktualisiert.

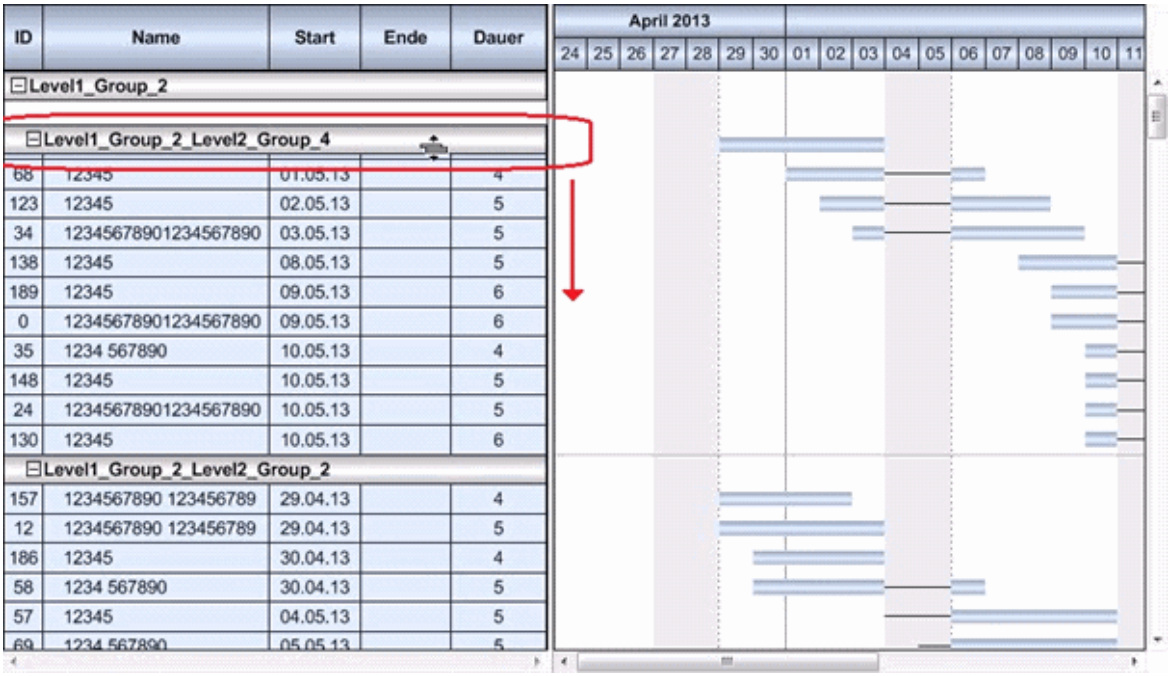
> Verschieben von Gruppen

VARCHART XGantt erlaubt dem Anwender, Gruppen innerhalb ihrer Ebene interaktiv zu verschieben. Dies geschieht, indem er im Diagramm entweder den Summenbalken oder Gruppenknoten vertikal verschiebt oder in

der Tabelle das jeweilige Tabellenformat vertikal verschiebt. Diese Strukturänderung entspricht einer manuellen Umsortierung, zu der es kein datenmäßiges Äquivalent gibt; es findet also keine Datenänderung statt.

Nach Durchführung der Änderung wird der verschobene Summenbalken/Gruppenknoten bzw. das verschobene Tabellenformat automatisch wieder unter den Cursor gescrollt. Dieses Scroll-Verhalten wird hier als Group-Tracking bezeichnet.

Im Diagrammbereich wird ein VARCHART-Knotenphantom mit Real-Darstellung des Summenbalkens/Gruppenknotens verwendet; im Tabellenbereich ein VARCHART-Knotenphantom mit Real-Darstellung der Tabellenbox. Da bei dem Verschiebevorgang keine Datenänderung stattfindet, bleibt die Real-Darstellung unverändert.



Timer-gesteuertes Live Update

Die gesamte Grafik wird durch die ständigen (ggf. umfangreichen) visuellen Änderungen sehr unruhig und der direkte Zustandswechsel ohne die Möglichkeit von Animationen kann verwirrend oder sogar störend wirken.

Daher ist eine Alternative zum direkten Zustandswechsel sinnvoll. Bei strukturellen Änderungen soll die Grafik nicht ständig aktualisiert werden, sondern gesteuert über einen Timer: Wenn der Anwender während der Interaktion mit der Maus einen Moment verharrt, wird erst nach kurzer, aber signifikanter Wartezeit die strukturelle Änderung durchgeführt und die Grafik aktualisiert. Der Anwender sieht dann eine zur jeweiligen Mausposition passende Grafik. Nun kann er die Interaktion fortsetzen, denn

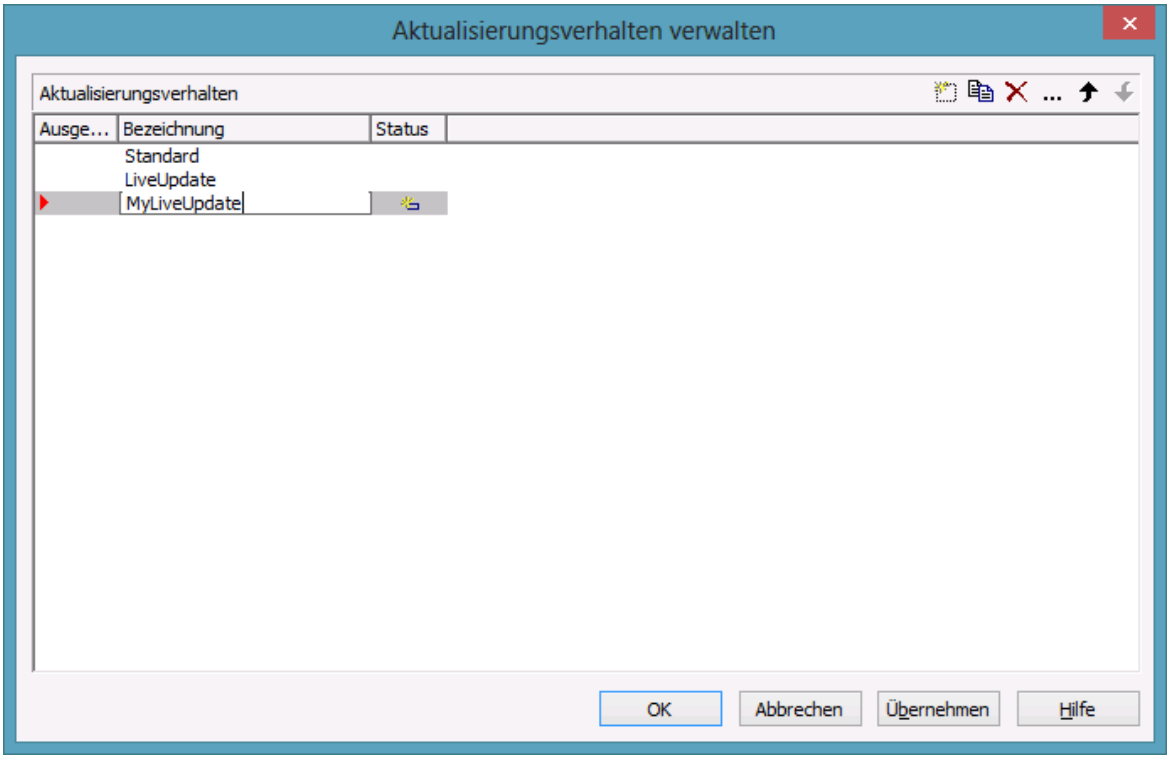
er bewegt die Maus noch immer mit gedrückter Taste. Dabei werden die strukturellen Änderungen wieder zurückgehalten bis er erneut verharrt. Wiederum werden die strukturellen Änderungen erst nach kurzer, aber signifikanter Wartezeit durchgeführt und die Grafik aktualisiert. Der Prozess wiederholt sich bis zum Ende der Interaktion (Loslassen der Maustaste). Bei diesem Verfahren bleibt die Grafik während der Interaktion relativ ruhig.

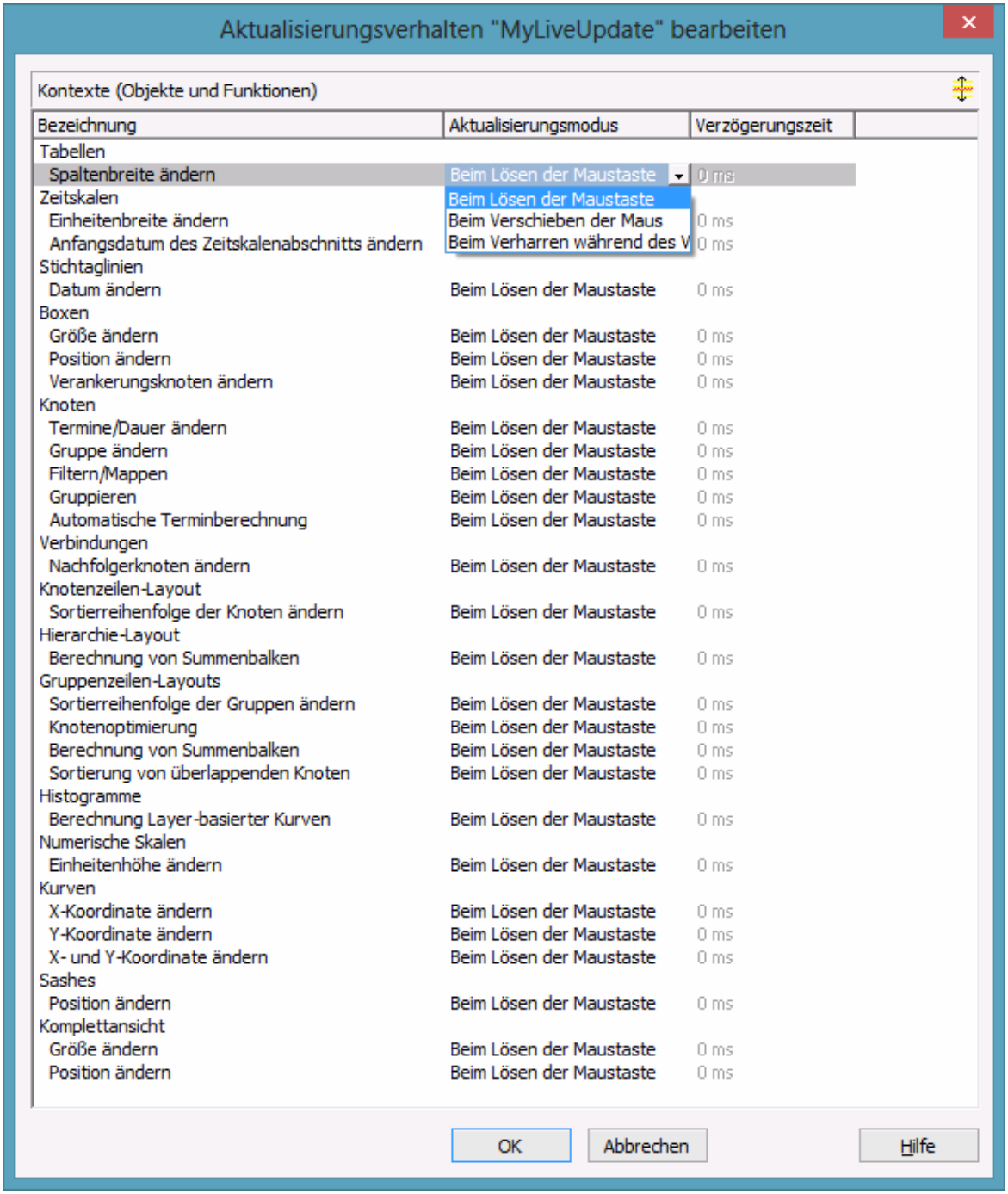
Live-Update in VARCHART XGantt einrichten

> Zur Designzeit

Die Einstellungen für das Live-Update werden zur Designzeit in den beiden Dialogen **Aktualisierungsverhalten verwalten** und **Aktualisierungsverhalten bearbeiten** vorgenommen. VARCHART XGantt verfügt bereits über die beiden Aktualisierungsverhalten **Standard** und **Live Update**, deren Einstellungen **nicht** vom Benutzer geändert werden können.

Es gibt jedoch die Möglichkeit, über die unten dargestellten Dialoge eigene Aktualisierungsverhalten zu erstellen, in denen dann individuelle Einstellungen vorgenommen werden können.





Hinweis: Bitte beachten Sie, dass die Zuweisung von individuellen Aktualisierungsverhalten für datengetriebene Objekte (Knoten, Links und Gruppen) **ausschließlich** über die API erfolgt.

> Zur Laufzeit

Die Einstellungen erfolgen über die Objekte:

- VcBox

- VcCurve
- VcDateLine
- VcGantt
- VcGroup
- VcLinks
- VcNode
- VcNumericScale
- VcTable
- VcTimeScale
- VcUpdateBehavior
- VcUpdateBehaviorCollection
- VcUpdateBehaviorContext
- VcWorldView

Weitere Informationenfinden Sie in der API-Referenz dieses Handbuchs.

3.19 MultiState-Felder

Was sind MultiState-Felder?

Im Tabellenteil gibt es die Möglichkeit, mit grafischen Feldern und Zuordnungstabellen verschiedene Datenfeldinhalte als Grafiken darzustellen. MultiState-Felder stellen eine Erweiterung dieses Konzepts dar, bei denen jeder Klick auf eine Grafik zum Zustandswechsel des zugehörigen Datenfeldes führt. Es handelt sich also um eine bequeme Möglichkeit, Datenfelder, die eine endliche Anzahl von Zuständen annehmen können, zu editieren. Deshalb funktionieren MultiState-Felder auch nur, wenn das Modul **Data Editing** lizenziert ist.

> Wie sie funktionieren

Mit jedem Klick wird zyklisch in der Zuordnungstabelle nach der nächsten Grafik gesucht, welche sich von der aktuellen Grafik unterscheidet. Der entsprechende Wert (Schlüssel aus der Zuordnungstabelle) wird dann dem Datenfeld zugewiesen. Falls neben der Zuordnungstabelle auch noch eine weitere Grafik (Default-Grafik) angegeben ist, wird diese ebenfalls beim zyklischen Durchlauf durch die Grafiken berücksichtigt. Erscheint nach einem Klick die Default-Grafik, dann wird als Wert der leere String ins Datenfeld eingetragen. Ansonsten erscheint die Default-Grafik immer dann, wenn im Datenfeld ein Wert steht, der nicht in der Zuordnungstabelle als Schlüssel vorkommt.

Die einfachste denkbare Anwendung von Multistate-Feldern sind boolesche Datenfelder, bei denen die Werte **true** und **false** beispielsweise durch ein Checkbox-Symbol mit bzw. ohne gesetzten Häkchen visualisiert werden. Bei jedem Klick auf das gerade sichtbare Symbol wechselt die Ansicht zum anderen Symbol, und damit ändert sich auch der Wert des zugehörigen Datenfeldes von **true** nach **false** bzw. umgekehrt.

> Tipps für die Programmierung

- Positionieren Sie in der Zuordnungstabelle Schlüssel, die auf ein und dieselbe Grafik verweisen, unmittelbar hintereinander. Nur so ist gewährleistet, dass dieselbe Grafik bei einem Zyklus durch die Zuordnungstabelle nur einmal angezeigt wird. Denn bei jedem Klick wird die nächste Grafik gewählt, die ungleich der aktuell gezeigten Grafik ist. So können Sie z.B. in der Map die Schlüssel **true**, **t** und **True** mit derselben Grafik verbinden. Wird diese Grafik angezeigt, dann erscheint beim

nächsten Klick in jedem Fall eine andere Grafik; es wird also nicht dreimal hintereinander dieselbe Grafik gezeigt.

- Aus dem gleichen Grund sollten Sie auch in der Zuordnungstabelle alle Schlüssel an den Anfang setzen, die auf dieselbe Grafik verweisen wie die Default-Grafik.
- Wenn eine Grafik mehrmals hintereinander in der Zuordnungstabelle vorkommt, dann wird beim Klicken als Wert in das Datenfeld immer der erste Schlüssel gesetzt. Stehen z.B. **true**, **t** und **True** hintereinander in der Map (mit gleicher Grafik), dann wird immer **true** und nicht **t** oder **True** als Wert ins Datenfeld eingetragen.
- MultiState-Felder ändern ihren Zustand nur, wenn das Editiern zugelassen ist (s. entsprechende VcGantt-Eigenschaften **InPlaceEditing-OnGroupsInDiagramEnabled, InPlaceEditingOnGroupsInTable-Enabled, InPlaceEditingOnNodesInDiagramEnabled, InPlace-EditingOnNodesInTableEnabled**).
- Damit die Grafiken bei verschieden hohen Tabellenzeilen nicht unterschiedlich groß dargestellt werden, sollte man die Höhe des Grafikfeldes auf einen festen Wert ungleich 0 mm setzen (s. Dialog **Tabellenformat bearbeiten** auf den Eigenschaftenseiten von XGantt).

Für weitere Informationen bzgl. der Grafikdateien und Zuordnungstabellen sei auf die Kapitel **Der Dialog "Tabellenformat bearbeiten"** und **Zuordnungstabellen** im Benutzerhandbuch sowie auf die Beschreibung der VcGantt-Eigenschaft **FilePath** im Referenzhandbuch verwiesen.

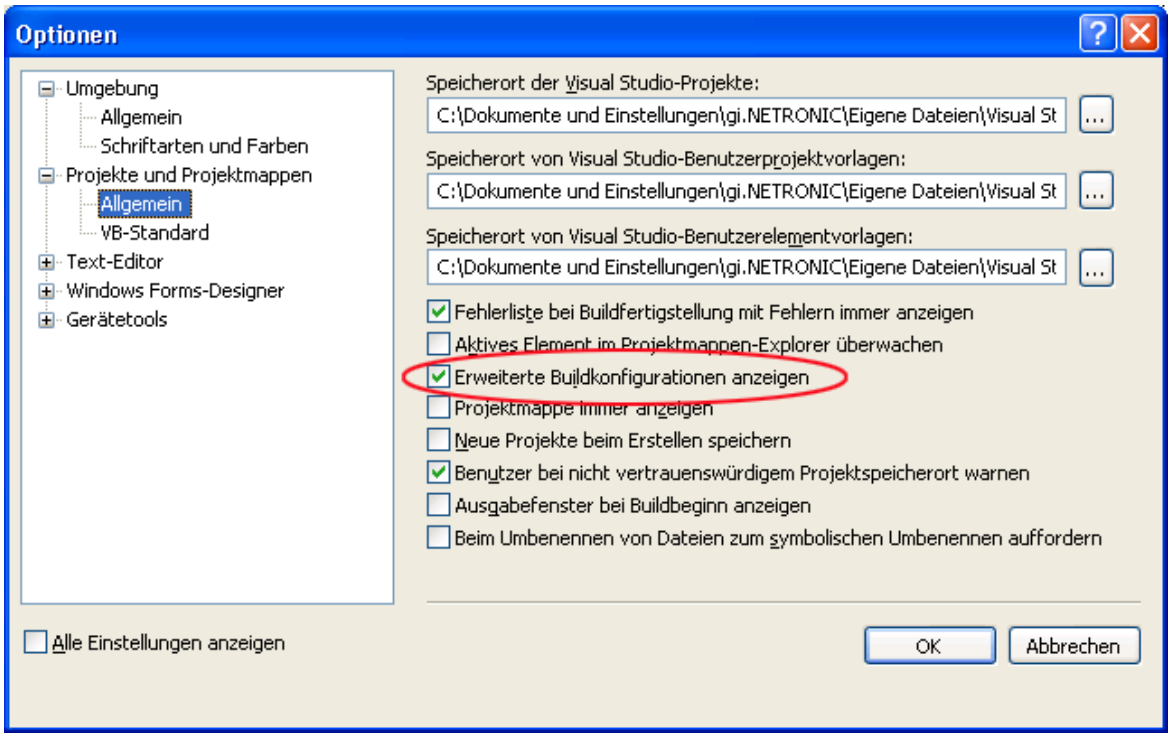
3.20 Plattformen x86 und x64

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 (Assemblys 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.

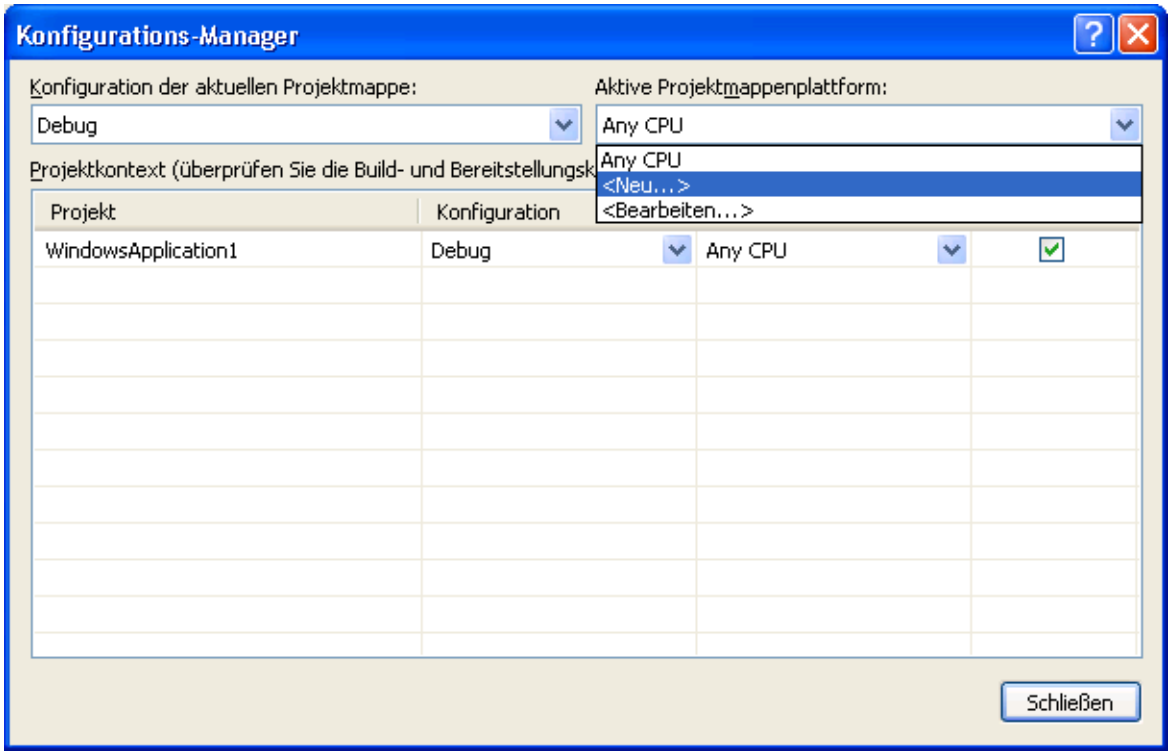
VARCHART XGantt liegt intern teilweise in reinem Maschinencode vor, was unter .NET als **Mixed Mode** bezeichnet wird. Daher muss XGantt für jeden Prozessor, mit dem es verwendet werden soll, neu übersetzt werden. Es sind Varianten für x86-Prozessoren und ab Version 4.3 auch für x64-Prozessore verfügbar.

Applikationen, in denen VARCHART XGantt 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.

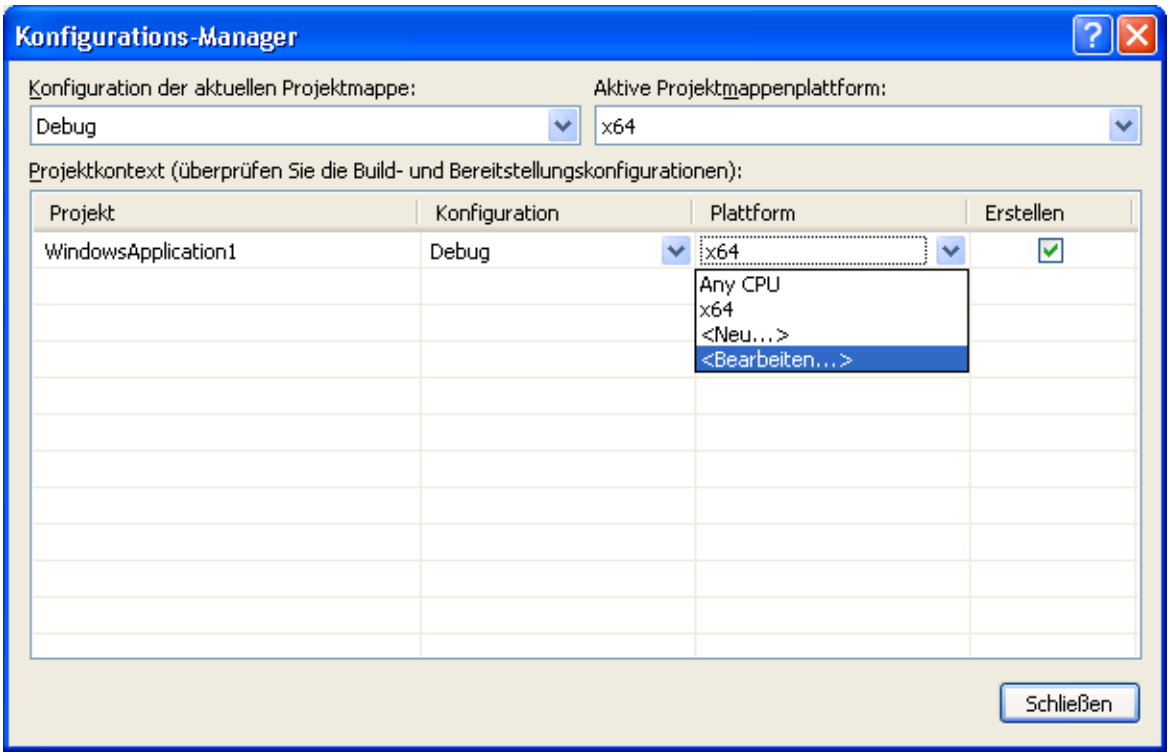


Zum Erstellen einer neuen Plattform wählen Sie bitte im Konfigurationsmanager den Eintrag <Neu> bei **Aktive Projektmappenplattform**.

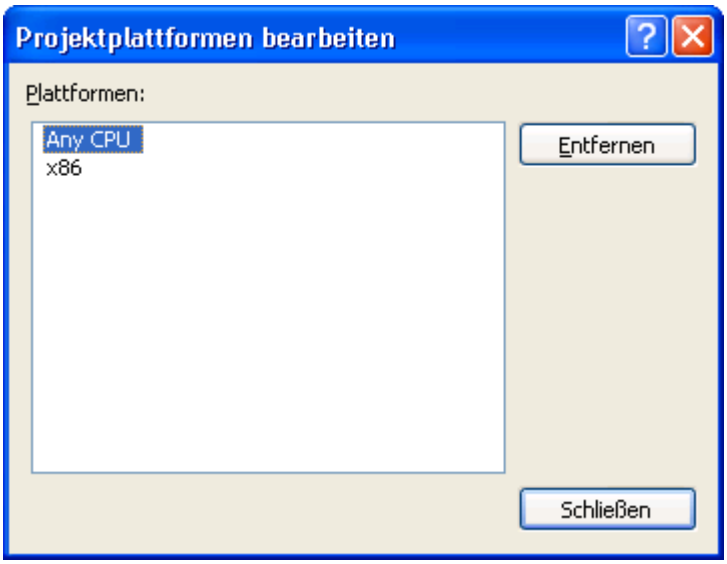


Folgender Dialog erscheint, in dem Sie nun die gewünschten Plattformen **x86** oder **x64** anlegen können:

Wenn Sie anschliessend die Plattform **Any CPU** löschen möchten, wählen Sie bitte zunächst im **Projektkontext** des Konfigurations-Managers in der Spalte **Plattform** den Menpünkt **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 XGantt 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 XGantt-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.XGantt.dll im neu entstandenen Verzeichnis C:\XGanttReference ein (evtl. müssen Sie vorher eine bestehende Referenz auf das XGantt-Installationsverzeichnis entfernen) und kompilieren Ihr Projekt noch einmal.

3.21 Ressourcenplanung

Der neue ResourceScheduler2 stellt eine wesentliche Erweiterung des ResourceScheduler1 aus Version 3.1 dar. Es werden die für eine Ressourcenplanung benötigten Objektarten nun in eigenen Datentabellen erwartet, was erst mit VARCHART XGantt 4.0 möglich wird. Im ResourceScheduler1 waren dagegen Aufgaben, Operationen, Zuweisungen und Ressourcen implizit in der Maindata-Tabelle zu definieren.

Zur Verfügung stehen die folgenden Objektarten, die jeweils in einer eigenen Datentabelle erwartet werden (bei Ressourcen können es bis zu 25 sein):

- **Aufgaben** (Tasks): Diese Objekte bilden eine Gruppierung von Operationen (s.u.) und halten elementare Eigenschaften wie das Freigabe- und das Fälligkeitsdatum sowie Priorität und Aufgabenmenge
- **Operationen**: Diese Objekte sind über Zuweisungen (siehe unten) auf Ressourcen (siehe unten) einplanbar und erhalten durch die Ressourcenplanung als Ergebnis Start- und Enddatum des Ausführungszeitraums. Operationen besitzen eine definierte Reihenfolge in der zugehörigen Aufgabe und können als bereits begonnen gekennzeichnet werden. Außerdem ist es möglich, mehrere einander ausschließende Operationsfolgen (sogenannte Routen) als Alternativen vorzugeben. Alle Operationen der von der Ressourcenplanung gewählten Route werden gemeinsam eingeplant.
- **Ressourcen**: Diese Objekte besitzen als Hauptmerkmal eine Kapazitätskurve und nach der Ausführung einer Ressourcenplanung auch eine Belegungskurve. Außerdem können Ressourcen für eine auf ihr eingeplanten Operation zeitdauerbestimmend sein, wovon augenblicklich einer Operation immer eine zugewiesen sein muss, um die Operation einplanen zu können. Daneben können einer Operation beliebig viele weitere Arbeits- und Material-Ressourcen zugewiesen sein. Weiteres Wesensmerkmal einer zeitdauerbestimmenden Ressource ist die Fähigkeit zur ein- oder mehrstufigen Gruppierung, wobei eine solche Ressource auch mehreren Gruppen gleichzeitig angehören kann.
- **Zuweisungen** (Assignments): Diese Objekte sind die eigentliche Verknüpfung zwischen Operationen und Ressourcen, wobei hier ein Übersetzungsfaktor für die Aufgabenmenge angegeben werden kann. Im Falle von zeitdauerbestimmenden Ressourcengruppen werden Zuweisungen von der Ressourcenplanung entsprechend als solche gekennzeichnet und konkrete Ersatzzuweisungen zu der ausgewählten Einzelressource erzeugt, so dass die Einplanung nachvollziehbar und im XGantt darstellbar wird.

- **Verbindungen** (Links): Diese Objekte beschreiben die Reihenfolge von Aufgaben, das heißt, vorangehende Aufgaben müssen beendet sein, bevor nachfolgende Aufgaben begonnen werden dürfen.

Überblick über die Objekte und die zugehörigen Eigenschaften

Aufgabentabelle	
TaskDataTableName	Name der Aufgabentabelle
TaskDueDateFieldIndex	Datum, bis zu dem eine Aufgabe beendet sein muss
TaskPlanningStrategyFieldIndex	Planungsstragie: ASAP oder JIT für einzelne Aufträge
TaskPriorityFieldIndex	Priorität, die die Gewichtung angibt, mit der die Aufgabe in die Planung mit einbezogen wird
TaskQuantityFieldIndex	Gewünschte Stückzahl, die durch den Arbeitsauftrag fertig gestellt werden soll.
TaskReleaseDateFieldIndex	Datum, ab dem ein Auftrag eingeplant werden darf.
TaskResultEndDateFieldIndex	Ermitteltes Enddatum der Aufgabe
TaskResultPostEndDateFieldIndex	Ermitteltes Enddatum der Nachlaufzeit
TaskResultPreparationsStartDateFieldIndex	Ermitteltes Startdatum der Vorlaufzeit
TaskResultProcessingStepFieldIndex	Ermittelte Abfolgenummer der Aufgabe
TaskResultProcessingTimeFieldIndex	Ermittelte Planungszeit der Aufgabe
TaskResultRouteFieldIndex	Ermittelte Route, bestehend aus verfügbaren Ressourcen, über die die Aufgabe abgearbeitet wird
TaskResultStartDateFieldIndex	Berechnetes Anfangsdatum der Aufgabe

Operationen-Tabelle	
OperationDataTableName	Name der Operationentabelle
OperationMaximumInterruptionTimeFieldIndex	Maximale Unterbrechungszeit der Operation (mit Belegung der Resource)
OperationLoadPerItemFieldIndex	Belegung der Resource pro Stück
OperationOverlapQuantityFieldIndex	Überlappung mit anderen Operationen
OperationPostLoadFieldIndex	Nachlaufbelegung der Operation
OperationPreparationLoadFieldIndex	Vorbereitungsbelegung der Operation

Operationen-Tabelle	
OperationResultPostEndDateFieldIndex	Ermitteltes Enddatum der Nachlaufphase
OperationResultProcessingTimeFieldIndex	Ermittelte Verarbeitungszeit der Operation
OperationResultPreparationStartDateFieldIndex	Ermitteltes Startdatum der Vorbereitungsphase
OperationResultSelectedTimingResourceIDFieldIndex	Ermittelte ID der Zeitdauer bestimmenden Ressource
OperationResultStatusFieldIndex	Fehler- oder Warnstatus
OperationRouteFieldIndex	Route, der die Operation angehört
OperationSequenceNumberFieldIndex	Reihenfolgewert der Operation innerhalb einer Route
OperationStartLockDateFieldIndex	Fixdatum
OperationTaskIDFieldIndex	Aufgabe, der die Operation angehört
OperationWorkInProcessFieldIndex	Fertigstellungsgrad der Operation

Ressourcen-Tabelle	
ResourceCalendarNameFieldIndex	Name des Ressourcen-Kalenders
ResourceCapacityType	Begrenzte oder unbegrenzte Kapaziäten für alle Ressourcen
ResourceCapacityTypeFieldIndex	Begrenzte oder unbegrenzte Kapaziäten für Einzelressourcen
ResourceConstraintTypeFieldIndex	Bedingung für Arbeits- und Materialressourcen
ResourceDataTableName	Name der Ressourcentabelle
ResourceEfficiencyFieldIndex	Effizienz in %
ResourceGroupDataTableName	Name der Tabelle für Gruppenressourcen
ResourceGroupIDFieldIndex	Gruppenzugehörigkeit der Ressource
ResourceNameFieldIndex	Name der Ressource
ResourceResultLoadCurveNamePrefix	Kurve, in die die ermittelte Auslastung von Arbeits- und Zeitressourcen eingetragen werden soll
ResourceResultStockCurveNamePrefix	Kurve, in die der ermittelte Lagerbestand von Materialressourcen eingetragen werden soll
ResourceSelectionStrategy	Auswahlstrategie für Ressourcen

Ressourcen-Tabelle	
ResourceSoftConstraintStartDateFieldIndex	Datum für den Statuswechsel einer Ressource von "hart" nach "weich"
ResourceType	Ressourcentyp
ResultProcessingStepCount	Ermittelte Anzahl von Aufgaben

Tabelle für Zuweisungen	
AssignmentDataTableName	Name der Zuweisungstabelle
AssignmentIsResultFieldIndex	Wurde der Datensatz als Ergebnis der Zeitplanung erzeugt?
AssignmentIsVisibleFieldIndex	Soll die Darstellung im Diagramm erfolgen?
AssignmentLoadOrConsumptionFieldIndex	Wert pro Stück
AssignmentMaximumLoadFieldIndex	Maximaler Auslastungsgrenzwert
AssignmentMinimumLoadFieldIndex	Minimaler Auslastungsgrenzwert
AssignmentOperationIDFieldIndex	Zugewiesene Operation
AssignmentResourceSelectionStrategyrFieldIndex	ASAP oder JIT für eine Ressource
AssignmentResourceIDFieldIndex	Zugewiesene Ressource

Tabelle für Verbindungen	
LinkDataTableName	Name der Tabelle
LinkDurationFieldIndex	Minimaler zeitlicher Abstand
LinkPredecessorTaskIDFieldIndex	Vorgängeraufgabe der Verbindung
LinkSuccessorTaskIDFieldIndex	Nachfolgeraufgabe der Verbindung

Allgemeine Eigenschaften und Methoden	
BaseTimeUnit	Mögliche eigene Zeiteinheit für die Ressourcenplanung
BaseTimeUnitsPerStep	Grob- oder Feinberechnung?
DataRecordEventsEnabled	Werden DataRecord-Ereignisse zugelassen?
DefaultOperationMaximumInterruptionTime	Maximale Zeitdauer einer einmaligen Unterbrechung für Operationen
DefaultResourceCalendarName	Zuweisung eines Default-Kalenders für die Zeitplanung
FullUsageOfPlanningUnitsEnabled	Nutzung von Restkapazitäten von Ressourcen

Allgemeine Eigenschaften und Methoden	
PlanningEndDate	Ende des Planungszeitraums
PlanningStartDate	Anfang des Planungszeitraums
PlanningStrategy	Planungsstragie: ASAP oder JIT für alle Aufträge
Process	Ressourcenplanung starten
ToleranceTimeOnASAPDueDates	Zeittoleranz beim Fälligkeitsdatum
ToleranceTimeOnJITReleaseDates	Zeittoleranz beim Freigabedatum
ToleranceTimeOnStartLockDates	Zeittoleranz für festes Startdatum
WorkInProcessType	Einheit des Fertigstellungsgrades
WritingDebugFilesEnabled	Verfügbarkeit von Debug-Dateien

Nachdem Sie die Eigenschaften der Tabellen gesetzt haben, können Sie die Ressourcenplanung über die Methode **Process** starten.

3.22 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:

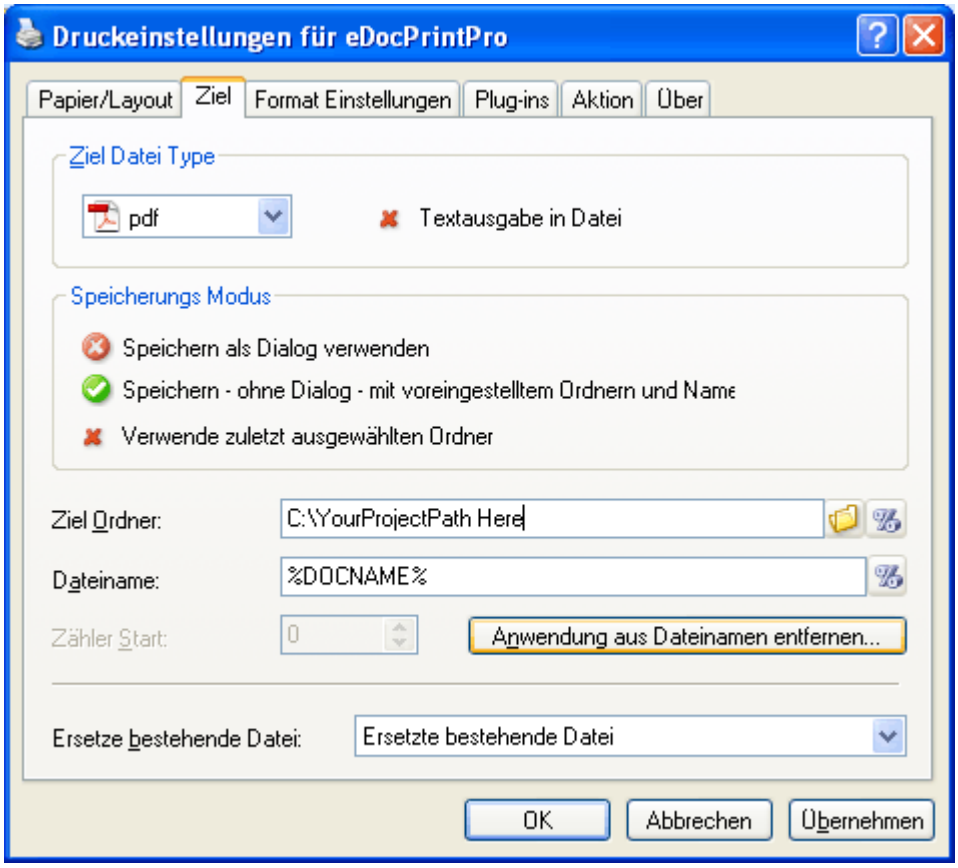
- Je nach Design der Anwendung kann es notwendig sein, den Treiber über die API so einzustellen, dass zur Laufzeit keine Dialoge und Messageboxen erscheinen. Dazu gehören insbesondere Dialoge zur Festlegung von Dateinamen und Pfaden.
- Soll das Setzen von Dateinamen und Pfad erst zur Laufzeit erfolgen, 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] sowie der kostenlose Treiber **eDocPrintPro** [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 Zieldatei 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

```
VcGantt1.Printer.PrinterName = "eDocPrintPro"
VcGantt1.Printer.DocumentName = "abc.pdf"
VcGantt1.PrintEx
```

Code-Beispiel C#

```
vcGantt1.Printer.PrinterName = "eDocPrintPro";
vcGantt1.Printer.DocumentName = "abc.pdf";
vcGantt1.PrintEx;
```

Ganz wenige Druckertreiber erfordern eine andere Code-Sequenz:

Code-Beispiel VB.NET

```
VcGantt1.Printer.PrinterName = "Win2PDF"
VcGantt1.PrintToFile "abc.pdf"
```

Code-Beispiel C#

```
vcGantt1.Printer.PrinterName = "Win2PDF";
vcGantt1.PrintToFile "abc.pdf";
```

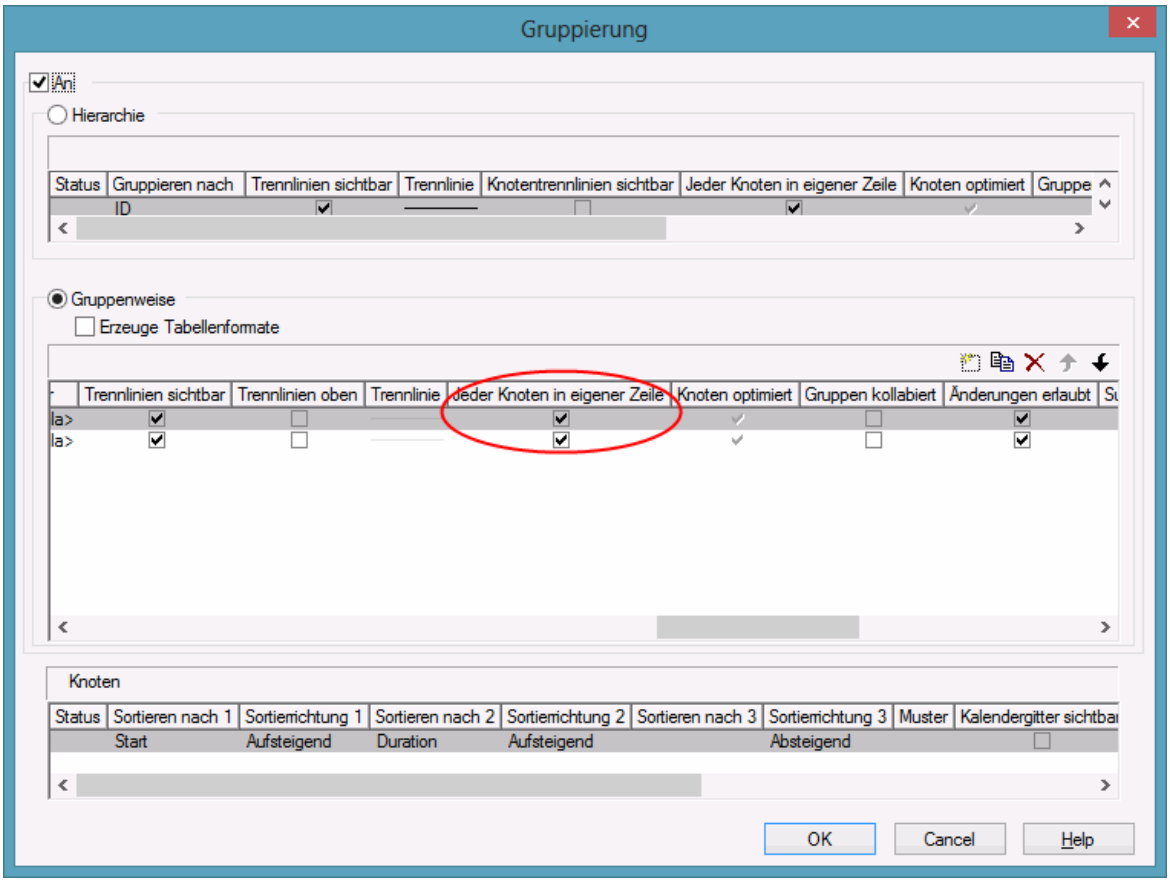
Für weitere Fragen zur Konfiguration und Benutzung von **eDocPrintPro** bitten wir Sie, sich mit dem Hersteller in Verbindung zu setzen.

3.23 Sortierung

Vorgänge werden in Anwendungen meist nach bestimmten Kriterien sortiert. Eine Sortierung kann nur bei Knoten stattfinden, die nicht in eine Hierarchie eingebunden sind, d.h. also nur bei Basisknoten oder solchen, die zu einer Gruppe gehören. Daher finden Sie Sortieroptionen nur dort, wo Sie Gruppen- und Basisknoten beeinflussen können. Beim Sortieren muss unterschieden werden, ob die Knoten in jeweils einer eigenen Zeile angeordnet sind oder ob sich mehrere Knoten in einer Zeile befinden.

Anordnung: Jeder Knoten in eigener Zeile

Wenn Sie die Knoten in einzelnen Zeilen anordnen wollen, rufen Sie den Dialog **Gruppierung** auf, den Sie bei den Eigenschaftenseiten auf der Seite **Objekte** unter **Gruppierung** finden:



Hier setzen Sie im mittleren Fenster die Einstellung **Jeder Knoten in eigener Zeile**. Sie können diese Einstellung auch über die Programmierschnittstelle mit der Eigenschaft **VcGroupLevelLayout.AllNodesInOneRow** setzen.

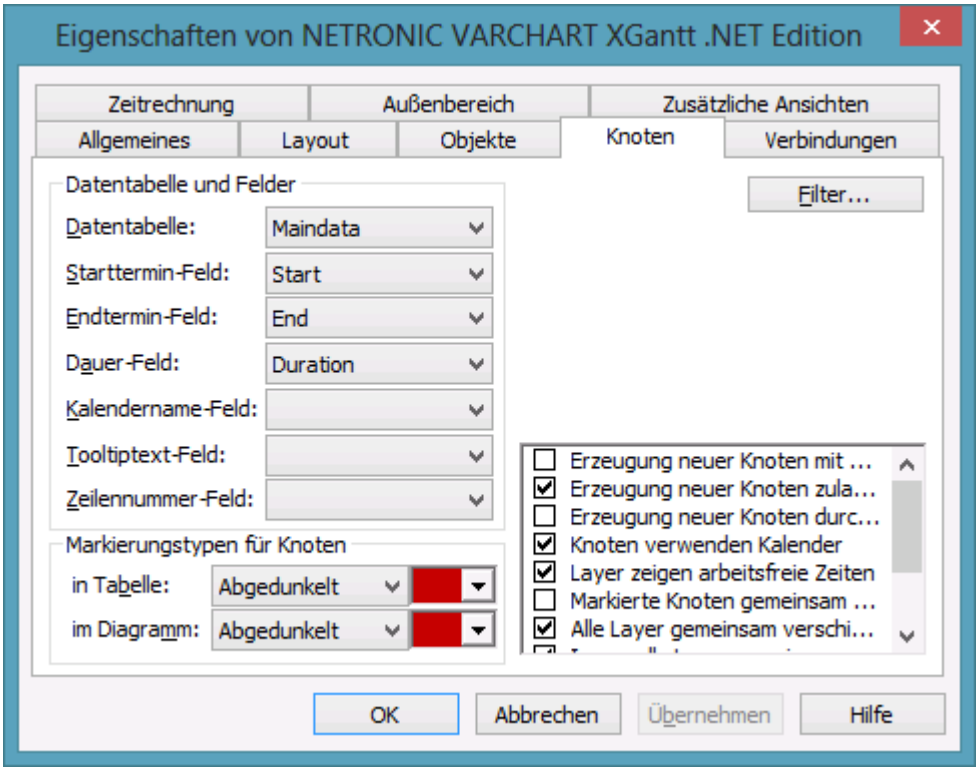
In dem darunterliegenden Fenster **Knoten** können Sie drei Datenfelder festlegen, nach denen die Vorgänge beim Öffnen des Diagramms sortiert werden sollen. Für jedes dieser Datenfelder können Sie außerdem bestimmen, ob aufsteigend oder absteigend sortiert werden soll.

Falls die Vorgänge gruppiert sind, gilt die Sortierung innerhalb jeder Gruppe.

Weiterhin stehen Ihnen folgende Möglichkeiten zur Verfügung, um das Aussehen der Knotenzeilen zu bestimmen:

- Auswahl eines **Musters**
- Anzeige, Position und Aussehen der **Trennlinien**
- über **Trennlinienschrittweite** können Sie festlegen, nach wie vielen Vorgängen jeweils eine Trennlinie gezogen werden soll. Falls die Vorgänge gruppiert sind, wird in jeder Gruppe neu zu zählen begonnen.

Weitere Optionen für die Sortierung finden Sie auf der Eigenschaftenseite **Knoten**:



- Sie können ein Datenfeld festlegen, in das die **Zeilennummer** jedes Vorgangs geschrieben werden soll. Das Zeilennummer-Feld wird immer erst aktualisiert, wenn die Daten mit der Methode **Save As** gesichert wurden.
- Über **Veränderung der Knotenreihenfolge über das Diagramm** bzw. **Veränderung der Knotenreihenfolge über die Tabelle** können Sie festlegen, ob dem Anwender Änderungen in der Vorgangsreihenfolge erlaubt

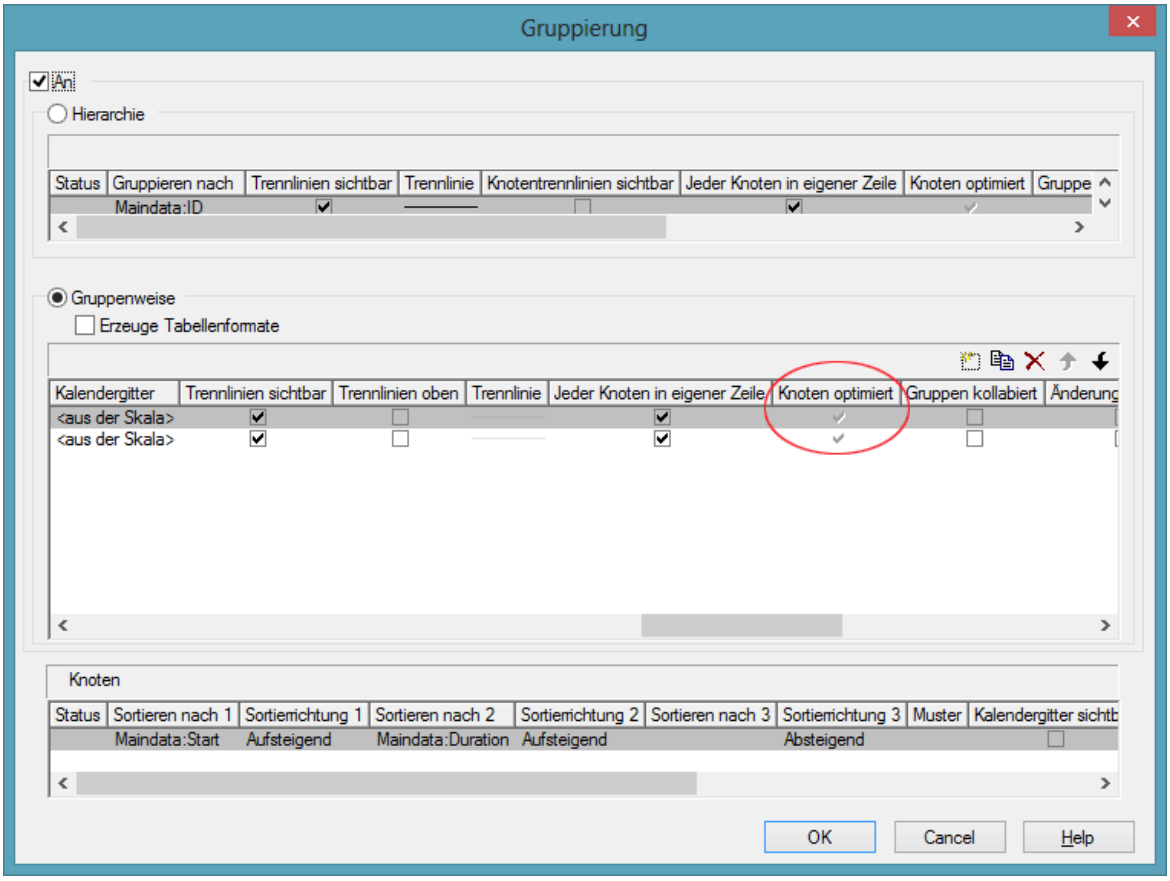
sein sollen. Dann lassen sich Vorgänge per Drag & Drop in eine andere Zeile ziehen. Wird ein Vorgang in eine andere Gruppe verschoben, werden sein Gruppierfeld und seine Farbe aktualisiert. Wird der Vorgang durch mehrere Layer repräsentiert, muss zusätzlich die Umschalt-Taste gedrückt werden.

Hinweis: Beachten Sie bitte, dass die Einstellungen im Dialog **Gruppierung** und auf der Eigenschaftenseite **Knoten** nur für die Sortierung der Vorgänge beim Öffnen des Diagramms gelten. Wenn Sie die Vorgänge danach neu sortieren möchten, verwenden Sie dazu bitte die Methode **SortNodes**. Die Aktualisierung der Sortierung muss also über diesen Aufruf angestoßen werden.

Anordnung: Knoten einer ganzen Gruppe in einer Zeile

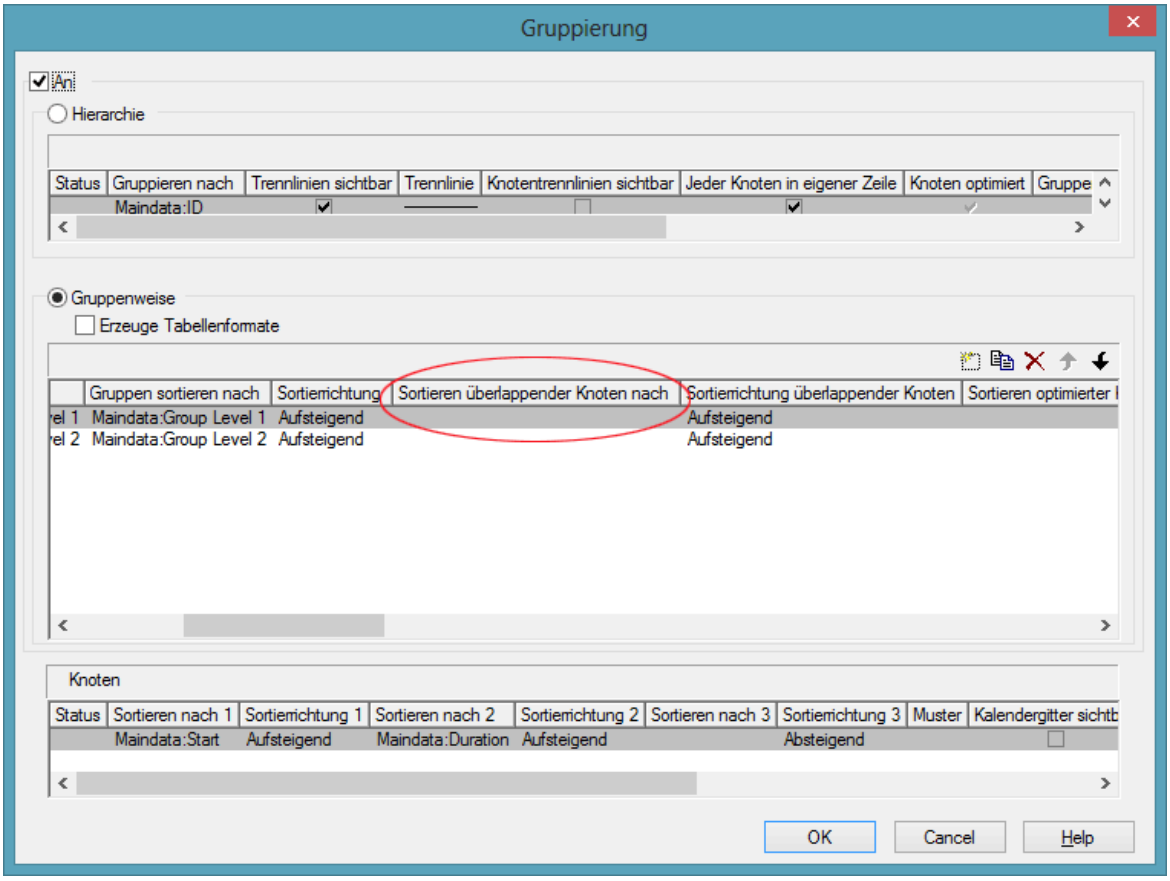
Wenn Sie mehrere Knoten (d.h. die Knoten einer Gruppe) in eine Zeile anordnen, können Sie auch für diese Knoten eine Sortierung einrichten (die einer Darstellungspriorität entspricht). Dabei ist zwischen der **überlappenden** und **optimierten** Darstellung zu unterscheiden, d.h. Knoten können sich entweder überlappen oder eine Überlappung durch Verbreiterung der Zeile vermeiden.

Mehrere Knoten in eine Zeile setzen Sie im Gruppierungsdialog im mittleren Fenster über das Feld **Knoten optimiert**; diese Einstellung wird standardmäßig angeschaltet, wenn Sie **Jeder Knoten in eigener Zeile** deaktivieren:



Sie können aber auch **Knoten optimiert** abschalten, dann erhalten Sie die Darstellung mit sich überlappenden Knoten. Sie können diese Einstellung alternativ über die API-Eigenschaft **VcGroup.NodesArrangedOptimized** setzen.

Die Darstellungspriorität können Sie über das Feld **Überlappende Knoten sortieren nach** im Dialog **Gruppierung** festlegen:

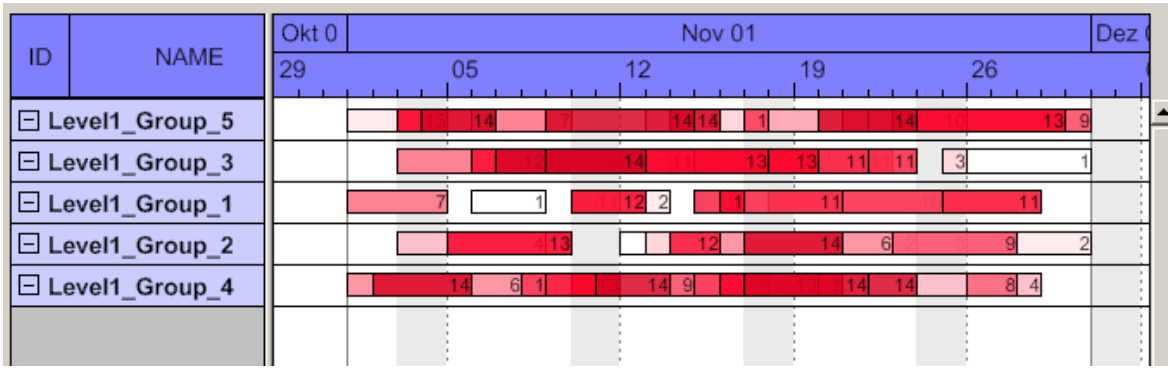


Analog zu den überlappenden Knoten können Sie über das Feld **Optimierte Knoten sortieren nach** auch optimierte Knoten sortieren.

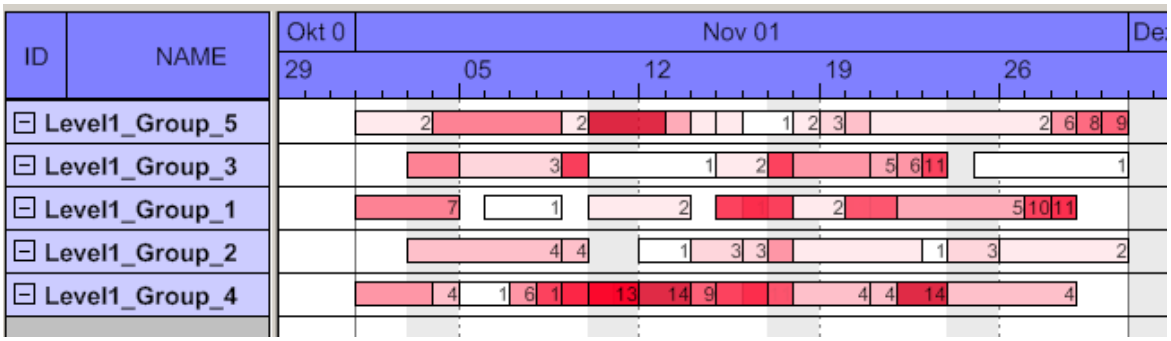
Wenn Sie keine Darstellungspriorität setzen, werden die Knoten standardmäßig nach Anfangsdatum und Dauer abgebildet: die spätesten und kürzesten werden am weitesten oben liegend dargestellt. Sie können die Darstellungs-Priorität alternativ über die Eigenschaften der Programmierschnittstelle **VcLevelLayout.OverlaidNodesSortDataFieldIndex** bzw. **VcLevel-Layout.OptimizedNodesSortDataFieldIndex** setzen.

Die Aktualisierung der Sortierung erfolgt automatisch; d.h. Sie brauchen diese nicht durch einen eigenen Aufruf anzustoßen.

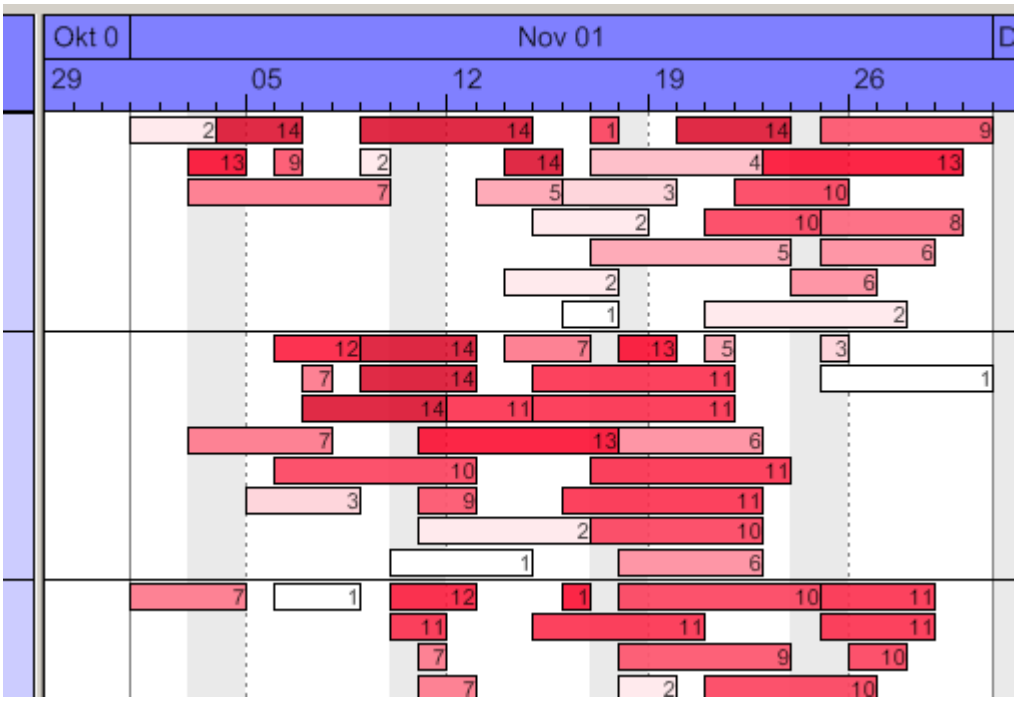
Außerdem haben Sie in im benachbarten Feld **Sortierrichtung überlappender Knoten** die Möglichkeit, die Reihenfolge aufsteigend oder absteigend zu wählen. Alternativ können Sie die Sortierrichtung über die Eigenschaften **OverlaidNodesSortOrder** bzw. **OptimizedNodesSortOrder** der Programmierschnittstelle setzen. Hier mögliche Ergebnisse:



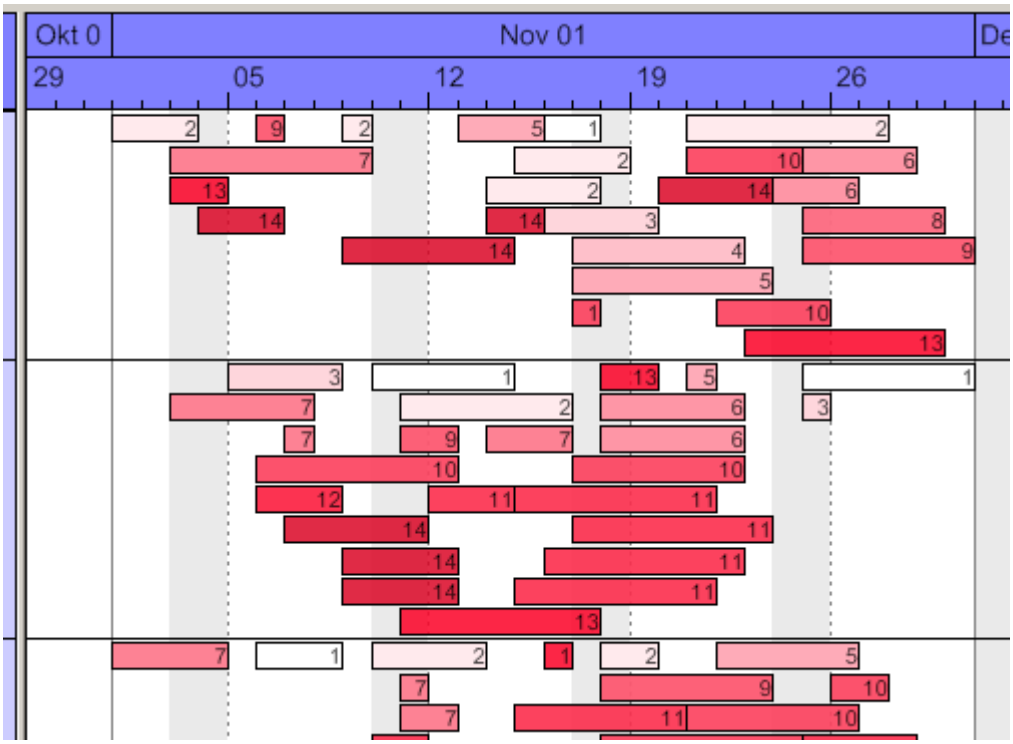
Überlappende Anordnung mit aufsteigender Darstellungspriorität von dunklen Knoten (dunkle Knoten im Vordergrund liegend)



Überlappende Anordnung mit absteigender Darstellungspriorität von dunklen Knoten (dunkle Knoten im Hintergrund liegend)



Optimierte Anordnung mit aufsteigender Darstellungspriorität von dunklen Knoten (dunkle Knoten im oberen Bereich der Zeile)



Optimierte Anordnung mit absteigender Darstellungspriorität von dunklen Knoten (dunkle Knoten im unteren Bereich der Zeile)

3.24 Sprachanpassung von Textausgaben

Sie können mit Hilfe des Ereignisses **VcTextEntrySupplying** die Texte aller zur Laufzeit erscheinenden Kontextmenüs, Dialogfelder, Infoboxen, Fehlermeldungen sowie Monats- und Tagesnamen 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

```
VcGantt1.TextEntrySupplyingEventEnabled = True
```

Code-Beispiel C#

```
vcGantt1.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
```

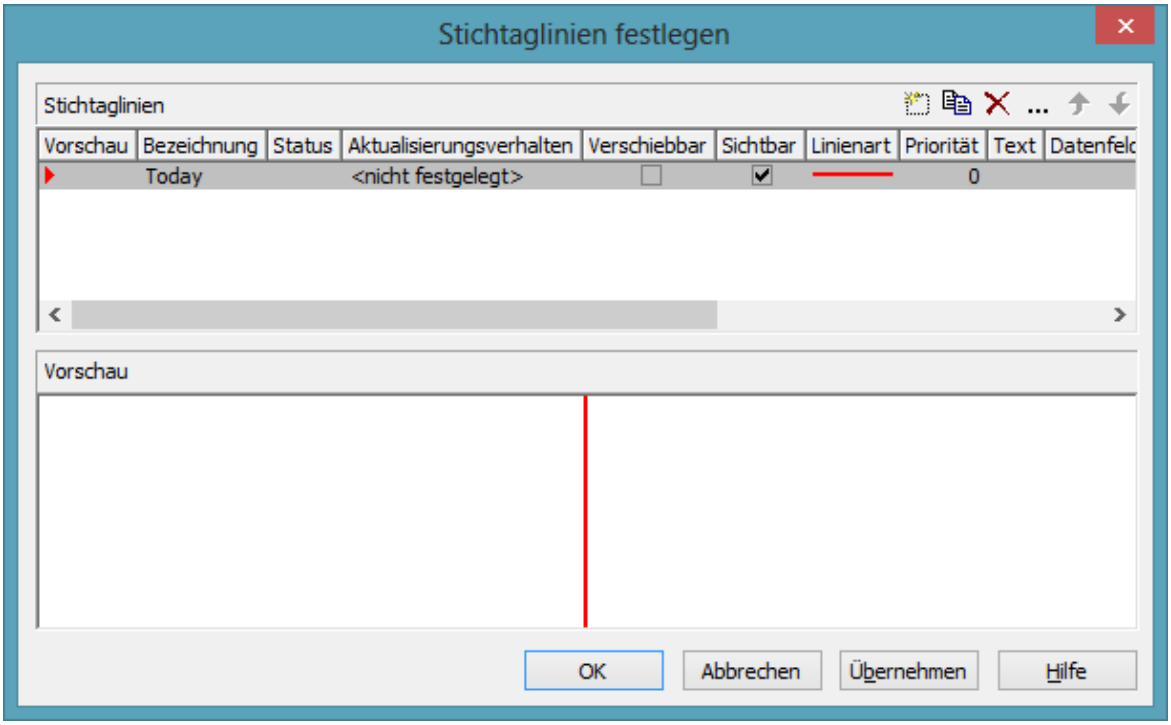
Code-Beispiel C#

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

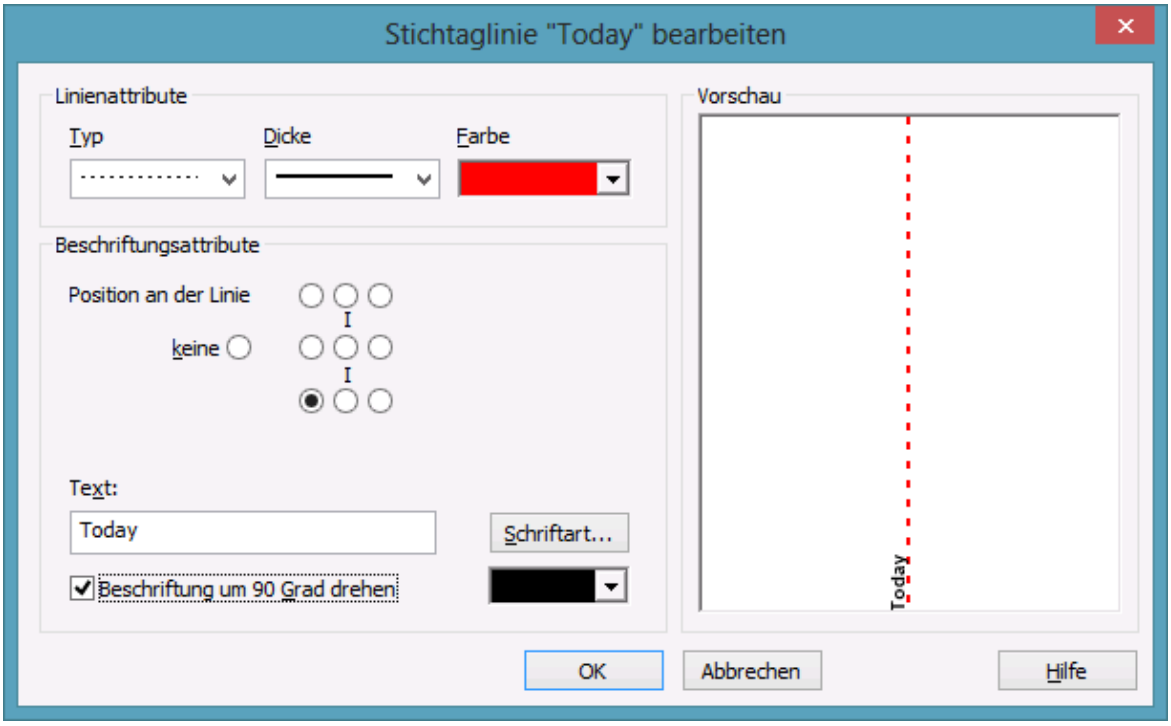
```
      case VcTextEntryIndex.vcTXERibMon8:
         e.Text = "September";
         break;
      case VcTextEntryIndex.vcTXERibQuar3:
         e.Text = "3. Quartal";
         break;
      }
   }
```

3.25 Stichtaglinien

Durch Stichtaglinien (vertikale Linien im Diagramm) können Sie bestimmte Daten besonders hervorheben. Die Eigenschaften der Stichtaglinien werden in zwei Dialogen festgelegt: Im Dialog **Stichtaglinien festlegen** bestimmen Sie Beschriftung, Datum, Priorität (relativ zueinander und zu den Rastern und Knoten) sowie Verschieb- und Sichtbarkeit. Im Dialog **Stichtaglinie bearbeiten** können Sie Linien- und Textattribute sowie die Beschriftungsposition festlegen. Sie erreichen diese Dialoge über die Eigenschaftenseite **Objekte**, Schaltfäche **Stichtaglinien**.

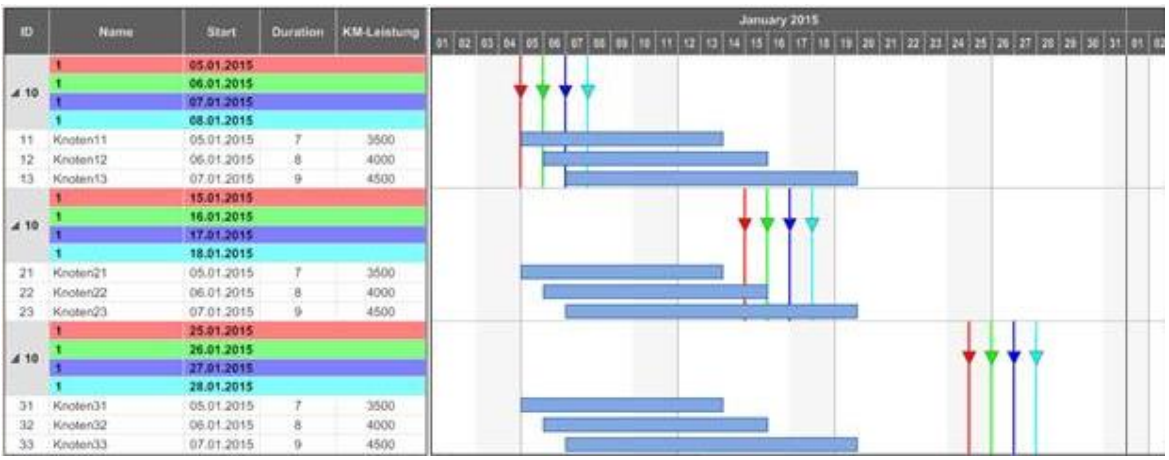


Von hier aus gelangen Sie durch Klick auf ... in den Dialog **Stichtaglinie bearbeiten**:

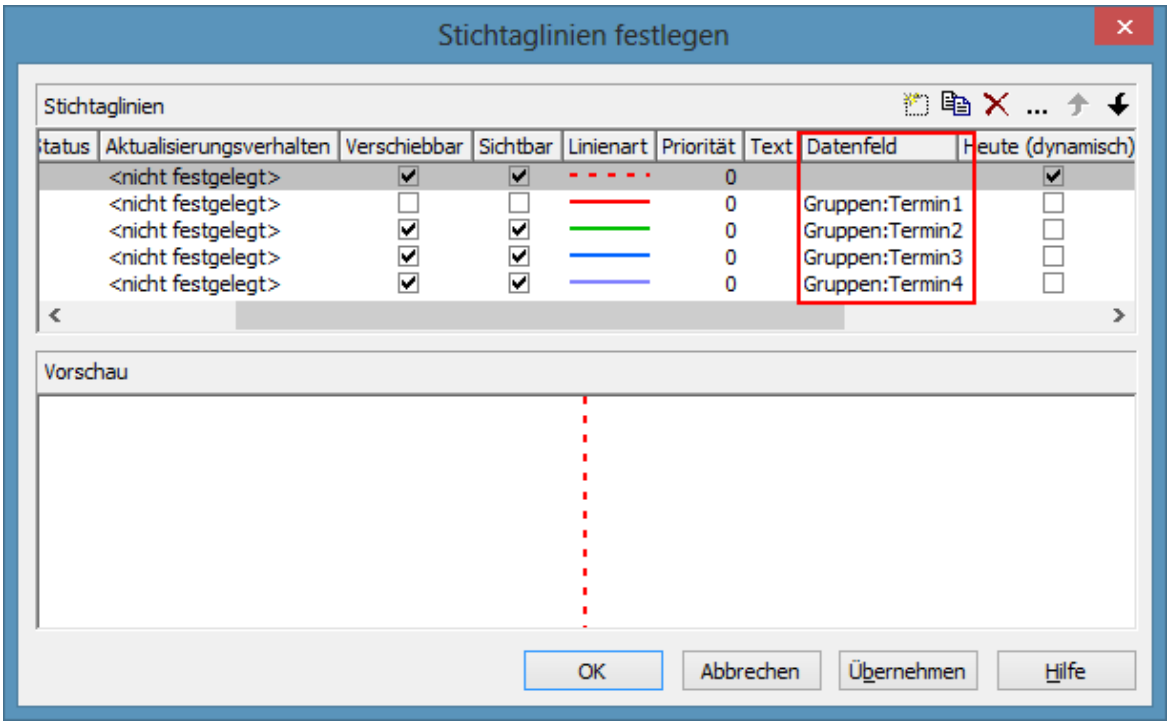


Individuelle, datenbasierte Stichtaglinien

Stichtaglinien können auch alternativ zu dem fixen Datum auf ein Datum aus einem Datensatz eines Knotens oder einer Gruppe zugreifen. Es können dann automatisch zu jedem Datensatz eines Knotens oder einer Gruppe individuelle Stichtaglinien als grafische Kopien entstehen, d.h. es werden die Eigenschaften (Farbe etc.) aus der zugrunde liegenden Stichtaglinien der DateLineCollection außer Datum verwendet, wobei Datum und Position im Plan jedoch individuell sind. Solche Stichtaglinien werden unter Anwendung des **NodeLevelLayout** bzw. **GroupLevelLayout** nur innerhalb der Bänder der Knoten bzw. Gruppen gezeichnet (siehe Bild: je vier Stichtaglinien wurden individuell für drei Gruppen erzeugt und positioniert; vier Symbollayer des eingeschalteten Gruppenknotens benutzen dieselben Termine wie die Stichtaglinien).



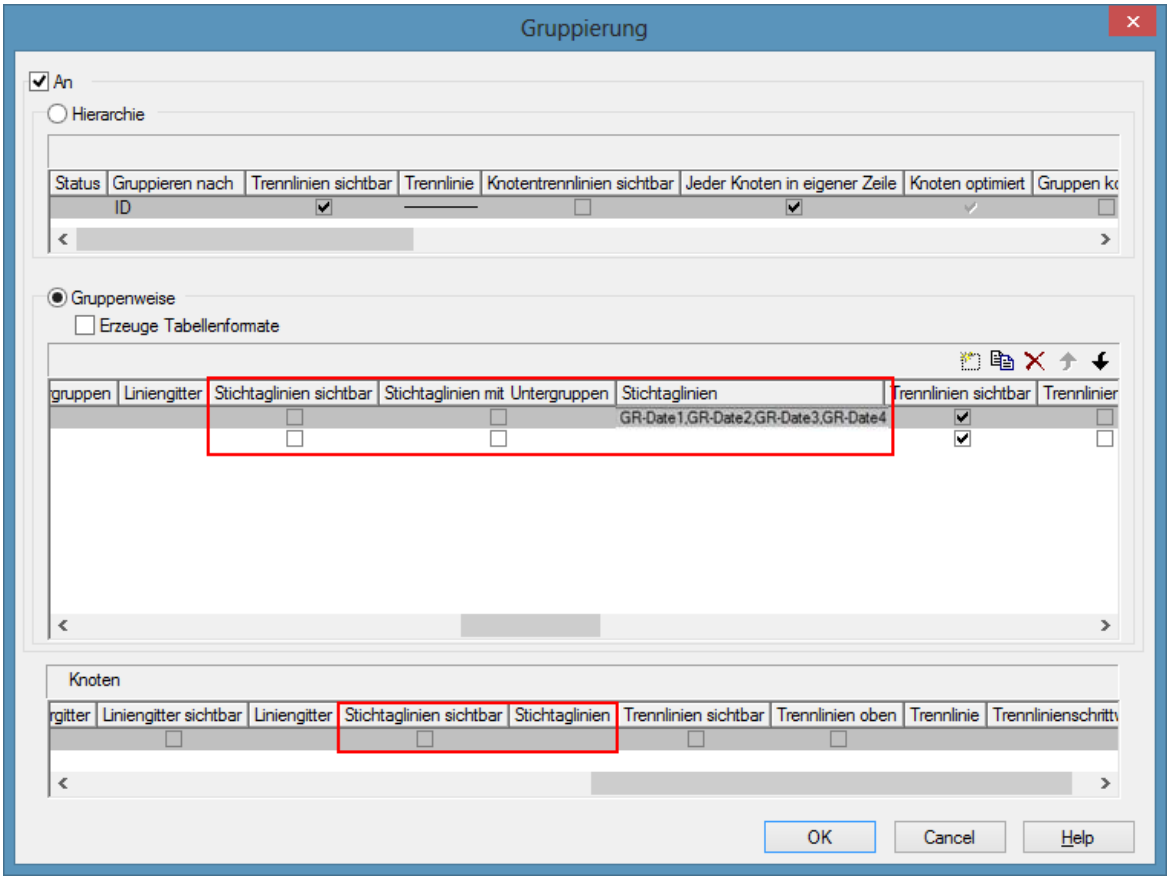
Um das zu erreichen, muss zunächst bei der Stichtaglinie ein Datenfeld angegeben werden, aus dem das Datum geholt werden soll.



Der enstprechende API-Befehl lautet: **VcDateLine.DateDataFieldIndex**

Wichtig: Bei einem individuell festgelegten Datenfeld hat das Datum aus dem Datensatz Priorität vor dem fixen Datum (**VcDateLine.Date**). Wenn kein Datum ermittelt werden konnte, z.B. weil das angegebene Datenfeld im Datensatz leer ist, erscheint keine Stichtaglinie.

Nachdem ein Datenfeld festgelegt wurde, muss die Stichtaglinie noch mit einem Datensatz verknüpft werden. Dies geschieht durch entsprechende Angaben im Dialog **Gruppierung**:



An der API geschieht dies durch die entsprechenden Eigenschaften:

VcGroupLevelLayout.DateLinesVisible

VcGroupLevelLayout.DateLineWithChildGroups

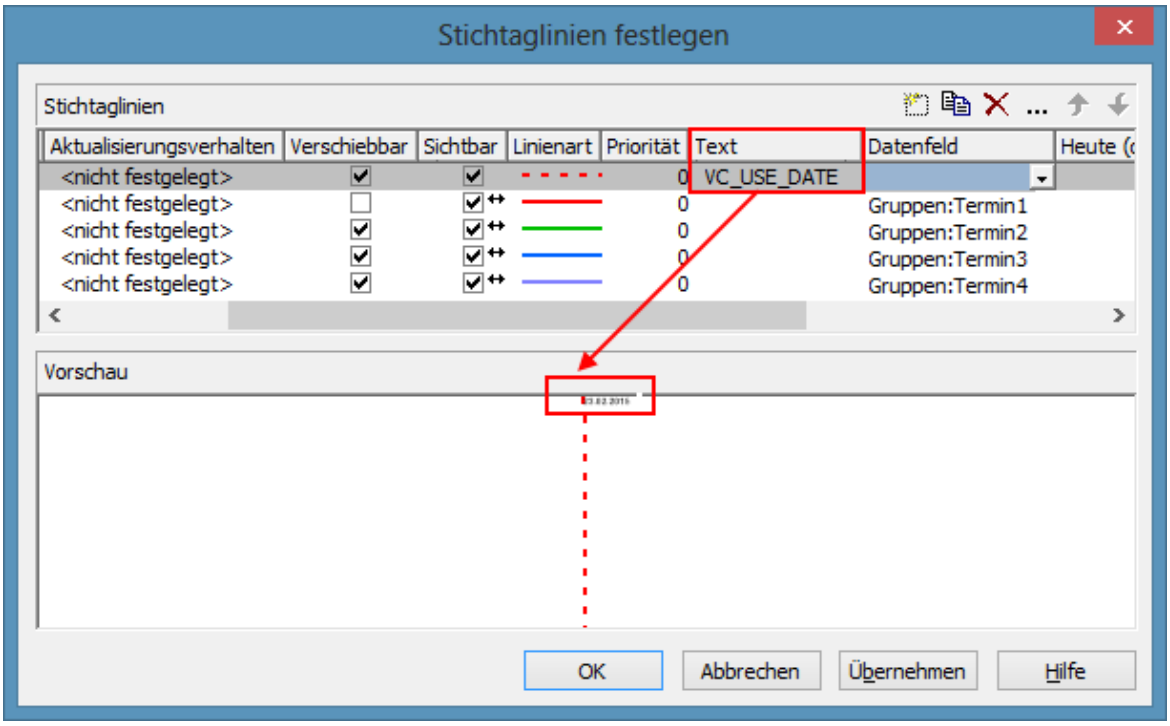
VcGroupLevelLayout.DateLineName

VcNodeLevelLayout:.DateLinesVisible

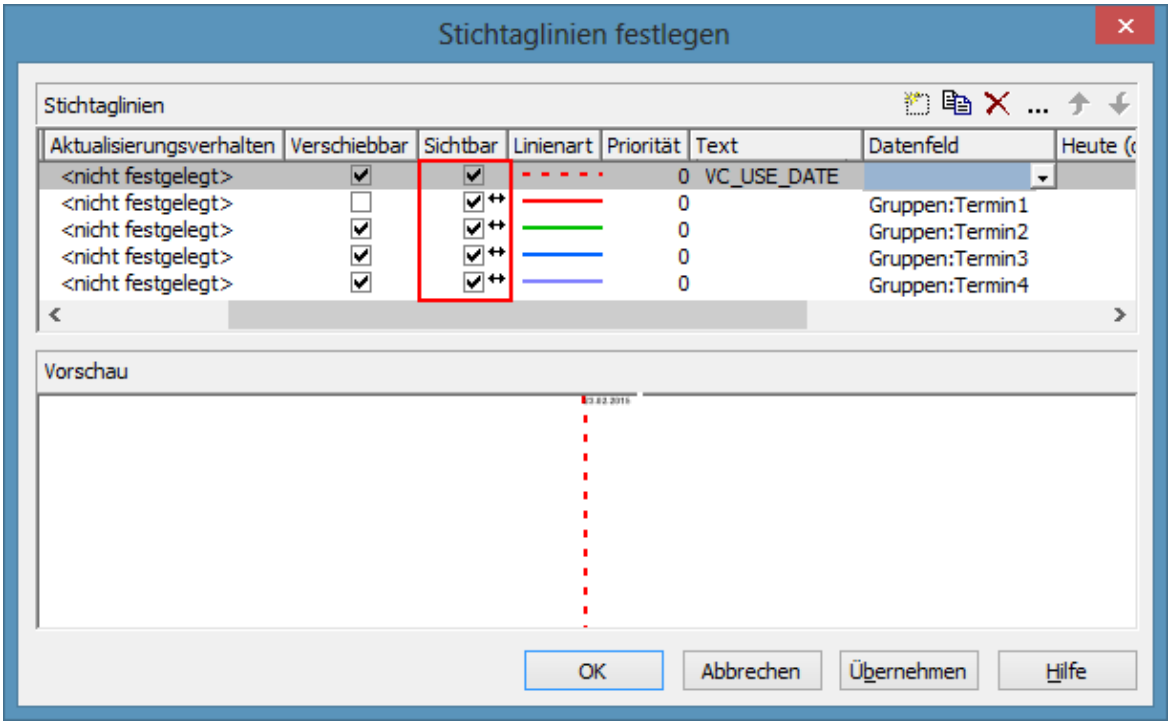
VcGroupLevelLayout.DateLineName

Beschriftung von Stichtaglinien

Stichtaglinien können mit einer Beschriftung versehen werden. Dies geschieht in der Regel durch einen festen Text. Generell bei allen, speziell aber bei individuellen Stichtaglinien ist ggf. die Ausgabe des individuellen Datums gewünscht. Durch das Schlüsselwort **VC_USE_DATE** wird das jeweilige Datum an der spezifizierten Stelle der Stichtaglinie (**VcDateLine.LabelPosition**) im eingestellten Datumsformat (**VcGantt.DateOutputFormat**) ausgegeben.



Damit die Stichtaglinien individuell sichtbar gemacht werden können, kann die Option **Sichtbar** datengetrieben und damit individuell festgelegt werden.



Die entsprechenden API-Eigenschaften:

VcDateLine.VisibleDataFieldIndex

VcDateLine.VisibleMapName.

3.26 Tabelle

Die Eigenschaften der Tabelle können Sie mit Hilfe von drei Dialogen festlegen, die Sie über die Eigenschaftenseite **Objekte** und die Auswahltaste **Tabelle** erreichen. Die dort aufrufbaren Dialoge heißen **Tabelle festlegen**, **Tabelle bearbeiten** und **Tabellenformat bearbeiten**. Sie können im ersten Dialog auch mehrere Tabellen anlegen.

Die Tabelle besteht aus standardmäßig sechs Spalten, die jeweils nur bei einer Breite größer 0 sichtbar sind. Die Zeilen der Tabelle werden durch sogenannte Tabellenformate beschrieben. Für jedes Tabellenformat können Sie Schriftart, Schriftfarbe, Hintergrundfarbe sowie Ausrichtung und Ränder festlegen. Die einzelnen Tabellenformate werden jeweils unter bestimmten Bedingungen verwendet:

- **StandardListCaption** für den Tabellenkopf
- **StandardList** für Vorgänge/Zeilen

Zusätzlich zu den Standardtabellenformaten können Sie weitere Formate anlegen, für die Sie Namen und Filter individuell festlegen können.

Tabellenformate für hierarchische Anordnung:

Die hierarchische Anordnung kann auf der Eigenschaftenseite **Objekte** gesetzt werden, indem Sie dort auf die Schaltfläche **Gruppierung** klicken.

- **Hierarchy:** Format für die Hierarchieebenen in expandiertem Zustand. Dabei wird das 2. Feld (normalerweise der Vorgangsname) ebenenweise eingerückt. Ein "-" zeigt an, dass kollabiert werden kann.
- **HierarchyCollapsed:** Format für die Hierarchieebenen in kollabiertem Zustand. Ein "+" zeigt an, dass expandiert werden kann.

ID	NAME	START
1	⊟ SW Development	02.09.98
1.2	⊞ *Design&Concept*	*02.09.98*
1.3	⊟ Coding	09.09.98
1.3.1	Phase A (DB)	09.09.98
1.3.2	Phase B (GUI)	15.09.98
1.4	⊞ *Testing*	*17.09.98*
1.5	Sales & Marketing	05.09.98
1.6	Delivery	24.09.98
1.7	Final Party	

*Bild oben: Das Format **HierarchyCollapsed** in der Zeile **Design&Concept** mit kollabierter Hierarchieebene und das Format **Hierarchy** in der Zeile **Coding** mit expandierter Hierarchieebene*

Tabellenformate für gruppierte Anordnung:

Eine Gruppierungs-Anordnung kann auf der Eigenschaftenseite **Objekte** gesetzt werden, indem Sie dort auf die Schaltfläche **Gruppierung** klicken.

- **Subtitle:** für die Gruppenüberschriften nicht kollabierter Gruppen. Die Gruppenüberschrift besteht aus nur einem Feld, das sich über die gesamte Tabellenbreite erstreckt. Ein "-" zeigt an, dass die Gruppe kollabiert werden kann.
- **Collapsed:** Format für die Gruppenüberschriften kollabierter Gruppen. Ein "+" zeigt an, dass die Gruppe expandiert werden kann.

ID	NAME	START
⊟ A		
1	SW Development	02.09.98
3	Requirements	02.09.98
7	Final Check	16.09.98
12	QA Requirement Check	23.09.98
⊞ *Group C*		
⊞ *Group B*		
⊟ E		
15	Final Party	30.09.98

*Bild oben: Das Format **Subtitle** in den Zeile **GroupC** und **GroupB** mit kollabierter Gruppen-Ebene und das Format **Collapsed** in der Zeile **E** mit expandierter Gruppenebene*

3.27 Tooltips zur Laufzeit

Sie können Tooltips verwenden, um Informationen über das mit der Maus berührte Objekt bereitzustellen. Mit Hilfe des Ereignisses **VcToolTipText-Supplying** können Sie die Texte aller zur Laufzeit erscheinenden Tooltips bearbeiten, z. B. um sie in unterschiedliche Sprachen zu übersetzen oder zu unterdrücken.

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

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

Code-Beispiel VB.NET

```
VcGantt1.ToolTipTextSupplyingEventEnabled = True
```

Code-Beispiel C#

```
VcGantt1.ToolTipTextSupplyingEventEnabled = true;
```

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcToolTipTextSupplying(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcToolTipTextSupplyingEventArgs) Handles
VcGantt1.VcToolTipTextSupplying

        Select Case e.HitObjectType
            Case VcObjectType.vcObjTypeDateLine
                e.Text = "Stichtaglinie"
            Case VcObjectType.vcObjTypeBox
                e.Text = "Box"
        End Select

End Sub
```

Code-Beispiel C#

```
private void VcGantt1_VcToolTipTextSupplying(object sender,
NETRONIC.XGantt.VcToolTipTextSupplyingEventArgs e)
   {
   switch (e.HitObjectType)
      {
      case VcObjectType.vcObjTypeDateLine:
         e.Text = "Stichtaglinie";
         break;
      case VcObjectType.vcObjTypeBox:
         e.Text = "Box";
         break;
      }
   }
```

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

3.28 Unicode-Zeichen

Damit zur Entwurfszeit in den Eigenschaftenseiten Unicode-Zeichen erscheinen, muss unter **Start / Systemsteuerung / Einstellungen / Anzeige / Darstellung** dem Bildelement **Fenster** ein geeigneter Schrifttyp zugewiesen werden.

Außerdem können trotz des Schrifttyps nur Zeichen angezeigt werden, deren zugehörige Sprache unter **Start / Systemsteuerung / Einstellungen / Region- und Spracheinstellungen** gewählt wurde.

Zur Laufzeit kann jedes Objekt in einer VARCHART-Komponente, das Texte enthält, Unicode-Zeichen darstellen, wenn ein geeigneter Schrifttyp bei der zugehörigen Eigenschaft **Font** gesetzt ist.

Den Kontextmenüs, Tooltipps und Laufzeit-Dialogen kann über die Eigenschaft **DialogFont** des Objektes **VcGantt** ein Unicode-Schrifttyp zugewiesen werden.

Eine Übersicht über alle verfügbaren Fonts, die zumindest einen Teil aller Unicode-Zeichen enthalten, ist bei "Wazu Japan's Gallery of Unicode Fonts" zu finden (http:// www.wazu.jp/index.html). Eine weitere gute Informationsquelle für den Unicode-Standard bietet die Homepage des Unicode-Konsortiums (http:// www.unicode.org) sowie die Einführung in Unicode auf der GlobalDev-Homepage von Microsoft (http:// www.microsoft.com / globaldev / getwr / steps / wrg_unicode.mspx). Unter **Start / Programme / Zubehör / Systemprogramme / Zeichentabelle** kann man sich in Windows 2000 und XP darüber informieren, welcher installierte Schriftarttyp welche Zeichen enthält.

Beim Import von CSV-Dateien erkennt die Methode **VcGantt.Load** automatisch, ob die Datei im Unicode- oder ANSI-Format vorliegt.

Hinweis: Die Entwicklungsumgebungen von Visual Studio 6 können in Quellcodedateien keine Unicode-Zeichen verwenden. Die interne Zeichendarstellung in Strings von Visual Basic 6 ist aber Unicode. Bei Visual C++ in Verbindung mit MFC muss man die Defines _UNICODE und UNICODE setzen, um Strings in Unicode zu verwenden. Ab Visual Studio .NET 2002 ist das Editieren von Quellcodedateien in Unicode-Kodierung möglich, beim Speichern muss man als Kodierung "Unicode" auswählen.

3.29 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 erzeugen

Zur Laufzeit können Sie mit der Maus Verbindungen zwischen zwei Vorgängen ziehen, wenn Sie den **Modus: Verbindung ziehen** gewählt haben.

Die Verbindungslinie geht dabei jeweils vom 1. Layer des Anfangsvorgangs zum 1. Layer des Endvorgangs.

Sie können Verbindungen auch über die API mit **InsertLinkRecord** anlegen.

Jedes interaktive Neuanlegen einer Verbindung wird der Applikation mit dem Ereignis **VcLinkCreating** mitgeteilt.

> 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 **VcGantt.DeleteLinkRecord** oder mit der Methode **VcLink.DeleteLink** löschen.

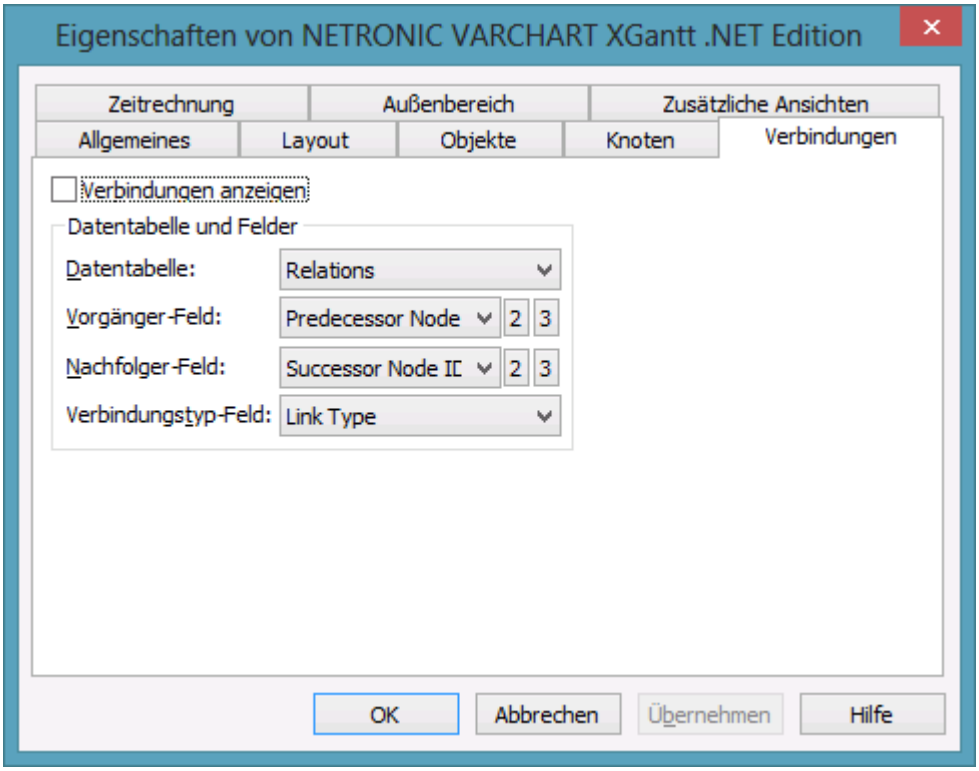
> Ereignisse

Auf folgende Ereignisse können Sie reagieren:

- **VcLinkCreating**
- **VcLinkCreated**
- **VcLinkDeleting**
- **VcLinkDeleted**
- **VcLinksLeftClicking**
- **VcLinksLeftDoubleClicking**
- **VcLinksRightClicking**

> Verbindungen festlegen

Auf der Eigenschaftenseite **Verbindungen** können Sie festlegen, ob die Verbindungen angezeigt werden sollen, und ggf. die Verbindungen spezifizieren. Darüberhinaus können im Dialog **Verbindungsaussehen verwalten** verschiedene Verbindungsaussehen erstellt und bearbeitet werden, für die Filter, Vorgänger- und Nachfolger-Layer, Linienart, Vorgänger- und Nachfolger-Portsymbole sowie die Art der Verbindungsführung festlegt werden können.



> Verbindungen festlegen

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.

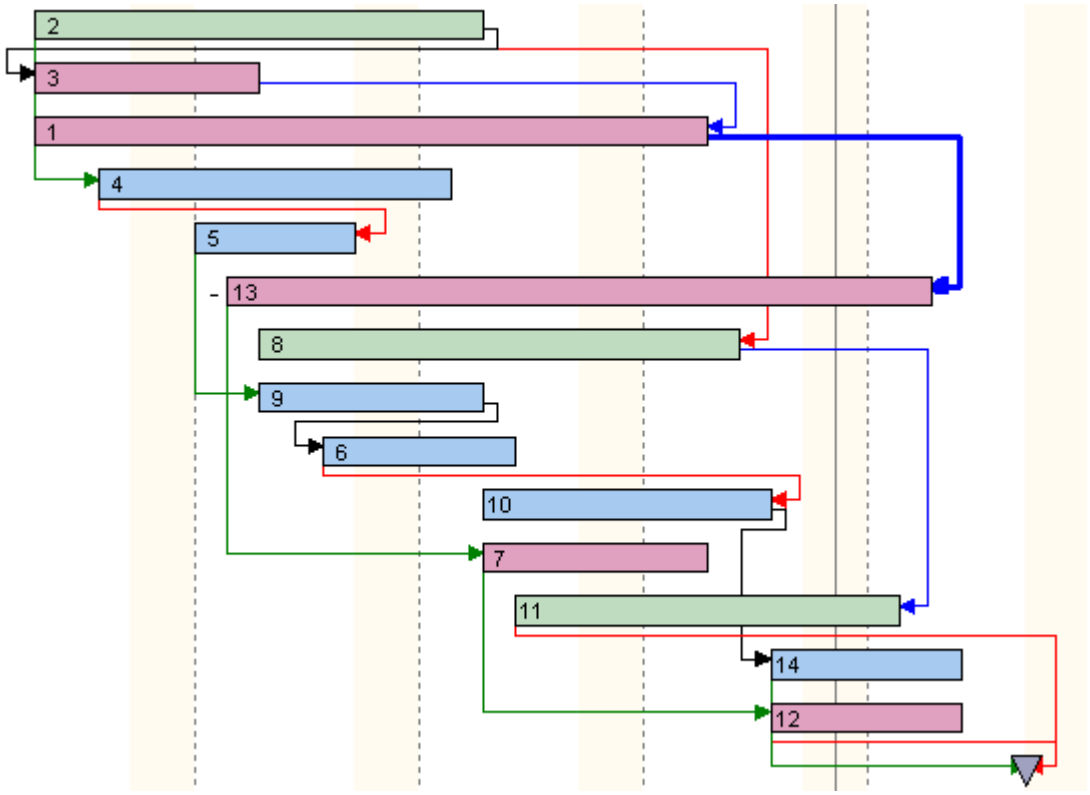
> Verbindungstypen

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

Verbindungstypen:

- FF: Ende-Ende
- FS: Ende-Anfang
- SF: Anfang-Ende
- SS: Anfang-Anfang

Sie können mit Hilfe dieses Datenfeldes den Verbindungstyp durch die entsprechende Linienführung darstellen.



Einige Beispiele für die typgerechte Darstellung von Verbindungen

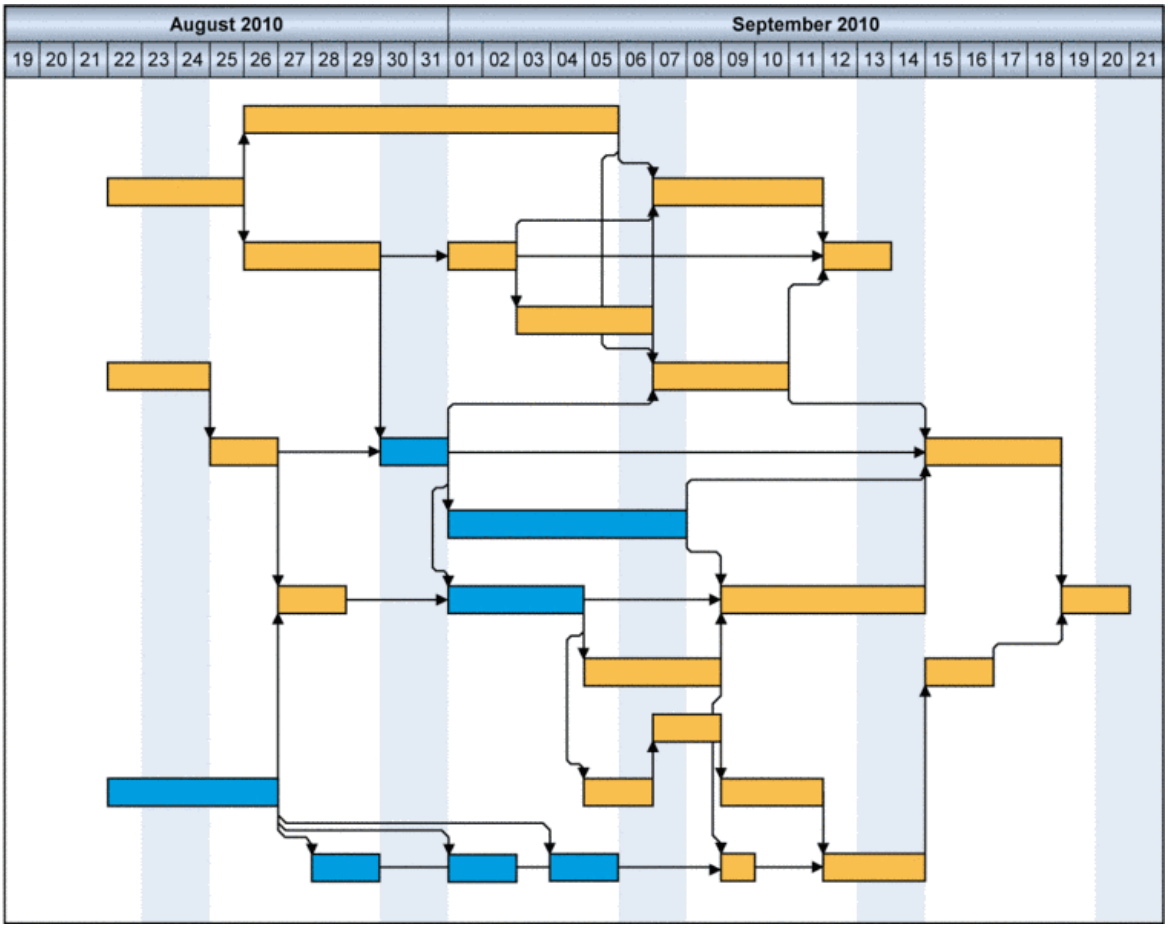
> Automatisches Layout

Für eine optimierte Linienführung der Verbindungen steht ein Layouter zur Verfügung. Durch diesen wird die Linienführung stets eindeutig und es ist immer klar erkennbar, woher die Verbindung kommt und wohin sie führt.

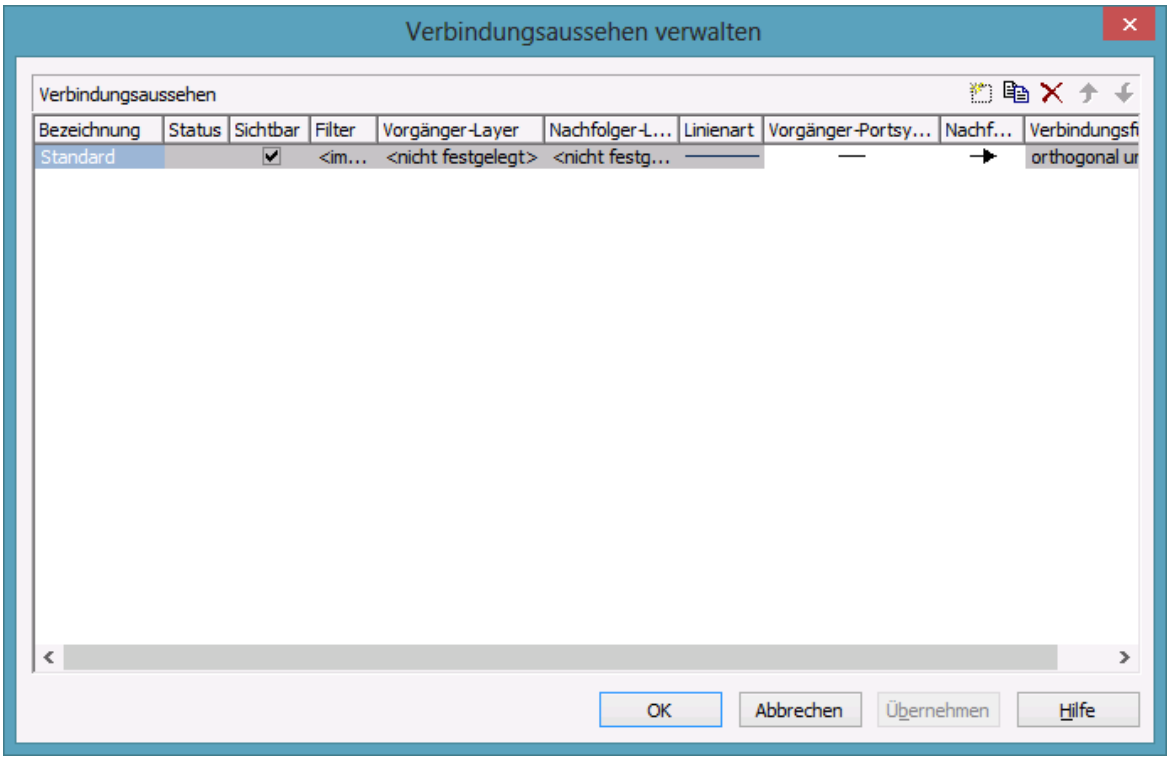
Zudem wird die Linienführung so optimiert, dass durch die Schachtelung von Krümmungen möglichst wenige Linienkreuzungen entstehen.

Die Höhe von Zeilen wird automatisch angepasst, um Platz für die parallelen horizontalen Verbindungsabschnitte zu schaffen.

Kleine Schrägen in den Ecken zeigen die Änderung der Verlaufsrichtung einer Verbindung an.



3.30 Verbindungsaussehen



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

> Weitere Festlegungen des Verbindungsaussehens

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

3.31 Verschiebehilfen

In Gantt-Charts kann einfach in die Planung eingegriffen werden, indem Aufträge, Vorgänge oder Ressourcen per Maus verschoben werden. Die exakte Positionierung ist jedoch nicht immer ganz einfach, da mit der Maus eine bestimmte Stelle im Gantt-Diagramm genau getroffen werden muss.

In vielen Gantt-Anwendungen gibt es zudem auch mehrstufige Gruppen. In großen Plänen und wenn Quell- und Zielgruppen weit voneinander entfernt liegen, kann so das Verschieben eines Knotens per Maus rasch unübersichtlich werden

Verschiebehilfen in horizontaler Richtung (Einrastwerkzeuge)

Aus Zeichenprogrammen oder Designer-Tools ist das sog. Einrastgitter bekannt, eine Hilfe zur exakten Positionierung von Objekten anhand eines vorgegebenen Gitters, meist in Pixel-Abständen gemessen. Eine ähnliche Funktionalität wurde in VARCHART XGantt implementiert. Dort werden die verschobenen Objekte aber nicht an einem festen Gitter ausgerichtet, sondern an anderen Objekten in der Grafik, die dann ein Einrastgitter mit unregelmäßigen Abständen festlegen.

> Objekte als Einrastziele

Bei Knoten (bzw. deren Layern), Stichtaglinien, Liniengittern und Kalendergittern ist die Definition von sog. Einrastzielen möglich, d.h. diese Objekte definieren bestimmte Stellen an sich selbst, die als Ziele für eine Einrastaktion anderer Objekte dienen. Beim horizontalen Verschieben oder Ändern der Größe eines Knotens oder Layers wird dessen Start- bzw. Endtermin zeitlich an den definierten Einrastzielen der anderen Objekte ausgerichtet. Der Start- oder Endtermin springt dabei innerhalb von 5 Pixeln neben einem Einrastziel an dieses heran und übernimmt damit dessen exakten Termin.

> Spezielles Verhalten für jedes Knotenlayout

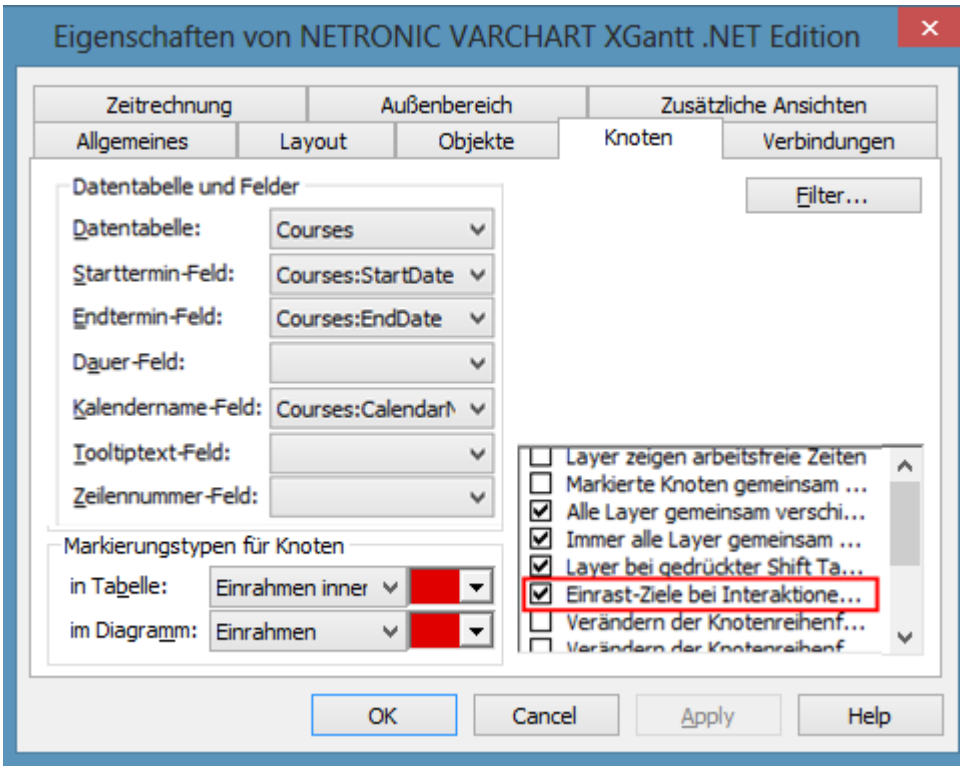
Für jedes Knotenlayout (ungruppiert, gruppiert, hierarchische Anordnung) wurde ein gesondertes Verhalten festgelegt (vorausgesetzt, die entsprechenden Objekte definieren Einrastziele):

- Für alle Knotenlayouts: Der zu verschiebende Layer wird an Stichtaglinien, Liniengittern und Kalendergittern ausgerichtet.

- Ungruppierte Anordnung: Der zu verschiebende Layer wird an den Layern aller Knoten ausgerichtet.
- Gruppierte Anordnung: Der zu verschiebende Layer wird an den Layern der Knoten derselben Gruppe (ohne Untergruppen) ausgerichtet. Bei einem eventuellen Wechsel der Gruppe während der Interaktion wird der Layer an den Objekten der neuen Gruppe ausgerichtet.
- Hierarchische Anordnung: Der zu verschiebende Layer wird an den Layern der Knoten desselben Astes (mit Unterästen) ausgerichtet. Bei einem evtl. Wechsel des Astes während der Interaktion wird der Layer an den Objekten des neuen Astes ausgerichtet.

> Neue Eigenschaften und API-Aufruf

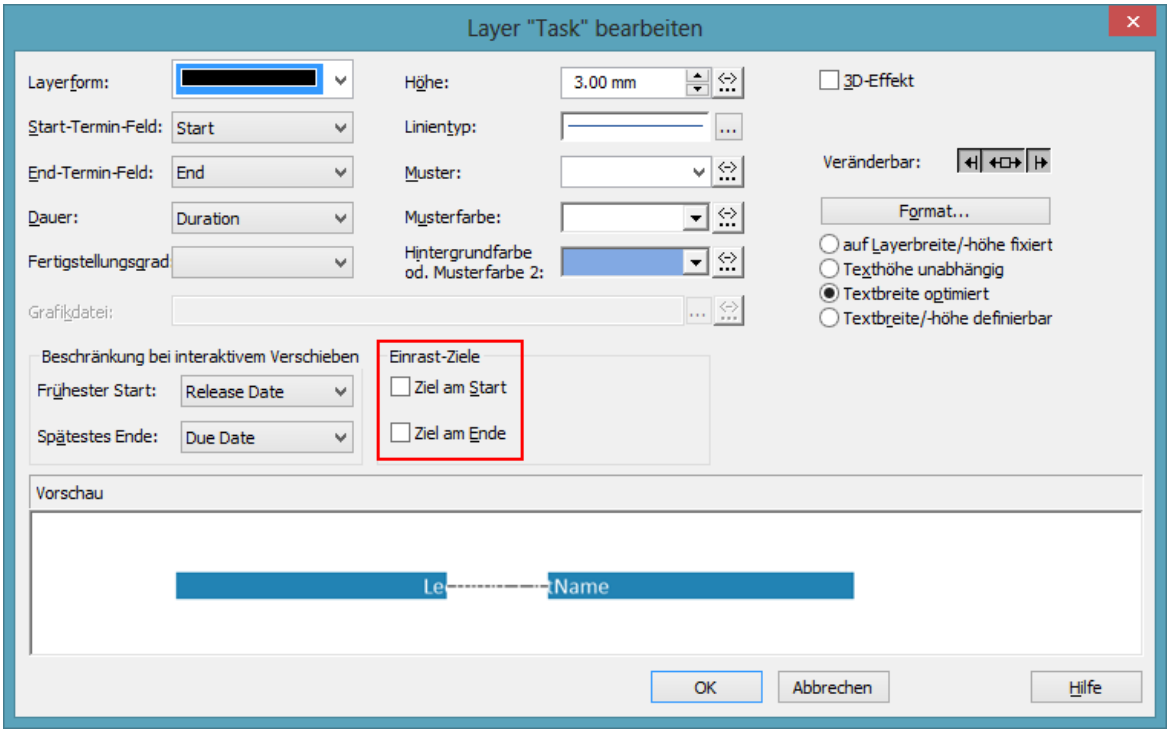
Damit die Einrastwerkzeuge wirksam werden, müssen sie auf der Eigenschaftenseite **Knoten** zugelassen werden:



API-Aufruf: **vcGantt.UseSnapTargetsInInteractions = true/false**

> Einrastziel LAYER

Im Dialog Layer bearbeiten kann durch Auswahl der Checkboxen **Ziel am Start-Termin** und **Ziel am End-Termin** bestimmt werden, dass der Layer seine Position (also seine Termine) als Einrastziele für das Verschieben eines Knotens bzw. Layers definiert.



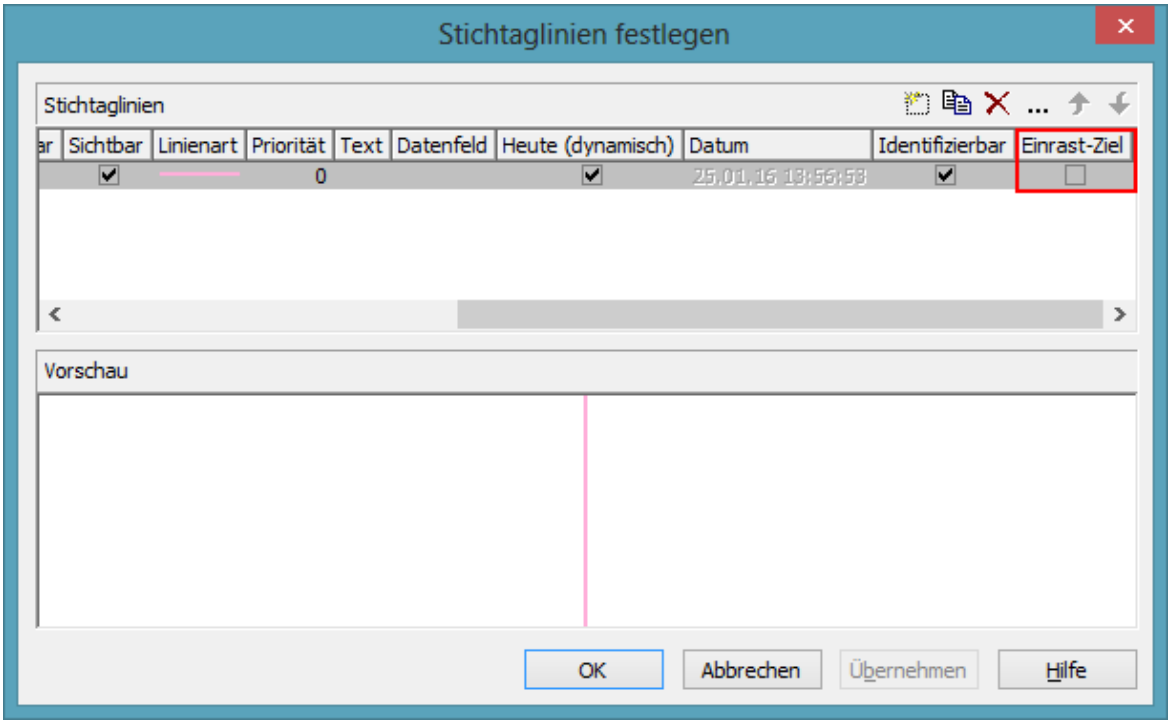
API-Aufrufe:

VcLayer.StartSnapTarget = true/false

VcLayer.EndSnapTarget = true/false

> Einrastziel STICHTAGLINIE

Im Dialog Stichtaglinien festlegen gibt es für jede Stichtaglinie eine entsprechende Checkbox Einrast-Ziel. Wenn sie ausgewählt ist, definiert die Stichtaglinie ihre Position (also ihr Datum) als Einrastziel für das Verschieben eines Knotens oder Layers.



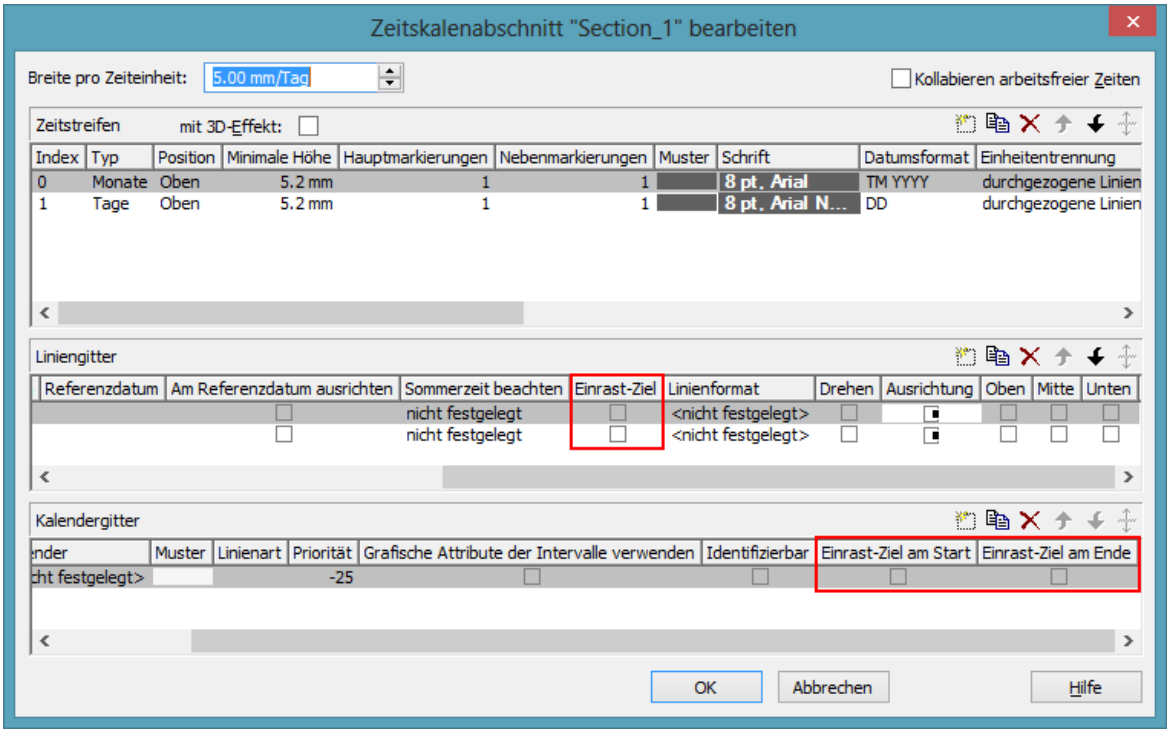
API-Aufruf: **VcDateLine.SnapTarget = true/false**

> Einrastziel LINIEN-/KALENDERGITTER

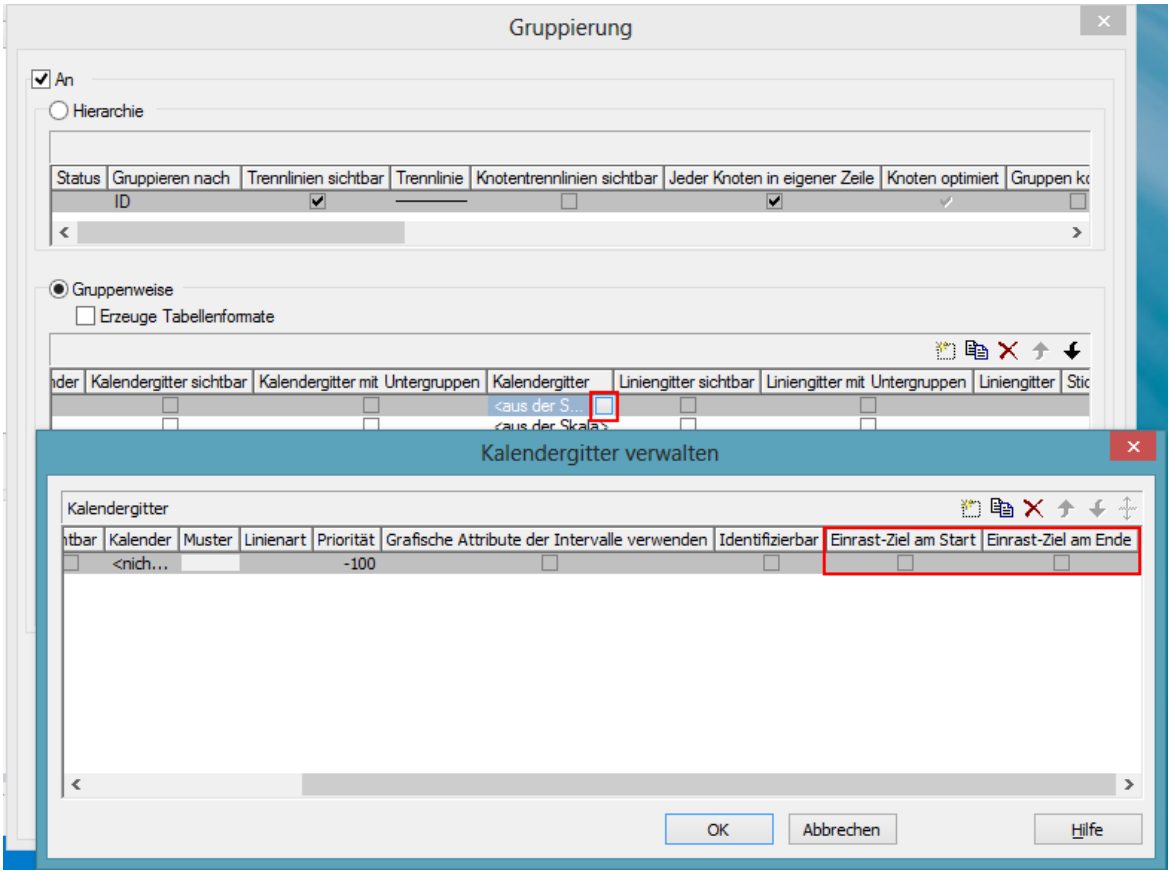
Für Liniengitter und Kalendergitter können die Einrastziele an zwei verschiedenen Stellen definiert werden:

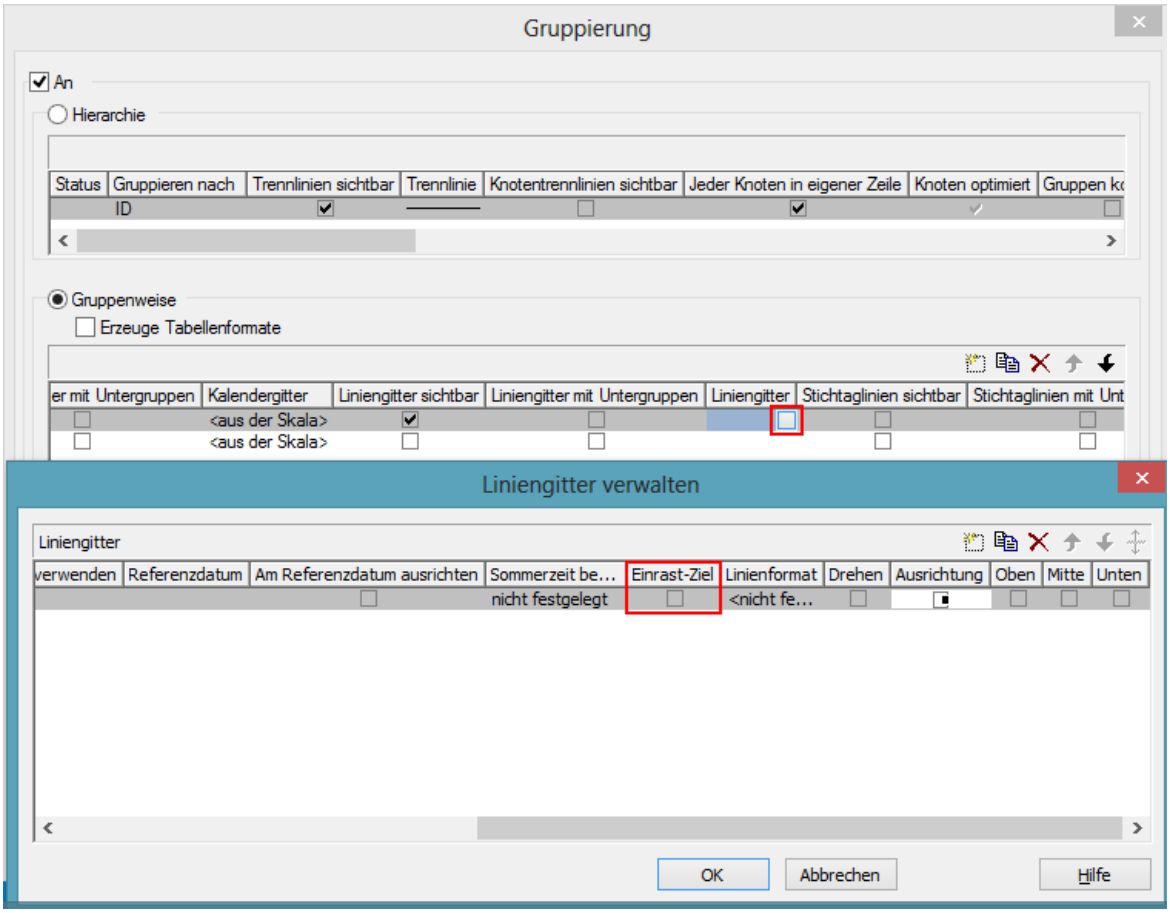
- im Dialog Zeitskalenabschnitt bearbeiten für nicht individuelle Objekte
- unterhalb des Gruppierung-Dialogs für individuelle, auf Gruppen oder Knoten bezogene Objekte

Wenn die entsprechenden Checkboxen im Dialog Zeitskalenabschnitt bearbeiten ausgewählt sind, definieren die jeweiligen Objekte ihre Positionen (also ihre Termine) als Einrastziele für das Verschieben eines Knotens oder Layers.



Aus dem Dialog **Gruppierung** kann man jeweils in die Dialoge **Kalendergitter verwalten** bzw. **Liniengitter verwalten** verzweigen. Dort kann durch Auswahl der entsprechenden Checkboxen bestimmt werden, dass die jeweiligen Objekte ihre Positionen (also ihre Termine) als Einrastziele für das Verschieben eines Knotens oder Layers definieren.


Gruppierung
An
Hierarchie
Status
Gruppieren nach
Trennlinien sichtbar
Trennlinie
Knotentrennlinien sichtbar
Jeder Knoten in eigener Zeile
Knoten optimiert
ID
Gruppenweise
Erzeuge Tabellenformate
Kalendergitter sichtbar
Kalendergitter mit Untergruppen
Kalendergitter
Liniengitter sichtbar
Liniengitter mit Untergruppen
Liniengitter
Kalendergitter verwalten
Kalendergitter
Kalender
Muster
Linienart
Priorität
Grafische Attribute der Intervalle verwenden
Identifizierbar
Einrast-Ziel am Start
Einrast-Ziel am Ende
OK
Abbrechen
Hilfe



API-Aufrufe:

VcDateLineGrid.SnapTarget = true/false

VcCalendarGrid.StartSnapTarget = true/false

VcCalendarGrid.StartSnapTarget = true/false

Achtung: Einrastziele von individuellen Objekten werden nur dann berücksichtigt, wenn ein einzelner Knoten verschoben wird, da es nicht sinnvoll ist, beim gleichzeitigen Verschieben mehrerer Knoten die Einrastziele aller Objekte (d.h. der Objekte aus mehreren Bändern) zu mischen. Ein separates Einrasten eines Knotens an den Einrastzielen des Bandes, in dem er liegt, ist nicht vorgesehen.

> Verschieben eines Knotens mit den Pfeiltasten

Knoten können nicht nur interaktiv mit der Maus verschoben werden, sondern auch über die Pfeiltasten auf der Tastatur. Zuvor muss allerdings noch folgende Einstellung vorgenommen werden:

vcGantt1.ArrowKeyMode = VcArrowKeyMode.vcResizeOrMoveNode.

Der Aufzählung **VcArrowKeyMode** wurde der Wert **vcNodeJumpToSnapTarget** hinzugefügt. Wenn dieser Wert gesetzt ist,

kann durch Drücken von STRG + Pfeiltaste links bzw. rechts mit einem zuvor markierten Knoten zum vorherigen bzw. nächsten Einrastziel gesprungen werden. Diese Aktion ist zyklisch, d.h. ist man an einem Ende angekommen, beginnt man wieder vorne.

Verschiebehilfen in vertikaler Richtung (Automatisches Kollabieren und Expandieren von Gruppen)

Im Windows-Explorer wird beim Verschieben einer Datei die Ordnerstruktur automatisch aufgeklappt. Dabei schiebt man die Datei auf einen kollabierten Ordner im Navigationsbereich, verharrt einen Moment, der Ordner öffnet sich und man kann weiter schieben, bis man den gewünschten Zielordner gefunden hat.

> Verhalten in älteren Versionen

In VARCHART XGantt musste man bisher beim vertikalen Verschieben eines Knotens in eine andere Gruppe recht lange nach der Zielgruppe suchen, wenn das Diagramm sehr viele Knoten in vielen expandierten Gruppen enthielt. Um die gesuchte Zielgruppe zu erreichen, war meist ein automatisches vertikales Scrollen notwendig, was mitunter langwierig und daher unkomfortabel war.

> Neu: Einfache Orientierung & schnelles vertikales Verschieben

Mit der neuen Funktionalität lässt sich die Suche nach der Zielgruppe deutlich abkürzen. Da hierbei die Kombinations- und Einstellmöglichkeiten ausgesprochen vielfältig sind, möchten wir uns an dieser Stelle darauf beschränken, eine mögliche Konfiguration vorzustellen.

Beispiel: Alle Gruppen außer der aktuellen kollabieren automatisch

Man kann VARCHART XGantt so konfigurieren, dass beim Verschieben eines Knotens alle Gruppen mit Ausnahme der gerade berührten Gruppe automatisch kollabiert werden. Der Zustand dieser Gruppe wird für den Fall beibehalten, dass man den Knoten nur innerhalb derselben Gruppe verschieben möchte. Durch das Kollabieren der anderen Gruppen reduziert sich die vertikale Ausdehnung des Plans auf einen Bruchteil der ursprünglichen Ausdehnung, wodurch wesentlich mehr Gruppen im Bild sichtbar sind als bisher. Im Idealfall sieht man schon jetzt die gesuchte Zielgruppe. Wenn nicht, kann VARCHART XGantt automatisch über die kollabierten Gruppen scrollen. Damit lässt sich die gewünschte Zielgruppe

wesentlich schneller finden als bisher. Ist die Zielgruppe erreicht, verharrt man einen Moment, die Zielgruppe wird expandiert und man kann weiterschieben. Dabei wird die zuvor berührte Gruppe kollabiert, so dass die Größe des Plans minimiert bleibt. Man schiebt weiter, ggf. noch einmal auf eine andere Gruppe. Diese wird expandiert, die zuvor expandierte Gruppe kollabiert usw., bis man sein Ziel erreicht hat. Mit dem Loslassen des Knotens in der Zielgruppe ist die Interaktion beendet und VARCHART XGantt kann, falls gewünscht, den alten Zustand der Gruppen wieder herstellen, wobei auf die neue Position des verschobenen Knotens gescrollt wird.

> Viele Kombinationsmöglichkeiten

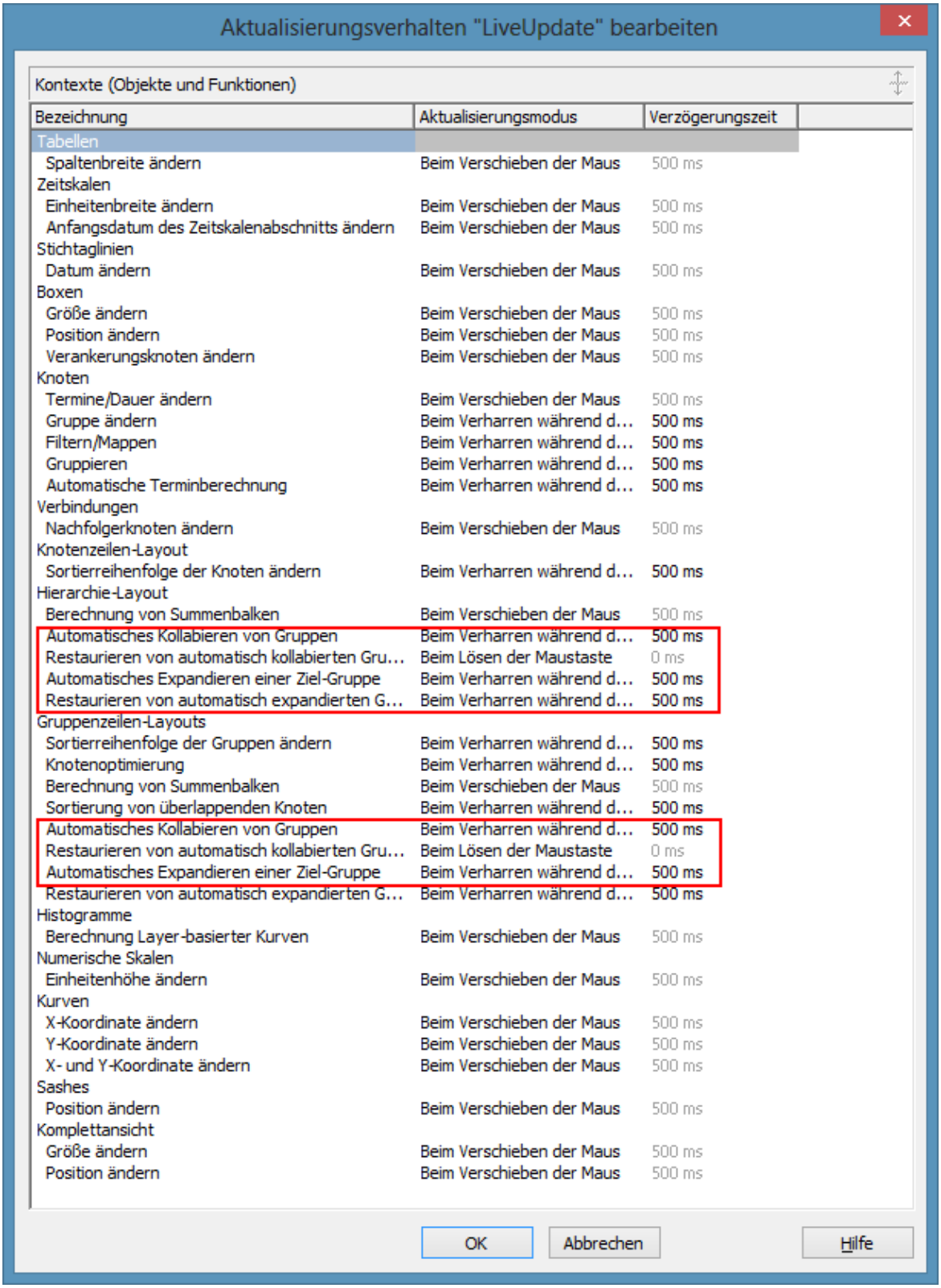
Dies war nur ein Beispiel für die neue Funktionalität. Weitere Einstellmöglichkeiten gibt es für:

- automatisches Kollabieren von Gruppen
- automatisches Expandieren von Gruppen
- automatisches Restaurieren von automatisch kollabierten oder expandierten Gruppen, die sich über ein Aktualisierungsverhalten zeitlich genau steuern lassen.

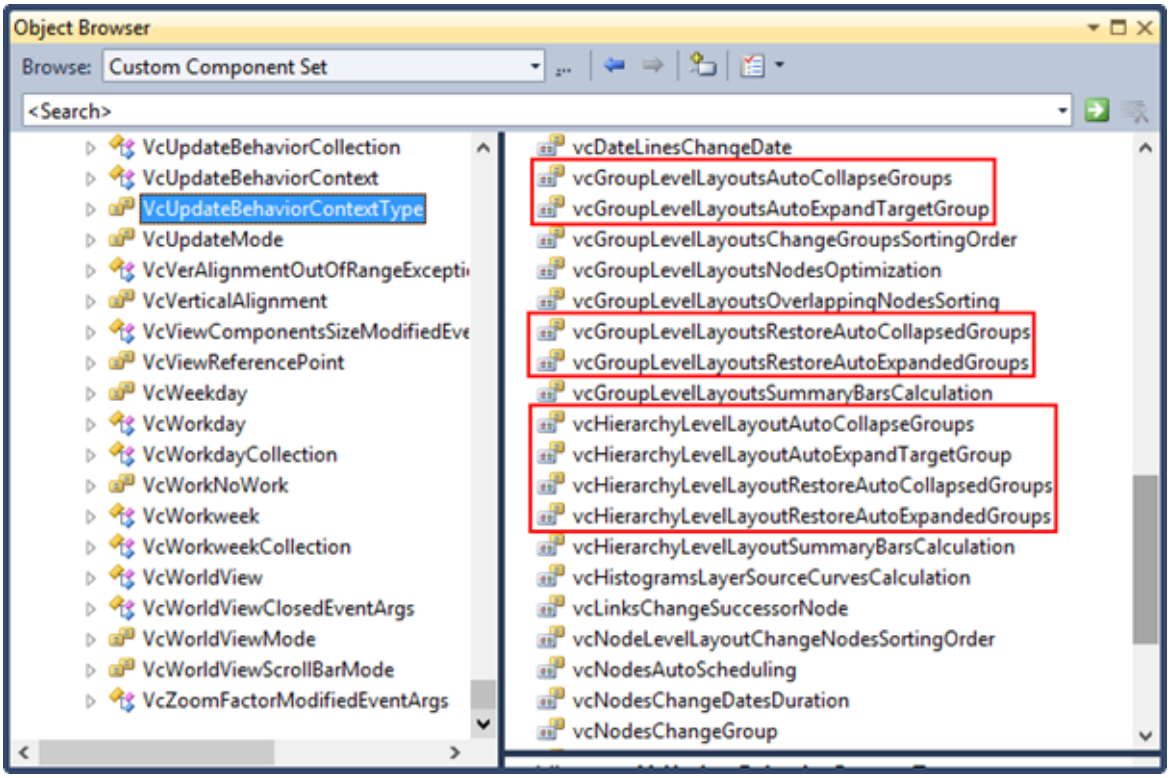
Diese Einstellungen können pro Gruppenebene und auch für die hierarchische Anordnung der Knoten vorgenommen werden, wodurch man die Verschiebevorgänge sehr detailliert gestalten kann.

> Neue Eigenschaften und API-Aufrufe

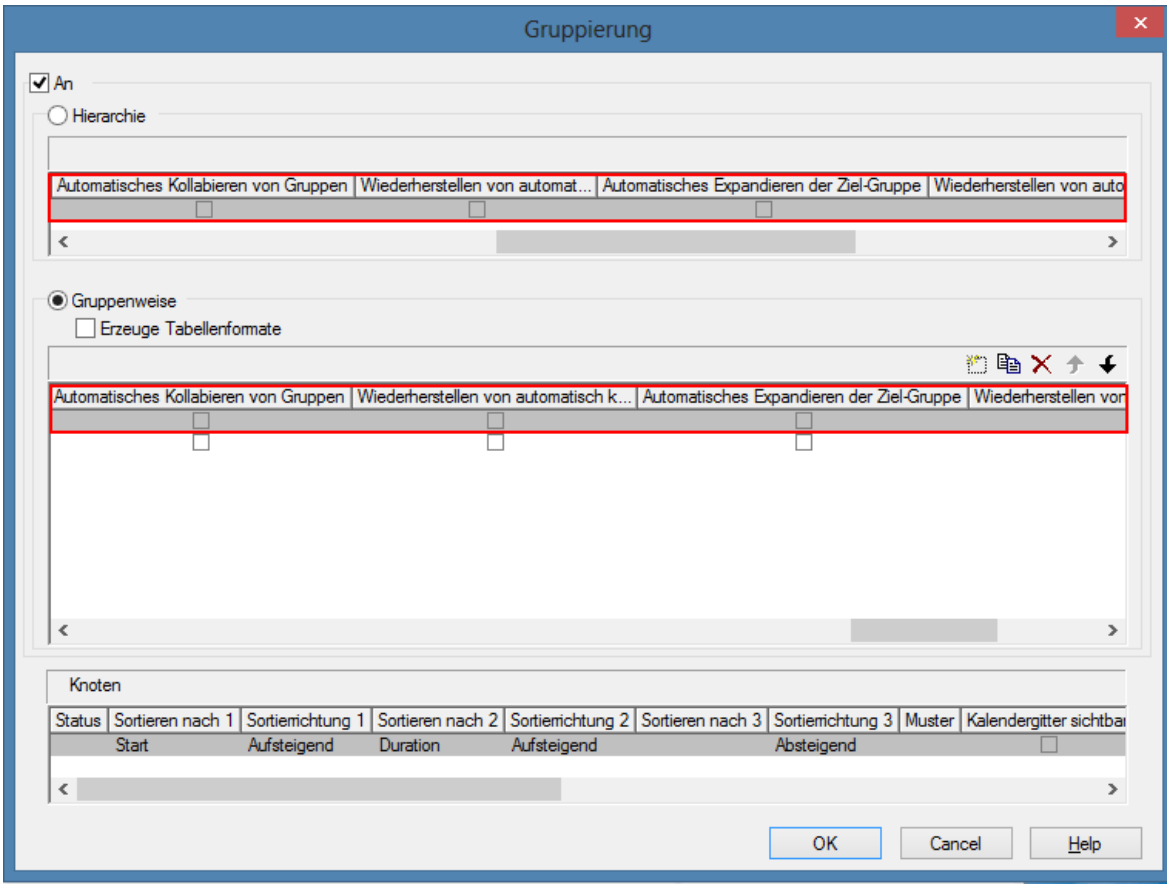
Im Dialog **Aktualisierungsverhalten bearbeiten** gibt es acht entsprechende Kontexte, jeweils vier bei **Gruppenzeilen-Layouts** und bei **Hierarchie-Layout**:



Um diese neuen Kontexte auch zur Laufzeit einstellen zu können, wurde die Aufzählung **VcUpdateBehaviorContextType** ebenfalls um 8 Werte ergänzt:



Die Funktionalitäten, die durch diese Kontexte durch Timer gesteuert aktiviert werden, können im Dialog **Gruppierung** ein- und ausgeschaltet werden. Dort wurden jeweils vier Spalten bei Hierarchie und Gruppenweise hinzugefügt.



API-Aufrufe:

VcGroupLevelLayout.AutoCollapseGroups = true/false

VcGroupLevelLayout.AutoExpandTargetGroup = true/false

VcGroupLevelLayout.RestoreAutoCollapsedGroups = true/false

VcGroupLevelLayout.RestoreAutoExpandedGroups = true/false

VcHierarchyLevelLayout.AutoCollapseGroups= true/false

VcHierarchyLevelLayout.AutoExpandTargetGroup = true/false

VcHierarchyLevelLayout.RestoreAutoCollapsedGroups = true/false

VcHierarchyLevelLayout.RestoreAutoExpandedGroups = true/false

3.32 Verwendung des Steuerelementes in HTML-Seiten

Der Microsoft Internet Explorer kann Windows-Forms-Steuerelemente, die in einer HTML-Seite eingebettet sind, anzeigen.

Um VARCHART XGantt in einer HTML-Seite zu verwenden, gehen Sie wie folgt vor:

Sie entwickeln zunächst Ihre Anwendung in Form einer Windows-Steuerelementbibliothek. Der wesentliche Unterschied zu einer Windows-Anwendung liegt darin, dass Sie in Microsoft Visual Studio ein andere Projektvorlage benutzen. Dadurch leitet sich die neue Klasse nicht von **System.Windows.Forms.Form**,

sondern von

System.Windows.Forms.UserControl

ab. Ziehen Sie das Steuerelement VARCHART XGantt auf das UserControl1. Der Code, den Sie schreiben müssen, entspricht dem einer Windows-Anwendung. Als Ergebnis erhalten Sie statt einer EXE-Datei eine Assembly in Form einer DLL. Das im vorherigen Kapitel entwickelte Einstiegsbeispiel finden Sie als fertige Windows-Steuerelementbibliothek im Ordner

UserGuideSamples\VB.NET\XGantt_Tutorial01_Web bzw. im Ordner

UserGuideSamples\CSharp\XGantt_Tutorial01_Web.

Alle für die Veröffentlichung über einen Webserver benötigten Dateien sollten Sie in einem Ordner zusammenstellen. Dies haben wir bereits für Sie im Ordner UserGuideSamples\Web getan.

Insgesamt sind fünf Dateien erforderlich:

Dateiname	Inhalt
NETRONIC.XGantt.dll	VARCHART XGantt Steuerelement
XGantt_Tutorial01_Web.dll	Steuerelementbibliothek mit UserControl1
Configuration.xml	Konfigurationsdatei für die Pfadsuche
Tutorial01.html	HTML-Startseite
xdependent.cab	Container für DLLs, die von XGantt benötigt werden

Mit der Datei **configuration.xml** stellen Sie sicher, dass die Assembly-Dlls im gleichen Verzeichnis gesucht werden, aus dem auch die HTML-Datei abgerufen wird.

Der Inhalt von **configuration.xml** sieht wie folgt aus:

Code-Beispiel

```
<configuration>
   <runtime>
      <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
         <probing privatePath=""/>
      </assemblyBinding>
   </runtime>
</configuration>
```

In der HTML-Datei **_Tutorial01.htm** müssen Verweise auf das Steuerelement und die Datei xdependent.cab eingebettet werden. Hierfür dienen die Object-Tags. Sie enthalten die Attribute **id**, **classid**, **height** und **width**.

Code-Beispiel

```
<html>
   <head>
      <title>VARCHART XGantt .NET WinForm</title>
      <link rel="Configuration" href="Configuration.xml">
   </head>
   <body>
      <object id="XDependentDummy" width=0 height=0
              standby="Please wait while loading the diagram
prerequisites..."
              classid="CLSID:544C9013-D784-472f-8EA6-BDF86ECF0427"
              codebase="xdependent.cab#version=8,4,2,0"/>
      <object id="XGantt_Tutorial01_WebLibrary"

classid="http:XGantt_Tutorial01_Web.dll#XGantt_Tutorial01_Web.UserContro
l1"
              height="500" width="910"/>
   </body>
</html>
```

Mit **id** wird ein frei wählbarer Bezeichner vergeben. Die **classid** spezifiziert die Herkunft. Mit dem HTTP-Protokoll wird die Assembly

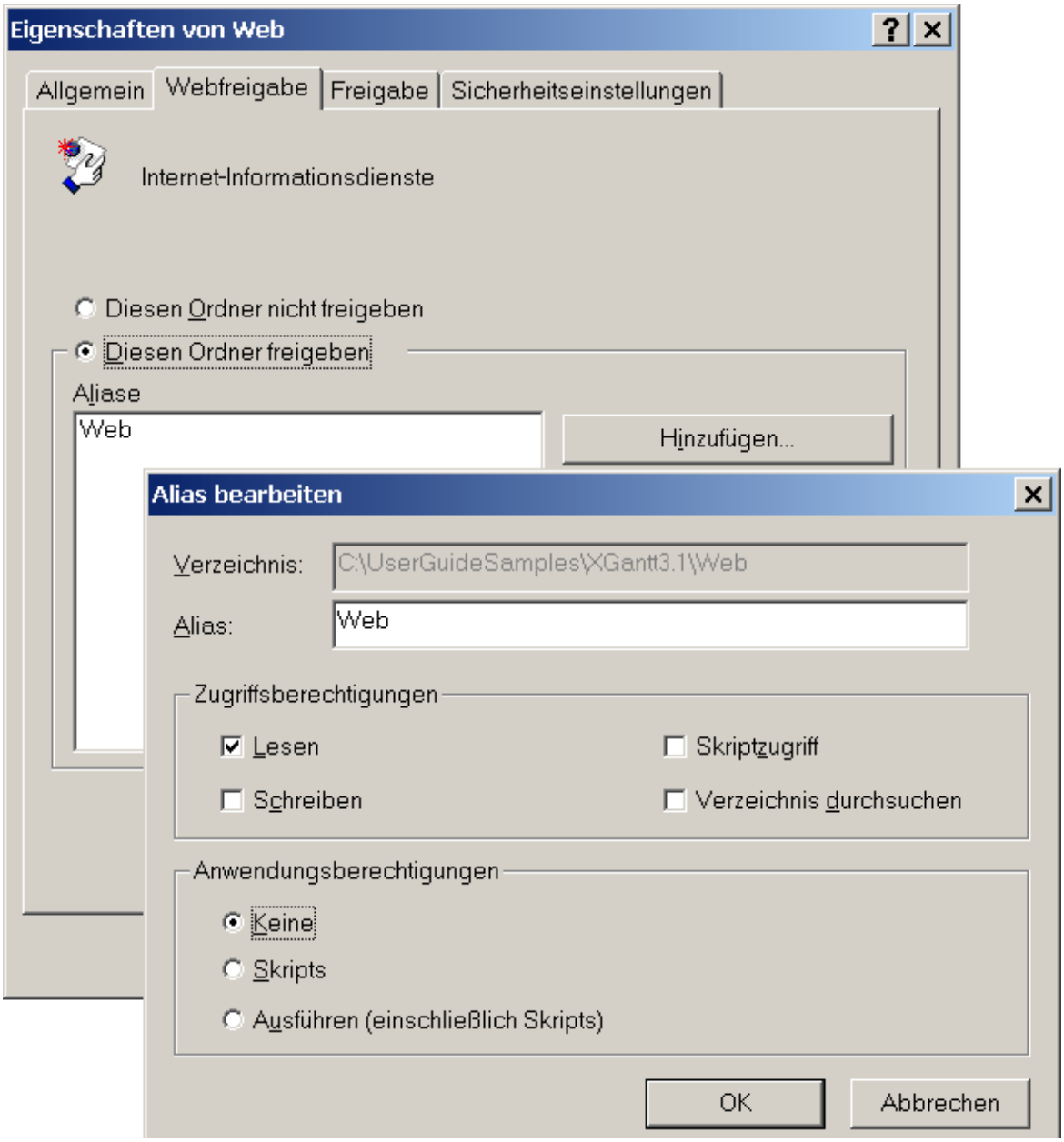
XGantt_Tutorial01_Web.dl

herunter geladen und innerhalb der Steuerelementbibliothek das Steuerelement **XGantt_Tutorial01_Web.UserControl1** angesprochen. Die Angabe des konkreten Steuerelementnamens ist notwendig, da eine Steuerelementbibliothek vom Prinzip her mehrere Steuerelemente enthalten kann. Über die Attribute **height** und **width** wird die Größe der Fläche spezifiziert, die das Steuerelement auf der HTML-Seite einnehmen wird.

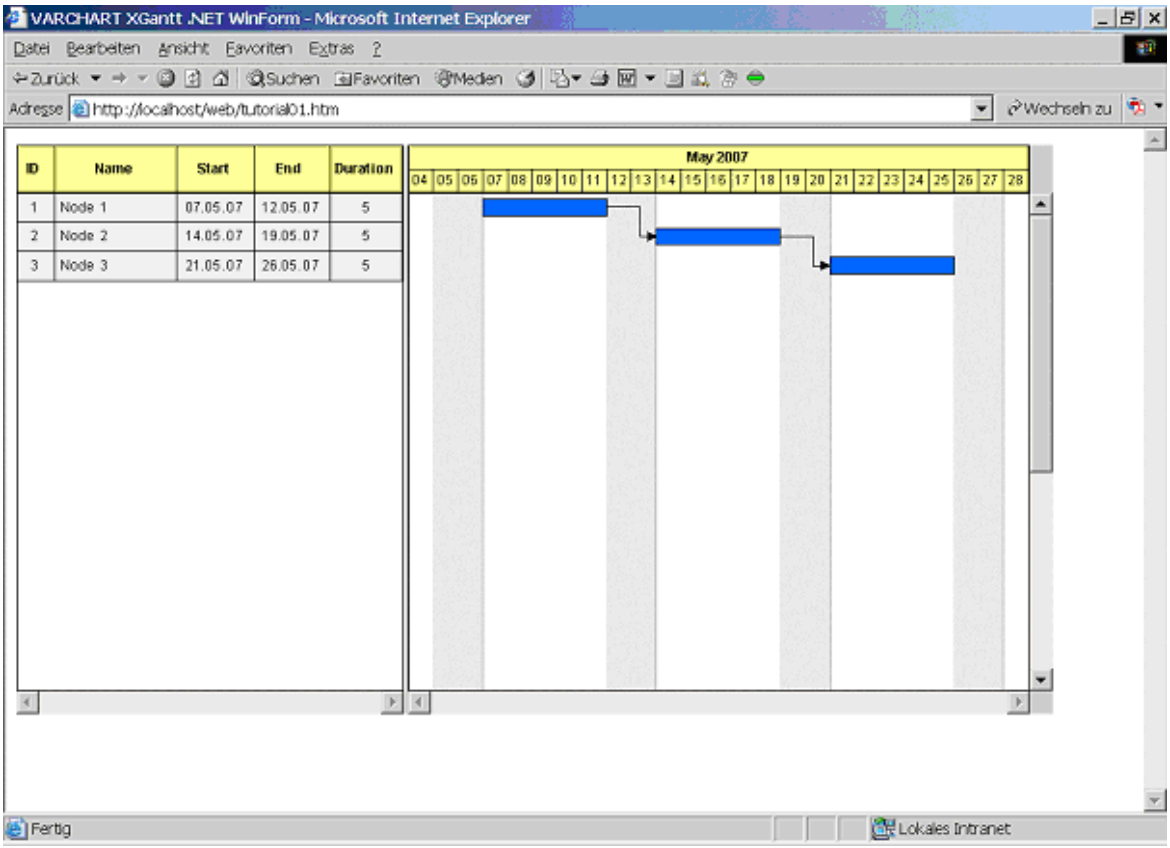
Das Object-Tag für die Datei xdependent.cab is nötig, da der Internet Explorer bzw. das .NET Framework 2.0 das automatische Herunterladen von Assemblys wie XGantt nicht unterstützen. XGantt benötigt die DLLs

MFC80U.DLL und MFVC80P.DLL, um ausgeführt werden zu können. Daher werden diese DLLs von einer CAB-Datei heruntergeladen, die nur beim ersten Aufruf ein kleines Dummy-ActiveX-Steuerelement installiert. Diese CAB-Datei ist signiert. Die Interneteinstellungen für die Server-Zone müssen geändert werden, damit signierte ActiveX-Steuerelemente heruntergeladen und installiert werden können. Weitere Informationen über die Internet-Einstellungen finden Sie in Kapitel 3.12 **Laufzeitsicherheitsrichtlinien für den Einsatz im Internet Explorer**.

Die Web-Anwendung lässt sich am einfachsten mit Hilfe des lokalen Internet Information Service testen. Für den Ordner **Web** wird eine Webfreigabe erstellt. Über das Kontextmenü des Ordners und den Eintrag **Eigenschaften** gelangen Sie zu dem entsprechenden Dialog.



Danach kann im Internet Explorer unter Angabe der Adresse http://localhost/web/tutorial01.htm die Seite abgerufen werden.



"

3.33 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 VcGantt oder mit dem Standard-Kontextmenü für das Diagramm können Sie ein Diagramm in einer Datei abspeichern.

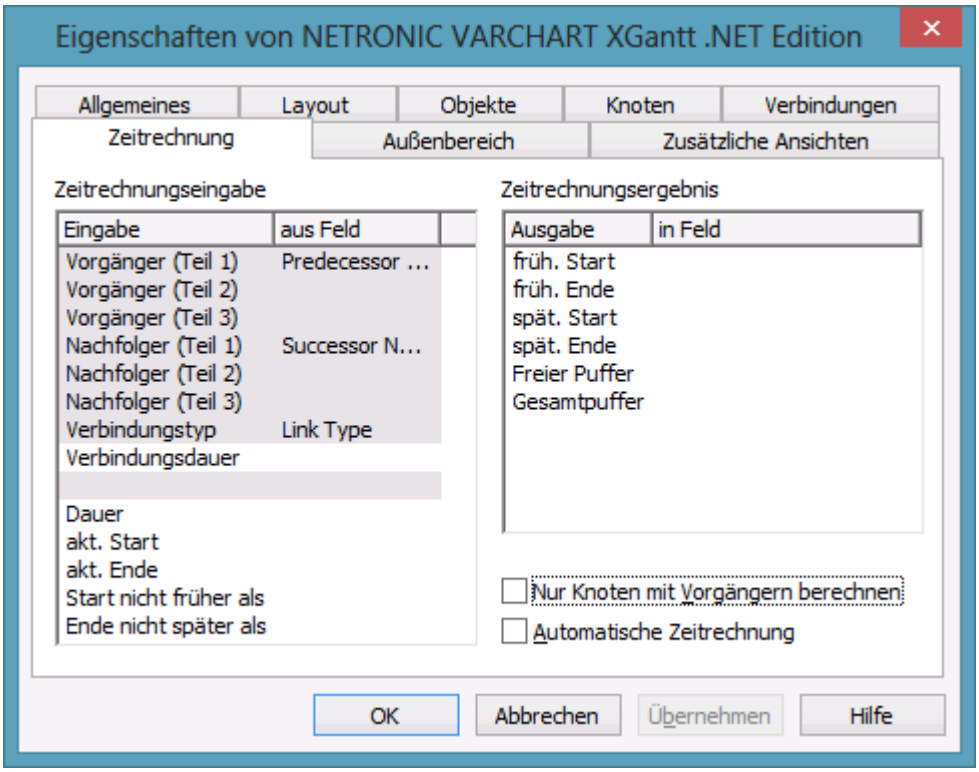
3.34 Zeitrechnung

Mit dem VARCHART-XGantt-Scheduler 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 XGantt an Ihre Schnittstelle anpassen, indem Sie festlegen, welche Datenfelder für die Eingabe (**Zeitrechnungseingabe**) und die Ausgabe (**Zeitrechnungsergebnis**) des Schedulers verwendet werden sollen.

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** und **Nachfolger** können in der **Zeitrechnungseingabe**-Tabelle nicht bearbeitet werden. Sie geben nur die auf der Eigenschaftenseite **Verbindungen** vorgenommenen Einstellungen wieder.



Die Ergebnisse werden wiederum in Datenfelder der Schnittstelle geschrieben. Als Ergebnisse stehen zur Verfügung: **früh. Start**, **früh. Ende**, **spät. Start**, **spät. Ende**, **Freier Puffer** und **Gesamtpuffer**. Jedem dieser

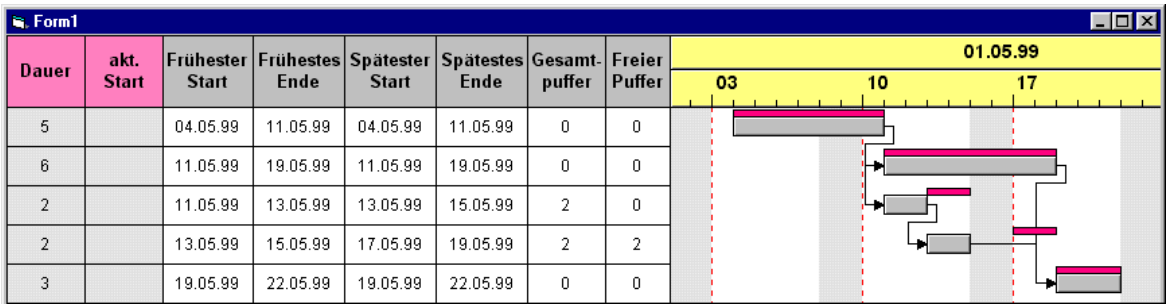
Ergebnisse können Sie ein Feld aus der in der Datendefinition vereinbarten Liste von Feldern zuweisen.

Alle folgenden Beispiele wurden für den Projektstart am 4.5.13 berechnet. Das erreichen Sie über die API mit der folgenden Codezeile:

Code-Beispiel VB.NET

```
VcGantt1.ScheduleProject ("04.05.2013", "11.05.2013")
```

Mit den oben dargestellten Einstellungen ergibt sich folgendes grafisches Ergebnis:



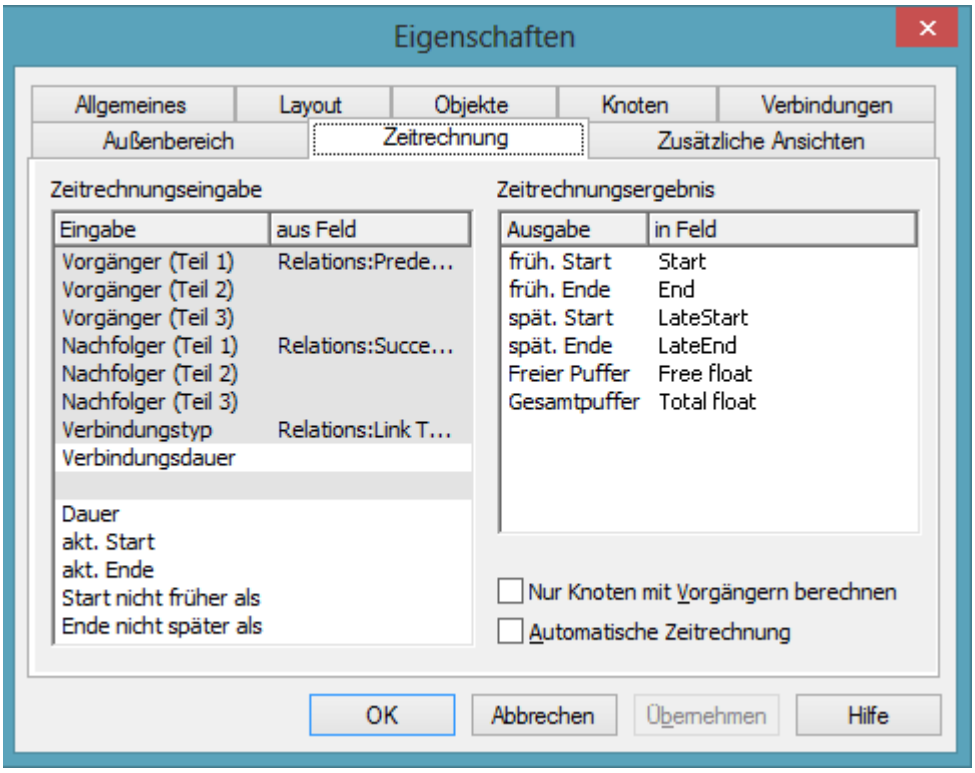
Dauer	akt. Start	Frühester Start	Frühestes Ende	Spätester Start	Spätestes Ende	Gesamt-puffer	Freier Puffer
5		04.05.99	11.05.99	04.05.99	11.05.99	0	0
6		11.05.99	19.05.99	11.05.99	19.05.99	0	0
2		11.05.99	13.05.99	13.05.99	15.05.99	2	0
2		13.05.99	15.05.99	17.05.99	19.05.99	2	2
3		19.05.99	22.05.99	19.05.99	22.05.99	0	0

In diesem Beispiel sind die frühen und späten Termine jeweils als Layer dargestellt.

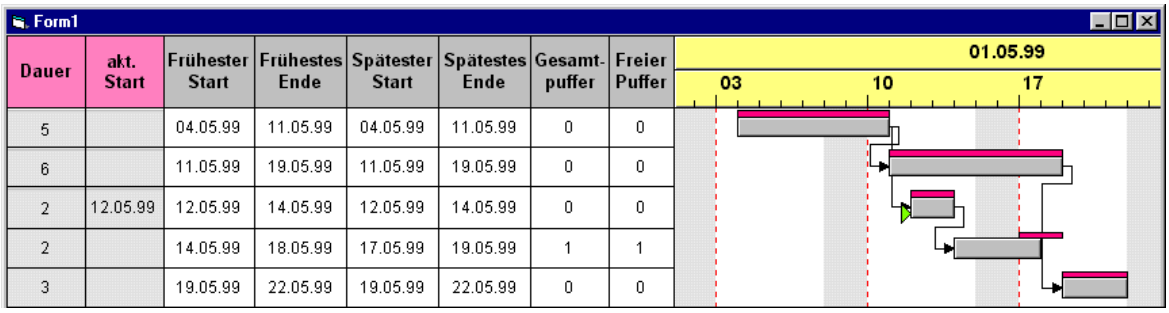
Es gibt noch weitere Möglichkeiten, die Zeitrechnung zu beeinflussen:

1. Sie können aktuelle Start- bzw. Endtermine angeben. Die Vorgänge sind dann unverrückbar.
2. Sie können für die Bedingungen **Start nicht früher als** und **Ende nicht später als** Referenztermine angeben. Dazu wird auf der Eigenschaften-seite **Zeitrechnung** in der linken Tabelle für die entsprechenden Werte auch jeweils ein Feld aus der Datendefinition festgelegt.

Für das folgende Beispiel wurden diese Einstellungen vorgenommen:

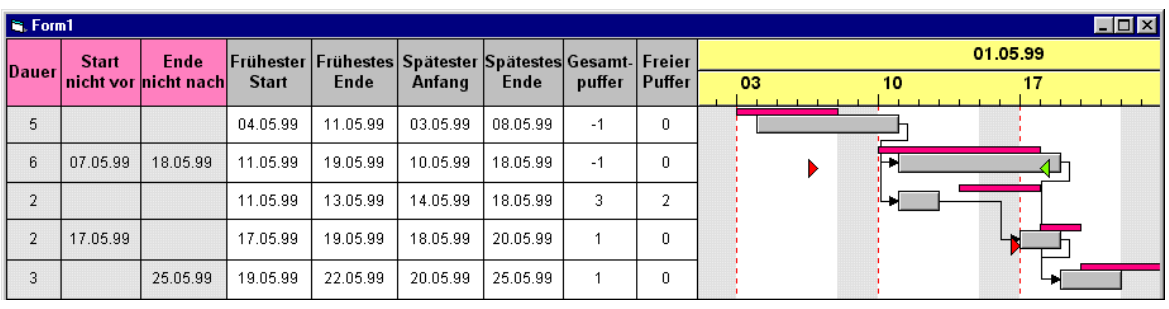


Wenn der aktuelle Start eines Vorgangs gesetzt ist, werden damit frühe und späte Termine fixiert. In dem folgenden Beispiel ist der gesetzte akt. Start durch ein grünes Dreieck markiert.



Dauer	akt. Start	Frühester Start	Frühestes Ende	Spätester Start	Spätestes Ende	Gesamt-puffer	Freier Puffer
5		04.05.99	11.05.99	04.05.99	11.05.99	0	0
6		11.05.99	19.05.99	11.05.99	19.05.99	0	0
2	12.05.99	12.05.99	14.05.99	12.05.99	14.05.99	0	0
2		14.05.99	18.05.99	17.05.99	19.05.99	1	1
3		19.05.99	22.05.99	19.05.99	22.05.99	0	0

Die Verwendung der Bedingungen **Start nicht früher als** und **Ende nicht später als** kann oder kann keine Auswirkungen haben. Im folgenden Beispiel sind die einschränkenden Termine durch rote und grüne Dreiecke visualisiert. Einige haben keinen Einfluss auf die Berechnung. Durch die Ende-Beschränkung des zweiten Vorgangs ergibt sich jedoch ein negativer Puffer für die ersten beiden Vorgänge.



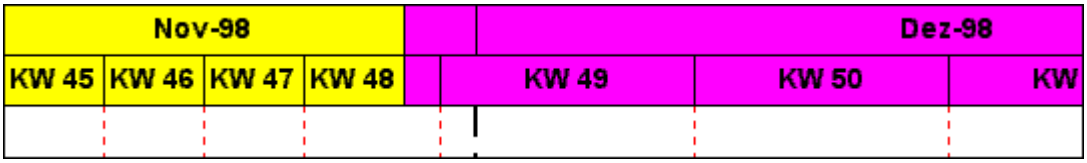
Dauer	Start nicht vor	Ende nicht nach	Frühester Start	Frühestes Ende	Spätester Anfang	Spätestes Ende	Gesamt-puffer	Freier Puffer
5			04.05.99	11.05.99	03.05.99	08.05.99	-1	0
6	07.05.99	18.05.99	11.05.99	19.05.99	10.05.99	18.05.99	-1	0
2			11.05.99	13.05.99	14.05.99	18.05.99	3	2
2	17.05.99		17.05.99	19.05.99	18.05.99	20.05.99	1	0
3		25.05.99	19.05.99	22.05.99	20.05.99	25.05.99	1	0

3.35 Zeitskala

Oberhalb des Diagrammbereichs wird die Zeitskala angezeigt. Ggf. können Sie einen oder mehrere beschriftete Zeitstreifen der Zeitskala auch unterhalb des Diagrammbereichs ausgeben (siehe Dialog **Zeitskalenabschnitt bearbeiten**, **Zeitstreifen**, **Position**).

Sie können aus verschiedenen Zeitskalen die zum dargestellten Zeitbereich passende Zeitskala auswählen.

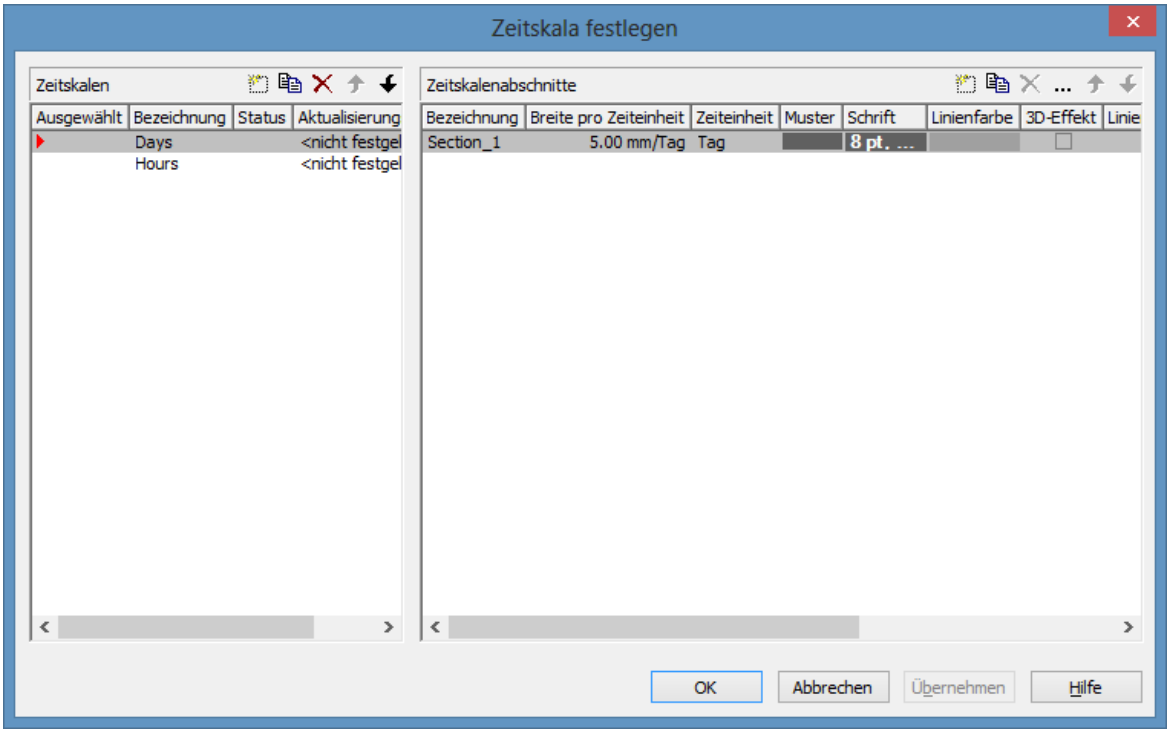
Sie können die Zeitskala in verschiedene Zeitskalenabschnitte unterteilen und deren Anzahl, Ausdehnung und Skalierung festlegen. Zeitabschnitte, die Sie besonders detailliert planen möchten, lassen sich dann gegenüber den übrigen gestreckt darstellen. So lässt sich beispielsweise die Planung für die nähere Zukunft detaillierter darstellen als die Planung für die fernere Zukunft oder der Vergangenheit. Auf diese Weise können Sie den Fokus auf die für Sie interessantesten Planungsabschnitte legen und diesen mit der Zeit verschieben. Sie können so ausgehend von einer Grobplanung sukzessive detaillierter planen.



Für die Darstellung der Zeitskala, Zeitskalenabschnitte und Gitter haben Sie vielfältige Gestaltungsmöglichkeiten. Sie können die Skalierungen, Notationen, Schriftattribute, die Ausrichtung der Beschriftung, die Farben, Strichdicken, Linientypen etc. festlegen, und zwar jeweils individuell für jedes Objekt.

Für die übersichtliche Gestaltung Ihrer Planung können Sie für jeden Zeitskalenabschnitt individuell Linien- und Kalendergitter einfügen.

Im Dialog **Zeitskala festlegen** können Sie verschiedene Zeitskalen definieren und auswählen, welche Sie für Ihre Darstellung verwenden möchten (**Ausgewählt**). Die Zeitskalen unterscheiden sich in der **Breite pro Zeiteinheit** und in den Zeitstreifen voneinander.



Die Auswahl der Zeitskala ist zur Laufzeit noch veränderbar.

> Anfang und Ende der Zeitskala festlegen

Auf der Eigenschaftenseite **Allgemeines** können Sie den standardmäßigen Anfang der Zeitskala festlegen (**Projektanfang** bzw. **Projektende**). Der Projektanfang wird zur Laufzeit mit der Eigenschaft **TimeScaleStart** oder der Methode **OptimizeTimeScaleStartEnd** an die aktuellen Daten angepasst. Das Datumsformat ist "DD.MM.YYYY;hh:mm:ss".

Hinweis: Das Enddatum ist nicht eingeschlossen. Geben Sie beispielsweise das Enddatum "31.12.02" an, ist der letzte Tag, der dargestellt wird, der 30.12.02.

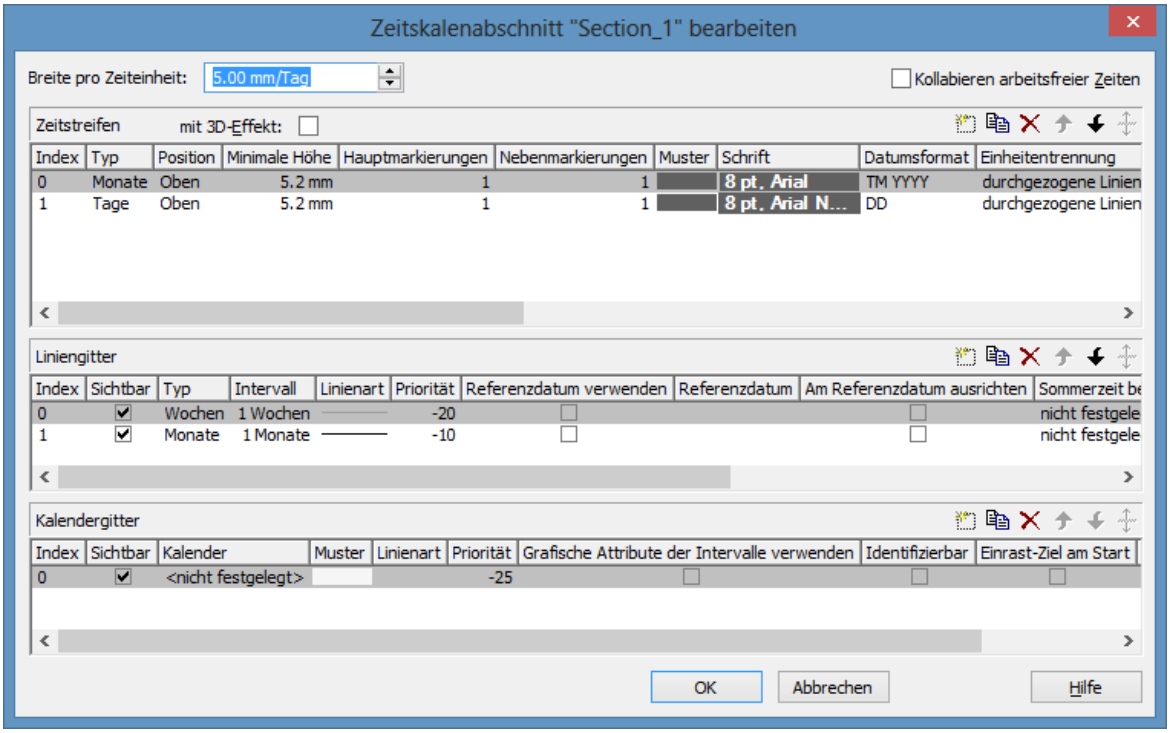
> Zeitskalenabschnitte

Sie können die Zeitskala in verschiedene Zeitskalenabschnitte unterteilen, um bestimmte Planungsabschnitte deutlich voneinander abzuheben, und für jeden Zeitskalenabschnitt verschiedene Zeitstreifen festlegen. Im Dialog **Zeitskala festlegen** können Sie für jeden Zeitskalenabschnitt individuell die **Zeiteinheit** und die **Breite pro Zeiteinheit** festlegen. Außerdem können Sie hier für jeden Zeitskalenabschnitt individuell **Farbe**, **Schriftart**, **Muster** und **3D-Effekt** festlegen, ein **Liniengitter** und ein **Kalendergitter** vereinbaren und festlegen, ob arbeitsfreie Zeiten ausgeblendet werden sollen.

Wenn Sie ein **Liniengitter** vereinbaren, werden in dem entsprechenden Zeitskalenabschnitt einstellbare vertikale Gitternetzlinien dargestellt.

Haben Sie für einen Zeitskalenabschnitt ein **Kalendergitter** vereinbart, werden in diesem Zeitskalenabschnitt die arbeitsfreien Zeiten durch vertikale farbige Flächen besonders markiert.

Vom Dialog **Zeitskala festlegen** gelangen Sie in das Dialogfeld **Zeitskalenabschnitt bearbeiten**, in dem Sie die einzelnen Zeitstreifen und Gitter jedes Zeitskalenabschnitts bearbeiten können.



> Breite pro Zeiteinheit

Die Zeiteinheit ist die kleinste Einheit, in die die Zeitskala unterteilt wird. Sie können Sie im Dialog **Zeitskala festlegen** in der Spalte **Zeiteinheit** auswählen. Möglich sind: Sekunde, Minute, Stunde und Tag.

Die **Breite pro Zeiteinheit** wird in Millimetern/Zeiteinheit angegeben und lässt sich in Schritten von Hundertstel Millimetern/Zeiteinheit verändern. Die minimale Breite beträgt 0,01 mm.

> Zeitstreifen

Zeitstreifen dienen zur Beschriftung der Zeitskala. Jeder Zeitskalenabschnitt kann mehrere Zeitstreifen besitzen (z. B. je einen mit einer Monats- und einer Tagesskala). Sie können für jeden Zeitstreifen die **Position** festlegen, d. h. ob er oberhalb oder unterhalb des Diagramms oder überhaupt nicht dargestellt werden soll. Außerdem können Sie für jeden Zeitstreifen Folgendes festlegen: Typ, Haupt- und Nebenmarkierungen, Farbe, Schriftart, Datumsformat,

Einheitentrennung, Ausrichtung, serielle Beschriftung, Referenzdatum und Kalender.

Für Datumsformat 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 "Uhr" (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

xC/XC: *Um dieses Datumsformat nutzen zu können, muss eine spezielle Einstellung in der Konfigurationsdatei vorgenommen werden. Bitte kontaktieren Sie bei Bedarf NETRONIC.* Sie können das Datum als maximal zehngliedrige, einfache Aufwärtszählung ab einem bestimmten Referenzdatum gestalten, z.B. "15:05:07:16:00". Dieses Beispiel datiert 15 Monate, 5 Tage, 7 Stunden, 16 Minuten und 0 Sekunden später als das gesetzte Referenzdatum. Die Notation dazu ist: **xC44:C33:C22:C11:C00**. Es sollen je 2 Stellen für die Glieder 4...0 vorgesehen werden, mit einem vorausgehenden "-" Symbol falls der Wert negativ ist. Die Trennzeichen sind variabel und könnten durch andere ersetzt werden. "x" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, aber kein "+"Symbol, falls er positiv ist. "X" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, und ebenso ein "+"Symbol, falls er positiv ist. Im Dialog **Zeitskalenabschnitt ... bearbeiten** sollten die Felder **Referenzdatum verwenden** und **Hauptmarkierung am Referenzdatum ausrichten** aktiviert sein, ebenso sollte **Serielle Beschriftungen** auf **Keine** gesetzt sein. Das Referenzdatum wird zur Laufzeit über die Methode **VcRibbon.set ReferenceDate** gesetzt und überschreibt die Werte des Dialogs.

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.

> Beispiel für die Beschriftung von Zeitstreifen

1. Zeitstreifen: **TWWW - TM - TQ - YYYY**, 2. Zeitstreifen: **TD**

KW37 - September - 3. Quartal - 2003							
Montag	Dienstag	Mittwoch	Donnerstag	Freitag	Samstag	Sonntag	Montag

Die vordefinierten Langtexte können durch benutzerdefinierte Texte ersetzt werden, indem die Eigenschaft **TextEntrySupplyingEventEnabled** auf "True" gesetzt wird. Dann kann auf die folgenden Werte des ControlIndex reagiert werden:

- vcTXERibDay0 bis vcTXERibDay6 (2212 bis 2218)

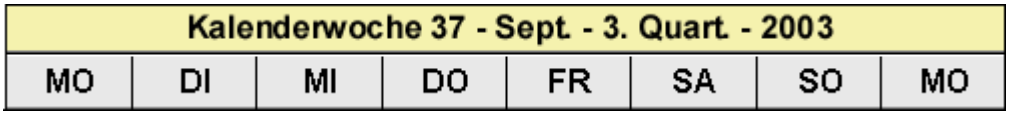
- vcTXERibCW (2223)
- vcTXERibMon0 bis vcTXERibMon11 (2200 bis 2211)
- vcTXERibQuar0 bis vcTXERibQuar2 (2219 bis 2222)

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.vcTXERibDay0
         e.Text = "MO"
    ...
      Case VcTextEntryIndex.vcTXERibCW
         e.Text = "KW"
      Case VcTextEntryIndex.vcTXERibMon8
         e.Text = "Sept."
      Case VcTextEntryIndex.vcTXERibQuar3
         e.Text = "3. Quart."
   End Select
End Sub
```

Code-Beispiel C#

```
private void VcGantt1_VcTextEntrySupplying(object sender,
NETRONIC.XGantt.VcTextEntrySupplyingEventArgs e)
   {
   switch(e.ControlIndex)
      {
      case VcTextEntryIndex.vcTXERibDay0:
         e.Text = "MO";
         break;
    ...
      case VcTextEntryIndex.vcTXERibCW:
         e.Text = "KW";
         break;
      case VcTextEntryIndex.vcTXERibMon8:
         e.Text = "Sept.";
         break;
      case VcTextEntryIndex.vcTXERibQuar3:
         e.Text = "3. Quart.";
         break;
      }
   }
```



3.36 Zuordnungstabellen

Zuordnungstabellen dienen dazu, ohne den Einsatz von Filtern bestimmte Eigenschaften datenfeldabhängig festzulegen. So können Sie z.B. Hintergrundfarbe, Muster, Musterfarbe und weitere Merkmale von Layern über Zuordnungstabellen datenabhängig gestalten.

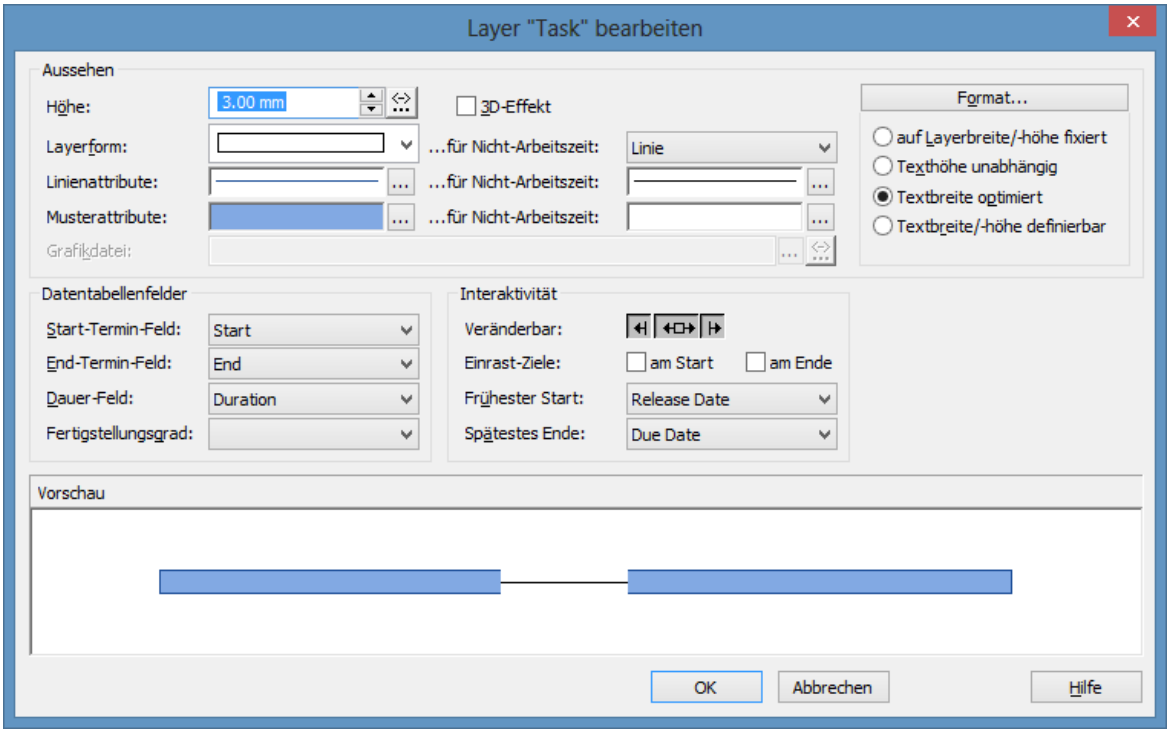
Zuordnungstabellen bestehen aus einer Liste von Zuordnungen. Jede Zuordnung besitzt einen Schlüssel und einen Wert. Der Wert ist je nach Typ der Zuordnungstabelle ein Grafikdateiname, ein Muster etc. Der Schlüssel ist jeweils ein möglicher Eintrag in einem Datenfeld. Zur Laufzeit werden die Schlüssel mit dem tatsächlichen Inhalt des angesprochenen Datenfelds verglichen und bei Gleichheit wird der Wert für die angesprochene grafische Eigenschaft eingesetzt.

Beispiel: dem Schlüssel "A" ist in der Zuordnungstabelle der Wert "grün" zugeordnet. Wenn nun die Zuordnungstabelle verwendet wird und irgendein Knoten den Feldwert "A" besitzt, wird die Farbe "grün" (z.B. als Hintergrundfarbe des Layers) zugewiesen. Als zweiter Wert könnte hier ein Legendentext "Ausführung termingerecht" zugeordnet sein. Grundsätzlich werden also die Inhalte der Datenfelder mit den Schlüsseln der Tabelle verglichen. Wenn sie sich entsprechen, werden die Werte verwendet, die den Schlüsseln zugeordnet sind.

Durch den Einsatz eines Filters statt eines Schlüssels lassen sich auch komplexere Zuordnungen angeben. Dabei werden generell bei der Auswertung einer Zuordnungstabelle zuerst die konkreten Schlüssel und dann bei Nichtzutreffen die Filter ausgewertet.

> Beispiel: Hintergrundfarbe von Layern

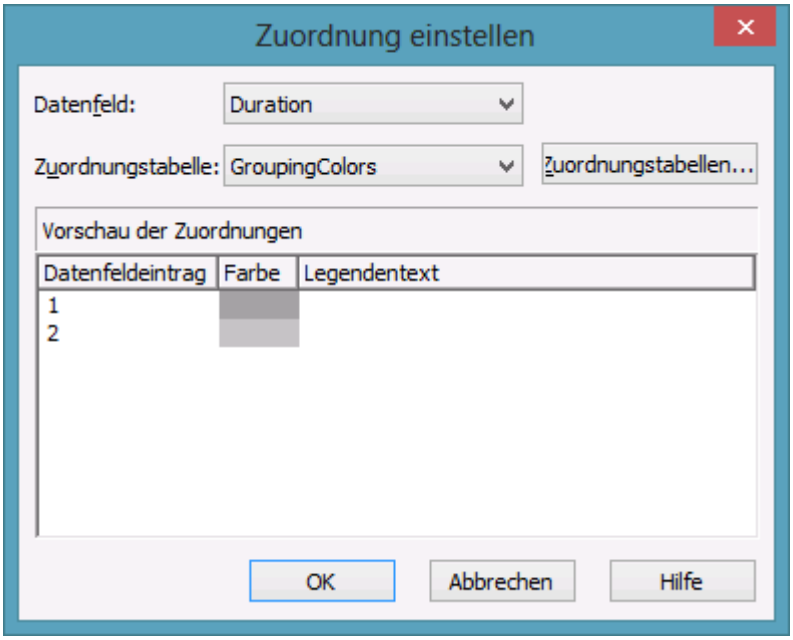
Im folgenden Beispiel soll die Hintergrundfarbe eines Layers über eine Zuordnungstabelle datenabhängig festgelegt werden. Klicken Sie dazu bitte im Dialogfeld **Layer bearbeiten** auf die Schaltfläche  für die **Hintergrundfarbe**.



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

> Zuordnung einstellen

IIm Dialogfeld **Zuordnung einstellen** verbinden Sie das Datenfeld eines Knotens mit einer Zuordnungstabelle, sodass die Werte des Datenfeldes und der Schlüssel der Tabelle miteinander verglichen werden können. So wird das gewünschte Attribut, im Beispiel die Hintergrundfarbe des Layers, datenabhängig festgelegt. Soll das Attribut nicht in Abhängigkeit nur eines einzelnen Wertes, sondern eines Wertebereiches oder noch komplexerer Kriterien festgelegt werden, so kann man einen Filter erstellen, den man im Dialog **Zuordnungstabelle bearbeiten** statt eines konkreten Wertes auswählt. Dieser Filter erscheint dann auch im Dialogfeld **Zuordnung einstellen** in der Auflistung der Datenfeldeinträge mit der Zuordnung des entsprechenden Attributes.

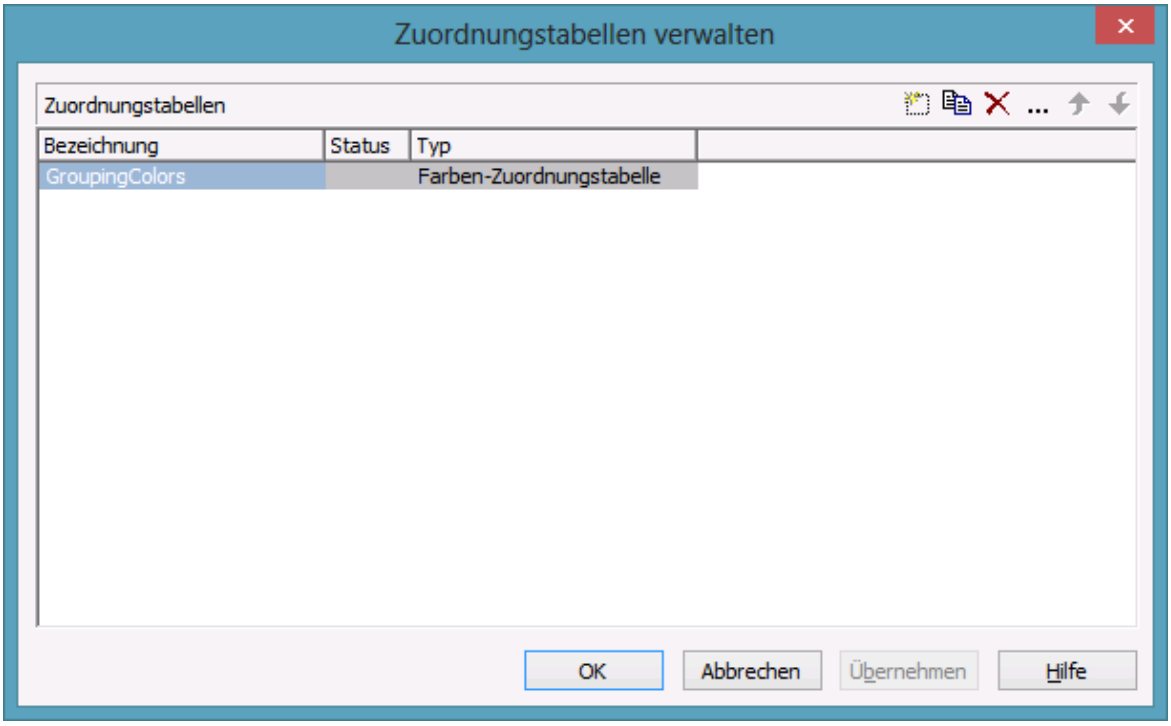


Um eine Zuordnung einzustellen, wählen Sie im obersten Feld des Dialogs das **Datenfeld** des Knotens, dessen Einträge mit den Schlüsseln der Tabelle verglichen werden sollen. Wählen Sie dann im Feld darunter die **Zuordnungstabelle**. (Es werden nur die Zuordnungstabellen angezeigt, die zu dem ausgewählten Attribut passen. Da Sie in unserem Beispiel im Dialog **Layer bearbeiten** die Hintergrundfarbe ausgewählt hatten, werden Ihnen hier auch nur Zuordnungstabellen vom Typ "Farben-Zuordnungstabelle" angeboten.) Nach der Auswahl werden die Werte der Zuordnungstabelle im Vorschaufenster des Dialogs sichtbar. Falls in Ihrem Beispiel noch keine Zuordnungstabelle vorhanden ist, legen Sie eine nach der Beschreibung des folgenden Kapitels an.

> Zuordnungstabellen verwalten

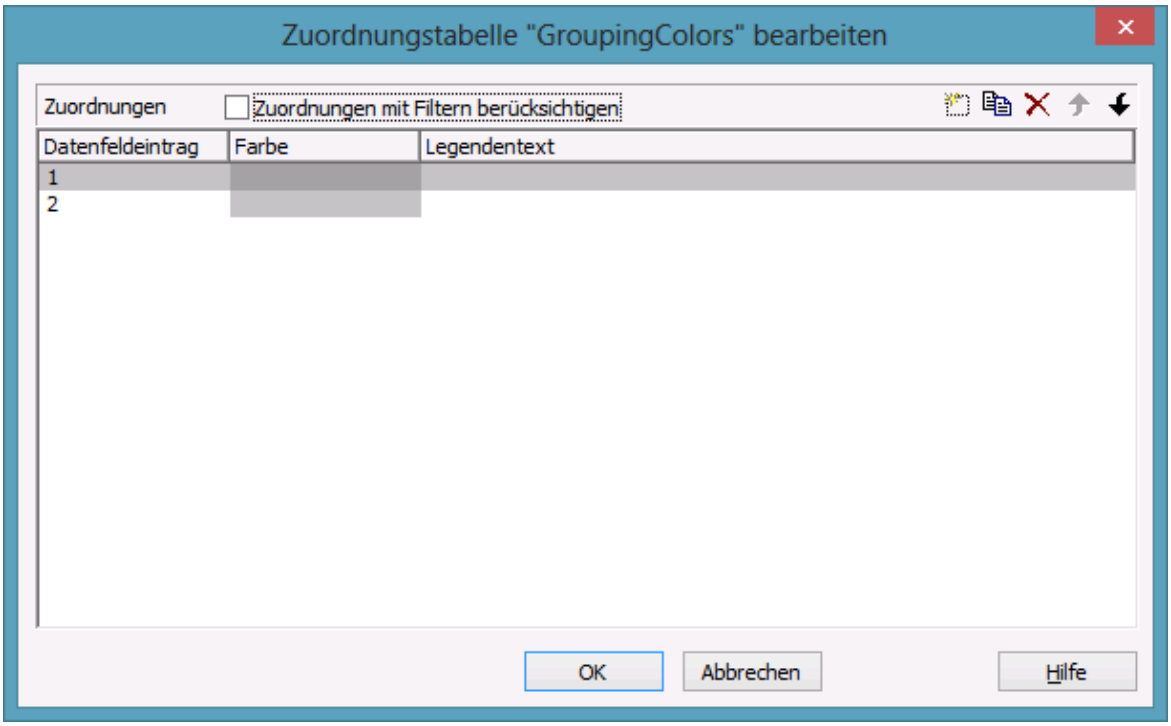
Im Dialogfeld **Zuordnungstabellen verwalten**, das Sie durch Klick auf die Schaltfläche **Zuordnungstabellen** oder über die Schaltfläche **Zuordnungstabellen** auf der Eigenschaftenseite **Objekte** erreichen, können Sie Namen und Typ einer Zuordnungstabelle durch direkte Eingabe verändern sowie über 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.

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

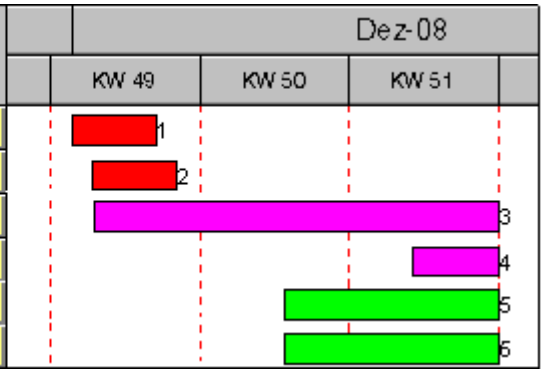
Wenn die Option **Zuordnungen mit Filtern berücksichtigen** ausgewählt ist, werden nicht nur die in der Liste der Datenfeldeinträge angegebenen festen Werte als Schlüssel berücksichtigt, sondern auch Filter, die aus der Dropdown-Liste ausgewählt werden können. Dadurch hängen die Ausführungswerte nicht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

> Beispiel

Bei dem hier definierten Layer werden die Vorgänge mit Datenfeldeintrag "A" in rot dargestellt, die Vorgänge mit Datenfeldeintrag "B" in rosa, etc. Die Standard-Hintergrundfarbe ist grau. Diese wird für Layer ohne Datenfeldeintrag bzw. mit einem Datenfeldeintrag, der nicht in der Zuordnungstabelle aufgeführt ist, verwendet.

ID	Beschreibung	Code 1
1	Vorgang 1	A
2	Vorgang 2	A
3	Vorgang 3	B
4	Vorgang 4	B
5	Vorgang 5	C
6	Vorgang 6	C



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

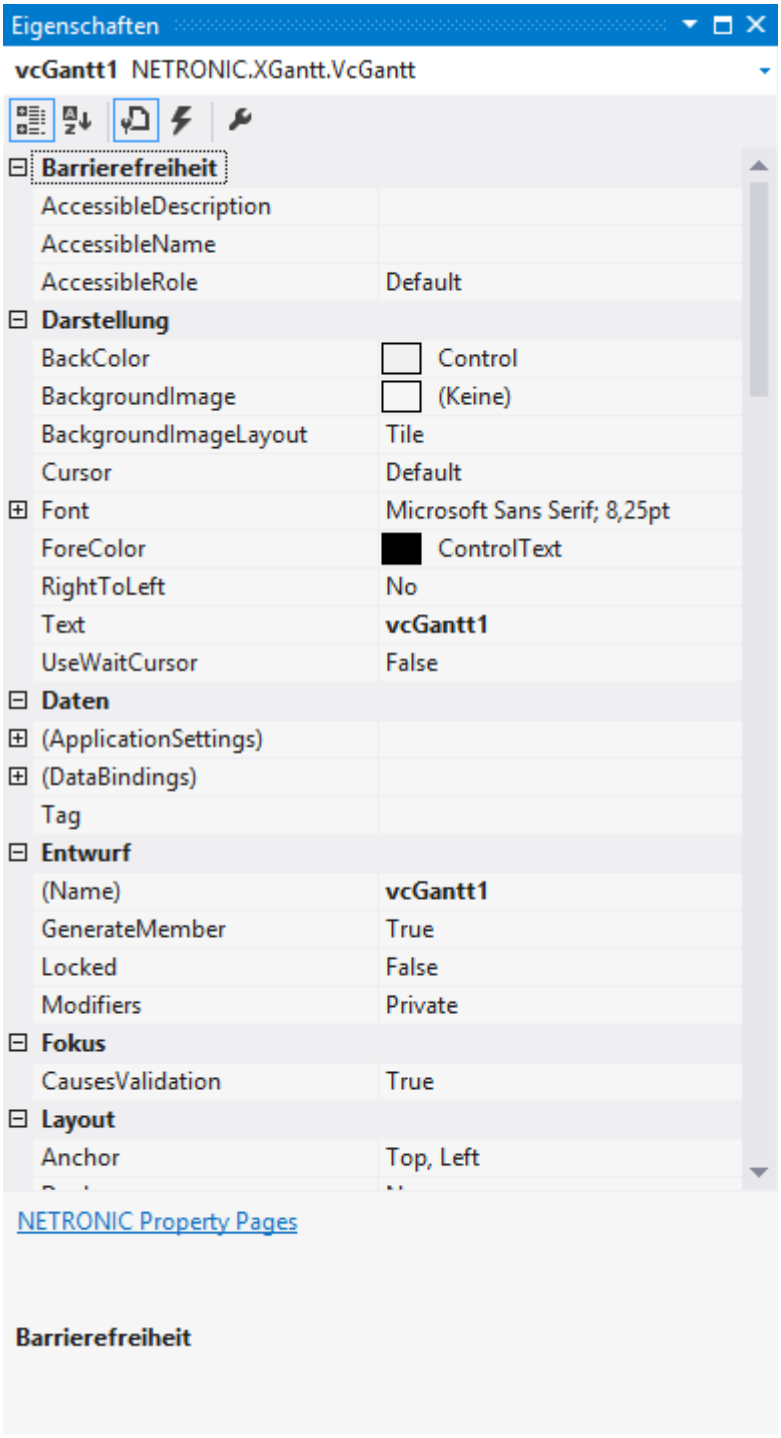
4.1 Allgemeines

Über die Eigenschaftenseiten kann VARCHART XGantt 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.

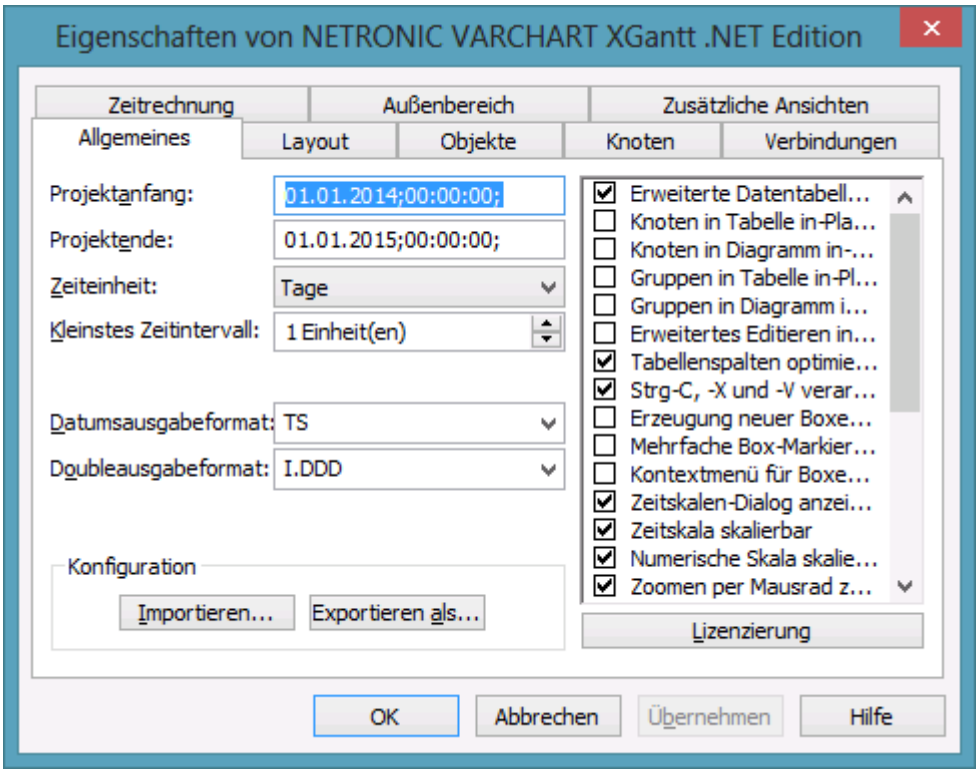
oder

- Im Eigenschaftenfenster (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 XGantt vornehmen.

Projektanfang

Legen Sie hier den standardmäßigen Anfang der Zeitskala fest. Der Anfang der Zeitskala kann zur Laufzeit mit der Eigenschaft **TimeScaleStart** oder der Methode **OptimizeTimeScaleStartEnd** an die aktuellen Daten angepasst werden. Das Datumsformat ist "DD.MM.YYYY;hh:mm:ss;".

Projektende

Legen Sie hier das standardmäßige Ende der Zeitskala fest. Das Ende der Zeitskala kann zur Laufzeit mit der Eigenschaft **TimeScaleEnd** oder der Methode **OptimizeTimeScaleStartEnd** an die aktuellen Daten angepasst werden. Das Datumsformat ist "DD.MM.YYYY;hh:mm:ss;".

Hinweis: Das angegebene Enddatum ist nicht eingeschlossen. Geben Sie beispielsweise das Enddatum "31.12.09" an, ist der letzte Tag der Zeitskala der 30.12.09.

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 **VcGantt.TimeUnit** gesetzt werden.

Kleinstes Zeitintervall

Legen Sie hier fest, wie viele Zeiteinheiten einem Schritt beim interaktiven Verändern eines Knotens entsprechen sollen.

Beispiel: Wenn Sie **Zeiteinheit** auf den Wert "Minuten" und **Kleinstes Zeitintervall** auf den Wert "30" setzen, beträgt das Zeitintervall für die Knotenpositionierung eine halbe Stunde. Der Balken bzw. Layer wird dann also immer zur vollen oder halben Stunde "einrasten".

Diese Option kann auch über die Eigenschaft **VcGantt.TimeUnitsPerStep** festgelegt 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 **VcGantt.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 **VcGantt.DoubleOutput-Format** 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 Dateiendungen. 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 **VcGantt.ExtendedDataTablesEnabled** gesetzt werden.

Knoten in Tabelle in-Place-Editieren

Aktivieren Sie dieses Kontrollkästchen, wenn das direkte Editieren von Knotendaten (bei eingeschalteter Gruppierung: von Blattknotendaten) in der Tabelle möglich sein soll. Diese Option kann auch über die Eigenschaft **VcGantt.InPlaceEditingOnNodesInTableEnabled** gesetzt werden.

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

Knoten in Diagramm in-Place-Editieren

Aktivieren Sie dieses Kontrollkästchen, wenn das direkte Editieren von Knotenlayern (bei eingeschalteter Gruppierung: von Blattknotenlayern) im Diagramm möglich sein soll. Diese Option kann auch über die Eigenschaft **VcGantt.InPlaceEditingOnNodesInDiagramEnabled** gesetzt werden.

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

Gruppen in Tabelle in-Place-Editieren

Aktivieren Sie dieses Kontrollkästchen, wenn das direkte Editieren von Gruppenknotendaten in der Tabelle möglich sein soll. Diese Option kann auch über die Eigenschaft **VcGantt.InPlaceEditingOnGroupsInTable-Enabled** gesetzt werden.

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

Gruppen in Diagramm in-Place-Editieren

Aktivieren Sie dieses Kontrollkästchen, wenn das direkte Editieren von Gruppenknotenlayern im Diagramm möglich sein soll. Diese Option kann auch über die Eigenschaft **VcGantt.InPlaceEditingOnGroupsInDiagram-Enabled** gesetzt werden.

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

Erweitertes Editieren in Tabelle zulassen

Aktivieren Sie dieses Kontrollkästchen, um erweiterte Möglichkeiten zum Bearbeiten des Tabelleninhalts und zum Navigieren nutzen zu können. Diese Option kann auch über die Eigenschaft **VcGantt.ExtendedEditingBehavior** gesetzt werden.

- **Vorgänge markieren und neue Feldinhalte eingeben:**

Beim Klick auf einen Vorgang werden nicht nur die entsprechende Tabellenzeile sowie der zugehörige Knoten im Diagrammbereich markiert, sondern Sie können auch direkt neue Feldinhalte eingeben.

Dabei sollten Sie folgendes beachten:

Wenn Sie in den **Diagrammbereich** klicken, ist immer das jeweils **erste** Feld der zugehörigen Tabellenzeile markiert und befindet sich damit im Editiermodus, unabhängig davon, welches Feld zuvor in der Tabelle markiert war. Beim Klick auf einen anderen Knoten verschiebt sich die Knoten-Markierung entsprechend und es wird wieder das erste Feld der entsprechenden Zeile markiert.

Durch Klick in den Tabellenbereich wird immer das Feld bearbeitet, auf das Sie geklickt haben.

Für beide Verfahren gilt:

Sie können die Vorgangs-Markierung mit den Cursor-Tasten nach oben/unten bzw. der ENTER-Taste verschieben und so die jeweils nächste/vorige Tabellenzeile markieren. Sollte vorher im Tabellenbereich eine andere als die erste Zelle markiert gewesen sein, so bleibt diese Auswahl auch für die neu markierte Zeile erhalten. Innerhalb einer markierten Tabellenzeile bewegen die Cursortasten nach rechts/links die Markierung jeweils ein Feld weiter/zurück.

Hinweis: Mit Hilfe der Taste ESC werden alle Markierungen aufgehoben.

- **Ändern eines Feldinhaltes**

Zum Ändern eines Feldinhaltes können Sie entweder noch einmal auf das Feld klicken oder die Taste F2 betätigen.

Bei einigen Feldtypen ist dies allerdings gar nicht mehr nötig. Bei Datumsfeldern lässt sich durch Klick auf den nebenstehenden Drop-Down-Pfeil das Datum-Steuerelement anzeigen. Für weitere Informationen zum

Gebrauch des Datum-Steuerelementes s. Kapitel 4.40 Dialogfeld "Stichtaglinien festlegen".

Bei numerischen Feldern finden sich Pfeil-Schaltflächen, mit deren Hilfe Sie den Wert jeweils erhöhen bzw. verringern können.

Hinweis: Mit Hilfe der Taste ESC verlassen Sie die editierten Felder, ohne die Änderungen zu übernehmen.

- **Einfügen von neuen Tabellenzeilen**

Mit Hilfe der Taste EINFG lässt sich oberhalb der jeweils markierten Zeile eine neue Tabellenzeile einfügen. Ist keine Zeile markiert, so wird hierdurch eine neue Zeile eingefügt.

Tabellenspalten optimierbar

Wenn Sie dieses Kontrollkästchen aktivieren, kann der Anwender durch einen Doppelklick auf die Trennlinie zwischen zwei Spalten erreichen, dass die Breite der linken Spalte automatisch an die Länge der darin enthaltenen Texte angepasst wird.

Diese Option kann auch über die Eigenschaft **VcGantt.TableColumn-WidthOptimizationAllowed** gesetzt 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. **PasteNodes-FromClipboard** 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 **VcGantt.CtrlCXVProcessing-Enabled** gesetzt werden.

Erzeugung neuer Boxen zulassen

Wenn Sie dieses Kontrollkästchen aktivieren, können zur Laufzeit neue Boxen angelegt werden. Zum Anlegen neuer Boxen zur Laufzeit muss im Kontextmenü (für das Diagramm) der **Modus: Box erzeugen** gewählt werden oder **InteractionMode** auf **VcCreateBox** gesetzt werden.

Diese Option kann auch über die Eigenschaft **VcGantt.BoxCreation-Allowed** 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 **VcGantt.MultipleBox-MarkingAllowed** gesetzt werden

Kontextmenü für Boxen anzeigen

Aktivieren Sie dieses Kontrollkästchen, um die Anzeige des Kontextmenüs für Boxen zur Laufzeit zu ermöglichen.

Diese Option kann auch über die Eigenschaft **VcGantt.ContextMenuFor-BoxesEnabled** gesetzt werden.

Zeitskalen-Dialog anzeigen

Aktivieren Sie dieses Kontrollkästchen, wenn das Dialogfeld **Zeitskala bearbeiten** bei einem Doppelklick auf die Zeitskala erscheinen soll.

Diese Option kann auch über die Eigenschaft **VcGantt.TimeScaleDialog-Enabled** gesetzt werden.

Zeitskala skalierbar

Nur wenn Sie diese Option wählen, lassen Sie zu, dass der Anwender die Auflösung der Zeitskala interaktiv verändern kann.

Diese Option kann auch über die Eigenschaft **VcGantt.TimeScaleRescaling-Allowed** gesetzt werden.

Numerische Skala skalierbar

Legen Sie hier fest, ob die numerische Skala des Histogramms skalierbar sein soll.

Diese Option kann auch über die Eigenschaft **VcGantt.NumericScale-RescalingAllowed** 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 **VcGantt.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.

Scroll-Ereignisse

Mithilfe dieses Kontrollkästchens können Sie die Scroll Events an- oder abschalten. Dies kann auch durch die Eigenschaft **VcGantt.ScrollEventsEnabled** geschehen.

Hinweis: Die Scroll-Events sind standardmäßig **nicht** aktiviert.

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 **VcGantt.TextEntrySupplyingEventEnabled** gesetzt werden.

Ereignis-Sicherheitsprüfung

Aktivieren Sie dieses Kontrollkästchen, wenn eine Sicherheitsprüfung der Ereignisses **VcNodeModifying** vorgenommen werden soll. Dann werden in diesen Ereignissen die Zugriffe auf Set-Calls des gleichen Objekttyps unterdrückt.

Diese Option kann auch über die Eigenschaft **VcGantt.EventsSecurityCheck** gesetzt werden.

Automatische Reduzierung der Zeilenhöhen

Diese Option beeinflusst das Verfahren zur Berechnung der Zeilenhöhe im Diagramm. Ist sie nicht ausgewählt, werden die vertikalen Offsets der Layer ausgehend von einer gedachten Nulllinie in der vertikalen Mitte einer Knotenzeile wirksam. Damit diese Nulllinie stets in der Mitte einer Zeile bleibt, kann so entweder der obere oder der untere Rand einer Zeile sehr groß erscheinen, wobei aber Layer mit vertikalem Offset von 0 immer vertikal zentriert bleiben.

Ist das Kontrollkästchen aktiviert, wird zwar weiterhin die gedachte Nulllinie verwendet, diese befindet sich jedoch nicht mehr zwingend in der Zeilenmitte, sondern wird so positioniert, dass für die Zeilenhöhe das kleinstmögliche Maß verwendet werden kann. Dadurch kann es vorkommen dass Layer mit einem vertikalen Offset von 0 nicht mehr auf gleicher Höhe liegen wie der vertikal zentrierte Text der zugehörigen Tabellenzeile.

Diese Option kann auch über die Eigenschaft **VcGantt.RowHeight-ReductionEnabled** gesetzt werden.

Anti-Aliasing bei Schriften

Mit dieser Option können Sie das Glätten von Schriftzeichen erlauben oder unterdrücken. Wenn dies bei bestimmten Schriftarten, insbesondere bei nichtlateinischen Zeichen, zu verminderter Lesefähigkeit führt, sollte die Eigenschaft abgeschaltet 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 Tabellenfeld 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 Option kann auch über die Eigenschaft **VcGantt.FontAntiAliasing-Enabled** gesetzt werden

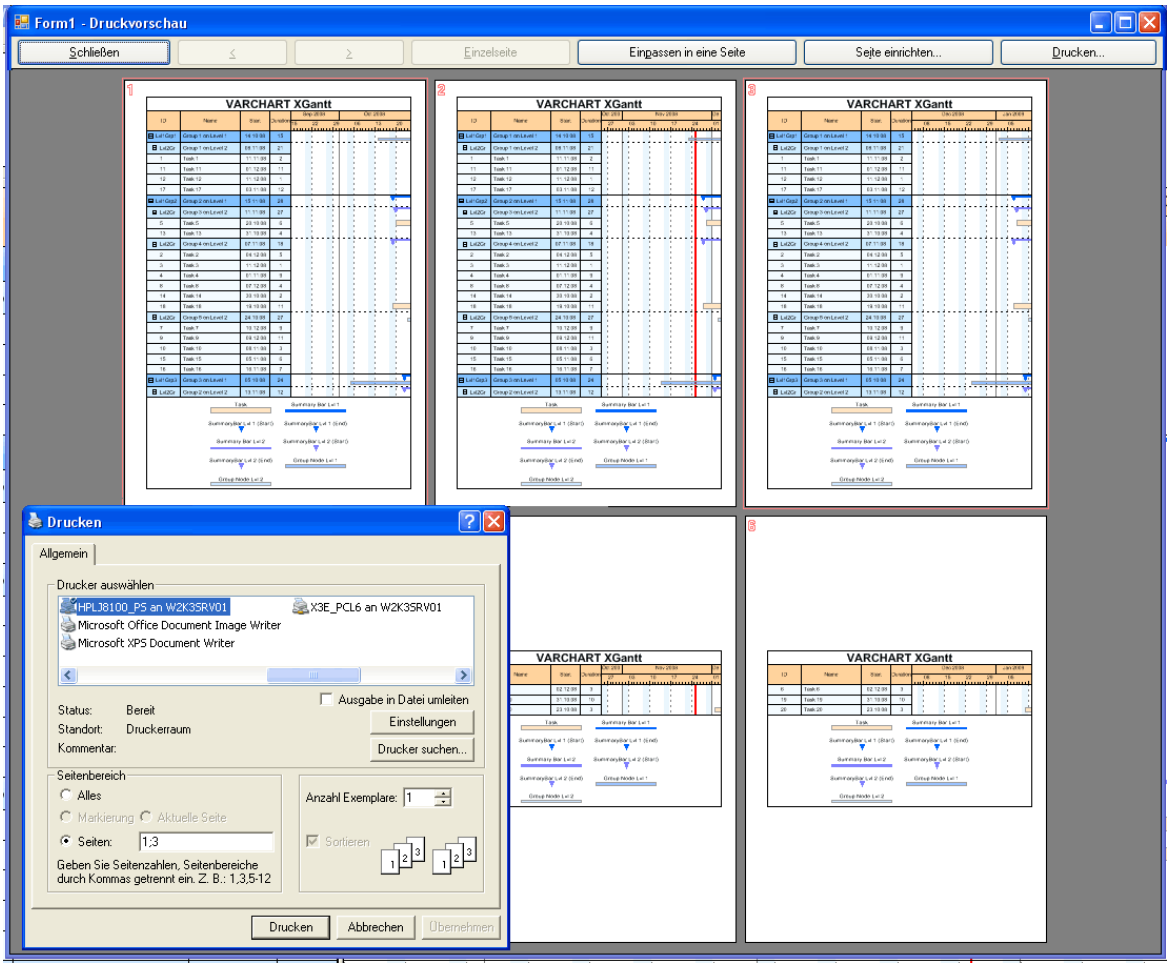
PrintDlgEx Dialog verwenden

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 **vcLRTOrthogonalDistinguishable** als Viertelkreise statt als gerade Linien dargestellt werden. Diese Option kann auch über die VcGantt-Eigenschaft **RoundedLinkSlantsEnabled** gesetzt werden.

Verbindungstyp berücksichtigen beim Knotenverschieben

Wählen Sie diese Option, damit beim Ziehen von Knoten und eingeschalteten Verbindungen die Phantomlinien, die die Verbindungen zu verschobenen Knoten und deren Anschlussknoten dargestellt werden, typgerecht dargestellt, d.h. sie beginnen/enden nicht in der Mitte der Knotenphantome, sondern auf deren linken oder rechten Seite.

Diese Option kann auch über die Eigenschaft **VcGantt.ConsiderLink-RelationTypesOnNodeDragging** gesetzt werden.

Optimierung von Gruppen bei Interaktionen

Ist diese Eigenschaft eingeschaltet, dann werden bei Interaktionen wie Aufziehen von Knoten, Knotenverschieben horizontal oder vertikal und Ändern des Start- oder Endtermins eines Layers die Knoten der gesamten Zielgruppe automatisch erneut optimiert, wenn sie bisher bereits optimiert dargestellt wird. Wird diese Eigenschaft abgeschaltet, dann wird bei denselben Interaktionen der Knoten unter der Cursorposition platziert, wenn dies nicht zu einer Überlappung führt. Bei einer möglichen Überlappung wird der Knoten in die nächste Zeile zu anderen Knoten platziert, wenn dies dort keine Überlappung verursacht. Wenn doch, dann bekommt der Knoten eine eigene Zeile unter der, an der der Mauscursor sich befindet.

Diese Option kann auch über die Eigenschaft **VcGantt.GroupOptimization-OnInteractionsEnabled** 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 **VcGantt.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 **VcGantt.PanningModeAllowed** gesetzt werden.

Selektion mit Gummmirechteck erlaubt

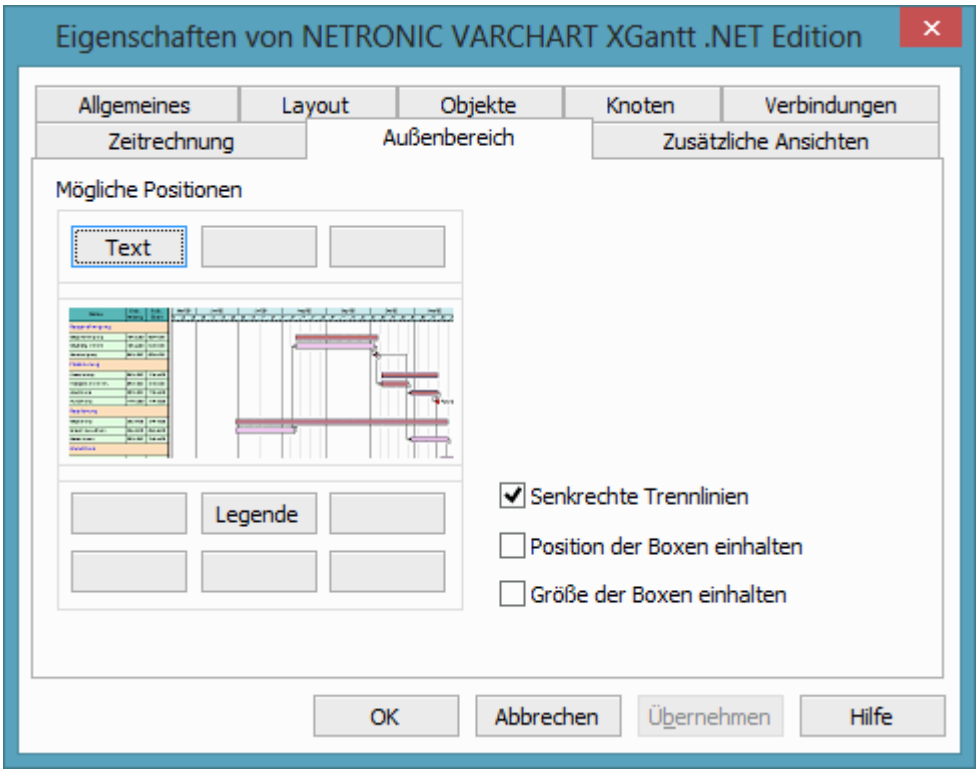
Wählen Sie diese Option, lässt sich die Mehrfachselektion von Knoten mithilfe eines Gummirechtecks aktivieren bzw. unterbinden.

Die Option kann auch über die Eigenschaft **VcGantt.AllowSelectionVia-RubberRect** 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.

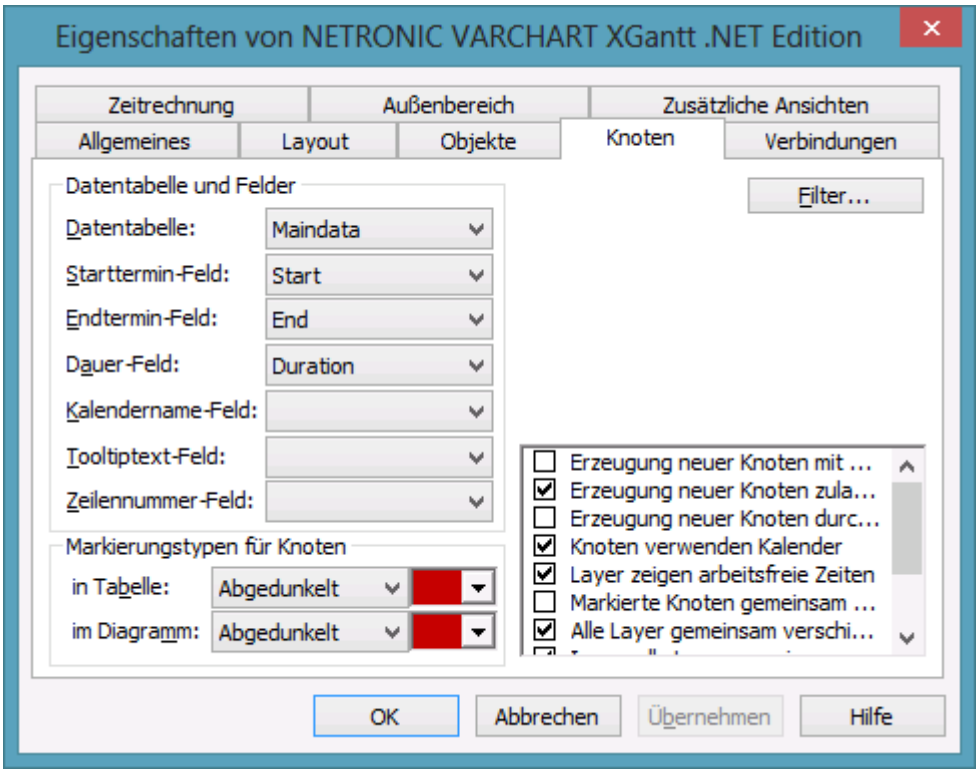
Position der Boxen einhalten

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 "Knoten"



Datentabelle

Wählen Sie hier die Datentabelle aus, die für die Darstellung der Knoten herangezogen werden soll.

Diese Option kann auch über die Eigenschaft **VcGantt.NodesDataTable-Name** festgelegt werden.

Starttermin-Feld

Vereinbaren Sie hier das Datenfeld, in das der Anfangstermin eines interaktiv angelegten Vorgangs geschrieben werden soll. Es werden nur Datumsfelder angeboten.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeStartDateData-FieldIndex** festgelegt werden.

Endtermin-Feld

Vereinbaren Sie hier das Datenfeld, in das der Endtermin eines interaktiv angelegten Vorgangs geschrieben werden soll. Es werden nur Datumsfelder angeboten.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeEndDateDataFieldIndex** festgelegt werden.

Dauer-Feld

Vereinbaren Sie hier das Datenfeld, in das die Dauer eines interaktiv angelegten Vorgangs geschrieben werden soll. Es werden nur numerische Felder angeboten

Diese Option kann auch über die Eigenschaft **VcGantt.NodeDurationDataFieldIndex** festgelegt werden.

Kalendername-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 die Option **Knoten verwenden Kalender** aktiviert sein. Außerdem müssen die Kalender vor dem Laden der Vorgänge erzeugt worden sein.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeCalendarNameDataFieldIndex** festgelegt werden..

Zeilennummer-Feld

Wählen Sie hier das Datenfeld aus, in das die Zeilennummer jedes Vorgangs geschrieben werden soll. Damit Änderungen wirksam werden, muss eine Aktualisierung über die Methode **VcGantt.UpdateRowNumberFields** vorgenommen werden.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeRowNumberDataFieldIndex** 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 VcGantt auf True setzen und im **VcToolTipTextSupplying**-Ereignis programmieren, welche Text angezeigt werden soll.

Markierungstypen für Knoten in der Tabelle

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

- Ohne
- Einrahmen innen
- Invertieren
- Abgedunkelt (um 25 %)
- Aufgehellt (um 25%)
- Pickmarks innen

In dem rechten Feld können Sie eine Farbe für die Markierung setzen.

Markierungstypen für Knoten im Diagramm

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

- Ohne
- Einrahmen
- Einrahmen innen
- Invertieren
- Abgedunkelt (um 25 %)
- Aufgehellt (um 25%)
- Pickmarks
- Pickmarks innen

In dem rechten Feld können Sie eine Farbe für die Markierung setzen.

Filter

Über diese Schaltfläche öffnen Sie das Dialogfeld **Filter verwalten**. Der für die Vorselektion der Knoten wirksame Filter kann ausschließlich zur Laufzeit

über die Eigenschaft **ActiveNodeFilter** des Objekts **VcGantt** eingestellt 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.

Nachdem ein Knoten erzeugt wurde, kann das Dialogfeld **Vorgänge bearbeiten** - auch wenn die Option deaktiviert ist - jederzeit durch einen Doppelklick auf den Knoten oder über sein Kontextmenü geöffnet werden.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeCreationWith-Dialog** festgelegt werden.

Erzeugung neuer Knoten zulassen

Nur wenn Sie diese Option wählen, lassen Sie zu, dass der Anwender in einem geöffneten Projekt interaktiv neue Knoten erzeugen kann. Neue Knoten lassen sich entweder im Modus **Knoten erzeugen** interaktiv erzeugen oder durch Doppelklick auf die gewünschte Position, wenn die Option **Erzeugung neuer Knoten durch Doppelklick** angeschaltet ist.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeCreation-Allowed** festgelegt werden.

Erzeugung neuer Knoten durch Doppelklick

Hier können Sie festlegen, dass neue Knoten per Doppelklick erzeugt werden können. Ein per Doppelklick erzeugter neuer Knoten wird auf der aktuellen Cursor-Position eingefügt und hat die Dauer eine Zeiteinheit.

Diese Option kann auch über die Eigenschaft **VcGantt.NodeCreationVia-DoubleClick** festgelegt werden.

Knoten verwenden Kalender

Wenn den Knoten Kalender zugewiesen werden sollen, aktivieren Sie dieses Kontrollkästchen. Die Kalenderzuweisung wirkt sich zum einen beim Verschieben von Vorgängen aus: Anfang und Ende der Vorgänge werden nicht auf arbeitsfreie Tage gelegt. Zum anderen werden beim Berechnen der Dauer von Vorgängen die arbeitsfreien Zeiten berücksichtigt. Auch bei der Zeitrechnung wirkt sich die Kalenderzuweisung aus. Derzeit ist standardmäßig ein Fünf-Tage-Kalender definiert ("BaseCalendar").

Diese Option kann auch über die Eigenschaft **VcGantt.NodesUseCalendars** festgelegt werden.

Ist kein individueller Kalender pro Knoten zugewiesen worden, dann wird der in der CalendarCollection als aktiv definierte Kalender verwendet.

Layer zeigen arbeitsfreie Zeiten

Aktivieren Sie dieses Kontrollkästchen, wenn arbeitsfreie Zeiten hervorgehoben werden sollen. Sie werden dann dann in der Form angezeigt, die im **Layer bearbeiten**-Dialog ausgewählt wurde (gilt nur für Rechtecklayer).

Diese Option kann auch über die Eigenschaft **VcGantt.LayersWithNonWorkInterval** festgelegt werden.

Markierte Knoten gemeinsam verschieben

Wählen Sie diese Option, damit mehrere markierte Knoten gemeinsam verschoben werden können. Wenn die Option nicht ausgewählt ist, können nur einzelne Knoten oder Layer (ja nach Einstellung der Option **Alle Layer gemeinsam verschieben** mit der Maus verschoben werden, selbst wenn mehrere Knoten markiert wurden.

Diese Option kann auch über die Eigenschaft **VcGantt.SelectedNodesMovingTogether** festgelegt werden.

Alle Layer gemeinsam verschieben

Aktivieren Sie dieses Kontrollkästchen, damit alle Layer eines markierten Knotens gemeinsam verschoben werden können. Ein Knoten kann durch einen Klick mit der linken Maustaste auf einen seiner Layer markiert werden.

Wenn dieses Kontrollkästchen nicht aktiviert ist, können die Layer eines markierten Knotens nur einzeln verschoben werden. Sollen trotzdem alle Layer eines markierten Knotens gemeinsam verschoben werden, muss beim Verschieben die Umschalt-Taste gedrückt werden. (Voraussetzung: Die Option **Layer bei gedrückter Shift-Taste als Knoten verschieben erlaubt** wurde ausgewählt).

Diese Option kann auch über die Eigenschaft **VcGantt.AllLayers**

MovingTogether festgelegt werden.

Immer alle Layer gemeinsam verschieben

Wenn dieses Kontrollkästchen aktiviert ist, können alle Layer eines Knotens gemeinsam verschoben werden ohne zuvor markiert werden zu müssen.

Diese Option kann auch über die Eigenschaft **VcGantt.AllLayersMoving-TogetherAlways** festgelegt werden.

Layer bei gedrückter Shift-Taste als Knoten verschieben erlaubt

Wenn Sie diese Option auswählen, können alle Layer eines markierten Knotens gemeinsam verschoben werden, indem während des Verschiebens die Umschalt-Taste gedrückt wird.

Diese Option kann auch zur Laufzeit über die Eigenschaft **VcGantt.Moving-LayersAsNodeWithShiftKeyAllowed** gesetzt werden.

Einrastziele bei Interaktionen

Wenn Sie diese Option auswählen, kann beim Verschieben von Knoten/Layern die Einrastfunktionalität genutzt werden, d.h. es wird festgelegt, ob ein Knoten/Layer beim Verschieben an den jeweils definierten Einrastzielen der Objekte einrasten soll oder nicht.

Diese Option kann auch zur Laufzeit über die Eigenschaft **VcGantt.Use-SnapTargetsInInteractions** gesetzt werden.

Einrastlinien anzeigen

Wenn Sie diese Option auswählen, können beim Verschieben von Knoten/Layern mit aktivierter Einrastfunktionalität entprechende Linien gezeigt werden, damit der/die definierten Einrastziele besser zu erkennen sind.

Diese Option kann auch zur Laufzeit über die Eigenschaft **VcGantt.Show-SnapLines** gesetzt werden.

Einrastmarkierungen anzeigen

Wenn Sie diese Option auswählen, können beim Verschieben von Knoten/Layern bei aktivierter Einrastfunktionalität am Zielknoten/-layer entprechende Markierungen angezeigt werden, damit der/die definierten Einrastziele besser zu erkennen sind.

Diese Option kann auch zur Laufzeit über die Eigenschaft **VcGantt.Show-SnapMarkings** gesetzt werden.

Verändern der Knotenreihenfolge über das Diagramm erlaubt

Wenn Sie diese Option auswählen, können Sie im Diagrammbereich Knoten mit der Maus von einer Zeile in eine andere verschieben und damit ihre Reihenfolge oder Gruppenzugehörigkeit ändern. Wenn ein Knoten aus mehreren Layern besteht, muss beim vertikalen Verschieben zusätzlich die Umschalt-Taste gedrückt werden.

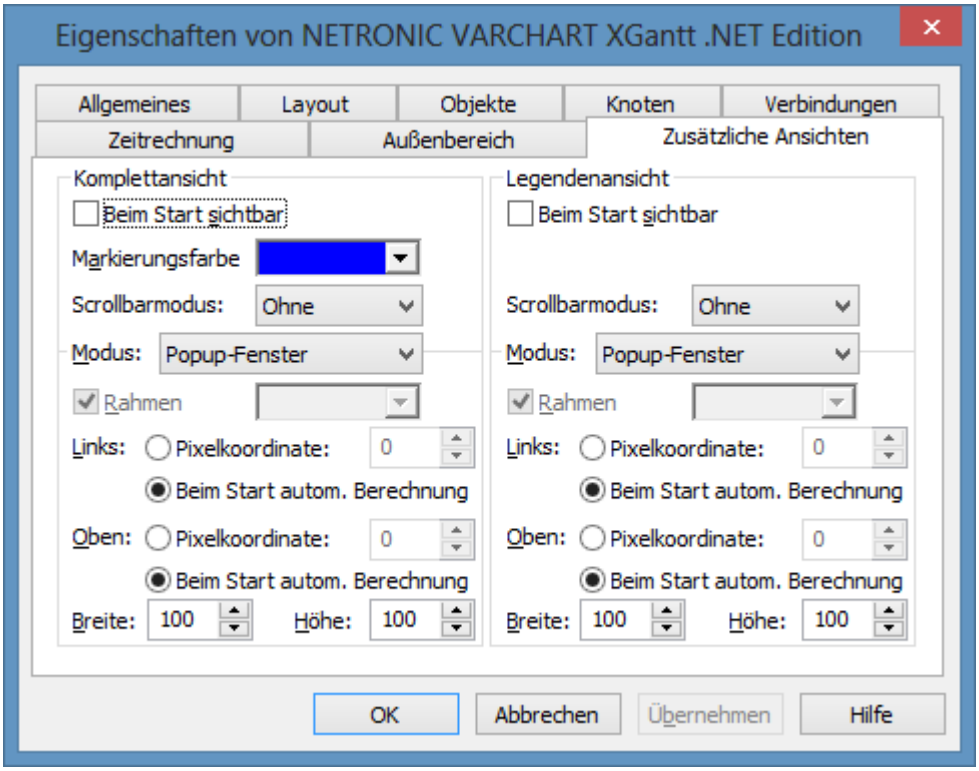
Diese Option kann auch zur Laufzeit über die Eigenschaft **VcGantt.Vertical-NodeMovementAllowed** gesetzt werden.

Verändern der Knotenreihenfolge über die Tabelle erlaubt

Wenn Sie diese Option auswählen, können Sie im Tabellenbereich Knoten mit der Maus von einer Zeile in eine andere verschieben und damit ihre Reihenfolge oder Gruppenzugehörigkeit ändern. Wenn ein Knoten aus mehreren Layern besteht, muss beim vertikalen Verschieben zusätzlich die Umschalt-Taste gedrückt werden.

Diese Option kann auch zur Laufzeit über die Eigenschaft **VcGantt.Vertical-NodeMovementViaTableAllowed** gesetzt werden.

4.5 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 und/oder das Histogramm angezeigt wird. Der eine Rahmen darin zeigt an, welcher Ausschnitt des Gantt-Graphen gerade im Hauptfenster dargestellt wird, der andere, welcher Ausschnitt des Histogramms.

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 oder für das Histogramm **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 OPpionen können auch über die Aufrufe **VcWorldView.Border** und **VcWorldView.Border.Color** bzw. **VcLegendView.Border** und **VcLegend-View.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.

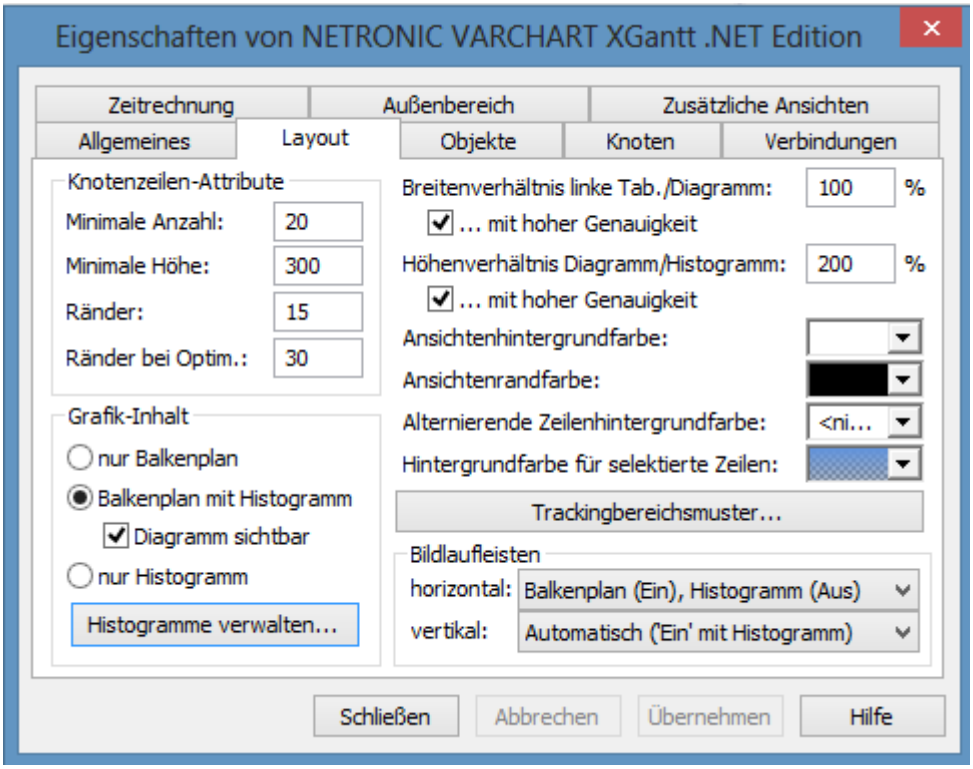
Diese Eigenschaft kann auch über die Aufrufe **VcWorldView.Width** bzw. **VcLegendView.Width** der Programmierschnittstelle gesetzt werden.

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.

4.6 Eigenschaftenseite "Layout"



Auf dieser Eigenschaftenseite können Sie das Layout Ihrer Grafik festlegen.

Minimale Anzahl

Legen Sie hier fest, wie viele Knotenzeilen beim Start des Programms im Diagrammbereich dargestellt werden sollen.

Diese Option kann auch über die Eigenschaft **VcGantt.InitialRowCount** festgelegt werden.

Minimale Höhe

Legen Sie hier die minimale Höhe der Knotenzeilen in 1/100 mm fest. Diese Eigenschaft kann auch zur Laufzeit über die Eigenschaft **MinimumRow-Height** des Objektes **VcGantt** gesetzt werden. Die einstellbaren Werte liegen zwischen 2 und 1000.

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.

Diese Option kann auch über die Eigenschaft **VcGantt.MinimumRow-Height** festgelegt werden.

Ränder

Legen Sie hier den minimalen vertikalen Abstand zwischen dem Knoten und der oberen oder unteren Knotenzeilengrenze in 1/100 mm fest.

Diese Option kann auch über die Eigenschaft **VcGantt.RowMargins** festgelegt werden.

Ränder bei optimierter Darstellung

Mit dieser Eigenschaft können Sie den vertikalen Abstand von Subzeilen in 1/100 mm festlegen. Diese Subzeilen existieren nur, wenn Gruppen optimiert dargestellt werden und daraus Knoten in einer jeweiligen Gruppenzeile in mehrere Subzeilen verteilt werden, um nicht zu überlappen.

Diese Option kann auch über die Eigenschaft **VcGantt.SubRowMargins** festgelegt werden.

Grafik-Inhalt

Wählen Sie aus, was in der Grafik dargestellt werden soll:

- nur der Balkenplan
- Balkenplan und Histogramm (hier kann die Sichtbarkeit des Balkenplans über eine Checkbox an- oder abgeschaltet werden)
- nur das Histogramm.

Histogramme verwalten

Der Dialog **Histogramme verwalten** erscheint.

Breitenverhältnis linkeTabelle/Diagramm

Legen Sie hier das prozentuale Verhältnis von der Breite der Tabelle zur Breite des Gesamtdiagramms (Tabelle plus Diagrammbereich) beim Start fest. Mit dem Eintrag "-1" legen Sie fest, dass die Tabelle beim Start komplett gezeigt wird.

Diese Option kann auch über die Eigenschaft **VcGantt.LeftTableDiagramWidthRatio** festgelegt werden.

…mit hoher Genauigkeit

Wenn diese Eigenschaft aktiviert ist, werden zur Ermittlung der Größenverhältnisse zwischen Tabelle und Diagramm die genaueren

Methoden **LeftTableDiagramWidthRatioEx** und **RightTableDiagram-WidthRatioEx** bzw. das Ereignis **VcTableWidthChangingEx** verwendet, die jeweils einen Wert vom Datentyp "Double" zurückgeben.

Ist die Eigenschaft nicht aktiviert, werden die Methoden **LeftTable-DiagramWidthRatio** und **RightTableDiagramWidthRatio** bzw. das Ereignis **VcTableWidthChanging** verwendet.

Diese Option kann auch über die Eigenschaft **VcGantt.UseHigherTable-DiagramWidthRatioPrecision** gesetzt werden.

Höhenverhältnis Diagramm/Histogramm

Legen Sie hier das prozentuale Verhältnis von der Höhe des Diagramm-bereichs (also des Diagramms ohne Histogramm) und der Höhe des Histogramms beim Start fest. Wenn Sie hier "-1" wählen, wird das Histogramm beim Start komplett gezeigt.

Diese Option kann auch über die Eigenschaft **VcGantt.DiagramHistogram-HeightRatio** festgelegt werden.

…mit hoher Genauigkeit

Wenn diese Option ausgewählt ist, werden zur Ermittlung des Höhenverhält-nisses zwischen Diagramm und Histogramm die genauere Methode **DiagramHistogramHeightRatioEx** bzw. das Ereignis **VcHistogram-HeightChangingEx** verwendet, die jeweils einen Wert vom Datentyp "Double" zurückgeben.

Beim Standardwert "False" wird die Methode **DiagramHistogramHeight-Ratio** bzw. das Ereignis **VcHistogramHeight** verwendet.

Diese Option kann auch über die Eigenschaft **VcGantt.UseHigherDiagram-HistogramHeightRatioPrecision** gesetzt werden.

Ansichtenhintergrundfarbe

Hier können Sie eine Farbe für den Diagrammhintergrund festlegen. Zusam-men mit der **Alternierenden Zeilenhintergrundfarbe** können Sie ein zeilenweise alternierendes Farbmuster einrichten.

Diese Option kann auch über die Eigenschaft **VcGantt.DiagramBack-groundColor** oder **VcGantt.ViewComponentsBackgroundColor** festgelegt werden.

Ansichtenrandfarbe

Hier können Sie eine Rahmenfarbe für alle Fenster gemeinsam festlegen.

Diese Option kann auch über die Eigenschaft **VcGantt.ViewComponents-BorderColor** festgelegt werden.

Alternierende Zeilenhintergrundfarbe

Hier können Sie eine zweite Hintergrundfarbe für das Diagramm setzen oder erfragen, die mit der **Diagrammhintergrundfarbe** ein zeilenweise alternierendes Farbmuster bildet.

Diese Option kann auch über die Eigenschaft **VcGantt.DiagramAlternatingRowBackgroundColor** festgelegt werden.

Hintergrundfarbe für selektierte Zeilen

Hier können Sie eine Hintergrundfarbe für die jeweils aktuelle Zeile des Diagramms, d.h. die Zeile, in der Sie gerade arbeiten, auswählen.

Diese Option kann auch über die Eigenschaft **VcGantt.SelectedRow-BackgroundColor** festgelegt werden.

Trackingbereichsmuster

Diese Schaltfläche öffnet den Dialog **Musterattribute bearbeiten**, in dem das Aussehen des freien Bereichs, der bei Interaktionen im LiveUpdate hin und wieder für kurze Zeit am oberen oder unteren Rand entsteht (Standardaussehen: rote Punkte auf gelbem Hintergrund) festgelegt werden kann.

Das Muster kann auch über die entsprechenden Eigenschaften **VcGantt.-TrackingSpaceBackgroundColor**, **VcGantt.TrackingSpacePattern** und **VcGantt.TrackingSpacePatternColor** festgelegt werden.

Bildlaufleisten

Hier können Sie festlegen, wo sich die Bildlaufleisten des Histogramms befinden sollen. Für den horizontalen Bereich können Sie zwischen folgenden Optionen wählen:

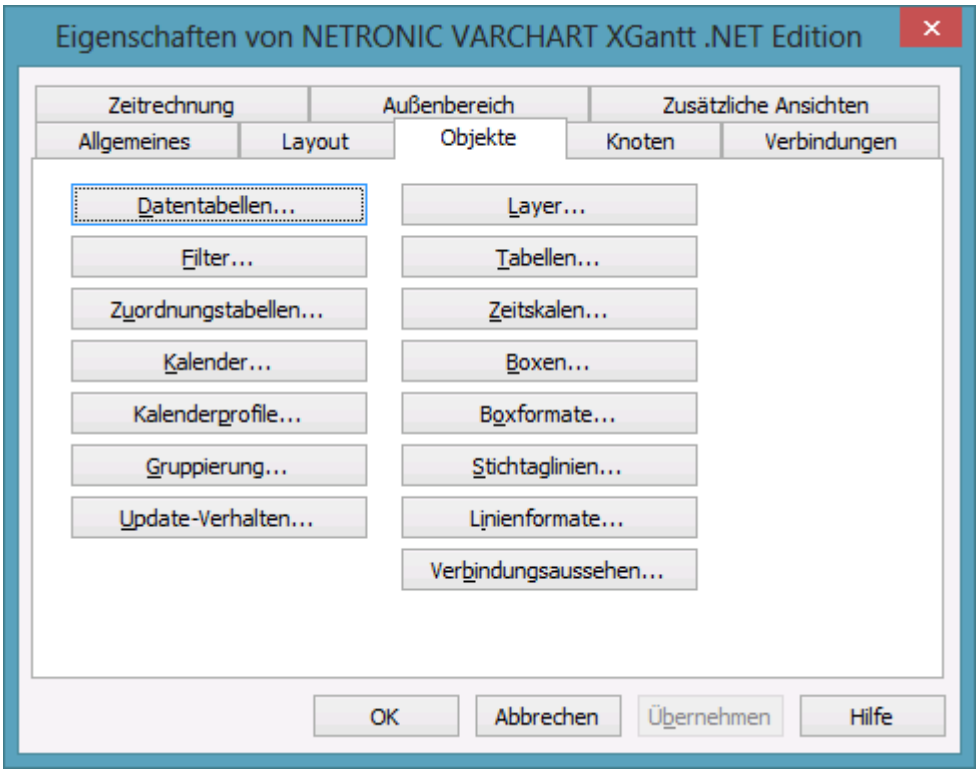
1. **Balkenplan (ein), Histogramm (aus)** - die horizontale Bildlaufleiste befindet sich zwischen Histogramm und Balkenplan
2. **Balkenplan (aus), Histogramm (ein)** - die horizontale Bildlaufleiste befindet sich unterhalb des Histogramms

3. **Keine** - es gibt keine horizontale Bildlaufleiste.

Für den vertikalen Bereich können Sie zwischen folgenden Optionen wählen:

1. **Automatisch (aber 'ein' mit Histogramm)** - eine vertikale Bildlaufleiste wird im Bedarfsfall automatisch rechts am Gantt-Graphen zugeschaltet; eine weitere ist stets am Histogramm vorhanden
2. **ein** - je eine vertikale Bildlaufleiste ist rechts am Gantt-Graphen und am Histogramm zugeschaltet
3. **aus** - die vertikalen Bildlaufleisten sind ausgeschaltet.

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**.

Kalenderprofile

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

Gruppierung

Über diese Schaltfläche öffnen Sie das Dialogfeld **Gruppierung**.

Update-Verhalten

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

Layer

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

Tabellen

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

Zeitskalen

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

Boxen

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

Boxformate

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

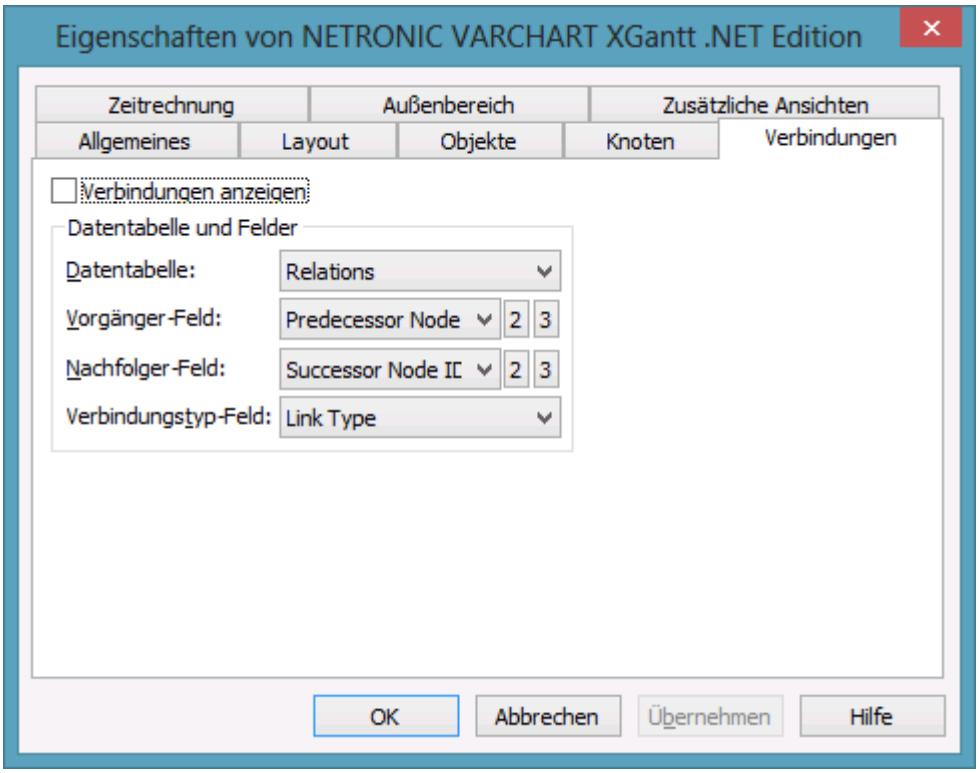
Stichtaglinien

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

Linienformate

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

4.8 Eigenschaftenseite "Verbindungen"



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

Verbindungen anzeigen

Wenn Sie dieses Kontrollkästchen aktivieren, werden sowohl Verbindungen als auch Phantomlinien für Links bei Interaktionen angezeigt.

Diese Option kann auch über die API- Eigenschaft **VcLinkAppearance-.Visible** festgelegt werden, betrifft jedoch nur die Verbindungen, nicht die Phantomlinien.

Datentabelle

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

Vorgänger-Feld

Wählen Sie hier das Datenfeld bzw. die Datenfelder aus der zuvor eingestellten Datentabelle aus, in dem die Identifizierung des Vorgängerknotens der Verbindung enthalten ist/enthalten sind.

Diese Option kann auch über die Eigenschaft **VcGantt.LinksPredecessor-DataFieldIndex** festgelegt werden.

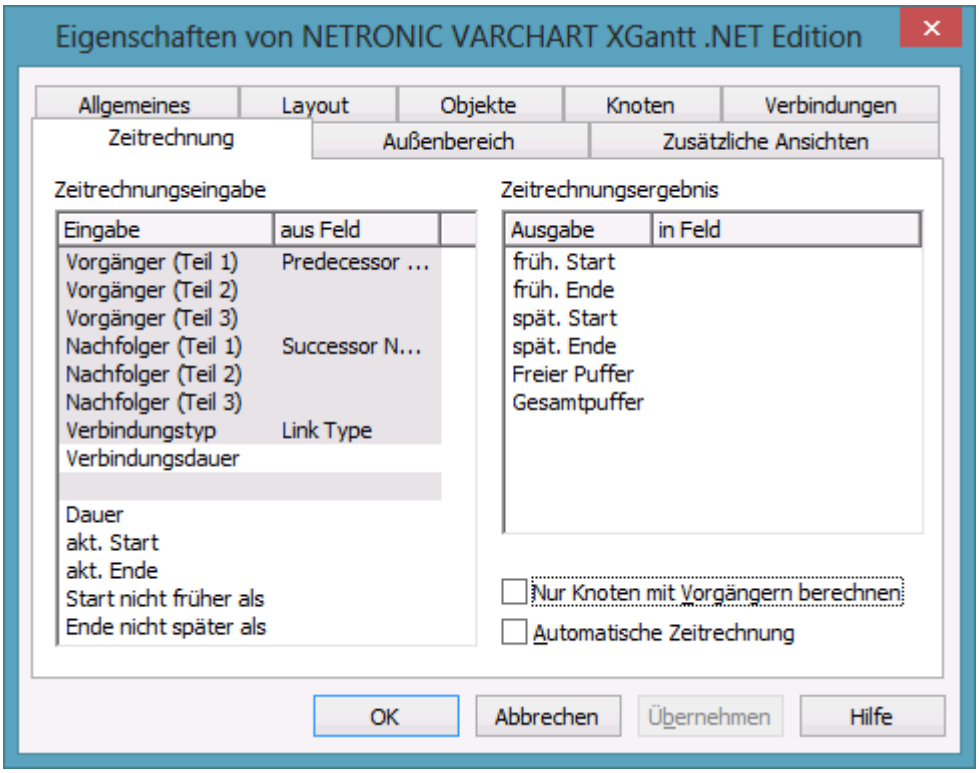
Nachfolger-Feld

Wählen Sie hier das Datenfeld bzw. die Datenfelder aus der zuvor eingestellten Datentabelle aus, in dem die Identifizierung des Nachfolgerknotens der Verbindung enthalten ist/enthalten sind. Diese Option kann auch über die Eigenschaft **VcGantt.LinksSuccessorDataFieldIndex** festgelegt werden.

Verbindungstyp-Feld

Wählen Sie hier das Datenfeld, das den Verbindungstyp enthält. Diese Option kann auch über die Eigenschaft **VcGantt.LinkTypeDataFieldIndex** festgelegt werden.

4.9 Eigenschaftenseite "Zeitrechnung"



Mit Hilfe dieser Eigenschaftenseite können Sie die Zeitrechnung des VARCHART XGantt an Ihre Schnittstelle anpassen, indem Sie festlegen, welche Datenfelder für die Eingabe (**Zeitrechnungseingabe**) und die Ausgabe (**Zeitrechnungsergebnis**) der Zeitberechnung (des Schedulers) verwendet werden sollen. (Vgl. "Wichtige Begriffe: Zeitrechnung".)

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.

Nur Knoten mit Vorgängern berechnen

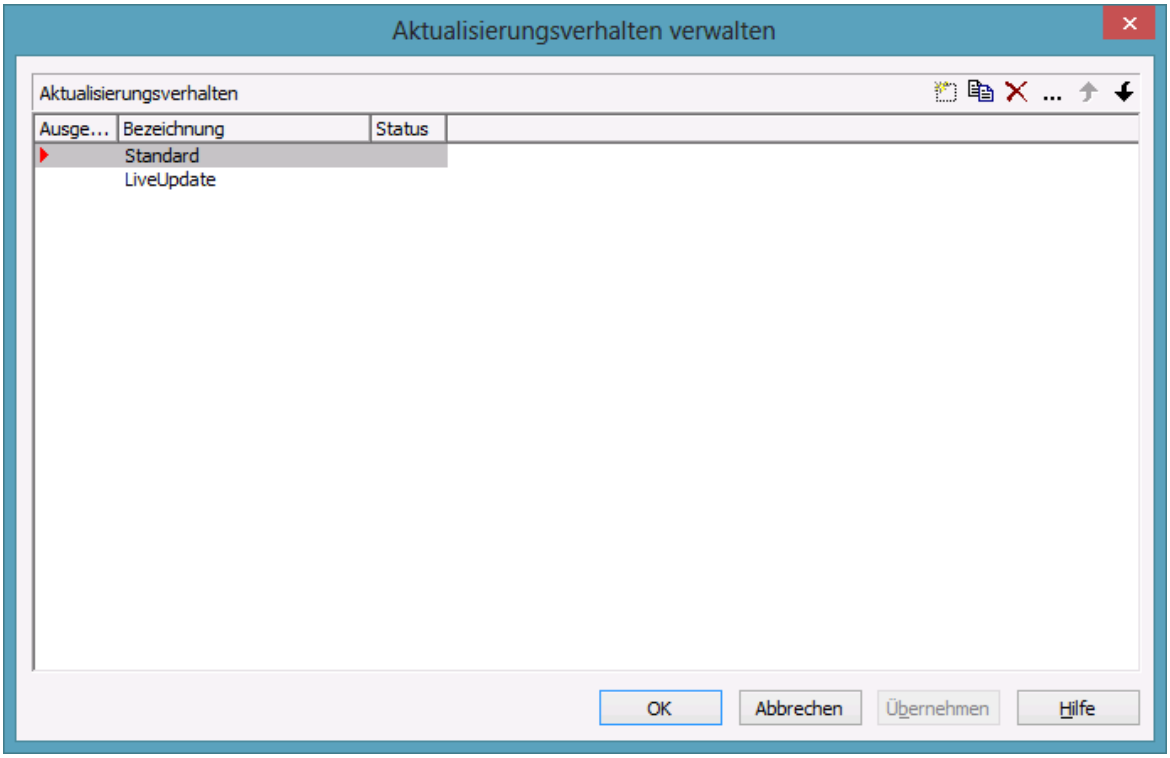
Wenn dieses Kontrollkästchen aktiviert ist, werden beim Zeitrechnen nur die Vorgänge, die Vorgänger besitzen, berechnet. Ein gesetzter Projektstart beim Aufruf der Zeitrechnung wird ignoriert.

Wenn dieses Kontrollkästchen nicht aktiviert ist, werden bei der Zeitrechnung alle Vorgänge berechnet.

Automatische Zeitrechnung

Wenn diese Option aktiviert ist, werden beim Anlegen oder Löschen einer Verbindung oder beim Ändern der Dauer eines Vorgangs automatisch die abhängigen Termine neu berechnet.

4.10 Dialogfeld "Aktualisierungsverhalten verwalten"



Sie gelangen in diesen Dialog über die Eigenschaftenseite **Objekte**. Hier können Sie eigene Aktualisierungsverhalten anlegen, kopieren, löschen und verschieben. Über die Schaltfläche ... öffnen Sie den Dialog **Aktualisierungsverhalten bearbeiten**.

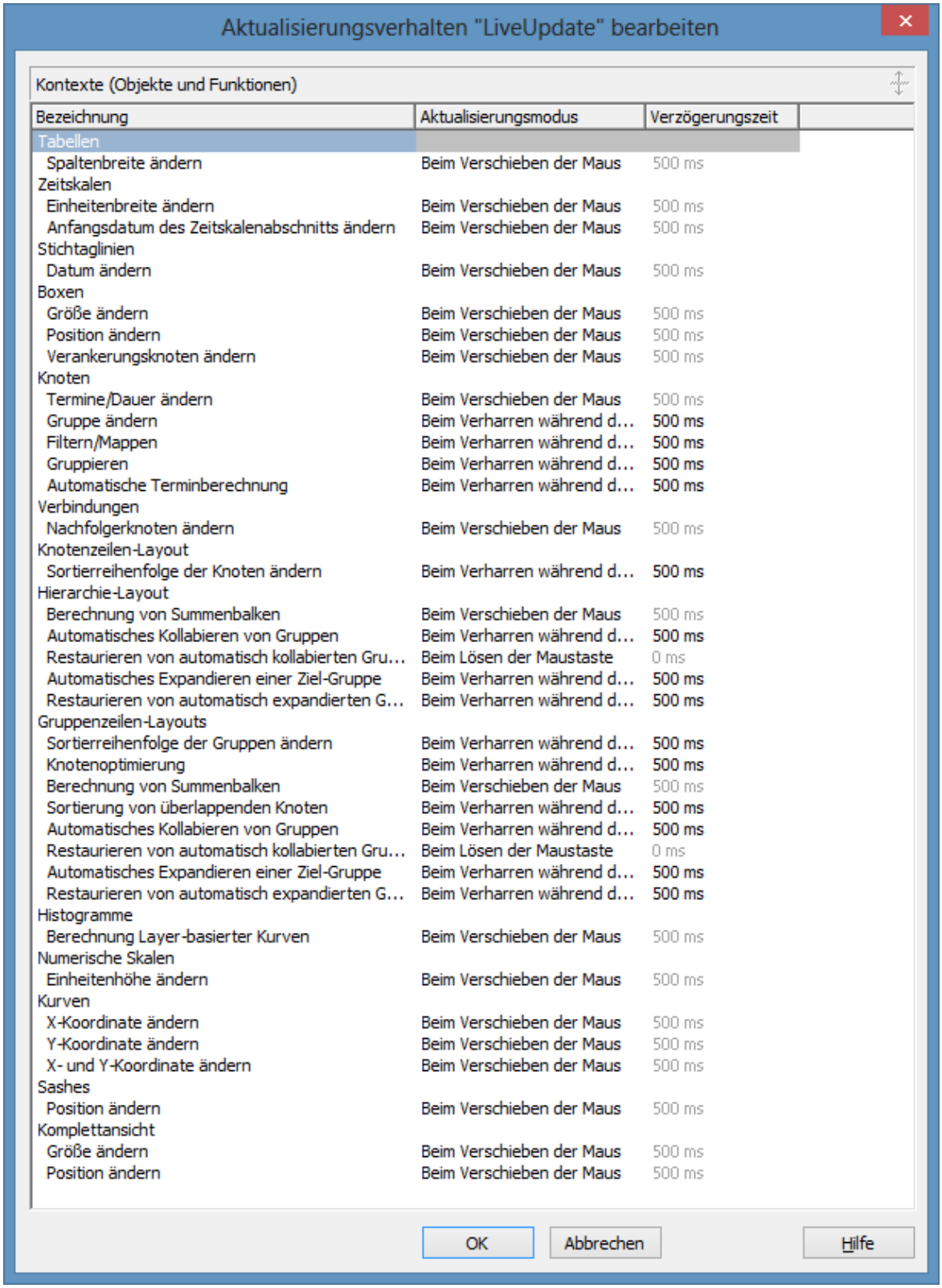
Aktualisierungsverhalten

- **Ausgewählt:** Ein rotes Dreieck kennzeichnet das gerade ausgewählte Aktualisierungsverhalten
- **Bezeichnung:** In dieser Spalte stehen die Namen aller vorhandenen Aktualisierungsverhalten. Die Namen er individuell erstellten Verhalten sind editierbar.
- **Status:** In der Spalte **Status** wird jedes Aktualisierungsverhalten gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Aktualisierungsverhalten hinzufügen / kopieren / löschen / nach oben / unten

 Mithilfe dieser Schaltflächen können Sie Aktualisierungsverhalten hinzufügen, kopieren, löschen und in der Liste nach oben oder unten verschieben.

4.11 Dialogfeld "Aktualisierungsverhalten bearbeiten"



Diesen Dialog erreichen Sie über den Dialog **Aktualisierungsverhalten verwalten**. Hier können Sie für individuell angelegte Aktualisierungsverhalten den Update-Modus und/oder die Verzögerungszeit ändern.

Bezeichnung

n dieser Spalte finden Sie die Namen der Objekte und zugehörigen Funktionen, die vom Live Update betroffen sind. Diese Bezeichnungen sind **nicht** editierbar.

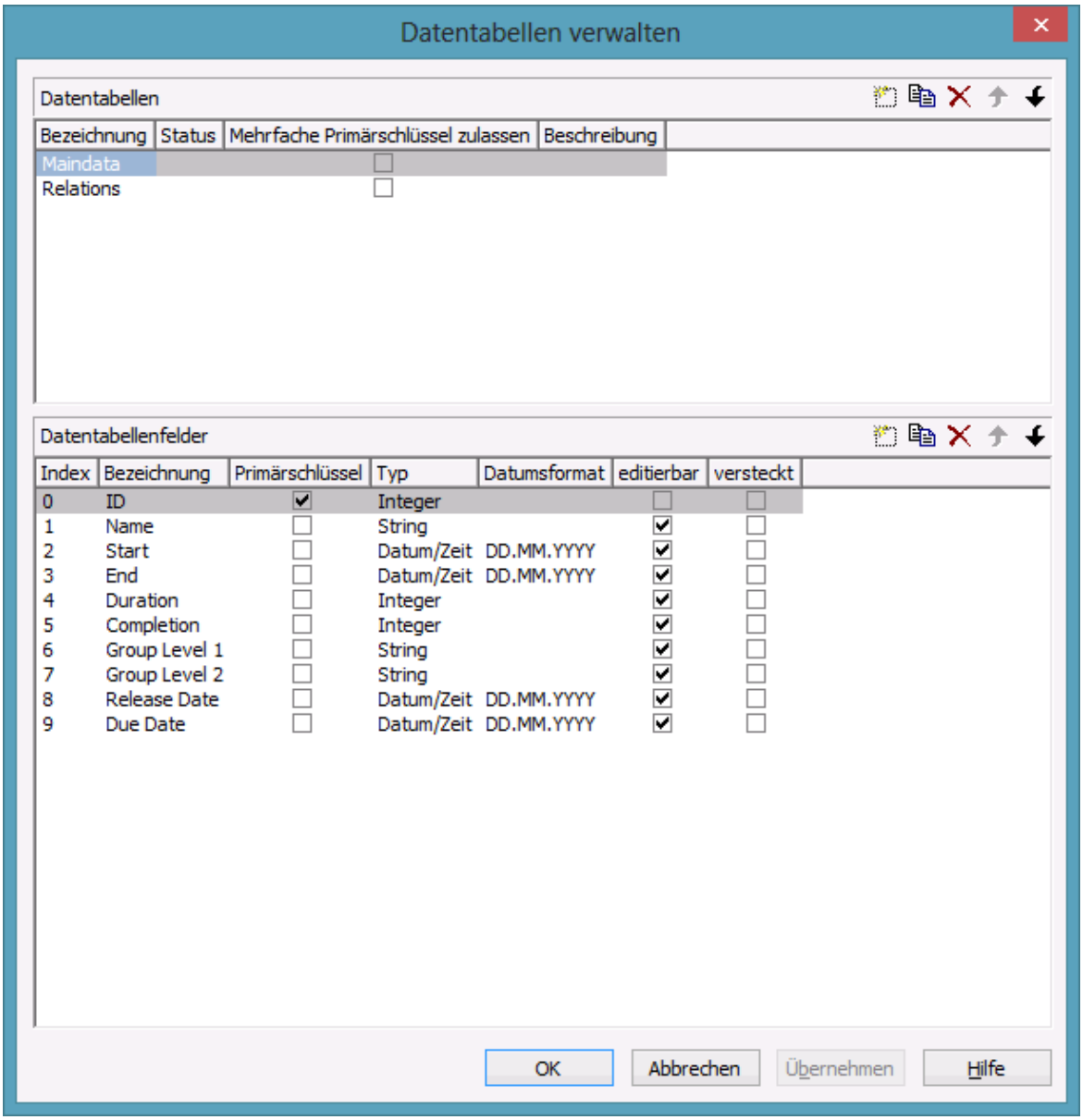
Update-Modus

Hier können Sie bei einem selbst erstellten Aktualisierungsverhalten auswählen, bei welcher Aktion des Cursors die Aktualisierung durch das Live Update erfolgen soll.

Verzögerungszeit

Nur wenn bei **Update-Modus** die Option **Verharren während des Verschiebens** ausgewählt wurde, kann hier die Verzögerungszeit, nach der während des Verschiebvorgangs des Mauszeigers die Aktualisierung durch das Live Update erfolgen soll, eingestellt werden.

4.12 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.

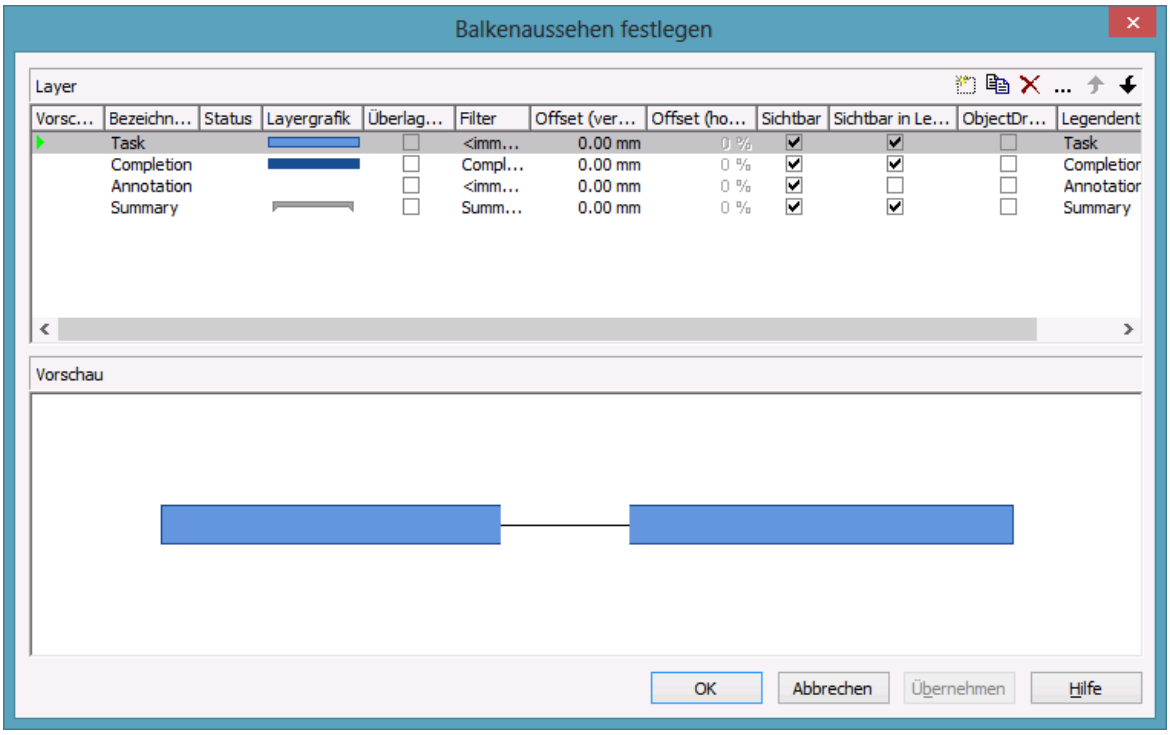
Dieses Datumsformat ist nur für die Dateneingabe, aber nicht für die Darstellung von Daten in Ihrer Grafik relevant.

- **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.

4.13 Dialogfeld "Balkenaussehen festlegen"



Vorgänge werden durch Balken dargestellt. Das grafische Design eines Balkens wird im Balkenaussehen festgelegt. Das Balkenaussehen setzt sich aus einem oder mehreren Layern zusammen, die dem Vorgang dynamisch über Filter zugewiesen werden.

Layer stellen Termine (Symbollayer) oder Terminpaare (Rechteck-Layer oder Linien-Layer) grafisch dar. Diese Termine kommen aus den Datenfeldern, die Sie im Dialog **Layer bearbeiten** auswählen.

Layer beinhalten sowohl grafische Attribute (Form, Linienfarbe, Muster, etc.) als auch eine Beschriftung. Außerdem können sie unterschiedliche Höhen und einen Offset haben, um sicherzustellen, dass alle Layer, die einem Vorgang zugewiesen sind, sichtbar sind.

Wenn ein Vorgang durch mehrere Layer dargestellt wird, werden die einzelnen Layer nacheinander gezeichnet, wobei sie einander u. U. überlagern. Der Layer, der ganz oben in der **Layer**-Tabelle steht, wird zuerst gezeichnet. Der letzte Layer in der **Layer**-Tabelle wird zuletzt gezeichnet und verdeckt u. U. die zuvor gezeichneten Layer. Aus der grafischen Überlagerung aller Layer, deren Filter auf einen Vorgang zutreffen, ergibt sich das Balkenaussehen.

Layer

In jeder Zeile dieser Tabelle können Sie einen Layer definieren.

Vorschau

Alle Layer, die in dieser Spalte durch eine kleine Pfeilspitze markiert sind, werden im Vorschaufenster angezeigt.

Durch eine grüne Pfeilspitze wird jeweils der Layer gekennzeichnet, auf dem der Cursor gerade steht. Dieser wird temporär nur so lange im Vorschaufenster angezeigt, wie der Cursor darauf verharrt.

Durch Anklicken können Sie die grüne in eine rote Pfeilspitze verwandeln und umgekehrt. Der rote Pfeil zeigt an, dass ein Layer im Vorschaufenster permanent dargestellt wird, d.h. auch wenn der Cursor gerade darauf steht.

Bezeichnung

In dieser Spalte werden die Namen aller verfügbaren Layer aufgeführt. Die Namen sind editierbar.

Status

In dieser Spalte wird jeder Layer gekennzeichnet, der seit dem Aufruf des Dialogs hinzugefügt () und/oder geändert () worden ist.

Layergrafik

In dieser Zeile wird das Aussehen jedes Layers (Layergrafik) angezeigt. Um die Layergrafik eines Layers zu verändern, klicken Sie auf die **Layer bearbeiten**-Schaltfläche oberhalb der Tabelle oder doppelklicken Sie auf den Eintrag **Layergrafik**. Sie gelangen in den Dialog **Layer bearbeiten**, in dem Sie die grafischen Attribute und die Beschriftung des Layers festlegen können.

Überlagerungslayer

Sie können im Modus **Alle Knoten in einer Zeile** und nicht **optimiert** die Überlagerung von Layern durch einen Überlagerungslayer anzeigen lassen. Es ist nur ein Aussehen für Überlagerungslayer möglich. Für den Überlagerungslayer wird kein Filter verwendet.

Filter

Der Filter, der mit einem Layer verbunden ist, bestimmt, welche Vorgänge durch den betreffenden Layer dargestellt werden. Um den Filter für einen Layer auszusuchen, markieren Sie das **Filter**-Feld. Dann erscheinen zwei Schaltflächen:

 Öffnen Sie die Kombobox mit allen verfügbaren Filtern und wählen Sie daraus einen Filter für den Layer aus.

 Oder klicken Sie auf die **Bearbeiten**-Schaltfläche, um den Dialog **Filter verwalten** aufzurufen. Dort können Sie Filter bearbeiten, kopieren, neu definieren oder löschen.

Beispiele für Filter sind "Standard", "Kritisch", "Meilenstein". Der gewählte Filter gibt die Bedingung an, unter der der Layer erscheinen soll. Haben Sie beispielsweise für den Layer "Früh" den Filter "kritisch" gewählt, so werden die "Früh"-Layer nur für kritische Vorgänge angezeigt.

Offset (vertikal)

Der vertikale Versatz des Layers gegenüber seiner horizontalen Mittellinie wird in Millimetern angegeben. Positive Werte bedeuten eine Verschiebung nach oben, negative nach unten.

Sobald Sie das **Offset (vertikal)**-Feld eines Layers markieren, erscheinen neben dem Eintrag Einstell-Schaltflächen, mit denen Sie den vertikalen Offset des markierten Layers schrittweise erhöhen oder verringern können.

 Außerdem erscheint diese Schaltfläche, über die Sie in den Dialog **Zuordnung einstellen** gelangen, in dem Sie eine datenabhängige Zuordnung des vertikalen Offsets vereinbaren können.

 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt.

Offset (horizontal)

(nur für Symbol-Layer) Sobald Sie das Feld **Offset (horizontal)** eines Symbollayers markieren, erscheint neben dem Eintrag eine Schaltfläche mit je einem Pfeil nach oben und nach unten. Sie können hier den horizontalen Offset des markierten Layers (seinen horizontalen Versatz gegenüber dem Layerdatum) in Schritten von jeweils 1 % erhöhen oder verringern (-50 bis +50 %).

Sichtbar

Deaktivieren Sie dieses Kontrollkästchen, damit der Layer unsichtbar ist. So können Sie einen Layer ausblenden, ohne ihn zu löschen.

Sichtbar in Legende

Aktivieren Sie dieses Kontrollkästchen, damit der Layer in der Legende dargestellt wird.

ObjectDraw-Ereignisse

Aktivieren Sie dieses Kontrollkästchen, damit die Ereignisse **VcObjectDrawing** und **VcObjectDrawn** bei Knoten, die mit diesem Layer gezeichnet werden, auftreten können.

Legendentext

Hier können Sie einen Legendentext für den Layer festlegen.

Layer hinzufügen

 Ein neuer Layer wird erzeugt.

Layer kopieren

 Der markierte Layer wird kopiert.

Layer löschen

 Der markierte Layer wird gelöscht.

Layer bearbeiten

 Der Dialog **Layer bearbeiten** wird geöffnet.

Layer früher/später zeichnen

Wenn ein Knoten aus mehreren Layern besteht, werden die einzelnen Layer dieses Knotens schichtweise übereinander gezeichnet. Dabei wird mit dem obersten Layer der Tabelle angefangen. Je tiefer die Position eines Layers in

der Liste, desto mehr andere Layer überlagert er. Die Zeichenreihenfolge der Layer entspricht also ihrer Reihenfolge in der Tabelle.

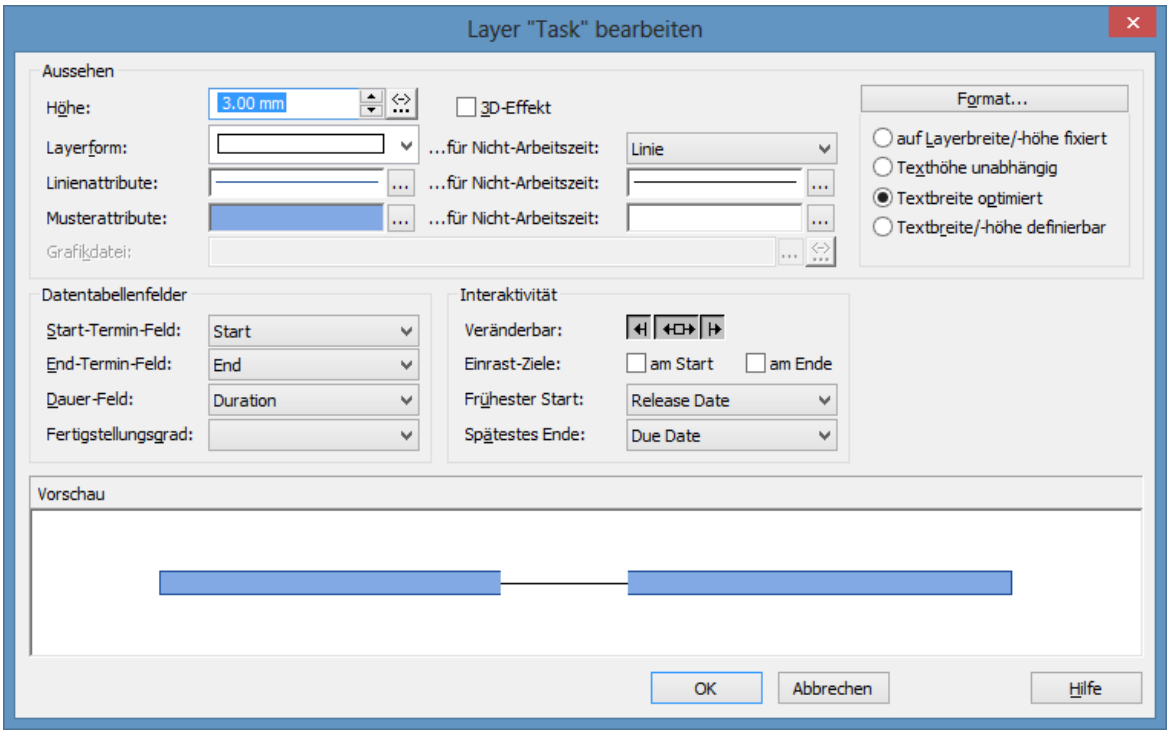
 Klicken Sie die Schaltfläche **Layer früher zeichnen** an, um den markierten Layer in der Tabelle eine Position höher zu setzen. In der Darstellung gelangt er dabei um eine Stelle weiter in den Hintergrund. Wenn er ganz oben in der Tabelle steht, wird er von allen anderen Layern überlagert.

 Klicken Sie die Schaltfläche **Layer später zeichnen** an, um den markierten Layer in der Tabelle eine Position tiefer zu setzen. In der Darstellung gelangt er dabei um eine Stelle weiter in den Vordergrund. Wenn er ganz unten in der Tabelle steht, überlagert er alle anderen Layer.

Vorschaufenster

Das Vorschaufenster zeigt alle Layer, die in der Tabelle markiert sind, mit den durch die Festlegung der Zeichnungsreihenfolge und des Offsets bestimmten Überlagerungen an.

4.14 Dialogfeld "Layer bearbeiten"



Sie erreichen dieses Dialogfeld über das Dialogfeld **Balkenaussehen festlegen**. In der Kopfzeile wird der Name des zu bearbeitenden Layers angezeigt.

Höhe

In diesem Feld können Sie die Höhe des Layers in Millimetern definieren.

Dies geschieht entweder durch direkte Eingabe des gewünschten Wertes in das Feld oder über die beiden Pfeil-Schaltflächen nach oben/nach unten.

 Über diese Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**, in dem Sie eine datenabhängige Zuordnung der Layer-Höhe vereinbaren können.

 Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche gefüllt dargestellt.

3D-Effekt

Sie können hier wählen, ob die Layer einen räumlichen Eindruck erhalten sollen oder nicht.

Layerform

Aus der **Layerform**-Liste können Sie die Form des Layers auswählen. Zur Auswahl stehen:

- **Bitmap-Layer**: Sie können unter **Grafikdatei** die zu verwendende Bitmapdatei auswählen.
- **Unsichtbares Symbol**: Der Layer ist mit Ausnahme seiner Beschriftung unsichtbar und wird auch nicht in der Legende dargestellt.
- **Rechtecklayer**
- **Keilförmige Layer**: aufsteigend oder absteigend keilförmig
- **Linienlayer**
- Verschiedene Typen von **Symbollayern**.

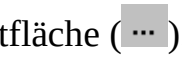
Rechtecklayer, keilförmige Layer und Linienlayer dienen zur Darstellung von Zeitspannen. Keilförmige Layer können verwendet werden, um ansteigende bzw. ausklingende Aktivitäten darzustellen, beispielsweise am Anfang bzw. am Ende von Vorgängen. Symbollayer dienen zur Darstellung von Zeitpunkten.

Form für arbeitsfreie Zeiten

Wählen Sie hier, in welcher Form arbeitsfreie Zeiten in Rechtecklayern dargestellt werden. Zuvor muss auf der Eigenschaftenseite **Knoten** die Option**Layer zeigen arbeitsfreie Zeiten anzeigen** ausgewählt werden.

Zur Auswahl stehen die Formen <Rechteck>, <Linie>, <freie Bereich> oder <keine>. <Keine> bewirkt, dass der Layer durchgezogen wird. In Kombination mit der oben erwähnten Option kann man so für einzelne Layer die arbeitsfreien Zeiten darstellen und für andere nicht.

Linienattribute

Hier wird die ausgewählte Linie der Layerumrandung angezeigt. Um die Linie zu ändern (Typ, Dicke, Farbe), klicken Sie auf die **Bearbeiten**-Schaltfläche (). Dann öffnet sich der Dialog **Linienattribute bearbeiten**.

Linienattribute für arbeitsfreie Zeiten

Bestimmen Sie hier, wie Layer für arbeitsfreie Zeiten umrandet werden. Durch Klick auf  gelangen Sie in den Dialog **Linienattribute bearbeiten**

Musterattribute

Hier wird das ausgewählte Muster des Layers angezeigt. Durch Klick auf ... gelangen Sie in den Dialog <bMusterattribute bearbeiten, in dem Sie Muster, Musterfarbe, Hintergrundfarbe oder die 2. Musterfarbe einstellen können.

Musterattribute für arbeitsfreie Zeiten

Bestimmen Sie hier, ob und wie Layer für arbeitsfreie Zeiten gefüllt werden. Durch Klick auf ... gelangen Sie in den Dialog **Musterattribute bearbeiten**

Grafikdatei

*(nur aktiv, wenn unter **Layerform** die Option <Bitmap-Layer> gewählt wurde*) Hier können Sie eine Grafikdatei auswählen, mit der der Layer dargestellt werden soll.

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

... Klicken Sie auf diese Schaltfläche, um den Windows-Dialog **Grafikdatei auswählen** zu öffnen.

 Über diese Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Grafik vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt ().

Bei der Darstellung der Grafik wird die Farbe des Pixels ihrer Ecke links oben durch die Diagramm-Hintergrundfarbe ersetzt, d. h. alle Pixel der Grafik in dieser Farbe werden transparent dargestellt.

auf Layerbreite/höhe fixiert

Wenn Sie diese Option wählen, werden Höhe und Breite der Beschriftung durch die Layerhöhe bzw. -breite bestimmt.

Texthöhe unabhängig

Wenn Sie diese Option wählen, ist die Höhe der Beschriftung außerhalb eines Layers unabhängig von der Layerhöhe, während ihre Breite durch die Layerbreite bestimmt wird. Die Höhe einer Beschriftung innerhalb eines Layers wird immer durch die Layerhöhe beschränkt.

Textbreite optimiert

Wenn Sie diese Option wählen, ist die Breite der Beschriftung außerhalb eines Layers unabhängig von der Layerbreite, während ihre Höhe durch die Layerhöhe bestimmt wird. Die Breite einer Beschriftung innerhalb eines Layers wird immer durch die Layerbreite beschränkt.

Textbreite/-höhe definierbar

Wenn Sie diese Option wählen, sind Breite und Höhe der Beschriftung unabhängig von der Layerbreite bzw. -höhe. Sie können dann die Breite und die Zeilenanzahl für die Felder einzeln im Dialog **Layerformat bearbeiten** oder über die Eigenschaften **MinimumWidth** und **TextLineCount** in Objekten vom Typ **VcLayerFormatField** festlegen.

Start-Termin-Feld

Definieren Sie hier den Anfangstermin des gewählten Layers, bspw. Frühester Anfang, Spätester Anfang, Geplanter Start.

Format

Über diese Schaltfläche öffnen Sie den Dialog **Layerformat bearbeiten**.

End-Termin-Feld

Definieren Sie hier den Endtermin des gewählten Layers, bspw. Frühestes Ende, Spätestes Ende, Geplantes Ende.

Ein Rechteck- bzw. Linien-Layer wird durch die Angabe eines Start-Termin-Feldes und eines End-Termin-Feldes oder eines Start-Termin-Feldes und einer Dauer definiert. Sind sowohl das End-Termin-Feld als auch die Dauer spezifiziert, so hat die Dauer Vorrang vor dem End-Termin-Feld. Bei Interaktionen wird dann aber nicht nur das Dauer-Feld aktualisiert, sondern auch das End-Termin-Feld.

Dauer

Die Einheit der Dauer wird in Abhängigkeit von der Einstellung der Zeiteinheit auf der Eigenschaftenseite **Allgemeines** interpretiert.

Wählen Sie aus der Liste das Datenfeld aus, das die Dauer des gewählten Layers enthält.

Fertigstellungsgrad

(nicht aktiv für Symbol- und Bitmaplayer) Wenn der aktuelle Layer den prozentualen Fertigstellungsgrad anzeigen soll, wählen Sie aus der Liste das Datenfeld aus, das den prozentualen Fertigstellungsgrad des gewählten Layers enthält.

Der durch den Layer dargestellte Endtermin wird aus dem Start-Termin-Feld, dem Endtermin-Feld bzw. der Dauer und dem Fertigstellungsgrad berechnet. Die dem Vorgang zu Grunde liegenden Daten bleiben unverändert.

Veränderbar

Sie können hier festlegen, ob der Anwender mit der Maus den gesamten Layer, den Layer-Anfang und/oder das Layer-Ende verschieben darf.

Sie können drei Optionen für den Anwender freigeben oder sperren:

1. den Layer-Anfang verändern
2. den gesamten Layer (d. h. Anfang und Ende des Layers gemeinsam) verschieben
3. das Layer-Ende verändern

Die gedrückten Schaltflächen zeigen die für den Anwender freigegebenen Optionen an.

Einrastziele

Bestimmen Sie hier, ob der Layer Start- und/oder Endedatum als Einrastziel definiert.

Frühester Start

Das im hier ausgewählten Datenfeld hinterlegte Datum mit Uhrzeit stellt die untere Grenze für die Startzeit des Layers beim interaktiven Verschieben des Layers bzw. Knotens dar.

Diese Option kann auch über die Eigenschaft **VcLayer.MinimumStartDataFieldIndex** festgelegt werden.

Spätestes Ende

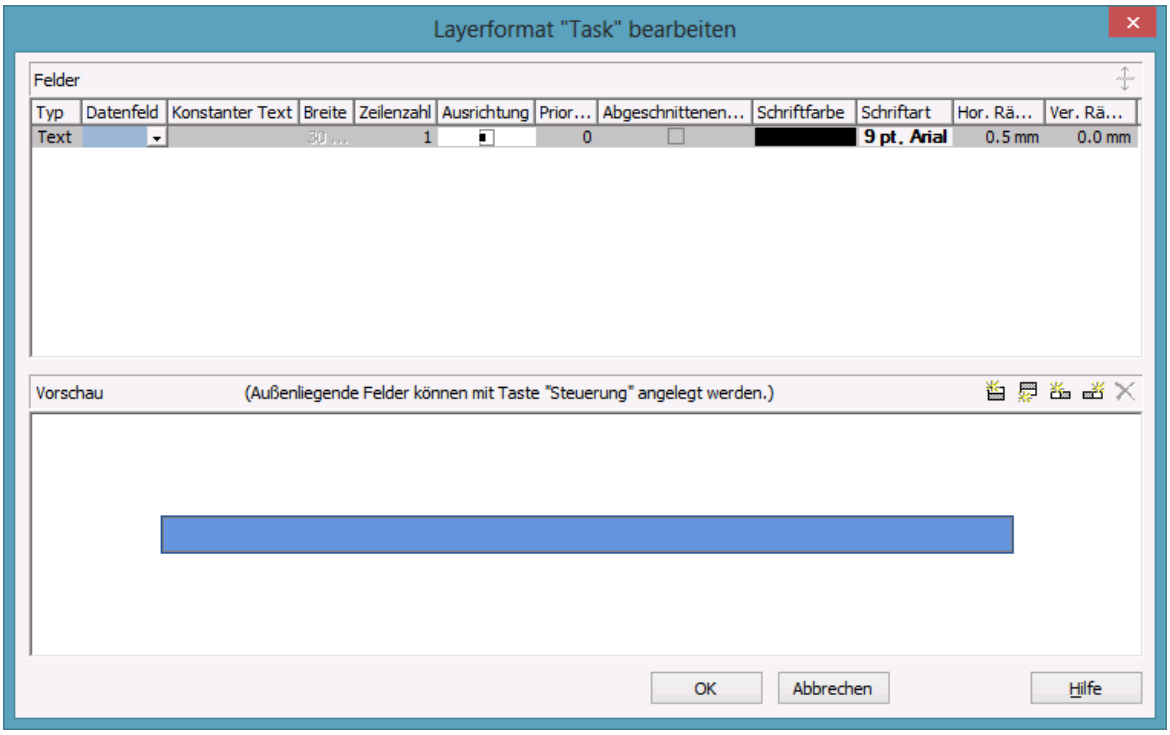
Das im hier ausgewählten Datenfeld hinterlegte Datum mit Uhrzeit stellt die obere Grenze für die Endzeit des Layers beim interaktiven Verschieben des Layers bzw. Knotens dar.

Diese Option kann auch über die Eigenschaft **VcLayer.MaximumEndDataFieldIndex** festgelegt werden.

Vorschau

In der Vorschau wird der Layer mit den hier gewählten Einstellungen dargestellt. Dabei werden Balkenlayer in der Vorschau immer von einer durchgezogenen Linie durchbrochen. Diese Linie zeigt an, wie der Layer zur Laufzeit aussieht, wenn arbeitsfreie Zeiten angezeigt werden und wenn den Knoten ein Kalender zugewiesen wird. (Diese beiden Einstellungen können Sie auf der Eigenschaftenseite **Knoten** vornehmen. Die Darstellung in der Vorschau ist allerdings unabhängig von diesen Einstellungen.)

4.15 Dialogfeld "Layerformat bearbeiten"



Sie erreichen dieses Dialogfeld, indem Sie im Dialog **Layer bearbeiten** auf die **Format**-Schaltfläche klicken.

Typ

Hier wird der Feldtyp (Text) angezeigt.

Datenfeld

Wählen Sie hier das Datenfeld, dessen Inhalt in dem aktuellen Feld ausgegeben werden soll. Außer den Datenfeldern, die Sie in der Datendefinition vereinbart haben, gibt es die Option <Zeilennummer>: Dabei wird die Zeilennummer des Balkens ausgegeben.

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 90 mm.

Hinweis: Nur nutzbar, wenn im Dialog **Layer bearbeiten** die Option **Textbreite/-höhe definierbar** gewählt ist.

Zeilenanzahl

Bestimmen Sie die Anzahl der Textzeilen des aktuellen Feldes.

Hinweis: Nur nutzbar, wenn im Dialog **Layer bearbeiten** die Option **Textbreite/-höhe definierbar** gewählt ist.

Nur für außenstehende Felder eines Layers: Sie können die Anzahl der Textzeilen hier auch dynamisch gestalten, d.h. abhängig von der Textlänge. Dazu haben Sie zwei Möglichkeiten:

1. Sie können die Zeilenzahl des Feldes direkt ermitteln lassen, in ein Feld schreiben und den Inhalt hier verwenden
2. Sie können die Zeilenzahl in einer Zuordnungstabelle niederlegen und hier zuordnen

Zu 1: Über die Methode **VcLayerFormatField.CalculateLineCount(...)** können Sie Anzahl der Zeilen ermitteln lassen und in einem Feld ablegen. Das Feld können Sie über den Zuweisungsdialog zuweisen, indem Sie in dem Feld **Zeilenanzahl** auf die rechte Schaltfläche mit dem Doppelpfeil drücken:

Daraufhin öffnet sich der Zuordnungsdialog, in welchem Sie oben den Namen des Datenfeldes auswählen. Das darunterliegende Feld für eine Zuordnungstabelle lassen Sie leer.

Zu 2: Die Verwendung einer Zuordnungstabelle setzt ihre vorherige Anlage und Bestückung voraus; zudem muss sie vom Typ **vcNumberMap** sein. Bei diesem Typ werden Texten Zahlen zugeordnet. Wenn ein hier niedergelegter Text in einem (noch zuzuweisenden) Datenfeld gefunden wird, erhält das Feld die zugeordnete Anzahl von Textzeilen. Zuordnungstabellen legen Sie über die Eigenschaftenseite **Objekte** und die dortige Schaltfläche **Zuordnungstabellen...** an. Die erstellte Zuordnungstabelle weisen Sie nun ebenfalls in dem Feld **Zeilenanzahl** durch Drücken auf die rechte Schaltfläche mit dem Doppelpfeil zu. In dem sich öffnenden Zuordnungsdialog können Sie das Datenfeld und die Zuordnungstabelle

auswählen und weisen auf diese Weise das Datenfeld zu, dessen Inhalt mit den Textzeichen der Zuordnungstabelle abgeglichen wird. Sie können in diesem Dialog den Inhalt der ausgewählten Tabelle einsehen und weiterführend auch ändern.

Ausrichtung

Bestimmen Sie hier die Ausrichtung des Inhalts in dem markierten Feld (rechtsbündig, zentriert oder linksbündig).

Priorität

Legen Sie hier die Priorität des Layerfeldes fest. Sie können Prioritätswerte zwischen -9 und +9 vergeben. Falls die Gesamtbreite des Layers nicht ausreichen sollte, um die Inhalte aller Layerfelder darzustellen, entscheidet die Priorität darüber, von welchen Layerfeldern der Inhalt vorrangig dargestellt wird. Zunächst wird der Inhalt des Feldes mit der höchsten Priorität nach Möglichkeit vollständig dargestellt. Dann wird der Inhalt von Feldern mit niedrigeren Prioritäten möglichst vollständig dargestellt. Kann der Inhalt eines Feldes nicht ganz dargestellt werden, wird er abgeschnitten oder unterdrückt, je nach Einstellung unter **Abgeschnittenen Text unterdrücken**.

Abgeschnittenen Text unterdrücken

Bestimmen Sie hier, ob ein Text, der nicht ganz in das Feld passt, unterdrückt oder abgeschnitten werden soll.

Schriftfarbe

Die Schriftfarbe des Feldes wird hier angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Die Farbzuordnungstabelle 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 hier angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**.

 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Schriftart vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt ().

selektierte Eigenschaft in alle Felder übernehmen

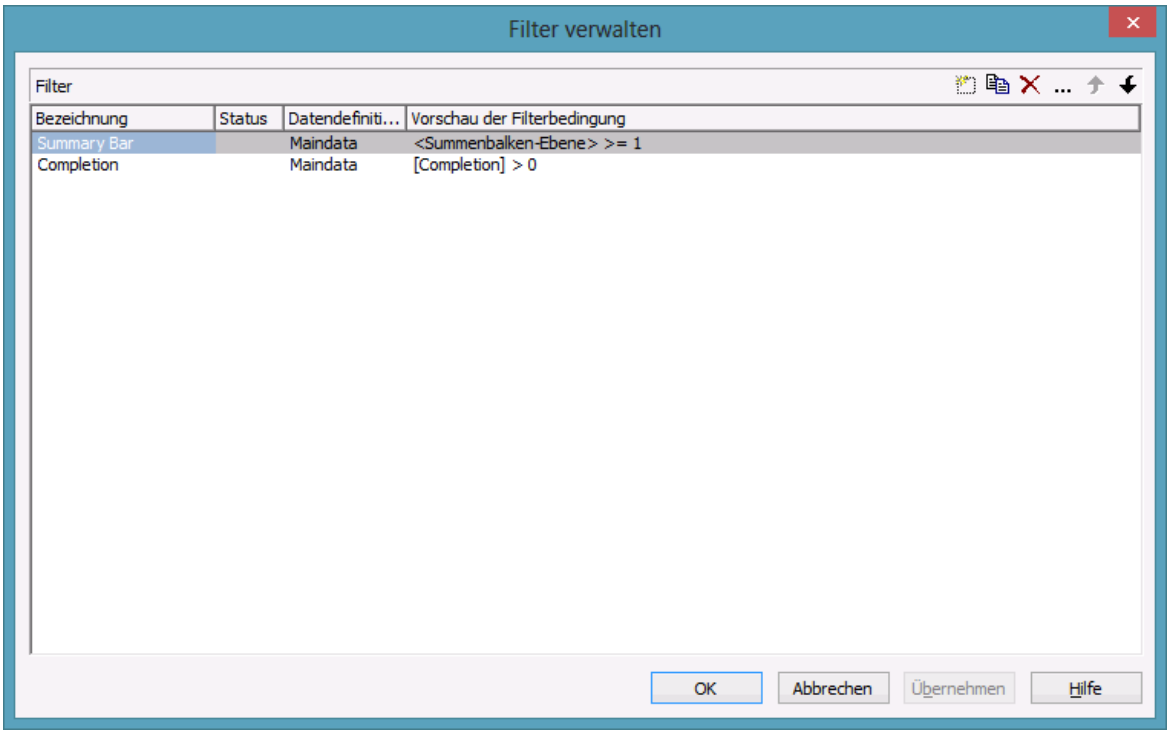
 Klicken Sie auf diese Schaltfläche, damit die selektierte Eigenschaft in alle Felder übernommen wird.

Vorschau

Hier werden die aktuellen Felder 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. Wenn Sie Felder außerhalb des Layers anlegen möchten, so ist zusätzlich die **Strg**-Taste zu drücken. Zum Löschen von Feldern können Sie auch die Entf-Taste benutzen.

4.16 Dialogfeld "Filter verwalten"



Sie erreichen dieses Dialogfeld

- von der Eigenschaftenseite **Objekte**
- für Layer: vom Dialogfeld **Balkenaussehen festlegen**
- für Tabellenformate: vom Dialogfeld **Tabelle bearbeiten**
- für Verbindungen: von der Eigenschaftenseite **Verbindungen** über die **Filter**-Schaltfläche
- für Histogrammkurven: vom Dialog **Histogramm bearbeiten** über die **Filter**-Kombobox
- für Knoten: von der Eigenschaftenseite **Knoten** über die **Filter**-Schaltfläche

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.

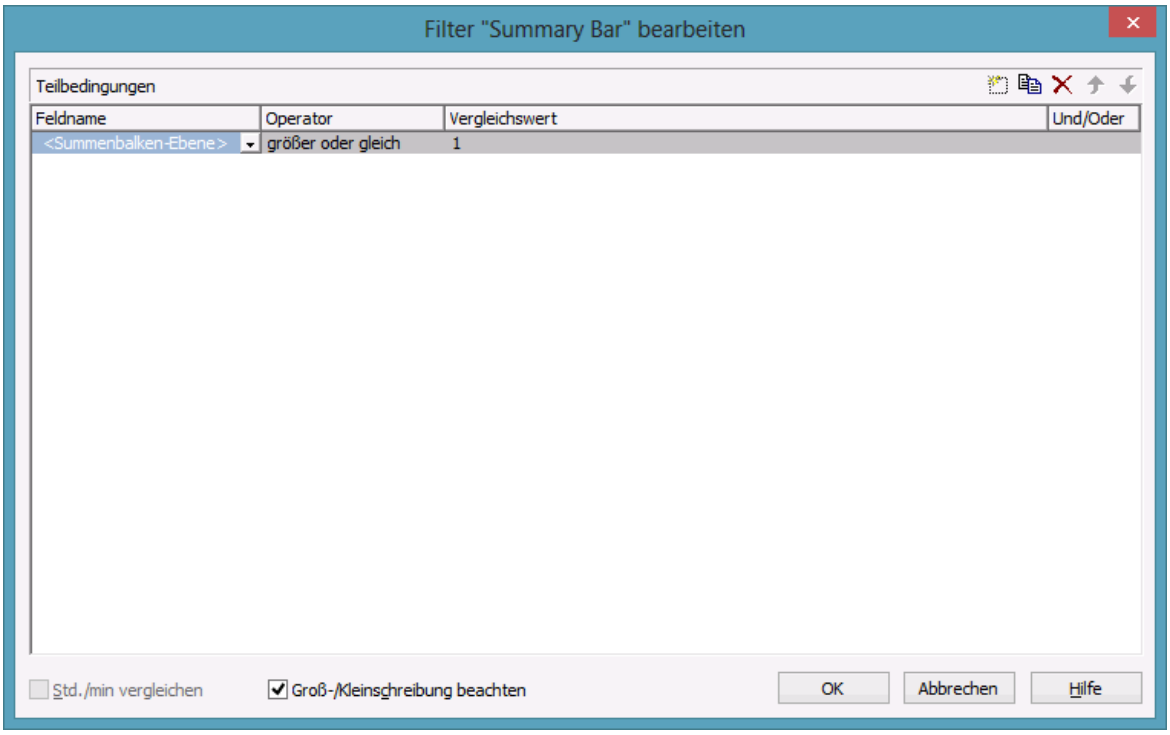
Filter bearbeiten

 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.

4.17 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 sowie eine Reihe von vordefinierten Einträgen:

- Der Eintrag <Gantt: Summenbalken-Ebene> kann für Gantt-Diagramme verwendet werden, um die Summenbalken darzustellen. Definieren Sie beispielsweise einen Filter mit der Bedingung "<Gantt: Summenbalken-Ebene> größer oder gleich 1" und weisen Sie ihn einem Layer (z.B. "Summenbalkenebene 1") zu, um damit die Summenbalken der 1. Ebene darzustellen. Beachten Sie bitte, dass auf der Eigenschaftenseite **Sortierung** die Option **Summenbalken anzeigen** aktiviert sein muss, damit die Summenbalken tatsächlich dargestellt werden.
- Filter mit dem Eintrag <Gantt: Gruppierungs-Ebene> können beispielsweise für Gantt-Diagramme im Dialog **Tabelle bearbeiten** als Zeilenfilter für Basiszeilen verwendet werden.
- <Gantt: Kollabiert>: für kollabierte Gruppen
- <Gantt: Jeder Knoten in eigener Zeile>: für eine Darstellung, bei der jeder Knoten in einer eigenen Zeile dargestellt wird
- <Gantt: Knoten überlappend>: für eine Darstellung bei der Knoten einander ggf. überlappen
- <Gantt: Zeile>: Eintrag, um Filter zeilenweise definieren zu können.
- <Gantt: Sammelvorgang>: Eintrag, um Filter für Sammelvorgänge definieren zu können.
- <Knoten schreibgeschützt>: Eintrag, um Filter für schreibgeschützte Knoten definieren zu können

Diese Option kann auch zur Laufzeit über die VcFilterSubCondition-Eigenschaft **DataFieldIndex** gesetzt werden.

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.)

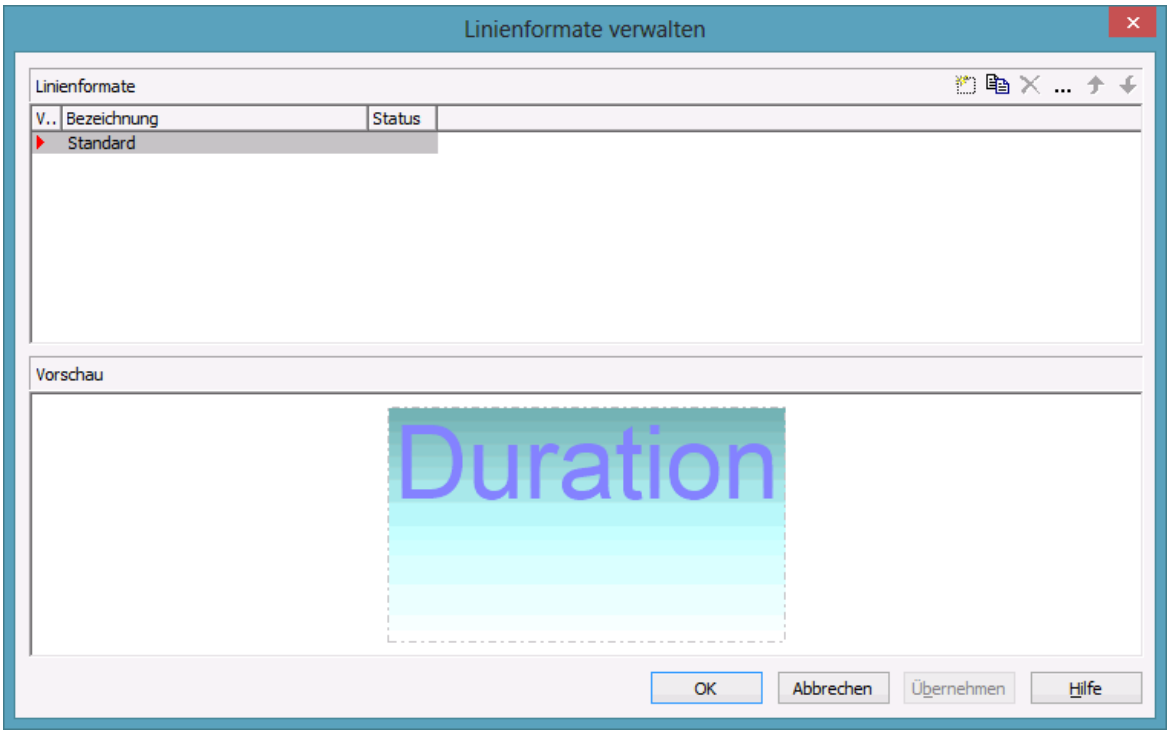
Std./min vergleichen

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.

4.18 Dialogfeld "Linienformate verwalten"



Sie erreichen dieses Dialogfeld

- von der Eigenschaftenseite **Objekte**
- aus dem Dialog **Liniengitter verwalten**

Vorschau

In dieser Spalte wird das jeweils im Vorschaufenster angezeigte Linienformat gekennzeichnet.

Bezeichnung

In dieser Spalte stehen die Namen aller vorhandenen Linienformate. Die Namen sind editierbar.

Status

In der Spalte **Status** wird jedes Linienformat gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Linienformat hinzufügen

 Ein neues Linienformat mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

Linienformat kopieren

 Das markierte Linienformat wird kopiert.

Linienformat löschen

 Der Filter, den Sie in der Liste markiert haben, wird gelöscht. Es können nur Filter gelöscht werden, die zur Zeit nicht benutzt werden.

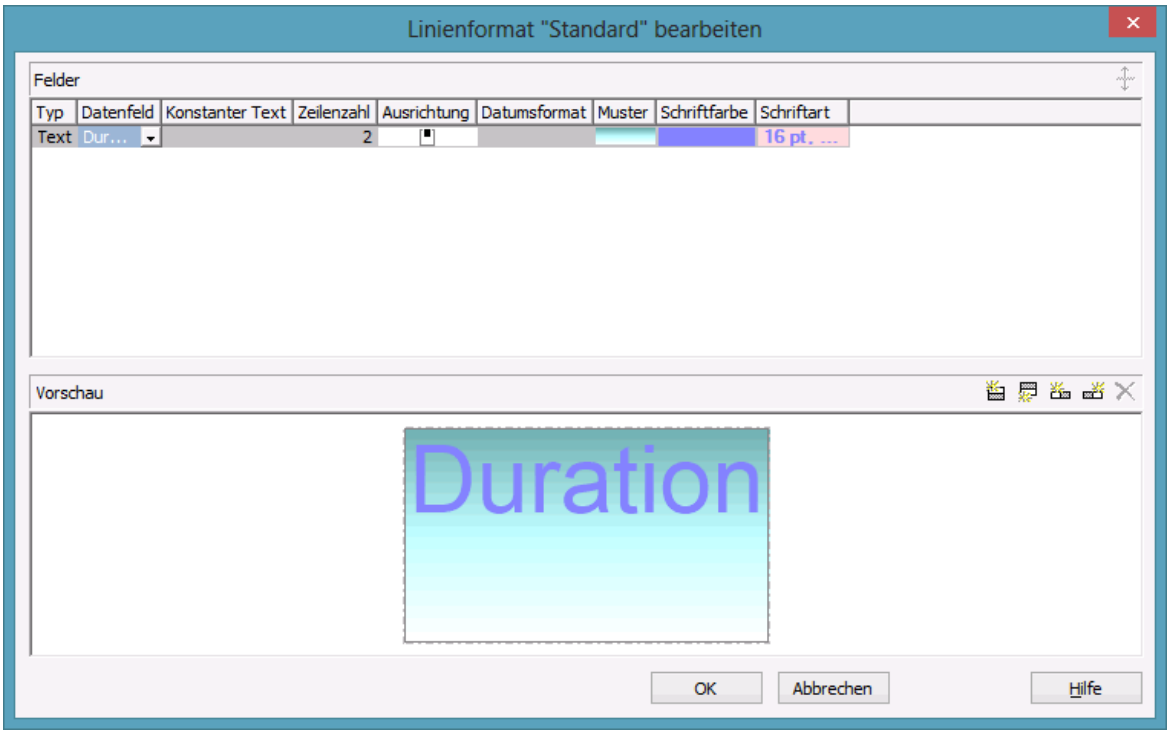
Linienformat bearbeiten

 Sie öffnen den Dialog **Linienformat bearbeiten**, in dem Sie die Attribute des Linienformats (Farbe, Muster etc.) auswählen können

Linienformat eine Zeile nach oben/unten

 Mit Hilfe dieser Schaltflächen können Sie das markierte Linienformat eine Zeile nach oben/unten schieben.

4.19 Dialogfeld "Linienformat bearbeiten"



Sie erreichen dieses Dialogfeld

- durch Klick auf **Linienformate** auf der Eigenschaftenseite **Objekte** und im dann folgenden Dialog **Linienformate verwalten** auf die Schaltfläche ...
- aus dem Dialog **Liniengitter verwalten**

Typ

Hier wird der Feldtyp (Text) angezeigt.

Datenfeld

Wählen Sie hier das Datenfeld, dessen Inhalt für die Beschriftung der Liniengitter genutzt werden soll. Außer den Datenfeldern, die Sie in der Datendefinition vereinbart haben, gibt es die Optionen <Datum> und <Gruppentitel>: Dabei wird das aktuelle Datum bzw. bei einer bestehenden Gruppierung der Gruppentitel ausgegeben.

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.

Zeilenanzahl

Bestimmen Sie die Anzahl der Textzeilen des aktuellen Feldes.

Ausrichtung

Bestimmen Sie hier die Ausrichtung des Inhalts in dem markierten Feld (rechtsbündig, zentriert oder linksbündig).

Datumsformat

Wenn Sie als Datenfeld für die Beschriftung der Liniengitter <Datum> gewählt haben, können Sie hier für dieses Datum ein Ausgabeformat wählen. Für das Datumsformat 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 "Uhr" (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

xC/XC: Sie können das Datum als maximal zehngliedrige, einfache Aufwärtszählung ab einem bestimmten Referenzdatum gestalten, z.B. "15:05:07:16:00". Dieses Beispiel datiert 15 Monate, 5 Tage, 7 Stunden, 16 Minuten und 0 Sekunden später als das gesetzte Referenzdatum. Die Notation dazu ist: **xC44:C33:C22:C11:C00**. Es sollen je 2 Stellen für die Glieder 4...0 vorgesehen werden, mit einem vorausgehenden "-" Symbol falls der Wert negativ ist. Die Trennzeichen sind variabel und könnten durch andere ersetzt werden. "x" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, aber kein "+"Symbol, falls er positiv ist. "X" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, und ebenso ein "+"Symbol, falls er positiv ist. Im Dialog **Zeitskalenabschnitt ... bearbeiten** sollten die Felder **Referenzdatum verwenden** und **Hauptmarkierung am Referenzdatum ausrichten** aktiviert sein, ebenso sollte **Serielle Beschriftungen** auf **Keine** gesetzt sein. Das Referenzdatum wird zur Laufzeit über die Methode **VcRibbon.set ReferenceDate** gesetzt und überschreibt die Werte des Dialogs.

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.

Muster

Legen Sie hier Hintergrundfarbe und -muster fest. Durch Klick auf  gelangen Sie in den Dialog **Musterattribute bearbeiten**. Hier können Sie durch Klick auf  neben dem jeweiligen Attribut ein Muster, eine Hintergrundfarbe sowie eine 2. Musterfarbe auswählen. Zusätzlich zu der vorgegebenen Farbpalette können Sie beliebige Farben definieren. Es stehen außerdem transparente Farben zur Verfügung.

Durch Klick auf die Schaltfläche  gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung des Musters und der Farben vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt ().

Schriftfarbe

Die Schriftfarbe des Feldes wird hier angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Die Farbzuordnungstabelle 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 hier angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**.

 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Schriftart vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt ().

selektierte Eigenschaft in alle Felder übernehmen

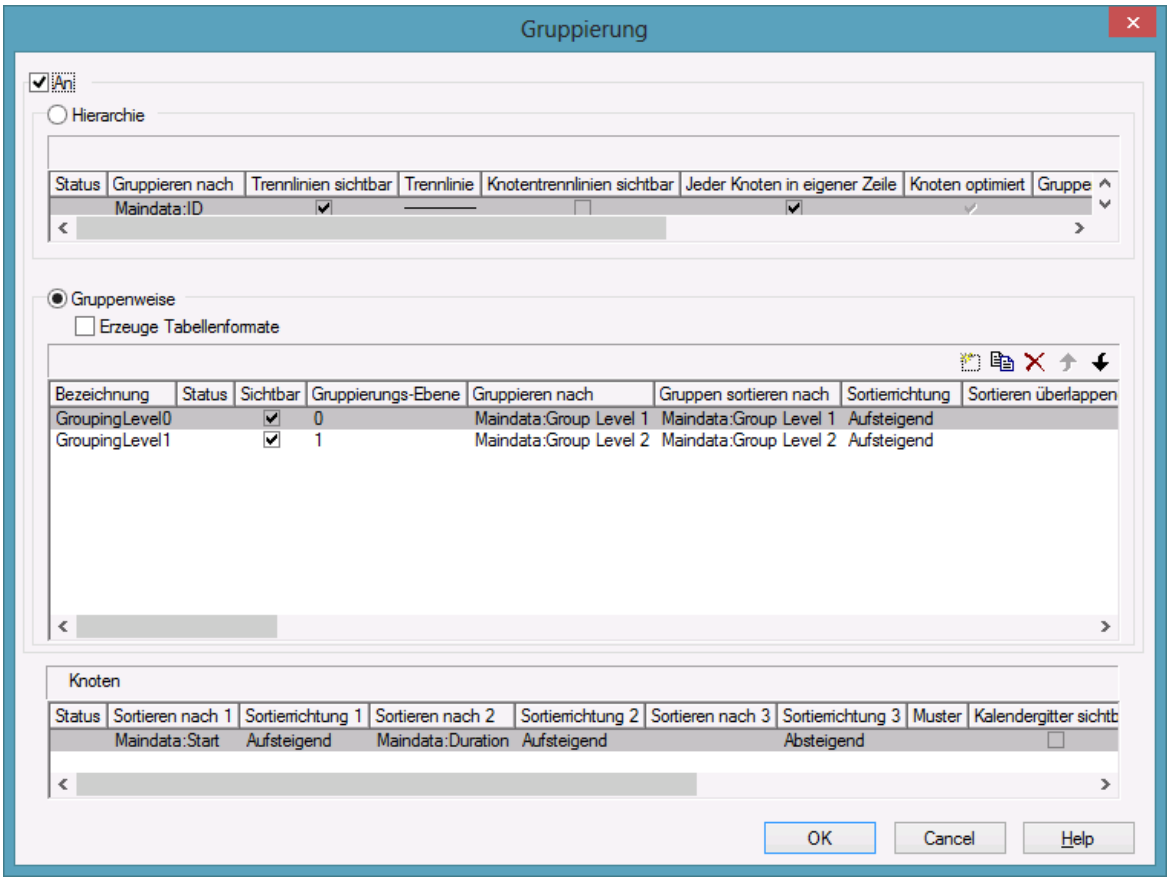
 Klicken Sie auf diese Schaltfläche, damit die selektierte Eigenschaft in alle Felder übernommen wird.

Vorschau

Hier werden die aktuellen Felder 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.

4.20 Dialogfeld "Gruppierung"



In diesem Dialog können Sie die Optionen zur Hierarchie, Gruppierung, Sortierung und zum Layout dieser Strukturen wählen.

Der Dialog enthält drei Bereiche: **Hierarchie**, **Gruppenweise** und **Knoten**, in denen Sie die entsprechenden Eigenschaften setzen können.

An

Das Gruppieren von Knoten in Form einer Hierarchie (mit einem Hierarchie-Code) oder das Gruppieren nach anderen Kriterien werden hier grundsätzlich an- oder ausgeschaltet.

> Hierarchie

Hier können Sie festlegen, ob die Anordnung der Vorgänge hierarchisch (nach einem Hierarchie-Code) erfolgen soll. Die einzelnen Hierarchieebenen werden im Hierarchie-Code durch Punkte voneinander getrennt. Wenn Sie diese Option wählen, wird die **Gruppenweise** Anordnung automatisch deaktiviert.

In der Tabelle unterhalb der Option **Hierarchie** können Sie weitere Einstellungen für die hierarchische Anordung vornehmen.

Gruppieren nach

Legen Sie das Feld fest, nach dem die Vorgänge hierarchisch angeordnet werden sollen.

Trennlinien sichtbar

Legen Sie hier fest, ob nach jeder Hierarchieebene Trennlinien ausgegeben werden.

Diese Option kann auch über die Eigenschaft **VcHierarchyLevelLayout.-SeparationLinesVisible** gesetzt werden.

Trennlinie

Durch Klick auf die Schaltfläche ··· öffnet sich der Dialog **Linienattribute**, in dem Sie Art und Aussehen der Trennlinien bestimmen können.

Die dort wählbaren Linienattribute lassen sich auch über die entsprechenden Eigenschaften **VcHierarchyLevelLayout.SeparationLineColor**, **VcHierarchyLeveLayout.SeparationLineThickness** und **VcHierarchy-LevelLayout.SeparationLineType** festlegen.

Knoten in Kopfzeilen

Bestimmen Sie hier ob jeder Knoten einer Gruppe in einer eigenen Zeile ausgegeben wird oder nicht.

Wenn die Option gewählt ist, entfällt der Tabellenteil für die Vorgänge und man muss die Layerbeschriftung oder den Tooltip verwenden, um die Vorgänge für den Anwender zu identifizieren.

Knoten überlagernd

Mithilfe dieser Option können Sie bestimmen, ob auf dieser Hierarchieebene das Knotenlayout optimiert ist, oder ob Knoten überlappen.

Rümpfe kollabiert

Wenn Sie diese Option wählen, werden bei Programmstart alle Ebenen ab der zweiten Hierarchieebene kollabiert dargestellt. Sie können danach interaktiv expandiert werden.

Summenbalken

Wenn Sie dieses Kontrollkästchen aktivieren, werden Summenbalken für alle Ebenen angezeigt. Wenn Sie Summenbalken ebenenweise definieren möchten, müssen Sie einen Layer mit einer entsprechenden Filterbedingung (<Summenbalken-Ebene> = ...) definieren.

Diese Option kann auch über die Eigenschaft **VcHierarchyLevelLayout.-SummaryBarsVisible** gesetzt werden.

Automatisches Kollabieren von Gruppen

Wenn Sie diese Option wählen, werden beim interaktiven Verschieben eines Knotens oder einer Gruppe alle Gruppen ausser der gerade berührten kollabiert.

Wiederherstellen von automatisch kollabierten Gruppen

Wenn Sie diese Option wählen, werden beim interaktiven Verschieben eines Knotens/einer Gruppe die zuvor automatisch kollabierten wiederhergestellt.

Automatisches Expandieren von Gruppen

Wenn Sie diese Option wählen, wird beim interaktiven Verschieben eines Knotens/einer Gruppe die Zielgruppe automatisch expandiert.

Wiederherstellen von automatisch expandierten Gruppen

Wenn Sie diese Option wählen, werden beim interaktiven Verschieben eines Knotens/einer Gruppe die zuvor automatisch expandierten wiederhergestellt.

Seitenumbruch nach jeder Gruppe

Nach Klick auf  stehen folgende Möglichkeiten zur Auswahl:

- **Kein**: es wird kein Seitenumbruch ausgeführt

- **bei voller Seite**: wenn eine Gruppe durch einen Seitenumbruch getrennt würde, wird der Seitenumbruch bereits nach der vorhergehenden Gruppe durchgeführt
- **nach jeder Gruppe**: nach jeder Gruppe wird ein Seitenumbruch ausgeführt

Diese Optionen können auch über die Eigenschaft **VcHierarchyLevel-Layout.PageBreakMode** gesetzt werden.

Maximaler Level für Seitenumbrüche

Geben Sie hier an, bis zu welcher Hierarchieebene höchstens Seitenumbrüche nach den Gruppen durchgeführt werden sollen. Geben Sie z. B. eine 4 an, wird ab Ebene 5 kein Seitenumbruch mehr eingefügt.

Ist der Standardwert -1 angegeben, werden auf allen Ebenen Seitenumbrüche durchgeführt.

Diese Option kann auch über die Eigenschaft **VcHierarchyLevelLayout.-LevelMaximumForPagebreaks** gesetzt werden.

> Gruppenweise

Hier legen Sie fest, ob die Anordnung der Vorgänge in Gruppen (gruppiert nach unterschiedlichen Kriterien) erfolgen soll. Wenn Sie diese Option wählen, wird die **Hierarchie** -Anordnung automatisch deaktiviert.

Im Bereich unterhalb von **Gruppenweise** können Sie alle weiteren Einstellungen für die Gruppierung - vor allem für das Aussehen (Muster, Kalendergitter, Liniengitter etc.) - treffen. Für jede Gruppierungsebene können eigene Kriterien definiert werden. Über die Schaltflächen  können Ebenen hinzugefügt, kopiert, gelöscht und verschoben werden.

Erzeuge Tabellenformate

Wenn dieses Kontrollkästchen aktiviert ist, werden für jede Gruppierungsebene, die Sie hier neu festlegen, eigene Tabellenformate angelegt: Subtitle_n, Collapsed_n. Sie müssen diese dann ggf. im Dialog **Tabellenformat bearbeiten** noch anpassen, insbesondere das Datenfeld.

Ist dieses Kontrollkästchen nicht aktiviert, werden keine Tabellenformate angelegt, wenn eine neue Gruppierungsebene definiert wird. Sie müssen sie dann ggf. selbst anlegen. Mit Hilfe dieser Option können Sie mit nur zwei Gruppierungstabellenformaten (Subtitle und Collapsed) auskommen, und

diese ggf. für jede Gruppierungsebene über Zuordnungstabellen und Filter modifizieren.

Gruppenknoten sichtbar

Wenn Sie diese Option wählen, wird für Gruppen, für die eine eigene Datentabelle angelegt wurde, im Diagramm je ein Balken dargestellt. Aktivieren Sie dazu vorher auch die Option **Erweiterte Datentabellen zulassen** auf der Eigenschaftenseite **Allgemeines**.

Bezeichnung

Legen Sie hier einen Namen für die Gruppierungsebene fest.

Sichtbar

Wählen Sie, ob die Gruppen dieses Levels angezeigt werden oder nicht.

Gruppierungs-Ebene

Hier wird die Ebene angegeben, für die die in dieser Zeile vorgenommenen Einstellungen gelten sollen. Wenn Sie die Reihenfolge der Ebenen ändern möchten, benutzen Sie die Pfeilschaltflächen oberhalb der Tabelle.

Gruppieren nach

Wählen Sie das Datenfeld aus, nach dem Vorgänge auf der gewählten Gruppierungsebene gruppiert werden sollen. Wenn Sie den Leereintrag wählen, werden die Vorgänge auf der eingestellten Ebene nicht gruppiert.

Gruppen sortieren nach

Wählen Sie hier das Datenfeld aus, nach dem die Gruppen beim Programmstart sortiert werden sollen. Wird hier nichts eingestellt, ergibt sich die Reihenfolge aus der Reihenfolge der Vorgänge beim Laden.

Sortierrichtung

Legen Sie hier die Sortierungsrichtung auf der gewählten Gruppierungsebene fest (aufsteigend oder absteigend).

Sortieren überlappendender Knoten nach

Wählen Sie hier das Datenfeld aus, nach dem die Knoten der Gruppe, die in einer Zeile angeordnet sind, sortiert werden sollen. Wird hier nichts eingestellt, ergibt sich die Reihenfolge aus dem Anfangsdatum und der Dauer der Vorgänge, d.h. die frühesten und kürzesten Vorgänge liegen am weitesten vorn. Diese Eigenschaft ist nur wirksam, wenn die Eigenschaft **VcGroupLevelLayout.NodesArrangedOptimized** auf **False** gesetzt wurde.

Sortierrichtung überlappender Knoten

Legen Sie hier die Sortierungsrichtung der sich überlappenden Knoten fest (aufsteigend oder absteigend).

Sortieren optimierter Knoten nach

Wählen Sie hier das Datenfeld aus, nach dem die Knoten der Gruppe, die in einer Zeile angeordnet sind, sortiert werden sollen. Wird hier nichts eingestellt, ergibt sich die Reihenfolge aus dem Anfangsdatum und der Dauer der Vorgänge, d.h. die frühesten und kürzesten Vorgänge liegen am weitesten vorn. Diese Eigenschaft ist nur wirksam, wenn die Eigenschaft **VcGroupLevelLayout.NodesArrangedOptimized** auf **True** gesetzt wurde.

Sortierrichtung optimierter Knoten

Legen Sie hier die Sortierungsrichtung der optimierten Knoten fest (aufsteigend oder absteigend).

Muster

Wenn Sie auf  klicken, öffnet sich das Dialogfeld **Musterattribute**. Hier können Sie ein Hintergrundmuster und zwei Musterfarben für die Gruppentitelzeile festlegen sowie über die Schaltfläche  eine datenabhängige Zuordnung der jeweiligen Eigenschaft vereinbaren.

Kalender

Wählen Sie hier das Datenfeld aus, das den Namen des Kalenders enthält, der für den Gruppenknoten verwendet werden soll.

Kalendergitter sichtbar

Bestimmen Sie, ob ein Kalendergitter angezeigt wird.

Kalendergitter mit Untergruppen

Wählen Sie, ob das Kalendergitter auch für Untergruppen dargestellt werden soll.

Kalendergitter

Hier können Sie entweder aus der Drop-Down-Liste ein bestehendes Kalendergitter für die Gruppe auswählen oder im Dialog **Kalendergitter vewalten** (zu erreichen über ···) ein neues Kalendergitter anlegen. Nähere Informationen zu diesem Dialog finden Sie im Kapitel **Dialogfeld Kalendergitter verwalten**.

Bei Auswahl der Option <aus der Skala> wird das erste nicht sichtbare Kalendergitter aus der Zeitskala angezeigt.

Liniengitter sichtbar

Hier können Sie festlegen, ob ein Liniengitter angezeigt wird.

Liniengitter mit Untergruppen

Wählen Sie, ob das Liniengitter auch für Untergruppen dargestellt werden soll.

Liniengitter

Hier können Sie entweder aus der Drop-Down-Liste ein bestehendes Liniengitter für die Gruppierungsebene auswählen oder im Dialog **Liniengitter verwalten** (zu erreichen über ···) ein neues Liniengitter anlegen. Nähere Informationen zu diesem Dialog finden Sie im Kapitel **Dialogfeld Liniengitter verwalten**.

Stichtaglinien sichtbar

Hier können Sie festlegen, ob ein Liniengitter angezeigt wird.

Stichtaglinien mit Untergruppen

Wählen Sie, ob die Stichtaglinie auch für Untergruppen dargestellt werden soll.

Stichtaglinien

Hier können Sie entweder aus der Drop-Down-Liste ein bestehende Stichtaglinie für die Gruppierungsebene auswählen oder im Dialog **Stichtaglinie festlegen** (zu erreichen über ···) eine neue Stichtaglinie anlegen. Nähere Informationen zu diesem Dialog finden Sie im Kapitel **Dialogfeld Stichtaglinie festlegen**.

Trennlinien sichtbar

Wenn Sie diese Option wählen, wird bei jedem Gruppenwechsel eine Trennlinie ausgegeben.

Trennlinien oben

Wenn Sie diese Option wählen, wird die Trennlinie oberhalb von Gruppen gezogen (statt unterhalb).

Trennlinie

Sie können das Aussehen der Trennlinien bearbeiten, indem Sie auf die **Bearbeiten**-Schaltfläche klicken und so das Dialogfeld **Linie bearbeiten** öffnen.

Knoten in Kopfzeilen

Bestimmen Sie hier ob jeder Knoten einer Gruppe in einer eigenen Zeile ausgegeben wird oder nicht.

Wenn die Option gewählt ist, entfällt der Tabellenteil für die Vorgänge und man muss die Layerbeschriftung oder den Tooltip verwenden, um die Vorgänge für den Anwender zu identifizieren.

Knoten überlagernd

Mithilfe dieser Option können Sie bestimmen, ob auf dieser Gruppierungsebene das Knotenlayout optimiert ist, oder ob Knoten überlappen.

Rümpfe kollabiert

Wenn Sie diese Option wählen, werden die einzelnen Gruppen zunächst kollabiert dargestellt, d. h. es sind nur die Gruppentitel sichtbar, aber nicht die Vorgänge.

Änderungen erlaubt

Wenn Sie dieses Kontrollkästchen aktivieren, kann der Anwender expandierte Gruppen kollabieren oder kollabierte Gruppen wieder expandieren. Der Anwender kann Gruppen durch einen Doppelklick auf die Gruppenüberschrift im Tabellenteil oder durch einen einfachen Klick auf das Minus- bzw. Plus-Zeichen neben der Gruppenüberschrift oder über das Kontextmenü für Gruppen kollabieren bzw. expandieren.

Summenbalken

Wenn Sie dieses Kontrollkästchen aktivieren, werden Summenbalken angezeigt. Um Summenbalken ebenenweise zu definieren, müssen Sie einen Layer mit einer entsprechenden Filterbedingung (<Summenbalken-Ebene> = ...) definieren.

Vertikal Verschieben über den Tabellenbereich

Wenn diese Option ausgewählt ist, können Sie Gruppen im Tabellenbereich mit der Maus verschieben und damit ihre Reihenfolge ändern.

Automatisches Kollabieren von Gruppen

Wenn Sie diese Option wählen, werden beim interaktiven Verschieben eines Knotens alle Gruppen ausser der gerade berührten kollabiert.

Wiederherstellen von automatisch kollabierten Gruppen

Wenn Sie diese Option wählen, werden beim interaktiven Verschieben eines Knotens die zuvor automatisch kollabierten wiederhergestellt.

Automatisches Expandieren von Gruppen

Wenn Sie diese Option wählen, wird beim interaktiven Verschieben eines Knotens die Zielgruppe automatisch expandiert.

Wiederherstellen von automatisch expandierten Gruppen

Wenn Sie diese Option wählen, werden beim interaktiven Verschieben eines Knotens die zuvor automatisch expandierten wiederhergestellt.

Vertikal Verschieben über den Diagrammbereich

Wenn Sie diese Option auswählen, können Sie Gruppen im Diagrammbereich mit der Maus verschieben und damit die Reihenfolge der Gruppen ändern.

Seitenumbruch nach jeder Gruppe

Nach Klick auf  stehen folgende Möglichkeiten zur Auswahl:

- **Kein**: es wird kein Seitenumbruch ausgeführt
- **bei voller Seite**: wenn eine Gruppe durch einen Seitenumbruch getrennt würde, wird der Seitenumbruch bereits nach der vorhergehenden Gruppe durchgeführt
- **nach jeder Gruppe**: nach jeder Gruppe wird ein Seitenumbruch ausgeführt

Diese Optionen können auch über die Eigenschaft **VcGroupLevelLayout.-PageBreakMode** gesetzt werden.

> Knoten

Die folgenden Einstellungen beschreiben die Optionen, die Sie für ungruppierte Knoten oder Knoten in Gruppen auswählen können. Dies betrifft vor allem die Sortierung sowie das Aussehen der Knotenzeilen.

Hinweis: Beachten Sie bitte, dass die Einstellungen nur für die Sortierung der Vorgänge beim Öffnen des Diagramms gelten. Wenn Sie die Vorgänge danach neu sortieren möchten, verwenden Sie dazu bitte die VcGantt-Methode **SortNodes**.

Sortieren nach 1-3

Legen Sie hier fest, nach welchen Datenfeldern die Vorgänge beim Öffnen des Diagramms sortiert werden. Sie können bis zu drei Datenfelder als Sortierungskriterien auswählen. Für jedes dieser Sortierungskriterien können Sie auswählen, ob absteigend oder aufsteigend sortiert werden soll

(**Sortierreihenfolge 1 bis 3**). Bei Angabe eines Feldes für **gruppiert nach** erfolgt die Sortierung innerhalb jeder Gruppe.

Muster

Wenn Sie auf ... klicken, öffnet sich das Dialogfeld **Musterattribute**. Hier können Sie ein Hintergrundmuster und zwei Musterfarben für die Knotenzeile festlegen sowie über die Schaltfläche  eine datenabhängige Zuordnung der jeweiligen Eigenschaft vereinbaren.

Kalendergitter sichtbar

Bestimmen Sie, ob ein Kalendergitter angezeigt wird.

Trennlinien sichtbar

Bestimmen Sie, ob eine Trennlinie angezeigt wird.

Trennlinien oben

Wenn Sie diese Option wählen, wird die Trennlinie oberhalb eines Knotens gezogen (statt unterhalb).

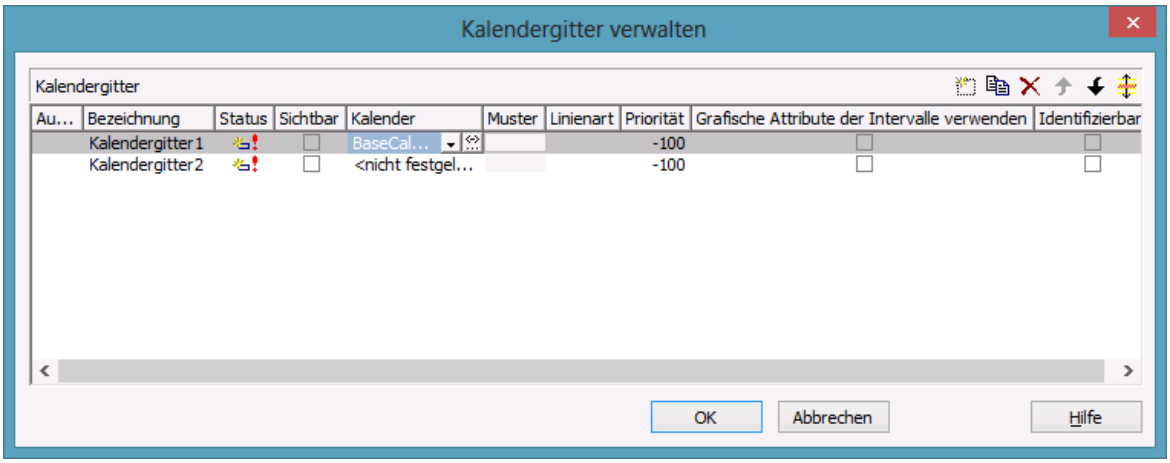
Trennlinie

Sie können das Aussehen der Trennlinien bearbeiten, indem Sie auf die **Bearbeiten**-Schaltfläche klicken und so das Dialogfeld **Linie bearbeiten** öffnen.

Trennlinienschrittweite

Hier können Sie festlegen, nach wie vielen Vorgängen jeweils eine Trennlinie gezogen werden soll.

4.21 Dialogfeld "Kalendergitter verwalten"



Sie gelangen in diesen Dialog, indem Sie im Dialog **Gruppierung** im Bereich **Gruppenweise** im Feld **Kalendergitter** auf ··· klicken.

Mithilfe der Schaltflächen können Sie Kalendergitter hinzufügen, kopieren oder löschen.

Mit den Pfeilschaltflächen können Sie ein Kalendergitter nach unten oder oben verschieben, während die Schaltfläche es ermöglicht, die ausgewählte Eigenschaft allen angezeigten Kalendergittern zuzuweisen.

Folgende Einstellungen können für die Kalendergitter vorgenommen werden:

Ausgewählt

Durch Klicken in dieses Feld können Sie dieses Kalendergitter für die Gruppierungsebene auswählen. Ein roter Pfeil zeigt die Auswahl an.

Bezeichnung

Vergeben Sie hier einen Namen für das Kalendergitter.

Status

In dieser Spalte wird jedes Kalendergitter gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Sichtbar

Aktivieren Sie dieses Kontrollkästchen für jedes Kalendergitter, das angezeigt werden soll.

Kalender

Ein hier ausgewählter Kalender wird auf alle Gruppen dieser Ebene angewendet. Wenn Sie keinen Kalender auswählen, wird der Kalender der Ebene verwendet, der das Kalendergitter zugeordnet ist.

Muster

Legen Sie hier die Hintergrundfarbe und -muster des Kalendergitters 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.

Linienart

Wenn Sie auf die Schaltfläche dieses Feldes (...) klicken, öffnet sich das Dialogfeld **Linienattribute des Kalendergitters**. Hier können Sie Farbe, Typ und Dicke der Randlinien der vertikalen Linien festlegen.

Priorität

Hier können Sie die Priorität jedes Kalendergitters in Relation zu den anderen Gittern und Layern festlegen (> 0: vor den Layern, < 0: hinter den Layern)

Grafische Attribute der Intervalle verwenden

Legen Sie fest, ob die für die Intervalle eingestellten grafischen Attribute dargestellt werden sollen.

Identifizierbar

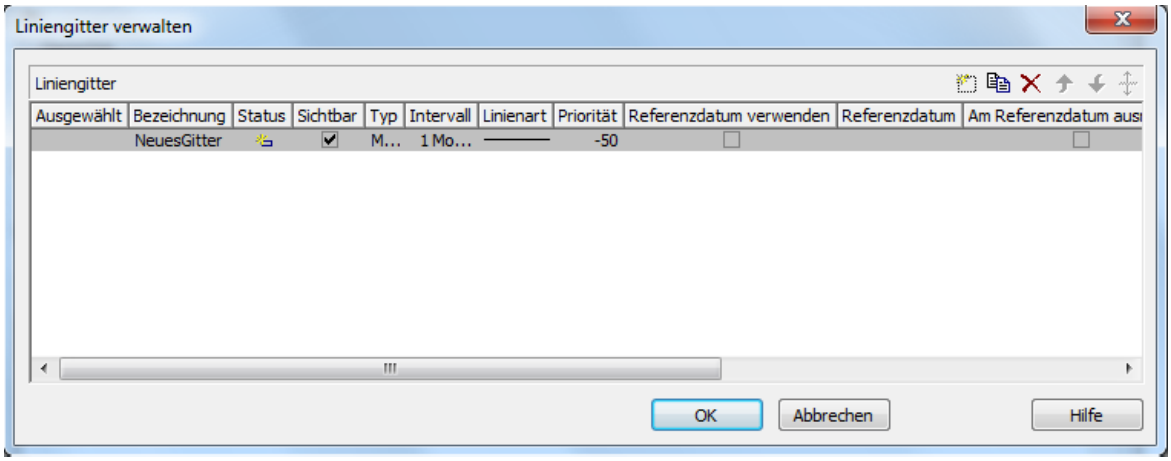
Hier können Sie festlegen, ob das Kalendergitter mittels der VcGantt-Methode **IdentifyObjectAt** identifizierbar ist. So kann z.B. ein Tooltip-Text nur bei einem identifizierbaren Gitter erscheinen; Analoges gilt für das Erscheinen des Kontext-Menüs bei einem Rechtsklick.

Für einen Tooltip-Text muss zudem das entsprechende Intervall identifiziert werden: s. **Kalendergitter**-Bereich des Dialogs **Zeitskalenabschnitt bearbeiten**.

Einrastziele Start/Ende

Wenn Sie diese Optionen wählen, definiert das Kalendergitter seine relevanten Positionen als "Einrastziele" für zu verschiebende Knoten/Layer.

4.22 Dialogfeld "Liniengitter verwalten"



Sie gelangen in diesen Dialog, indem Sie im Dialog **Gruppierung** im Bereich **Gruppenweise** im Feld **Liniengitter** auf ... klicken.

Mithilfe der Schaltflächen können Sie Liniengitter hinzufügen, kopieren oder löschen.

Mit den Pfeilschaltflächen können Sie ein Liniengitter nach unten oder oben verschieben, während die Schaltfläche es ermöglicht, die ausgewählte Eigenschaft allen angezeigten Liniengittern zuzuweisen. Folgende Einstellungen können für die Liniengitter vorgenommen werden:

Ausgewählt

Durch Klicken in dieses Feld können Sie dieses Liniengitter für die Gruppierungsebene auswählen. Ein roter Pfeil zeigt die Auswahl an.

Bezeichnung

Vergeben Sie hier einen Namen für das Liniengitter.

Status

In dieser Spalte wird jedes Liniengitter gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Sichtbar

Aktivieren Sie dieses Kontrollkästchen für jedes Liniengitter, das angezeigt werden soll.

Typ

Hier können Sie die Grundeinheit jedes Liniengitters festlegen, z. B. Tage, Wochen, etc.

Intervall

Hier können Sie das Intervall zwischen den einzelnen Linien in ganzzahligen Vielfachen der Grundeinheit des Liniengitters festlegen.

Linienart

Wenn Sie auf die Schaltfläche dieses Feldes (···) klicken, öffnet sich das Dialogfeld **Linienattribute des Liniengitters**. Hier können Sie Farbe, Typ und Dicke der Randlinien der Streifen festlegen.

Priorität

Hier können Sie die Priorität jedes Liniengitters in Relation zu anderen Gittern und zu den Layern festlegen (> 0: vor den Layern, < 0: hinter den Layern).

Referenzdatum verwenden

Aktivieren Sie dieses Kontrollkästchen, falls ggf. der Ursprung des Liniengitters auf das Referenzdatum gelegt werden soll.

Referenzdatum

Wählen Sie ggf. das Referenzdatum mit Hilfe des Datumsdialogs aus.

Am Referenzdatum ausrichten

Aktivieren Sie dieses Kontrollkästchen, wenn die die Linien des Liniengitters am Referenzdatum ausgerichtet werden sollen, d.h. die Position kann vom Anfang einer Zeiteinheit abweichen, z.B. um13:17 jedes Tages.

Ist diese Option nicht gewählt, werden die Linien eines Liniengitters standardmäßig am Anfang der jeweiligen Zeiteinheit positioniert, z.B. immer um 00:00 Uhr jedes Tages.

Linienformat

Hier können Sie entweder aus der Drop-Down-Liste ein bestehendes Linienformat für das Liniengitter auswählen oder im Dialog **Linienformate vewalten** (zu erreichen über ...) ein neues Kalendergitter anlegen. Nähere Informationen zu diesem Dialog finden Sie im Kapitel **Dialogfeld Linienformate verwalten**.

Drehen

Wenn Sie diese Option auswählen, kann die Beschriftungen an den Linien des Liniengitters um 90 Grad (vertikal) gedreht werden.

Ausrichtung

Hier können Sie die horizontale Ausrichtung für die Beschriftungen der Linien festlegen.

Oben

Wenn Sie diese Option wählen, erscheint die Beschriftung der Linien im Liniengitter am oberen Rand des Gantt-Graphen.

Mitte

Wenn Sie diese Option wählen, erscheint die Beschriftung der Linien im Liniengitter in der Mitte des Gantt-Graphen.

Unten

Wenn Sie diese Option wählen, erscheint die Beschriftung der Linien im Liniengitter am unteren Rand des Gantt -Graphen.

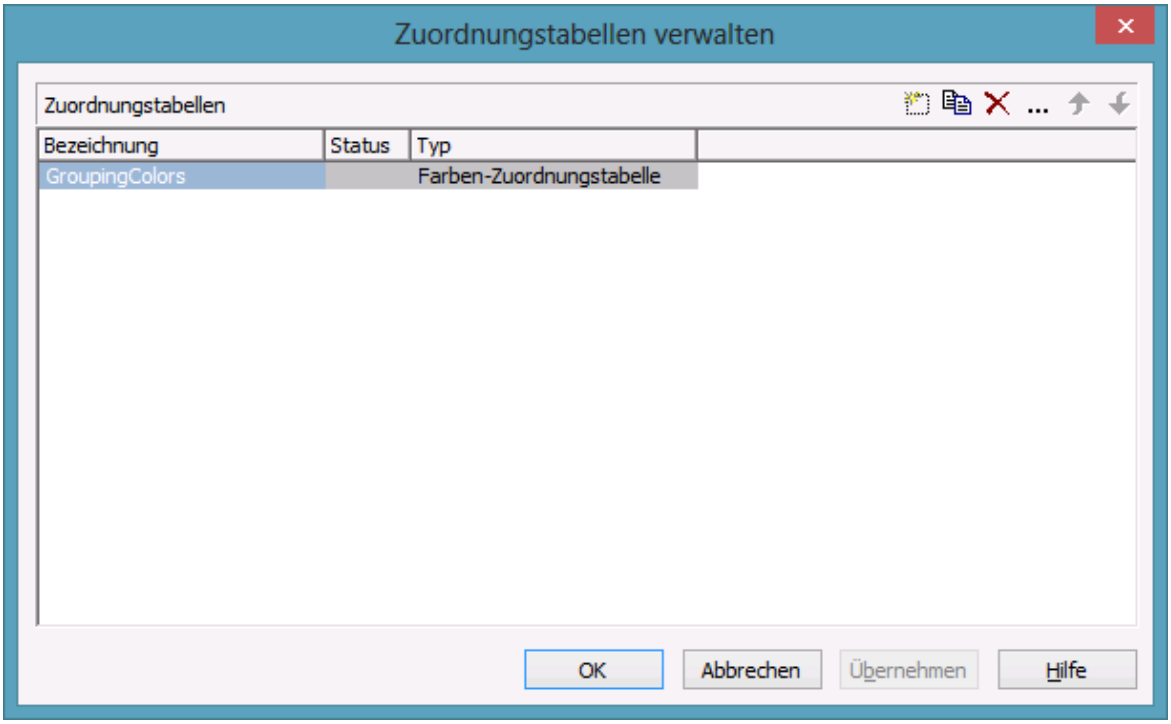
Sommerzeit beachten

Wenn Sie diese Option wählen, wird die Sommerzeit berücksichtigt.

Einrastziel

Wenn Sie diese Option wählen, definiert das Liniengitter seine relevanten Positionen als "Einrastziele" für zu verschiebende Knoten/Layer.

4.23 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

- Grafikdateien-Zuordnungstabelle
- Schriften-Zuordnungstabelle
- Millimeter-Zuordnungstabelle
- Nummern-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.

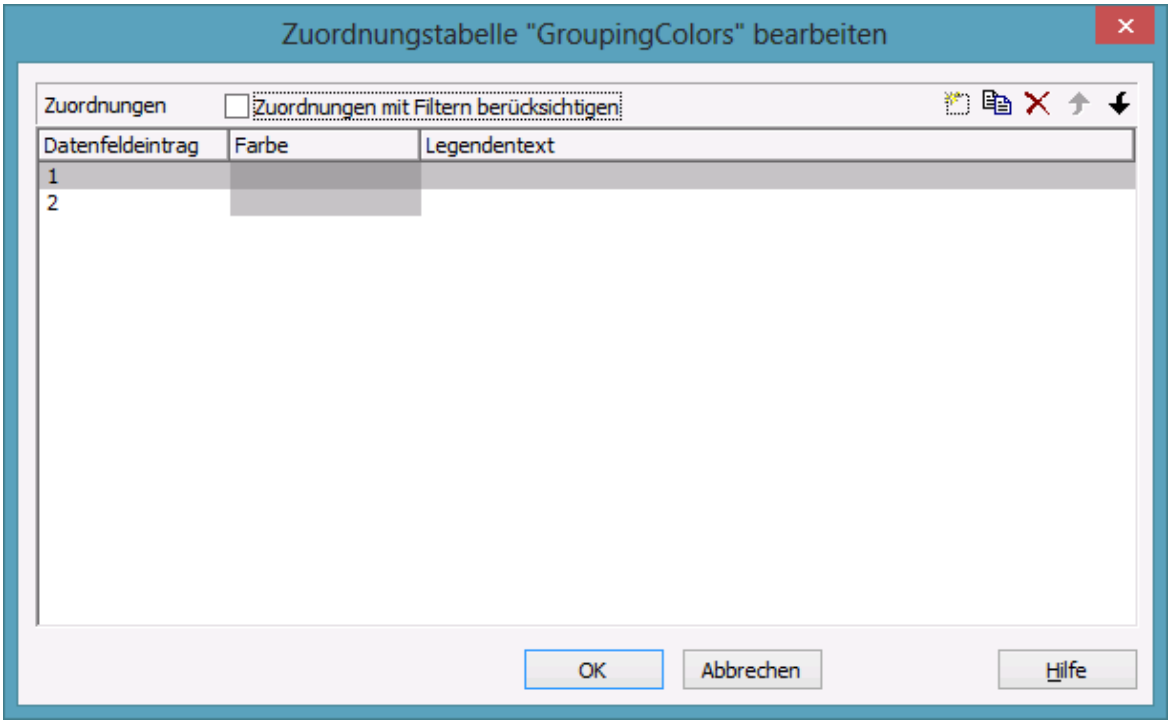
Zuordnungstabelle bearbeiten

 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.

4.24 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. ein Muster und den Legendentext vereinbaren können.

Farbe/Muster

Um die Farbe oder das Muster 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. des Musters gelangen. Im Farben-Dialog stehen auch transparente Farben zur Verfügung.

Legendentext

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.

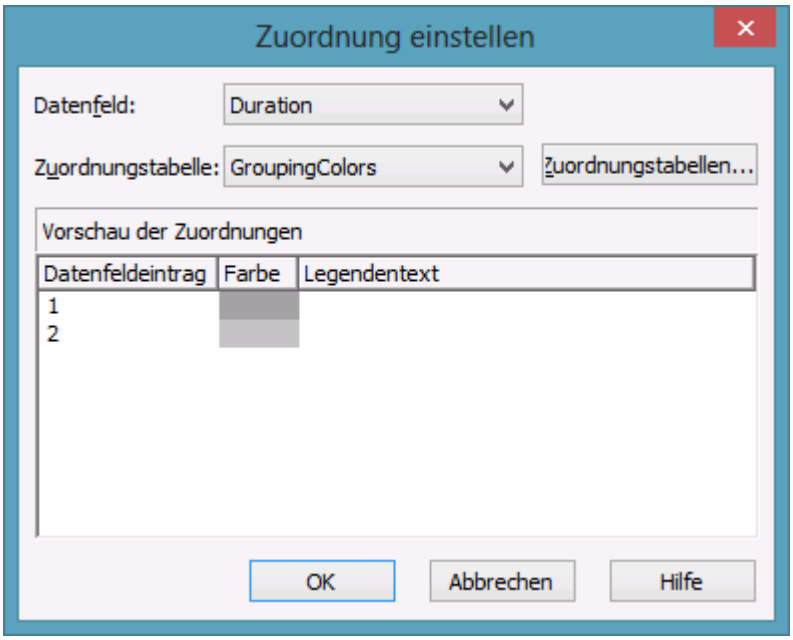
Zuordnung löschen

 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.

4.25 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 in Abhängigkeit ihresTyps den einzelnen Datenfeldeinträgen die entsprechenden Attribute zuordnet.

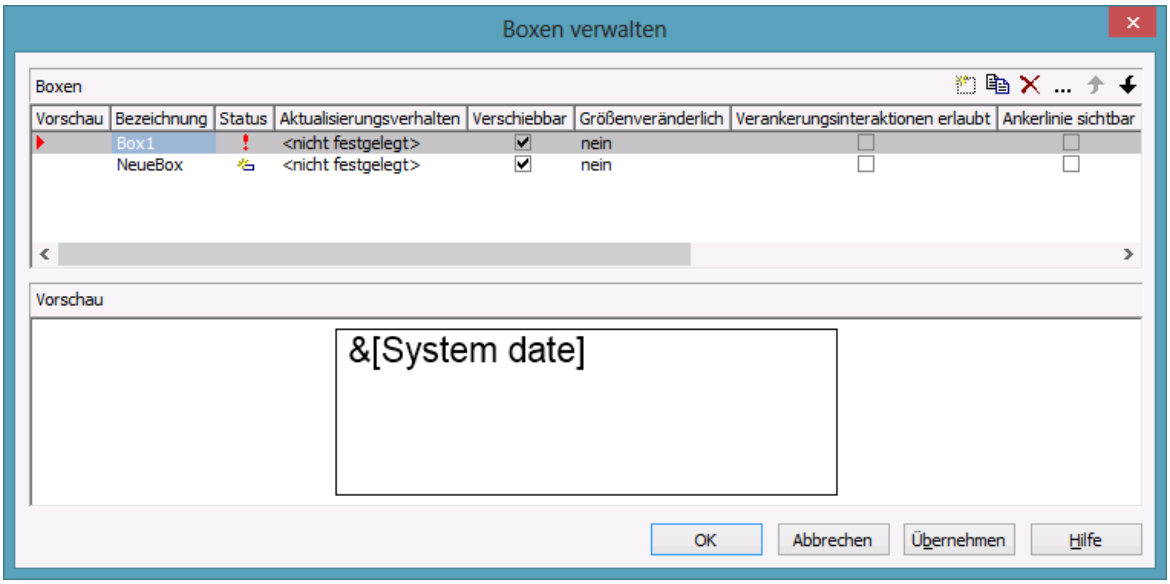
Zuordnungstabellen

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 sowie die diesen zugeordneten Attribute.

4.26 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.

Größenveränderlich

Legen Sie hier fest, ob die Größe einer Box interaktiv geändert werden kann, wobei sie wählen können, ob nur die Höhe, nur die Breite oder Höhe und Breite veränderbar sein sollen. Sobald Sie mit der Maus auf den Rand der Box zeigen, wird der Mauszeiger zu einem Doppelpfeil. Vergrößern oder verkleinern Sie nun die Box, indem Sie den Rahmen mit gedrückter linker Maustaste in die gewünschte Richtung ziehen.

Tipp: Wenn Sie **Breite und Höhe** ausgewählt haben, können Sie den Mauszeiger auch auf eine Ecke der Box positionieren und Höhe und Breite gleichzeitig ändern.

Verankerungsinteraktionen erlaubt

Legen Sie fest, ob Ver- oder Entankerungsinteraktionen (per Maus oder Kontextmenü) durchgeführt werden können. Dadurch wird es dem Anwender ermöglicht, Boxen an Knoten zu verankern bzw. sie wieder zu lösen.

Sie können eine Box entweder interaktiv (mit der Maus + gedrückter Umschalt-Taste oder über das Kontextmenü) oder über die API mit einem Knoten verankern.

- **Verankern mit der Maus:** Zeigen Sie mit der Maus auf die gewünschte Box und drücken Sie die Umschalt-Taste. Ein kleiner Anker erscheint. Nun ziehen Sie mit gedrückter linker Maustaste eine Linie von der Box bis zu dem gewünschten Knoten und lassen die Maustaste wieder los. Die Box ist nun mit dem Knoten verankert und bei aktivierter Option **Ankerlinie sichtbar** erscheint eine Verbindungslinie. Um die Verankerung wieder zu lösen, verfahren sie genauso.
- **Verankern über das Kontextmenü:** Markieren Sie den Knoten, mit dem die Box verankert werden soll und wählen dann aus dem Kontextmenü der Box den Befehl **Box am markierten Knoten verankern**. Sollte das Kontextmenü nicht erscheinen, muss auf der Eigenschaftenseite **Allgemeines** die Option **Kontextmenü für Boxen anzeigen** aktiviert werden.

Zum Lösen der Box von dem Knoten wählen Sie aus dem Kontextmenü den entsprechenden Befehl.

Wenn Sie die Box mit einem anderen Knoten verankern möchten, verfahren Sie genauso wie oben beschrieben

- **Verankern über API:** Bitte lesen Sie dazu in der API-Referenz die Beschreibung der Eigenschaft **AnchoringInteractionsAllowed** sowie der Methode **AnchorToNode** des Objektes **VcBox**

Eine verankerte Box kann weiterhin interaktiv verschoben werden (Voraussetzung: die Option **Verschiebbar** ist aktiviert).

Wird ein Knoten mit einer verankerten Box verschoben, so wird die Box entsprechend mitverschoben. Wird der Knoten kollabiert, wird auch die Box kollabiert d.h. sie ist nicht mehr sichtbar. Sobald der Knoten expandiert wird, ist auch die Box wieder sichtbar.

Bei interaktiver Ver/Entankerung bleibt die Position der Box auf dem Bildschirm erhalten - die zu Grunde liegenden Offset-Werte werden entsprechend der Referenzpunkte (Origin, ReferencePoint) umgerechnet. Wenn sich also eine Textbox mit einem bestimmten Offset z.B. auf das Diagramm oben links (Origin) bezieht und dann mit einem Knoten verankert wird, wird automatisch ein Offset zum Knoten oben links berechnet und dieser Offset führt dazu, dass die Position auf dem Bildschirm beibehalten wird. Wird die Box vom Knoten gelöst, erfolgt eine Rückberechnung.

An der API geschieht dies genauso bei **AnchorToNode**, nicht aber beim Setzen der Eigenschaft **NodeID**.

Ankerlinie sichtbar

Legen Sie hier fest, ob bei Verankerungen die eingestellten Referenzpunkte beim Knoten und bei der Box durch eine Linie verbunden werden sollen.

Knoten-ID

Hier können Sie eine beliebige Zeichenkette eingeben, die dann als Knoten-ID interpretiert wird und dazu dient, den Knoten zu identifizieren, an dem die

jeweilige Box verankert werden soll. Eine leere Zeichenkette bedeutet, dass die Box nicht verankert ist.

Hinweis: Es wird nicht überprüft, ob die Syntax korrekt ist oder ob der Knoten existiert. Wenn der Knoten nicht existiert, wird nichts verankert.

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.

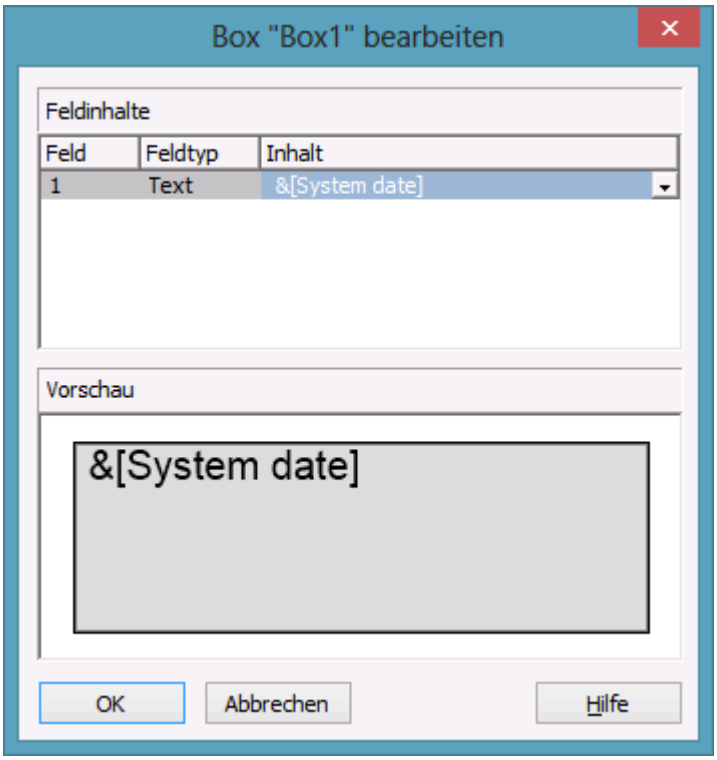
Box bearbeiten

 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.

4.27 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).

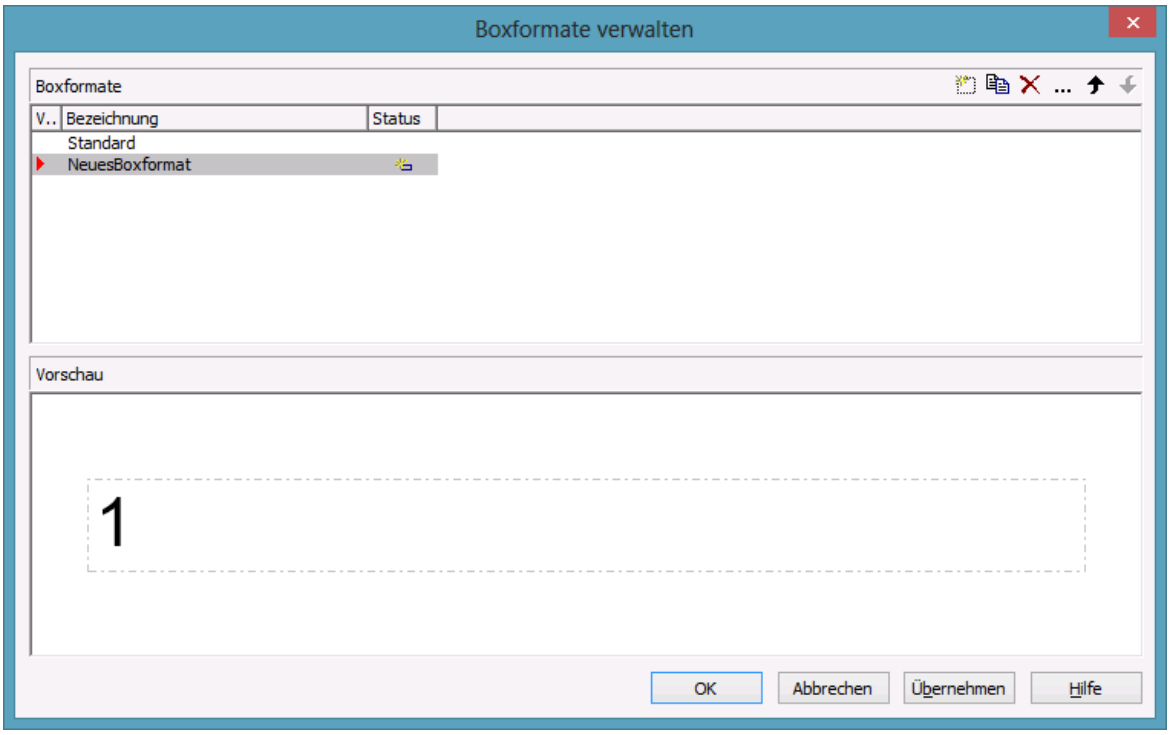
Inhalt

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.

Mögliche Grafikformate: WMF, JPG, BMP, GIF, PCX, PNG, TIF.

4.28 Dialogfeld "Boxformate verwalten"



Sie gelangen in dieses Dialogfeld über die Eigenschaftenseite **Objekte**.

Vorschau

Das in der Vorschauspalte markierte Format wird im Vorschaufenster 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.

Boxformat hinzufügen

Ein neues Format mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

Boxformat kopieren

 Das markierte Format wird kopiert.

Boxformat 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.

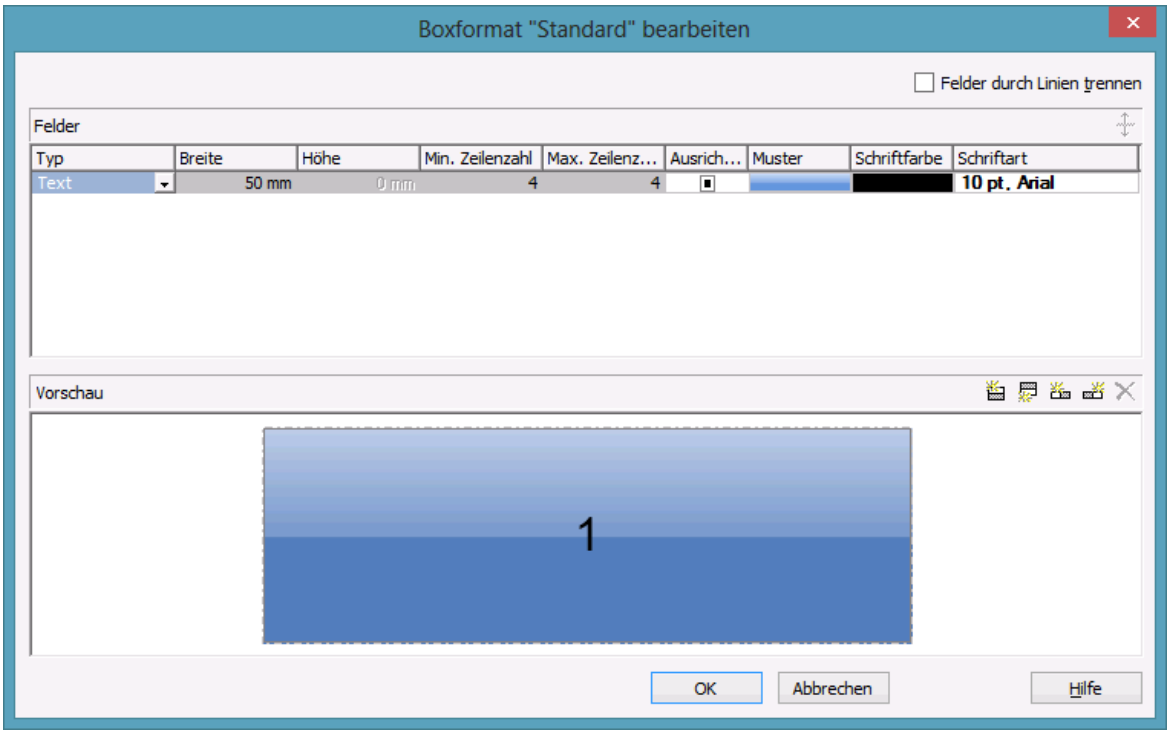
Boxformat bearbeiten

 Sie gelangen in das Dialogfeld **Boxformat bearbeiten**.

Boxformat eine Zeile nach oben / unten

 Mit Hilfe dieser Schaltflächen können Sie das markierte Format in der Tabelle eine Zeile nach oben bzw. unten schieben.

4.29 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

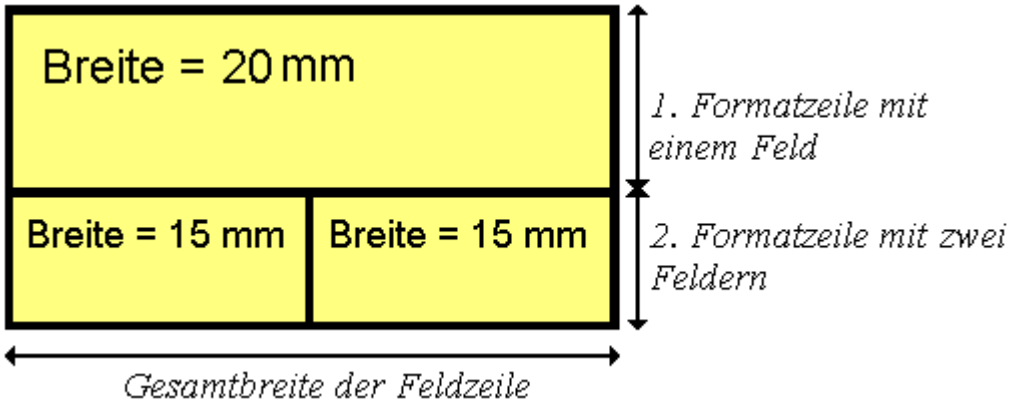
Aktivieren Sie dieses Kontrollkästchen, wenn die Felder der Box durch Linien getrennt werden sollen.

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

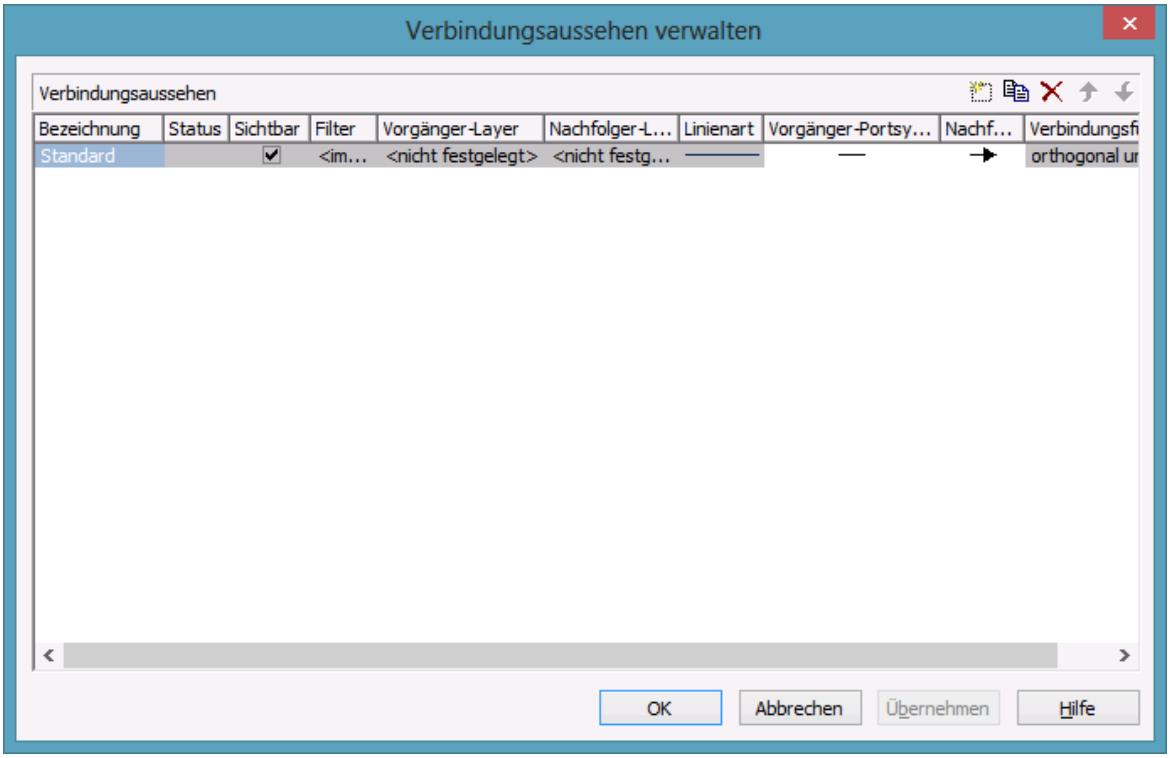
 Klicken Sie auf diese Schaltfläche, damit die selektierte Eigenschaft in alle Felder übernommen wird.

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.

4.30 Dialogfeld "Verbindungsaussehen verwalten"



Sie gelangen in dieses Dialogfeld, über die Eigenschaftenseite **Objekte**.

Bezeichnung

In dieser Spalte werden die Namen der verfügbaren Verbindungsaussehen angezeigt. Die Namen sind editierbar.

Diese Option kann auch über die Eigenschaft **LinkAppearanceName** festgelegt werden.

Status

In der Spalte **Status** wird jedes Verbindungsaussehen 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.

Vorgänger-Layer

Wählen Sie hier aus, an welchen Layer des Vorgängerknotens die Verbindung angeknüpft werden soll. Falls der gewählte Layer einem Knoten nicht zugewiesen wurde, wird die Verbindung an den ersten sichtbaren Layer des jeweiligen Knotens geknüpft.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.PredecessorLayerName** festgelegt werden.

Nachfolger-Layer

Wählen Sie hier aus, an welchen Layer des Nachfolgerknotens die Verbindung angeknüpft werden soll. Falls der gewählte Layer einem Knoten nicht zugewiesen wurde, wird die Verbindung an den ersten sichtbaren Layer des jeweiligen Knotens geknüpft.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.-SuccessorLayerName** 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

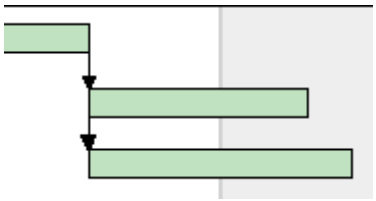
Wählen Sie hier für jedes Verbindungsaussehen ein Portsymbol, das die Einmündung zum Nachfolgerknoten kennzeichnet.

Diese Option kann auch über die Eigenschaft **VcLinkAppearance.-SuccessorPortSymbol** festgelegt werden.

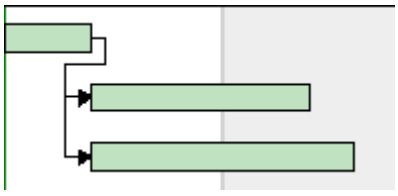
Verbindungsführung

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.

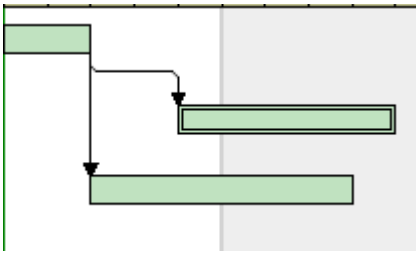
Sie können die Art der Verbindungsführung auch über die **VcLinkAppearance**-Eigenschaft **RoutingType** setzen.



Geradliniger Verbindungstyp



Orthogonaler Verbindungstyp



Orthogonal unterscheidbarer Verbindungstyp

Verbindungsaussehen hinzufügen

 Ein neues Verbindungsaussehen mit einem Standardnamen wird angelegt. Diesen Namen können Sie editieren, indem Sie darauf doppelklicken und ihn dann verändern.

Verbindungsaussehen kopieren

 Das markierte Verbindungsaussehen wird kopiert.

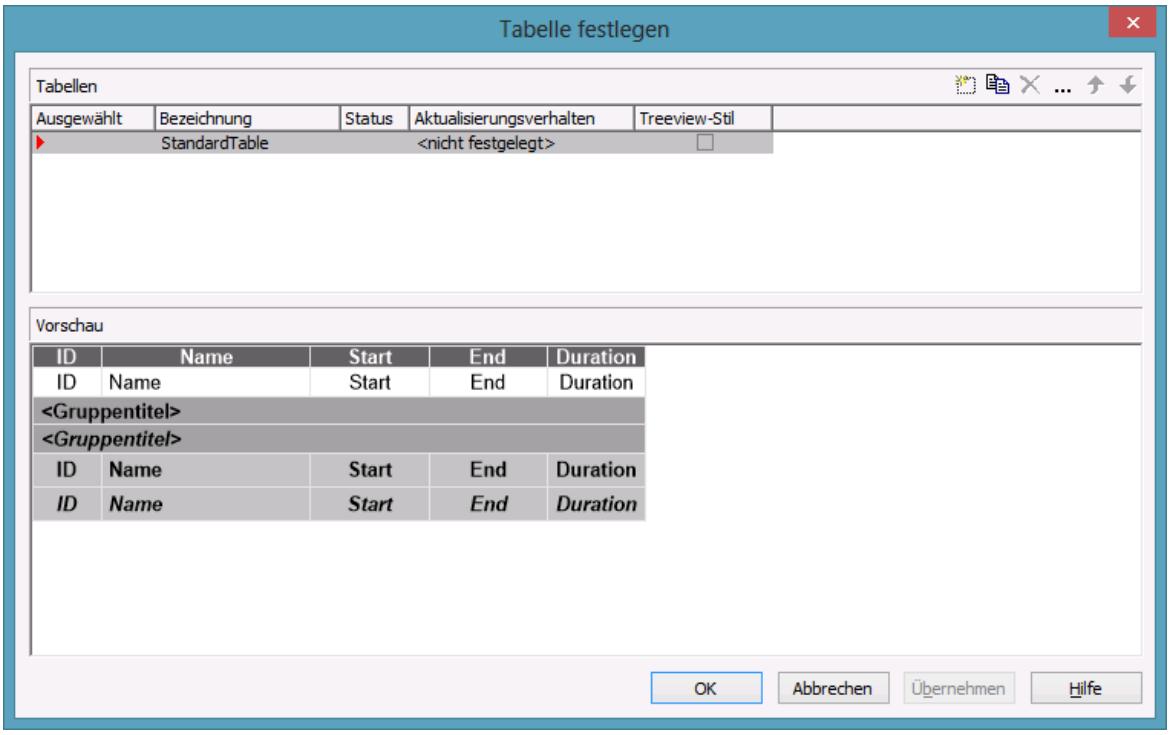
Verbindungsaussehen löschen

 Das Verbindungsaussehen, das Sie in der Liste markiert haben, wird gelöscht. Es können nur Verbindungsaussehen gelöscht werden, die zur Zeit nicht benutzt werden.

Verbindungsaussehen eine Zeile nach oben/unten

  Mit Hilfe dieser Schaltflächen können Sie das markierte Verbindungsaussehen eine Zeile nach oben/unten schieben.

4.31 Dialogfeld "Tabelle festlegen"



In diesem Dialog können Sie die Tabellen festlegen und verwalten.

Vorschau

Die in dieser Spalte mit einem roten Pfeil markierte Tabelle wird im Vorschaufenster in der unteren Hälfte des oben dargestellten Dialogs angezeigt. Sie ist gleichzeitig die Tabelle, die aktuell bearbeitet wird.

Bezeichnung

In dieser Spalte werden die Namen aller verfügbaren Tabellen aufgeführt. Die Namen sind editierbar.

Status

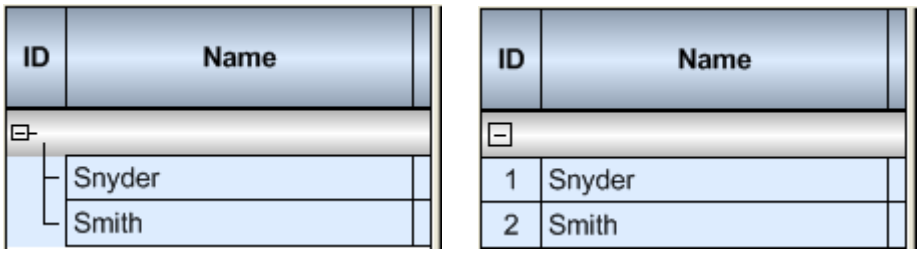
In dieser Spalte wird jede Tabelle gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Aktualisierungsverhalten

Wählen Sie hier das gewünschte Aktualisierungsverhalten aus. Bei der Einstellung <nicht festgelegt>, gilt das Aktualisierungsverhalten, das im Dialog **Aktualisierungsverhalten bearbeiten** für Tabellen festgelegt wurde.

Treeview-Stil

Wenn dieses Kontrollkästchen aktiviert ist, werden Knoten im TreeView-Stil angeordnet, bei dem die logische Baumstruktur durch Linen gezeichnet wird. Ebenen werden in beiden Darstellungen mit Plus- oder Minus-Zeichen versehen.

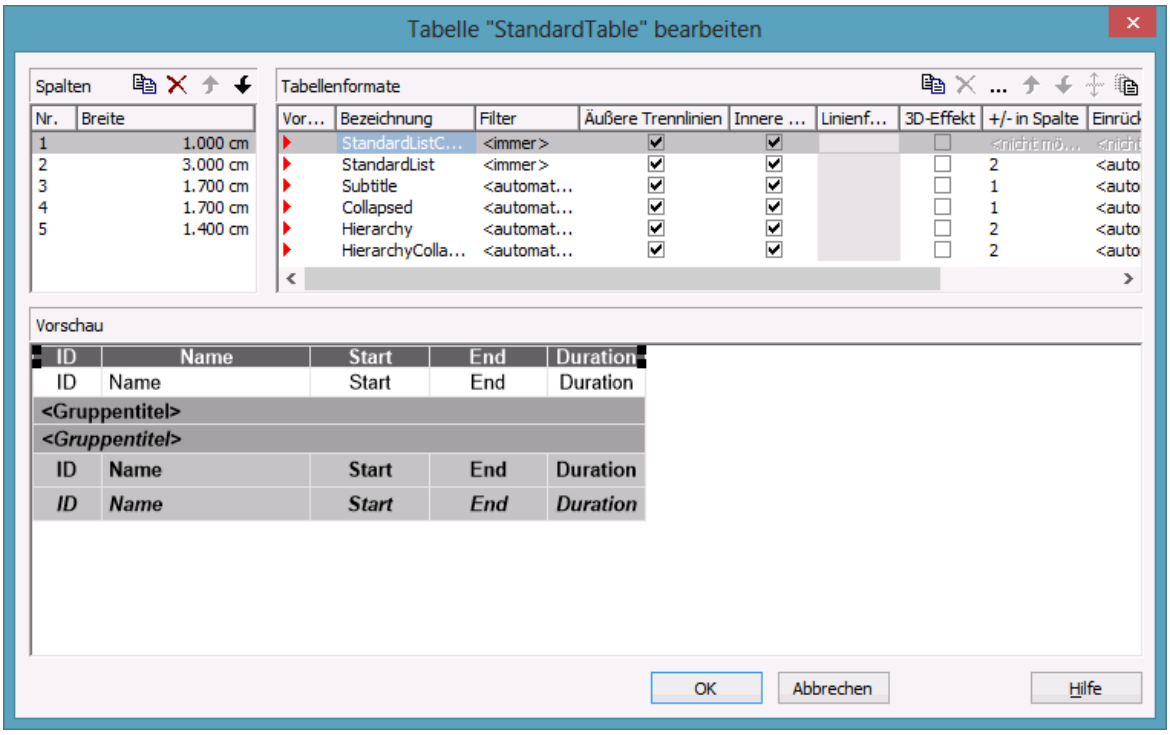


Bilder oben: Gruppierung mit und ohne Treeview-Stil

Tabelle hinzufügen / kopieren / löschen / nach oben / unten

 Mit Hilfe dieser Schaltflächen können Sie die markierte Tabelle hinzufügen, kopieren, löschen und in der Liste nach oben oder unten verschieben. Letzteres kann z.B. der Sortierung und damit einer besseren Übersicht dienen, hat aber keinerlei Funktion hinsichtlich der Priorität.

4.32 Dialogfeld "Tabelle bearbeiten"



In diesem Dialog können Sie eine Tabelle bearbeiten.

Spalten

In der **Spalten**-Liste werden die **Nr.** und die **Breite** jeder Tabellenspalte aufgelistet. Die Breite kann im Bereich von 0 bis 10 cm in Schritten von 1 mm variiert werden.

Maximal 100 Spalten sind möglich. Die Reihenfolge der Tabellenspalten in der **Spalten**-Liste entspricht der Reihenfolge der Spalten in der Grafik.

 Mit Hilfe der Schaltflächen oberhalb der **Spalten**-Liste können Sie Tabellenspalten kopieren, löschen oder ihre Position verändern.

Tabellenformate

In der **Tabellenformate**-Liste können Sie verschiedene Zeilentypen festlegen:

- **Vorschau:** Jedes Tabellenformat, das in dieser Liste mit einem roten Dreieck markiert ist, wird in der Vorschau dargestellt.
- **Bezeichnung:** Jedes Tabellenformat hat standardmäßig eine Bezeichnung. **StandardListCaption** ist das Tabellenformat für die Tabellenüberschrift. Die Bezeichnung der Tabellenformate ist editierbar

bei den Tabellenformaten **ListFormat2** und **ListFormat3** sowie bei allen selbst definierten Tabellenformaten.

- **Filter:** Mit jedem Tabellenformat ist ein Filter verbunden, der die Vorgänge auswählt, für die das entsprechende Tabellenformat verwendet werden soll. Wenn mehrere Filter dieser Liste auf einen Vorgang zutreffen, wird das Tabellenformat mit der jeweils höchsten Priorität verwendet. Die Reihenfolge der Filter in der Auflistung des Fensters von unten nach oben bestimmt ihre Priorität. Der obere Filter besitzt somit die niedrigste Priorität. Es gibt folgende vordefinierte Filter: <Anschlussknoten>: das Format wird nur auf Knoten angewendet, die an die selektierten Knoten angeschlossen sind. <Nie>: der Filter kommt nie zur Anwendung und kann praktischerweise als Kopiervorlage für neue, selbstdefinierte Filter dienen. <Automatisch>: das Format wird auf Knoten angewendet, die einer zu spezifizierenden Gruppierungsebene angehören. <Immer>: das Format ist das Default-Format und wird auf alle Knoten angewendet, die von keinem anderen Filter abgefangen werden. Der Filter <immer> ist daher nicht löschbar. Es ist sinnvoll, diesen Filter ganz nach oben zu setzten.
- **Äußere/Innere Trennlinien:** Legen Sie für jedes Tabellenformat fest, ob die Felder darin durch äußere/innere Trennlinien voneinander getrennt werden sollen.
- **Linienfarbe:** Hier können Sie die Farbe der Trennlinien für jedes Format individuell setzen.
- **3D-Effekt:** Legen Sie hier fest, ob die einzelnen Tabellenfelder durch einen 3D-Effekt hervorgehoben werden sollen.
- **+/- in Spalte:**Legen Sie hier fest, ob in einer Spalte +/- zum Ein- bzw. Ausblenden untergeordneter Zeilen angezeigt werden soll. Wählen Sie aus der Liste die gewünschte Spaltennummer.
- **Einrückungsspalte:**Legen Sie hier fest, welche Spalte einen Einzug bekommen soll. Dies wirkt sich nur aus, wenn der Zeile bzw. dem Knoten andere Zeilen/Knoten untergeordnet sind. Dann wird die erste untergeordnete Zeile eingerückt. Bei Auswahl von **automatisch** wird die Spalte eingerückt, in der +/- angezeigt wird.
- **Einrückungsbreite:** Geben Sie hier das Maß des Einzugs in mm an.

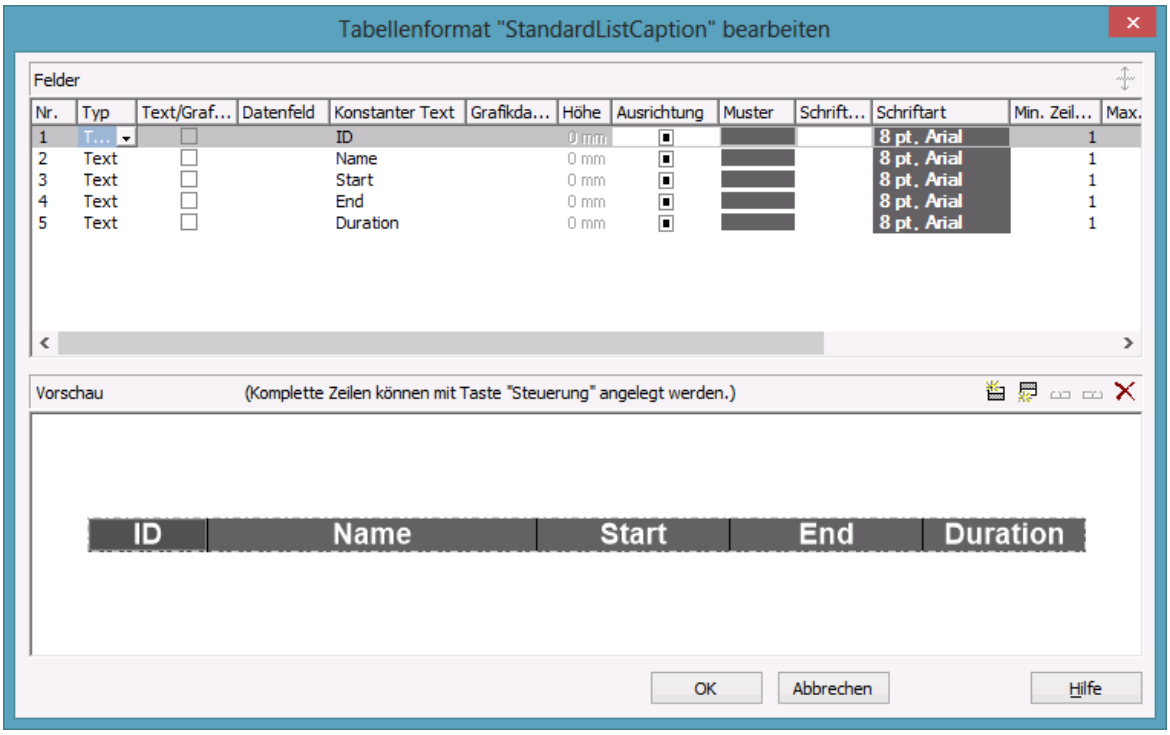
 Mit Hilfe dieser Schaltflächen oberhalb der **Tabellenformate**-Liste können Sie das markierte Tabellenformat kopieren oder löschen oder dafür den Dialog **Tabellenformat bearbeiten** aufrufen.

Hinweis: Beim Tabellenformat **StandardListCaption** (Tabellenüberschrift) sind die Attribute der Tabellenformate nicht über Zuordnungstabellen einstellbar.

 Mit Hilfe der Pfeil-Schaltflächen können Sie alle Tabellenformate mit Ausnahme der beiden ersten in der Liste positionieren.

 Wenn Sie die Eigenschaft **Äußere Trennlinien** oder **Innere Trennlinien** für ein Tabellenformat geändert haben und dann auf diese Schaltfläche klicken, wird diese Änderung auf alle anderen Tabellenformate übertragen.

4.33 Dialogfeld "Tabellenformat bearbeiten"



In diesem Dialog können Sie ein Tabellenformat (einen Zeilentyp) bearbeiten.

Nr.

Nummer des Tabellenformatfeldes: Diese Nummer ist nicht editierbar. Sie dient als Index, über den Sie über die API auf dieses Tabellenformatfeld zugreifen können.

Wenn Sie im Vorschaufenster ein neues Tabellenformatfeld anlegen, erhält dieses in der Liste zunächst statt einer Nummer ein Fragezeichen. Erst nachdem Sie den Dialog mit **OK** verlassen und wieder geöffnet haben, wird das Fragezeichen durch eine neue Nummer ersetzt.

Typ

Wählen Sie hier den Feldtyp:**Text**, **Grafik** oder **MultiState**. MultiState-Felder werden verwendet, um z.B. beim Anklicken eines Tabellenfeldes eine zyklische Abfolge von Zuständen wie auch der zugehörigen Datenfelder auszulösen.

Text/Grafik kombiniert

Wenn dieses Kontrollkästchen aktiviert ist, können in dem Tabellenformatfeld 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. Außer den Datenfeldern, die Sie in der Datendefinition vereinbart haben, gibt es folgende Optionen:

- <Gruppentitel>: der bei der jeweiligen Gruppierungsebene eingestellte Code
- <Zeilennummer>: fortlaufende Zeilennummer

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 ausgeben werden soll.

Grafikdateiname

Hier werden Name und Pfad der Grafikdatei angezeigt, die in dem gewählten Tabellenfeld dargestellt werden soll.

Wenn Sie auf ein **Grafikdateiname**-Feld klicken, erscheinen zwei Schaltflächen:

Wenn Sie auf die Schaltfläche drücken, erscheint der Windows-Dialog **Grafikdatei auswählen**. 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 VcGantt-Eigenschaft **FilePath** gesetzt ist. Wird sie dort nicht gefunden, wird sie zuerst im gerade aktiven Arbeitsverzeichnis der Applikation und dann im Installationsverzeichnis des VARCHART XGantt Steuerelementes gesucht.

Klicken Sie auf diese Schaltfläche (**Zuordnungen einstellen**), um unter Verwendung einer Zuordnungstabelle Grafiken abhängig vom Inhalt eines Datenfelds in einem Tabellenfeld 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 in der linken oberen Ecke durch die Diagramm-Hintergrundfarbe ersetzt; d. h. alle Pixel der Grafik in der Farbe des Eck-Pixels sind transparent.

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 99 mm.

Ausrichtung

Wählen Sie hier die Ausrichtung des Inhalts in dem markierten Feld (9 Möglichkeiten).

Muster

Legen Sie hier die standardmäßige Hintergrundfarbe und -muster des Tabellenformates fest. Durch Klick auf gelangen Sie in den Dialog **Musterattribute bearbeiten**. Hier können Sie durch Klick auf neben

dem jeweiligen Attribut ein Muster, eine Hintergrundfarbe sowie eine 2. Musterfarbe auswählen. Zusätzlich zu der vorgegebenen Farbpalette können Sie beliebige Farben definieren. Es stehen außerdem transparente Farben zur Verfügung.

Durch Klick auf die Schaltfläche  gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung des Musters und der Farben vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt ().

Schriftfarbe

Die Schriftfarbe des Feldes wird hier angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Die Farbzuordnungstabelle 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 wurde, erscheint nach Verlassen des Feldes statt  ein Pfeil auf der Schaltfläche ().

Schriftart

Die Schriftart des Feldes wird hier angezeigt. Wenn Sie auf dieses Feld klicken, erscheinen zwei Schaltflächen:

 Über diese Schaltfläche öffnen Sie den Windows-Dialog **Schriftart**.

 Über die zweite Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**. Hier können Sie eine datenabhängige Zuordnung der Schriftart vereinbaren. Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt ().

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.

Zeilenabstand

Wählen Sie hier den prozentualen Zeilenabstand.

Umbruch

Legen Sie fest, ob ggf. die Zeilen umgebrochen werden sollen.

Hor. Ränder (links/ rechts)/ Ver. Ränder (oben/ unten)

Legen Sie die Breite der Ränder der Tabellenformatfelder fest.

+/- in Spalte

Wählen Sie aus, ob in der Spalte +/- zum Ein- bzw. Ausblenden der weiteren Zeilen angezeigt werden soll

Indentierungsspalte

Legen Sie hier fest, ob die Spalte einen Einzug bekommen soll.

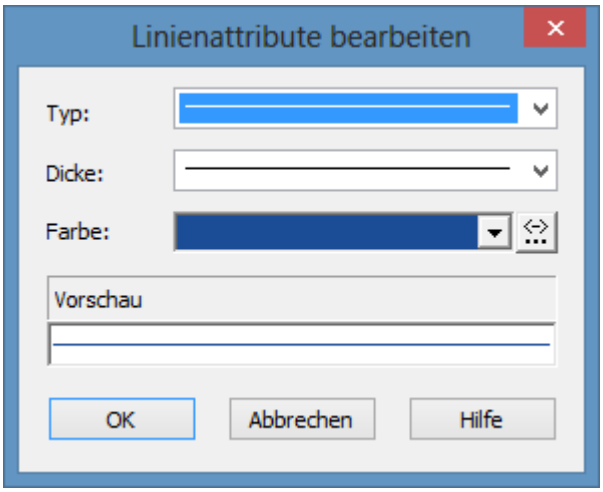
Vorschau

Hier werden die aktuellen Felder des Tabellenformats dargestellt. Wenn Sie im Vorschaufenster ein Feld anklicken, können Sie dessen Attribute in der **Felder**-Tabelle bearbeiten.

 Mit Hilfe der Schaltflächen oberhalb des Vorschaufensters können Sie neue Felder anlegen oder das markierte Feld löschen. Zum Löschen von Feldern können Sie auch die Entf-Taste benutzen.

Die vier ersten Schaltflächen, die zum Hinzufügen neuer Felder dienen, sind immer nur dann aktiviert, wenn an der markierten Stelle tatsächlich Felder hinzugefügt werden können. Dies hängt davon ab, wie viele Spalten für das betreffende Tabellenformat im übergeordneten Dialog **Tabelle bearbeiten** festgelegt worden sind, wie viele Felder also maximal nebeneinander in diesem Tabellenformat möglich sind.

4.34 Dialogfeld "Linienattribute bearbeiten"



Der Dialog zum Bearbeiten der Linienattribute, der jeweils über die Schaltfläche ... aufgerufen werden kann, steht zur Verfügung für Hierarchie und Gruppierung, für Kalendergitter, für das Balkenaussehen, für Kurvenfüllung und die numerischen Skalen beim Histogramm, für das Verbindungsaussehen, für Intervalle 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.

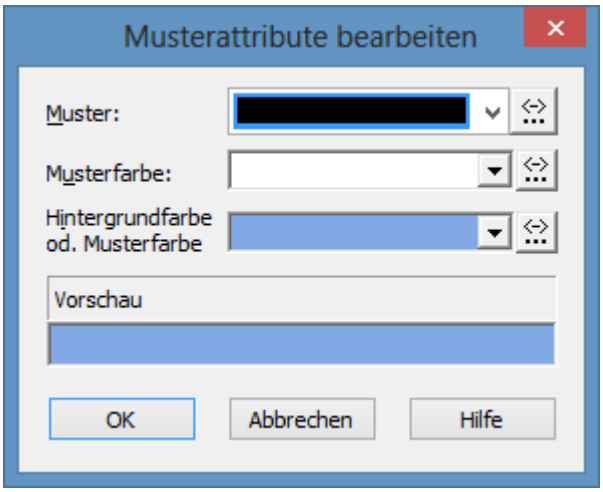
Über diese Schaltfläche gelangen Sie in den Dialog **Zuordnung einstellen**, in dem Sie eine datenabhängige Zuordnung der Linienfarbe vereinbaren können.

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.

4.35 Dialogfeld "Musterattribute bearbeiten"



Der Muster-Dialog, der über die Schaltfläche ... aufgerufen werden kann, steht zur Verfügung für die Kurvenfüllung beim Histogramm, für Kalendergitter, die Titelzeile einer Gruppe, Intervalle, Zeitskalenabschnitte, Box-, Linien- und Tabellenformate, Layer sowie für Knotenzeilen.

Über diese Schaltfläche gelangen Sie jeweils in den Dialog **Zuordnung einstellen**, in dem Sie eine datenabhängige Zuordnung für Muster, Musterfarbe oder Hintergrundfarbe oder Musterfarbe2 vereinbaren können.

Wenn eine Zuordnung vorgenommen worden ist, wird der Pfeil auf der Schaltfläche fett dargestellt.

Muster

Hier können Sie ein Hintergrundmuster auswählen.

Musterfarbe

Wählen Sie hier die Vordergrund-Farbe des Musters aus.

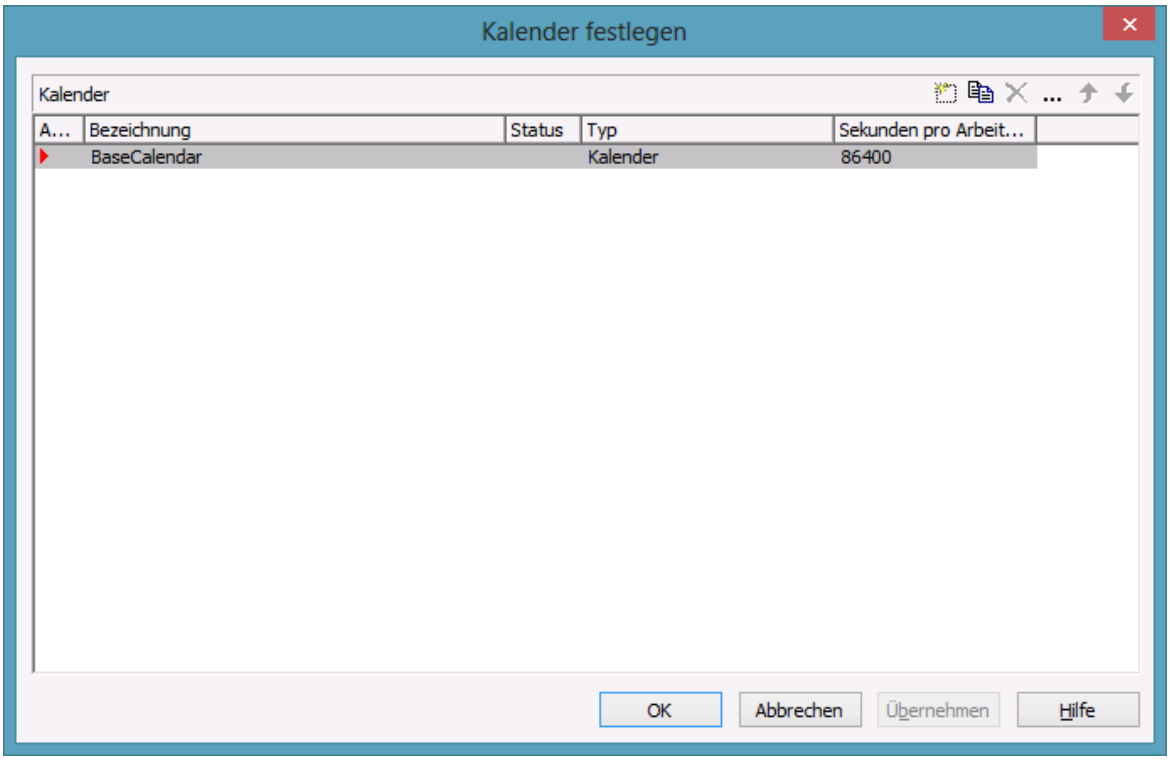
Hintergrundfarbe oder Musterfarbe 2

Wählen Sie hier die Hintergrund-Farbe oder eine zweite Musterfarbe aus.

Vorschau

Hier wird das Aussehen des Musters mit den gewählten Einstellungen angezeigt.

4.36 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.

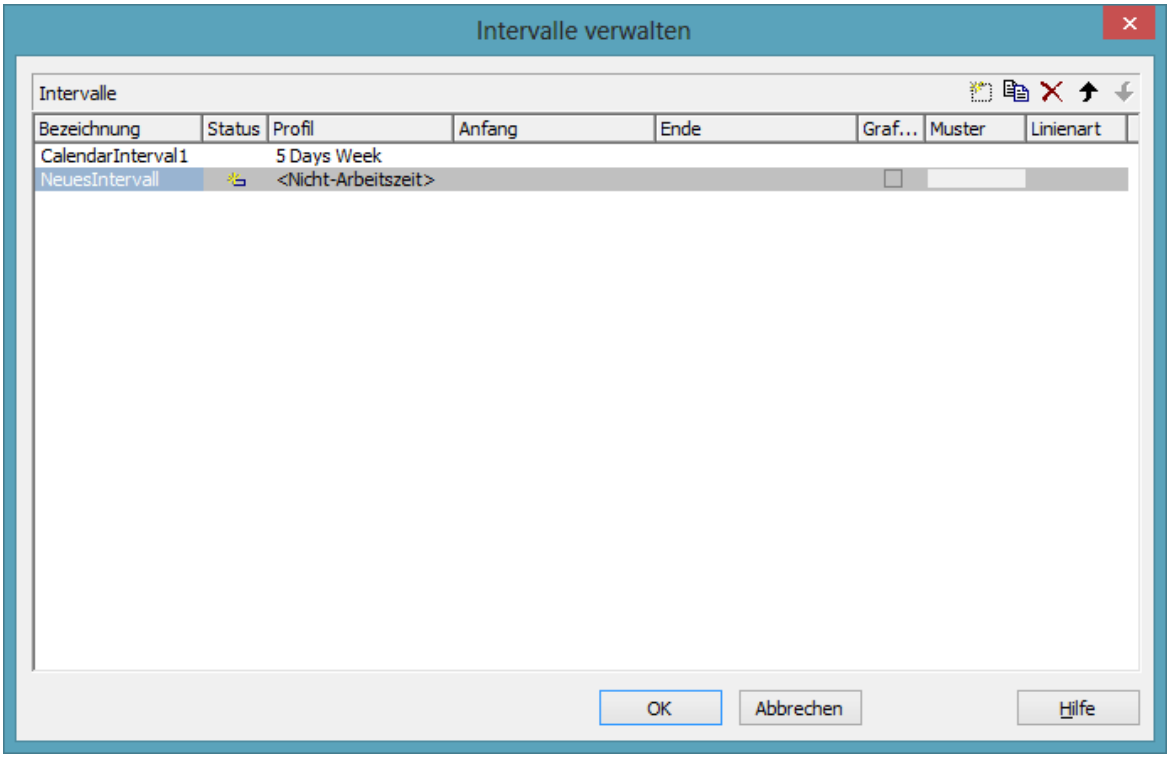
Kalender löschen

 Der markierte Kalender wird gelöscht.

Kalender bearbeiten

 Wenn Sie diese Schaltfläche anklicken, öffnet sich der Dialog **Kalender bearbeiten**.

4.37 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.

Grafische Attribute verwenden

Wenn Sie diese Option wählen, können für das Intervall Muster und Linientyp eingestellt und angezeigt werden. Die Option ist nur aktiv für die Profiltypen <Arbeitszeit> und <Nicht-Arbeitszeit>.

Muster

Durch Klick auf  gelangen Sie in den Dialog **Musterattribute bearbeiten**.

Linienart

Durch Klick auf  gelangen Sie in den Dialog **Linienattribute bearbeiten**.

Intervall hinzufügen

 Ein neues Intervall mit einem Standardnamen wird angelegt. Doppelklicken Sie auf den markierten Namen um ihn zu verändern.

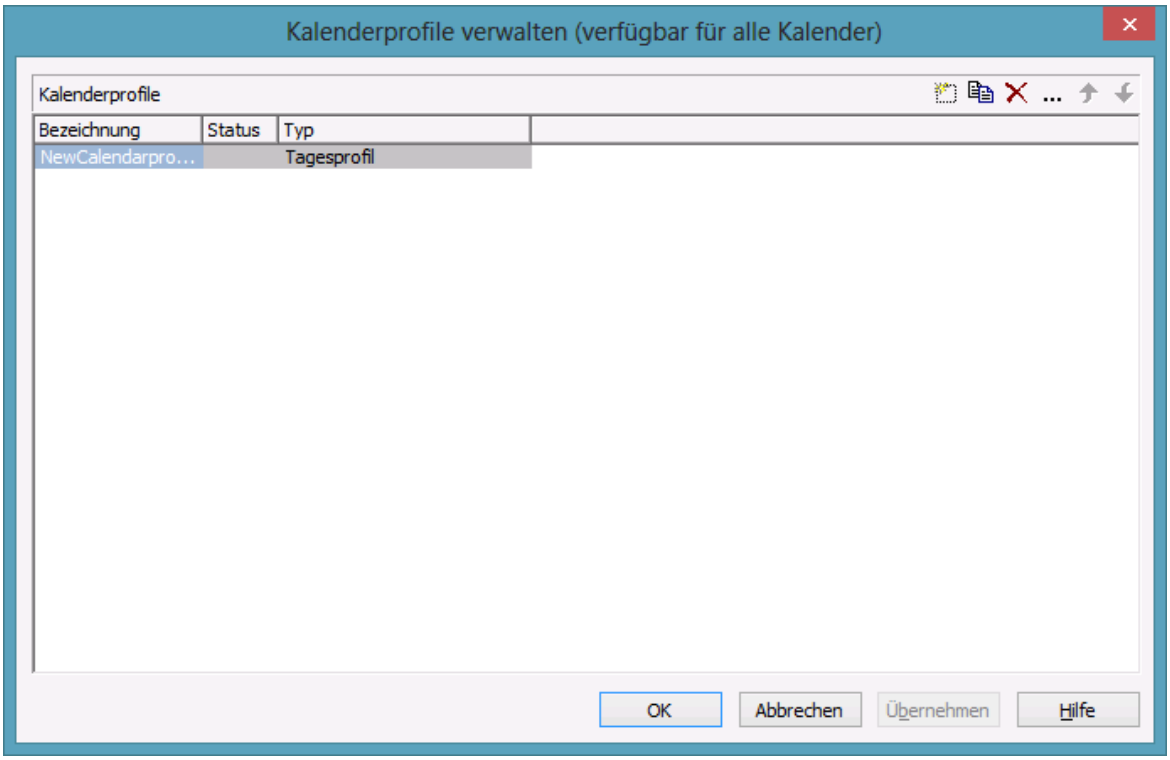
Intervall kopieren

 Es wird eine Kopie des markierten Intervalls angelegt.

Intervall löschen

 Das markierte Intervall wird gelöscht.

4.38 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>.

Kalenderprofil hinzufügen

Ein neues Kalenderprofil mit einem Standardnamen wird angelegt. Doppelklicken Sie auf den markierten Namen um ihn zu verändern.

Kalenderprofil kopieren

 Es wird eine Kopie des markierten Kalenderprofils angelegt.

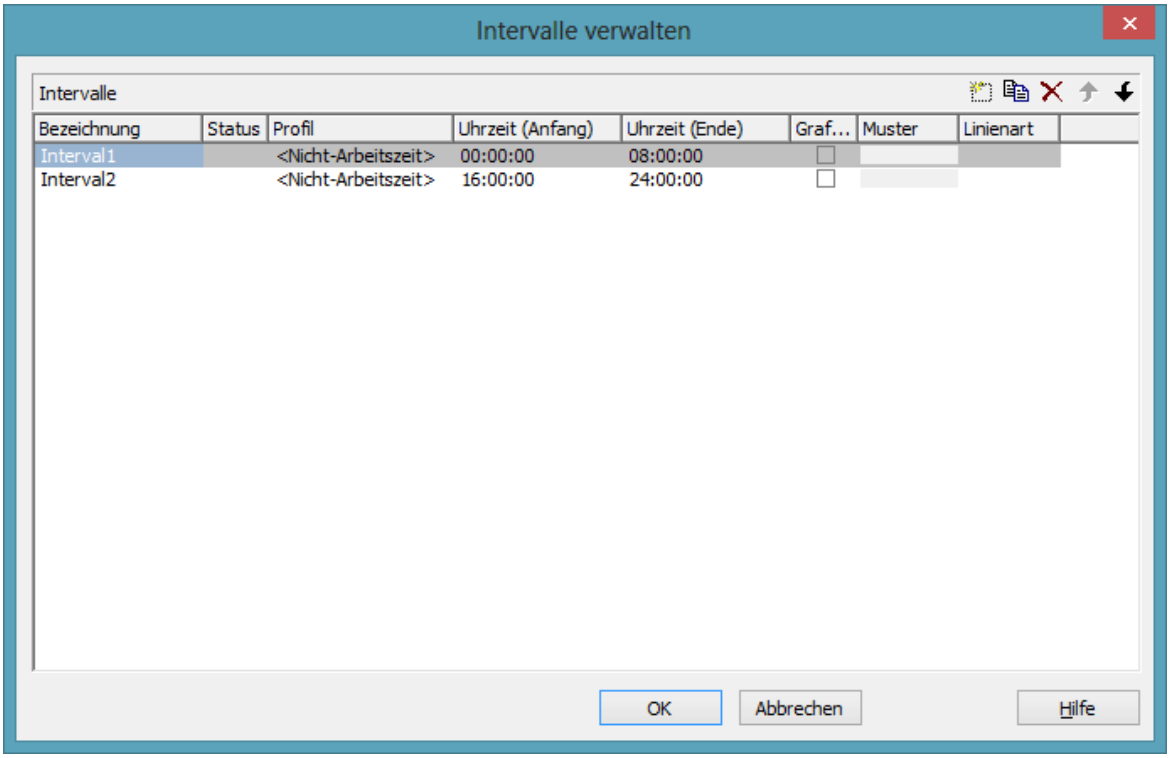
Kalenderprofil löschen

 Das markierte Kalenderprofil wird gelöscht.

Kalenderprofil bearbeiten

 Wenn Sie diese Schaltfläche anklicken, öffnet sich der Dialog **Intervalle verwalten** (Kalenderprofile).

4.39 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.

Grafische Attribute verwenden

Wenn Sie diese Option wählen, können für das Intervall Muster und Linientyp eingestellt und angezeigt werden. Die Option ist nur aktiv für die Profiltypen <Arbeitszeit> und <Nicht-Arbeitszeit>.

Muster

Durch Klick auf ... gelangen Sie in den Dialog **Musterattribute bearbeiten**.

Linienart

Durch Klick auf ... gelangen Sie in den Dialog **Linienattribute bearbeiten**.

Intervall hinzufügen

Ein neues Intervall mit einem Standardnamen wird angelegt. Doppelklicken Sie auf den markierten Namen um ihn zu verändern.

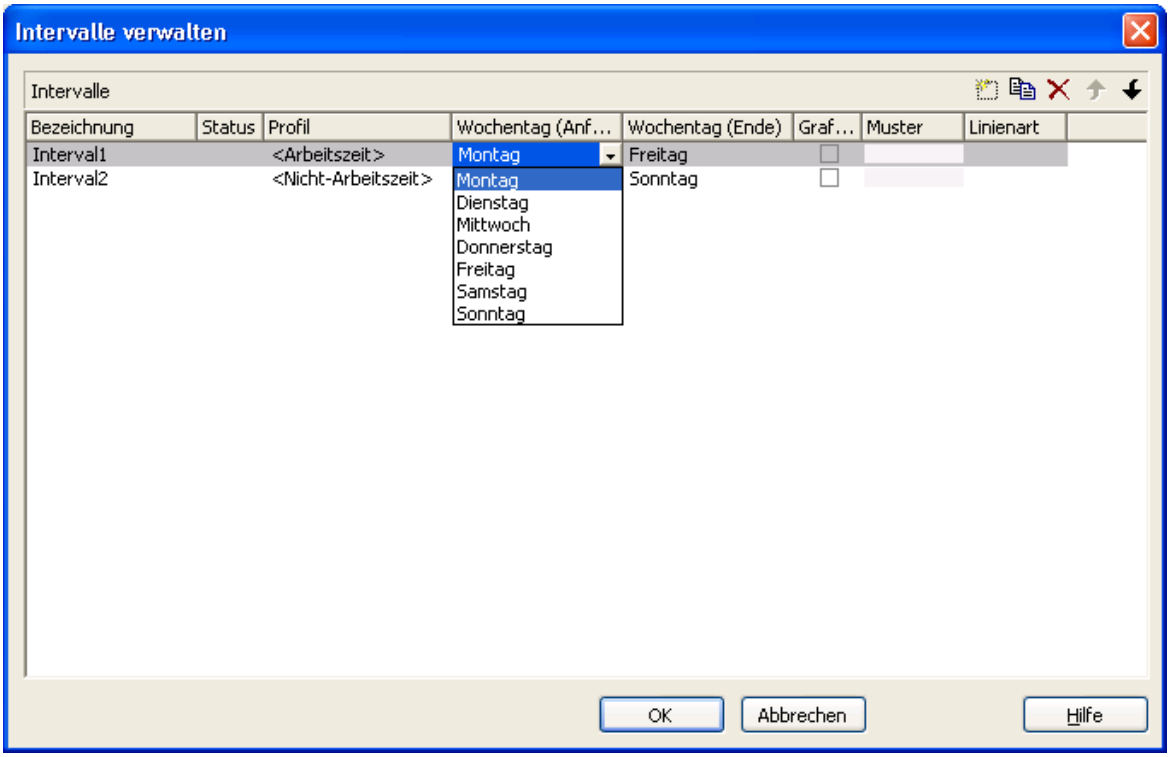
Intervall kopieren

Es wird eine Kopie des markierten Intervalls angelegt.

Intervall löschen

Das markierte Intervall wird gelöscht.

4.40 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.

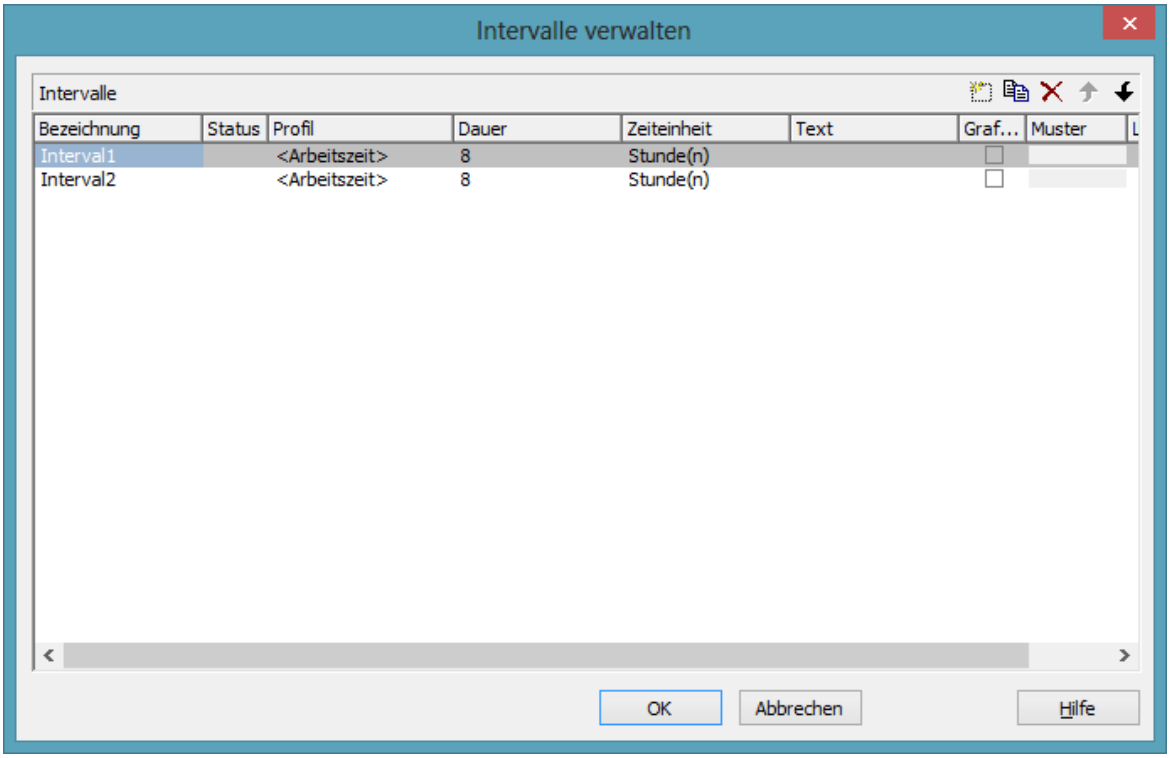
Wochentag Anfang/Wochentag Ende

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.

4.41 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.

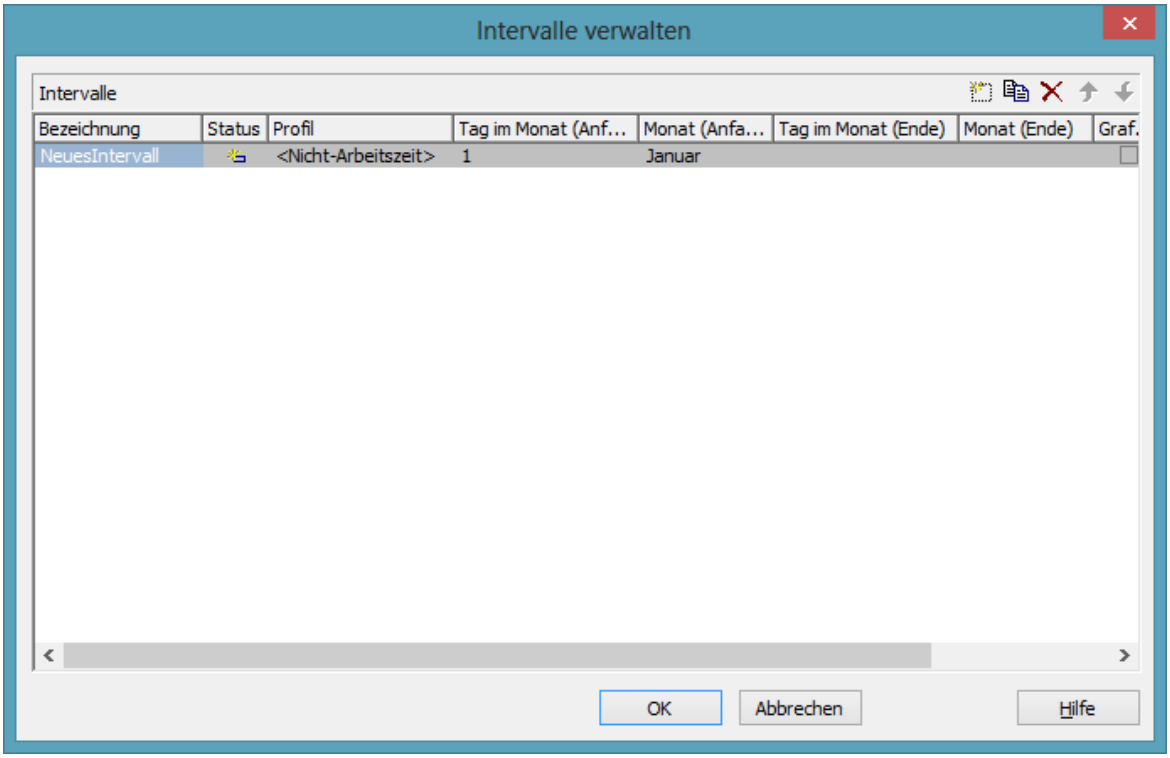
Zeiteinheit

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.

4.42 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.

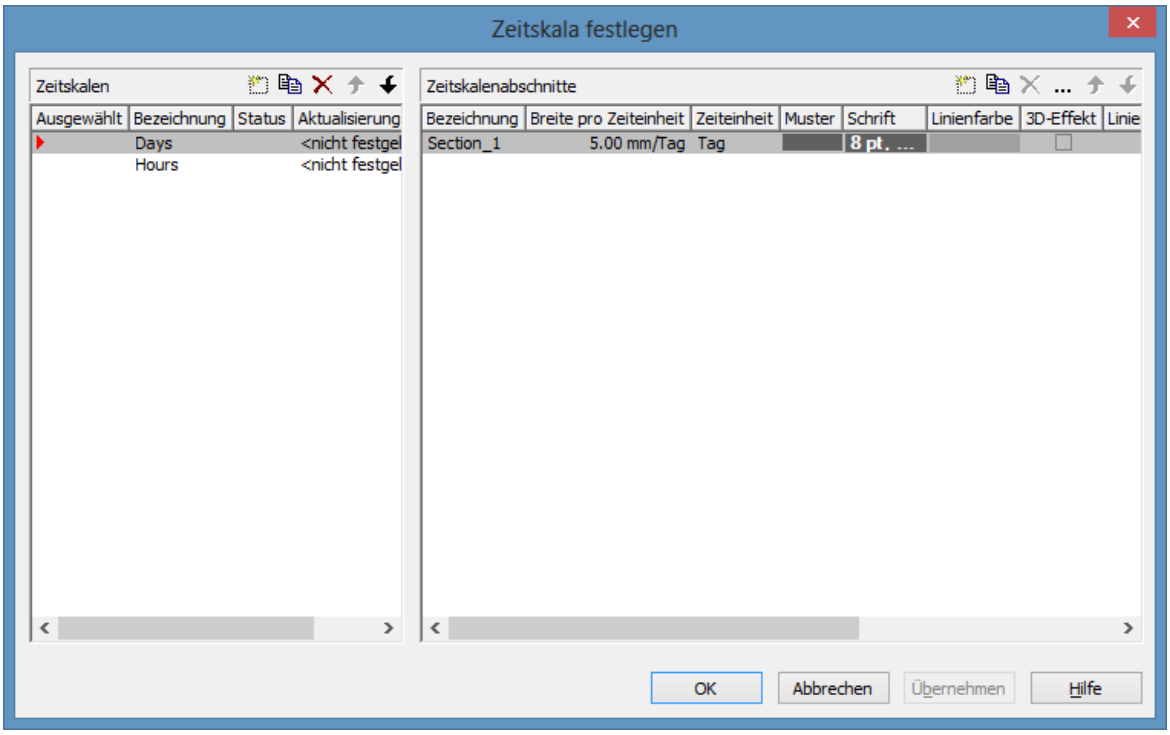
Monat (Anfang)/Monat (Ende)

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.

4.43 Dialogfeld "Zeitskala festlegen"



Sie erreichen dieses Dialogfeld von der Eigenschaftenseite **Objekte**. Sie können hier verschiedene Zeitskalen und deren Zeitskalenabschnitte festlegen.

Zeitskalen

- **Ausgewählt:** Die Zeitskala, die Sie in dieser Spalte durch eine kleine Pfeilspitze markieren, wird für das Diagramm verwendet. Beachten Sie bitte, dass die Zeitskala, die Sie hier wählen, und die Zeiteinheit (Eigenschaftenseite
- **Allgemeines**) zueinander passen sollten.
- **Aktualisierungsverhalten:> Wählen Sie hier das gewünschte Aktualisierungsverhalten aus. Bei der Einstellung <nicht festgelegt>, gilt das Aktualisierungsverhalten, das im Dialog Aktualisierungsverhalten bearbeiten** für Zeitskalen festgelegt wurde.
- **Bezeichnung:** In dieser Spalte werden die Namen aller verfügbaren Zeitskalen aufgeführt. Die Namen sind editierbar.
- **Status:** In dieser Spalte wird jede Zeitskala gekennzeichnet, die seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Zeitskala hinzufügen / kopieren / löschen / nach oben / unten

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

Zeitskalenabschnitte

Diese Tabelle enthält alle für die ausgewählte Zeitskala definierten Zeitskalenabschnitte mit ihren Eigenschaften. Die folgenden Eigenschaften können Sie hier festlegen:

- **Bezeichnung** des Zeitskalenabschnitts
- **Breite pro Zeiteinheit:** Die Breite der Grundeinheit eines Zeitskalenabschnitts wird in mm/Zeiteinheit festgelegt. Die Grundeinheit ist die kleinste Einheit, in die die Zeitskala unterteilt wird. Die Breite der Grundeinheit wird in Millimetern angegeben und lässt sich in Schritten von Hundertstel Millimetern verändern. Die minimale Breite der Grundeinheiten beträgt 0,01 mm.
- **Zeiteinheit:** des Zeitskalenabschnitts: Sekunden, Minuten, Stunden, Tage.
- **Muster** des Zeitskalenabschnitts: Sie können ein anderes Muster auswählen, indem Sie auf die Schaltfläche () klicken und damit das Dialogfeld **Musterattribute bearbeiten** öffnen. Das hier gewählte Muster wird für alle Zeitstreifen dieses Zeitskalenabschnitts verwendet. Falls die Muster der Zeitstreifen zuvor unterschiedlich waren, werden sie damit gleich gesetzt.
- **Schrift:** Hier werden die Schriftart und -farbe angezeigt. Über die erste Schaltfläche () gelangen Sie in das Dialogfeld **Farbe**, in dem Sie die Schriftfarbe festlegen können. Über die zweite Schaltfläche () gelangen Sie in den Windows-Dialog **Schriftart**, in dem Sie die Schriftart für die Beschriftung des Zeitskalenabschnitts auswählen können. Die hier gewählte Schrift (Schriftfarbe und -art) wird für alle Zeitstreifen dieses Zeitskalenabschnitts verwendet. Falls die Schriften der Zeitstreifen zuvor unterschiedlich waren, werden sie damit gleich gesetzt.
- **Linienfarbe:** Bestimmen Sie hier, in welcher Farbe der Zeitskalenabschnitt umrahmt werden soll.

- **3D-Effekt:** Mit Hilfe dieser Option können Sie wählen, ob der Zeitskalenabschnitt durch einen 3D-Effekt einen räumlichen Eindruck erhalten soll.
- **Liniengitter:** Hier können Sie festlegen, ob die vordefinierten vertikalen Gitternetzlinien im Diagrammbereich unter dem Zeitskalenabschnitt dargestellt werden sollen.
- **Kalendergitter:** Hier können Sie festlegen, ob im Diagrammbereich unter dem Zeitskalenabschnitt ein vordefiniertes Kalendergitter angezeigt werden soll. Dadurch werden Wochenenden und sonstige arbeitsfreie Tage durch Flächen besonders markiert.
- **Kollabieren der arbeitsfreien Zeiten:** Hier können Sie für den Zeitskalenabschnitt festlegen, ob arbeitsfreie Zeiten ausgeblendet werden sollen oder nicht. Der Kalender, der die arbeitsfreien Zeiten bestimmt, wird im Dialog **Kalender festlegen** ausgewählt.

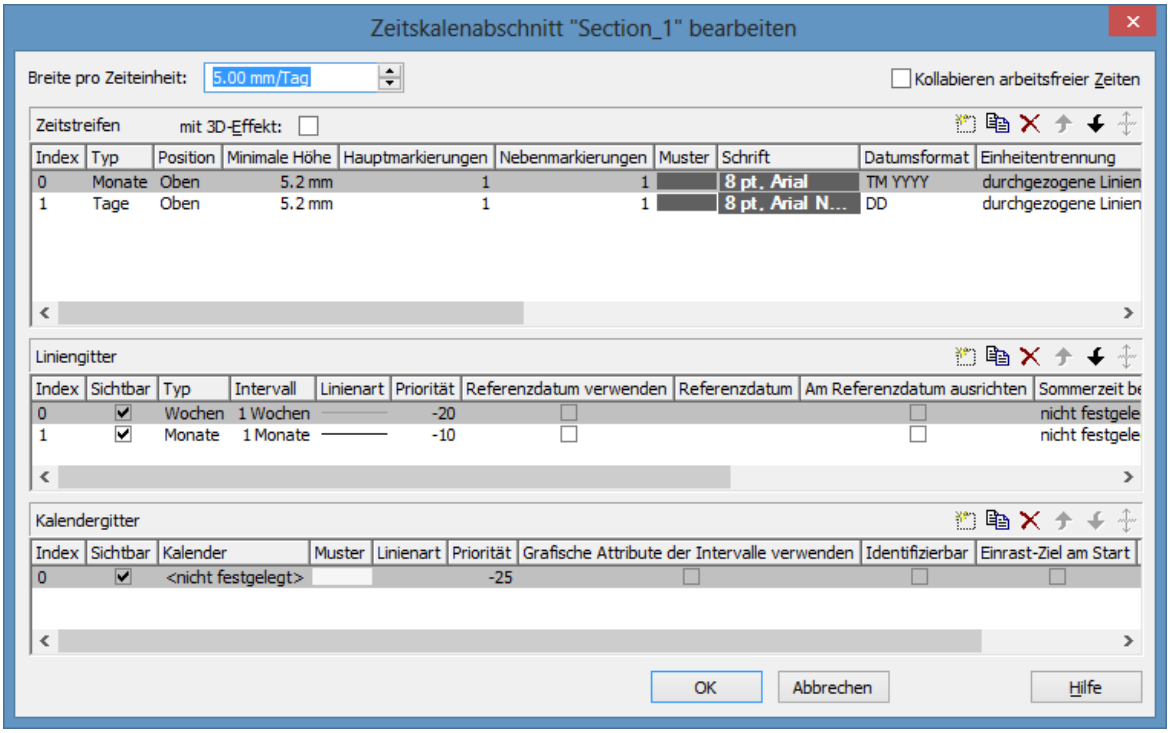
Zeitskalenabschnitt hinzufügen/ kopieren/ löschen/ bearbeiten/ früher/ später

Mit Hilfe dieser Schaltflächen können Sie Zeitskalenabschnitte hinzufügen, kopieren, löschen, bearbeiten und in der Tabelle nach oben bzw. unten verschieben.

Die Reihenfolge der Zeitskalenabschnitte in der Tabelle entspricht der Reihenfolge in der Darstellung.

Wenn Sie einen neuen Zeitskalenabschnitt definiert haben, werden anschließend alle Zeitskalenabschnitte in ihrer zeitlichen Ausdehnung ungefähr gleich breit dargestellt. Durch Ziehen mit der Maus können Sie diese Aufteilung noch modifizieren. Über die API können Sie den Anfang jedes Zeitskalenabschnitts festlegen bzw. bearbeiten.

4.44 Dialogfeld "Zeitskalenabschnitt bearbeiten"



Breite pro Zeiteinheit

Hier kann die Breite pro Zeiteinheit für den gewählten Zeitskalenabschnitt eingestellt werden. Der neue Wert wird in das entsprechende Feld der **Zeitskalenabschnitte**-Tabelle des Dialogs **Zeitskala festlegen** übertragen.

Kollabieren arbeitsfreier Zeiten

Legen Sie für den gewählten Zeitskalenabschnitt fest, ob arbeitsfreie Zeiten ausgeblendet werden sollen oder nicht. Der Kalender, der die arbeitsfreien Zeiten bestimmt, ist der, der im Dialog **Kalender festlegen** ausgewählt ist.

Wenn Sie hier eine Festlegung treffen, wird sie in die **Zeitskalenabschnitte**-Tabelle des Dialogs **Zeitskala festlegen** übernommen.

Zeitstreifen

Zeitstreifen dienen zur Beschriftung der Zeitskala. Jeder Zeitskalenabschnitt kann mehrere Zeitstreifen besitzen (beispielsweise je einen mit einer Monats- und einer Tagesskala).

 Mit Hilfe dieser Schaltflächen können Sie Zeitstreifen hinzufügen, kopieren, löschen und in der Tabelle nach oben bzw. unten verschieben.

In der Tabelle können Sie Folgendes für die einzelnen Zeitstreifen des gewählten Zeitskalenabschnitts festlegen:

- **Index**: fortlaufende Nummer der Zeitstreifen (nicht editierbar)
- **Typ** des Zeitstreifens: Sekunden, Minuten, Stunden, Tage, Wochen, Monate, Quartale, Jahre, Schichten, Fiskalische Quartale, Fiskalische Jahre.
- **Position:** Legen Sie hier für jeden Zeitstreifen fest, ob er oberhalb (**oben**) oder unterhalb (**unten**) des Diagrammbereichs oder überhaupt nicht dargestellt werden soll (**keine**).
- **Minimale Höhe** des Zeitstreifens in mm
- **Hauptmarkierungen:** Legen Sie hier fest, nach wie vielen Zeiteinheiten eine Hauptmarkierung erscheint, z. B. nach jeweils sieben Tagen. (Die Zeiteinheit hängt vom Typ des verwendeten Zeitstreifens ab.) Die Hauptmarkierungen werden beschriftet, wenn ausreichend Platz vorhanden ist.
- **Nebenmarkierungen:** Legen Sie hier fest, nach wie vielen Zeiteinheiten eine unbeschriftete Nebenmarkierung erscheint, z. B. nach jeweils einem Tag. (Die Zeiteinheit hängt vom Typ des verwendeten Zeitstreifens ab.)
- **Muster:**Legen Sie hier das Muster des Zeitstreifens fest. Durch Klick auf  gelangen Sie in den Dialog **Musterattribute bearbeiten**, in dem Sie ein Muster, eine Hintergrundfarbe sowie eine 2. Musterfarbe auswählen können. Solange Sie kein anderes Muster festlegen, haben alle Zeitstreifen des Zeitskalenabschnitts das Muster, das im Dialog **Zeitskala festlegen** ausgewählt wurde. Wenn Sie für den ersten Zeitstreifen eines Zeitskalenabschnitts ein neues Muster auswählen, wird dieses auch für das **Muster**-Feld der **Zeitskalenabschnitte**-Tabelle im Dialog **Zeitskala festlegen** übernommen.
- **Schrift:** Hier werden die Schriftart und -farbe jedes Zeitstreifens angezeigt. Wenn dieser Wert nicht gesetzt wird, haben alle Zeitstreifen des Zeitskalenabschnitts standardmäßig die Schriftart, die im Dialog **Zeitskala festlegen** dafür ausgewählt wurde. Um für einen Zeitstreifen eine andere Schriftfarbe festzulegen, klicken Sie auf die Pfeil-Schaltfläche (). Sie gelangen dann in das Dialogfeld **Farbe**. Um für einen Zeitstreifen eine andere Schriftart festzulegen, klicken Sie auf die zweite Schaltfläche (). Sie gelangen dann in den Windows-Dialog

Schriftart. Die Schrift, die Sie für den ersten Zeitstreifen eines Zeitskalenabschnitts vereinbaren, wird in das **Schrift**-Feld der **Zeitskalenabschnitte**-Tabelle im Dialog **Zeitskala festlegen** übertragen.

- **Datumsformat** des Zeitstreifens: Welche Formate möglich sind, hängt von dem gewählten Typ des Zeitstreifens ab. 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 "Uhr" (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

xC/XC: Sie können das Datum als maximal zehngliedrige, einfache Aufwärtszählung ab einem bestimmten Referenzdatum gestalten, z.B. "15:05:07:16:00". Dieses Beispiel datiert 15 Monate, 5 Tage, 7 Stunden, 16 Minuten und 0 Sekunden später als das gesetzte Referenzdatum. Die Notation dazu ist: **xC44:C33:C22:C11:C00**. Es sollen je 2 Stellen für die Glieder 4...0 vorgesehen werden, mit einem vorausgehenden "-" Symbol falls der Wert negativ ist. Die Trennzeichen sind variabel und könnten durch andere ersetzt werden. "x" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, aber kein "+"Symbol, falls er positiv ist. "X" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, und ebenso ein "+"Symbol, falls er positiv ist. Im Dialog **Zeitskalenabschnitt ... bearbeiten** sollten die Felder **Referenzdatum verwenden** und **Am Referenzdatum ausrichten** aktiviert sein, ebenso sollte die **Serielle Beschriftungen** auf **Keine** gesetzt sein. Das Referenzdatum wird zur Laufzeit über die Methode **VcRibbon.set ReferenceDate** gesetzt und überschreibt die Werte des Dialogs.

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.

- **Einheitentrennung:** Zur Unterteilung des Zeitstreifens können Sie zwischen drei Möglichkeiten wählen: **durchgezogene Linien**, **Ticks** und **keine Linien**.
- **Tick-Position:** Legen Sie fest, ob die Ticks und deren Beschriftungen am oberen oder unteren Rand ausgegeben werden sollen.
- **Markierungsfarbe:** Legen Sie die Farbe der Markierungen fest.
- **Ausrichtung:** Für die Ausrichtung der Beschriftung an den Hauptmarkierungen stehen folgende Möglichkeiten zur Verfügung: **Zentriert**, **Rechts**, **Links**, **An den Ticks** (Hauptmarkierungen).

- **Serielle Beschriftung:** Legen Sie fest, ob auf dem Zeitstreifen serielle Nummern statt des Datums angezeigt werden sollen, und falls ja, ob Null der Beginn am ggf. gesetzten Referenzdatum sein soll.
- **Referenzdatum verwenden:** Aktivieren Sie dieses Kontrollkästchen, falls ggf. der Ursprung der seriellen Nummern (oder des fiskalischen Jahres oder Quartals) auf das Referenzdatum gelegt werden soll. Andernfalls wird er auf den Anfang des Zeitskalenabschnitts gelegt.
- **Referenzdatum:** Wählen Sie ggf. das Referenzdatum mit Hilfe des Datumsdialogs aus.
- **Am Referenzdatum ausrichten:** Aktivieren Sie dieses Kontrollkästchen, wenn die Linien des Liniengitters am Referenzdatum ausgerichtet werden sollen, d.h. die Position kann vom Anfang einer Zeiteinheit abweichen, z.B. um13:17 jedes Tages. Ist diese Option nicht gewählt, werden die Linien eines Liniengitters standardmäßig am Anfang der jeweiligen Zeiteinheit positioniert, z.B. immer um 00:00 Uhr jedes Tages.
- **Kalender:** Falls Sie einen Schichten-Zeitstreifen darstellen möchten, wählen Sie hier einen der Schichtkalender, die Sie im Dialog **Kalender festlegen** definiert haben.
- **Sommerzeit beachten:** Aktivieren Sie dieses Kontrollkästchen, wenn auf dem Zeitstreifen die Sommerzeit berücksichtigt werden soll.

Liniengitter

Im Diagrammbereich und im Histogramm können ein oder mehrere Liniengitter (bestehend aus vertikalen Linien) unter dem gewählten Zeitskalenabschnitt dargestellt werden.

 Mit Hilfe dieser Schaltflächen können Sie Liniengitter hinzufügen, kopieren, löschen und in der Tabelle nach oben bzw. unten verschieben.

In der Tabelle können Sie die Liniengitter für den gewählten Zeitskalenabschnitt festlegen:

- **Index**: fortlaufende Nummer der Liniengitter (nicht editierbar)
- **Sichtbar**: Aktivieren Sie dieses Kontrollkästchen für jedes Liniengitter, das angezeigt werden soll.
- **Typ:** Hier können Sie die Grundeinheit jedes Liniengitters festlegen, z. B. Tage, Wochen, etc.
- **Intervall:** Hier können Sie das Intervall zwischen den einzelnen Linien in ganzzahligen Vielfachen der Grundeinheit des Liniengitters festlegen.

- **Linienart:** Wenn Sie auf die Schaltfläche dieses Feldes (…) klicken, öffnet sich das Dialogfeld **Linienattribute des Liniengitters**. Hier können Sie Farbe, Typ und Dicke der Randlinien der Streifen festlegen.
- **Priorität:** Hier können Sie die Priorität jedes Liniengitters in Relation zu anderen Gittern und zu den Layern festlegen (> 0: vor den Layern, < 0: hinter den Layern).
- **Referenzdatum:**Das Referenzdatum versetzt den Anfang des Liniengitters um den angegebenen Abstand zu dem standardmäßigen Beginn am Montag 00.00 Uhr.
- **Sommerzeit beachten:** Aktivieren Sie dieses Kontrollkästchen, wenn für dieses Liniengitter die Sommerzeit berücksichtigt werden soll.
- **Einrastziel** Das Liniengitter definiert seine relevanten Positionen als "Einrastziele" für zu verschiebende Knoten/Layer.

Kalendergitter

Im Diagrammbereich und im Histogramm können unter dem gewählten Zeitskalenabschnitt ein oder mehrere Kalendergitter dargestellt werden, die arbeitsfreie Zeiten durch vertikale Streifen darstellen.

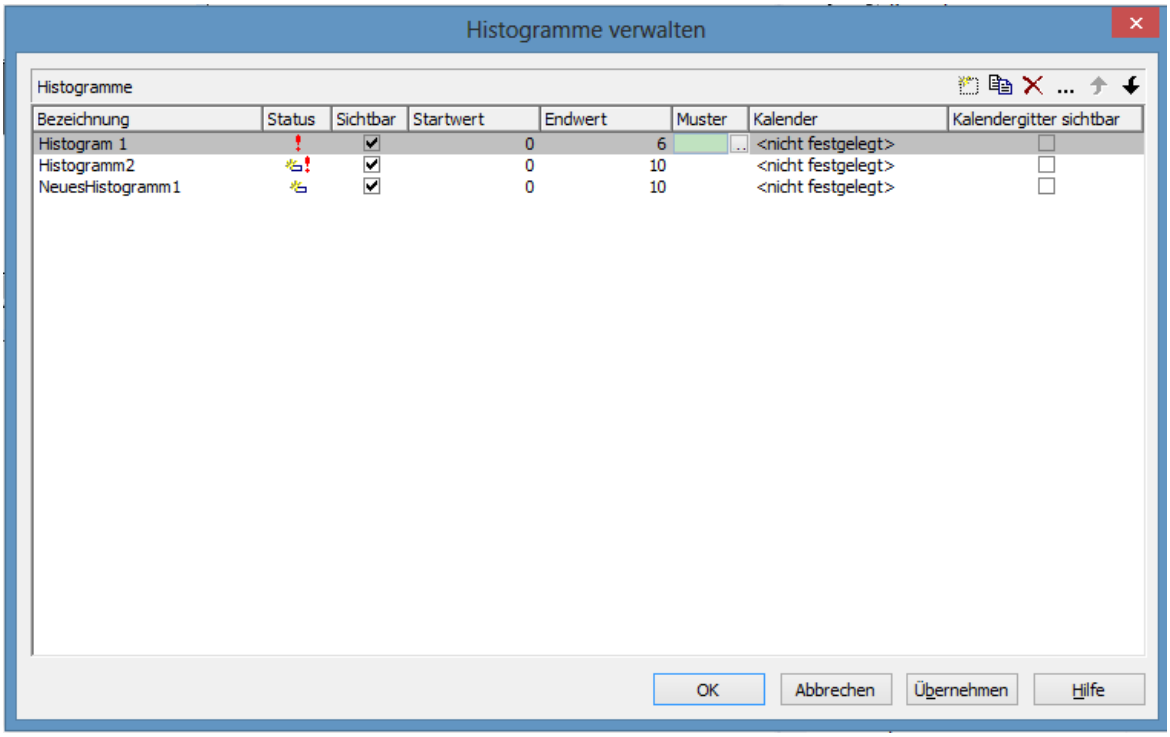
 Mit Hilfe dieser Schaltflächen können Sie Kalendergitter hinzufügen, kopieren, löschen und in der Tabelle nach oben bzw. unten verschieben.

Sie können hier Folgendes für jedes Kalendergitter festlegen:

- **Index**: fortlaufende Nummer der Kalendergitter (nicht editierbar)
- **Sichtbar**: Aktivieren Sie dieses Kontrollkästchen für jedes Kalendergitter, das angezeigt werden soll.
- **Kalender**: Wählen Sie den Kalender aus, der die arbeitsfreien Zeiten bestimmen soll. Beim Eintrag <nicht festgelegt> wird der Kalender verwendet, der im Dialog **Kalender festlegen** ausgewählt ist.
- **Muster:** Wenn Sie auf die Schaltfläche dieses Feldes (…) klicken, öffnet sich das Dialogfeld **Musterattribute des Kalendergitters**. Hier können Sie Typ, Vordergrund- und Hintergrundfarbe des Kalendergitters festlegen. Auch transparente Farben sind hier möglich.
- **Linienart:** Wenn Sie auf die Schaltfläche dieses Feldes (…) klicken, öffnet sich das Dialogfeld **Linienattribute des Kalendergitters**. Hier können Sie Farbe, Typ und Dicke der Randlinien der vertikalen Linien festlegen.

- **Priorität:** Hier können Sie die Priorität jedes Kalendergitters in Relation zu den anderen Gittern und Layern festlegen (> 0: vor den Layern, < 0: hinter den Layern).
- **Kalendergitter>!: Das Kalendergitter definiert seine relevanten Positionen als "Einrastziele" für zu verschiebende Knoten/Layer.**

4.45 Dialogfeld "Histogramme verwalten"



Sie erreichen dieses Dialogfeld über die Eigenschaftenseite Layout.

Sie können hier mehrere Histogramme festlegen und auswählen, welche(s) davon angezeigt werden soll(en).

Bezeichnung

In dieser Spalte werden die Namen aller verfügbaren Histogramme aufgeführt. Die Namen sind editierbar.

Status

In dieser Spalte wird jedes Histogramm gekennzeichnet, das seit dem Aufruf des Dialogs hinzugefügt ( ) und/oder geändert ( ) worden ist.

Sichtbar

Mit Hilfe dieses Kontrollkästchens können Sie festlegen, ob das gewählte Histogramm dargestellt werden soll oder nicht.

Startwert

Legen Sie hier den niedrigsten Wert der numerischen Skala des Histogramms fest. Gegebenenfalls wird der niedrigste Wert der numerischen Skala an die Kurvenwerte angepasst.

Endwert

Legen Sie hier den höchsten Wert der numerischen Skala des Histogramms fest. Gegebenenfalls wird der höchste Wert der numerischen Skala an die Kurvenwerte angepasst.

Muster

Hier können für jedes Histogramm Muster und Farbe festgelegt werden.

Histogramm hinzufügen

 Ein neues Histogramm wird erzeugt.

Histogramm kopieren

 Das markierte Histogramm wird kopiert.

Histogramm löschen

 Das markierte Histogramm wird gelöscht.

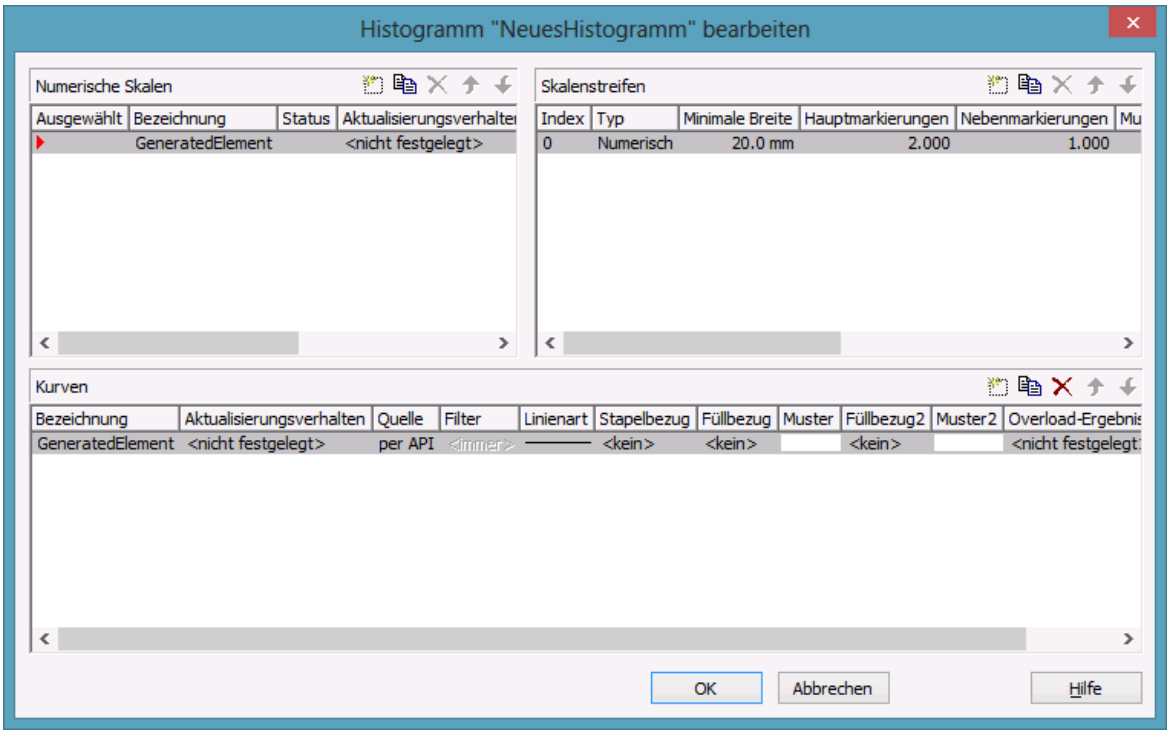
Histogramm bearbeiten

 Der Dialog **Histogramm bearbeiten** wird geöffnet.

Histogramm eine Zeile nach oben/unten

 Mit Hilfe dieser Schaltflächen können Sie das markierte Histogramm eine Zeile nach oben/unten schieben. Die Position der Histogramme in der Tabelle bestimmt deren Ausgabereihenfolge.

4.46 Dialogfeld "Histogramm bearbeiten"



Sie erreichen dieses Dialogfeld, indem Sie im Dialog **Histogramme verwalten** auf die **Histogramm bearbeiten**-Schaltfläche (…) klicken.

Sie können hier für das zu bearbeitende Histogramm verschiedene numerische Skalen mit jeweils einem oder mehreren Skalenstreifen definieren und auswählen, welche davon dargestellt werden soll.

Das Histogramm kann mehrere Kurven enthalten. Sie können für jede Kurve festlegen, aus welcher Quelle die Daten dafür stammen sollen. Durch Filter kann für jede Kurve festgelegt werden, welche Vorgänge zu der Kurve beitragen sollen. Außerdem können Sie das Aussehen der Kurven definieren.

Numerische Skalen

- **Ausgewählt:** Die rote Pfeilspitze markiert die zu verwendende numerische Skala.
- **Bezeichnung** der numerischen Skala
- **Status:** In dieser Spalte wird jede numerische Skala 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 Aktu-

alisierungsverhalten, das im Dialog **Aktualisierungsverhalten bearbeiten** für die numerische Skala festgelegt wurde.

- **Breite pro Einheit** in mm, legt den Abstand zwischen den Hauptmarkierungen fest
- **Einheit gibt die Schrittweite der Hauptmarkierungen an**
- **Linienfarbe** Legen Sie hier die Markierungsfarbe aller numerischer Skalenstreifen von Histogrammen fest
- **Liniengitter:** Legen Sie hier fest, ob ein Liniengitter angezeigt werden soll oder nicht.
- **Linienart:** Hier wird der Linientyp des Liniengitters angezeigt. Um ihn zu ändern, klicken Sie auf die Schaltfläche (···). Dann öffnet sich der Dialog **Linie bearbeiten**.

Skalenstreifen

Legen Sie hier für jeden Skalenstreifen der markierten numerischen Skala Folgendes fest:

- **Index**: fortlaufende Nummer des Skalenstreifens (nicht editierbar)
- **Typ** des Skalenstreifens: **Numerisch** oder **Textuell**. Über die Schaltfläche öffnen Sie einen Dialog, in dem Sie den Typ festlegen können.
- **Minimale Breite** in mm
- **Hauptmarkierungen:** Legen Sie hier fest, nach wie vielen Einheiten eine Hauptmarkierung (beschrifteter Teilstrich) erscheint.
- **Nebenmarkierungen:** Legen Sie hier fest, nach wie vielen Einheiten eine Nebenmarkierung (ein kleinerer, unbeschrifteter Teilstrich) erscheint.
- **Muster:** Durch Klick auf ··· gelangen Sie in den Dialog **Musterattribute bearbeiten**, in dem Sie ein Muster, eine Musterfarbe sowie eine Hintergrundfarbe oder 2. Musterfarbe für den Skalenstreifen auswählen können. Zusätzlich zu der vorgegebenen Farbpalette können Sie beliebige Farben definieren. Es stehen außerdem transparente Farben zur Verfügung.
- **Schrift:** Die Schriftart des Skalenstreifens wird hier angezeigt. Über die Schaltfläche (···) erreichen Sie den Windows-Dialog **Schriftart**.

- **Doubleformat:** Hier können Sie aus verschiedenen Zahlenausgabeformaten wählen, wobei **I** für die Ziffern vor dem Dezimaltrenner und **D** für je eine Ziffer hinter dem Dezimaltrenner steht.
- **Markierungsfarbe:** Legen Sie hier die Markierungsfarbe aller numerischer Skalenstreifen fest
- **ObjectDraw-Ereignisse:** Wählen Sie diese Option, wenn Sie die beiden Ereignisse **VcObjectDrawing** und **VcObjectDrawn** auslösen möchten. Mit dem Ereignis **VcObjectDrawing** wird der Standardskalenstreifen durch einen benutzerdefinierten ersetzt, das Ereignis **VcObjectDrawn** wird hierbei nicht benötigt. Das Ereignis **VcObjectDrawn** ermöglicht es Ihnen, dem von VARCHART XGantt gezeichneten Skalenstreifen noch etwas hinzuzufügen.
- **Einheitenbezeichnung:** Beschriftung der Einheiten der numerischen Skala.

Kurven

- **Bezeichnung:** In dieser Spalte werden die Namen aller verfügbaren Kurven angezeigt.
- **Aktualisierungsverhalten:** Wählen Sie hier das gewünschte Aktualisierungsverhalten aus. Bei der Einstellung <nicht festgelegt>, gilt das Aktualisierungsverhalten, das im Dialog **Aktualisierungsverhalten bearbeiten** für Kurven festgelegt wurde.
- **Quelle:** Hier können Sie für jede einzelne Kurve festlegen, aus welcher Quelle die Daten für die Berechnung der einzelnen Kurven stammen sollen. Sie können zwischen zwei grundsätzlichen Alternativen wählen:

  1. per Layer: Die Kurve wird aus den Daten der gewählten Layer aller Vorgänge, die die Layer-Filterkriterien erfüllen, generiert. Über die Wahl eines Filters können Sie diese Vorgänge noch weiter eingrenzen.

  2. per API: Sie können die Werte über die Programmierschnittstelle (API) mit der VcCurve-Methode **SetValues** eingeben.

  Über die **Bearbeiten**-Schaltfläche (···) können Sie in beiden Fällen den Dialog **Datenquelle der Kurve einstellen** aufrufen.
- **Filter:** Wählen Sie für jede Kurve den Filter aus, der bestimmt, welche Vorgänge zu der Kurve beitragen sollen. Über die **Bearbeiten**-Schaltfläche (···) können Sie ggf. den Dialog **Filter verwalten** aufrufen.

- **Linienart:** Wenn Sie auf den Eintrag dieses Feldes klicken, öffnet sich das Dialogfeld **Linie bearbeiten**, in dem Sie das Aussehen der Kurvenlinie festlegen können.
- **Stapelbezug:** Wenn Sie die Kurven stapeln möchten, geben Sie für jede einzelne Kurve an, auf welche Kurve sie gestapelt werden soll. Sie können so sukzessive die Kurven aufeinander stapeln, um so beispielsweise die Gesamtauslastung Ihrer Ressourcen (z. B. Arbeitsgruppen) darzustellen. Soll eine Kurve nicht auf eine andere Kurve gestapelt werden, wählen Sie hierfür den Eintrag "<kein>". Wählen Sie für alle Kurven den Eintrag "<kein>", werden die Kurven nicht aufeinander gestapelt, sondern überlagern einander gegebenenfalls. Um die Kurven dennoch voneinander unterscheiden zu können, sollten Sie sie durch unterschiedliche Muster kennzeichnen.
- **Füllbezug:** Hier können Sie festlegen, bis wohin die Füllung unterhalb jeder Kurve reichen soll. Wählen Sie hier für eine Kurve den Eintrag "<kein>", wird keine Füllung unter dieser Kurve dargestellt. Wählen Sie "<Null-Linie>", so reicht die Füllung unter dieser Kurve bis zur Null-Linie. Geben Sie hier eine andere Kurve an, reicht die Füllung nur bis zur dieser Kurve.
- **Muster:** Hier können Sie das Muster der Füllung unter jeder Kurve festlegen. Über die **Bearbeiten**-Schaltfläche (…) erreichen Sie den Dialog **Muster**, in dem Sie das Muster bearbeiten können.
- **Füllbezug 2:** Wählen Sie hier eine 2. Bezugskurve aus. Die Füllung zur 2. Bezugskurve wird nur dargestellt, wenn die y-Werte der in dieser Zeile definierten Kurve größer sind als die der 2. Bezugskurve.
- **Muster 2:** Hier können Sie das Muster und die Farbe der Füllung über der 2. Bezugskurve festlegen.
- **Overload-Ergebnis Wählen Sie aus der Drop-Down-Liste einen von Ihnen zuvor erstellten Kalender, in den die Intervalle, die sich aus den berechneten Overload-Terminen ergeben, geschrieben werden. Dieser Kalender kann z.B. für die Darstellung eines Kalendergitters in der Gruppe verwendet werden.**

Beispiele zur Verwendung von Histogrammen finden Sie unter "Tutorium: Histogramme erstellen" sowie "Tutorium: Kapazitätsengpässe darstellen".

Numerische Skala/Zeitstreifen/Kurve hinzufügen

 **Ein neues Objekt wird erzeugt.**

Numerische Skala/Zeitstreifen/Kurve kopieren

 **Das markierte Objekt wird kopiert.**

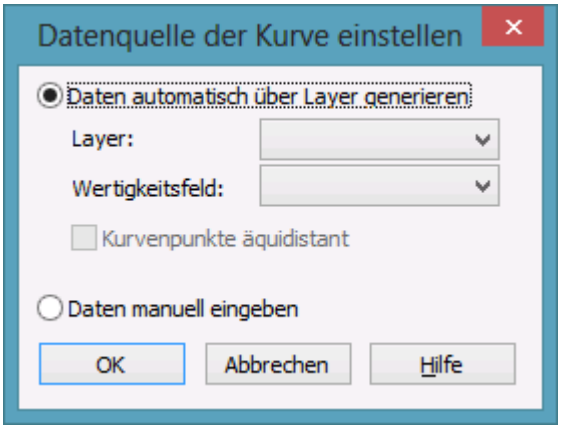
Numerische Skala/Zeitstreifen/Kurve löschen

 **Das markierte Objekt wird gelöscht.**

Numerische Skala/Zeitstreifen/Kurve eine Zeile nach oben/unten

 **Mit Hilfe dieser Schaltflächen wird das markierte Objekt in der Liste um eine Zeile nach oben bzw unten geschoben.**

4.47 Dialogfeld "Datenquelle der Kurve einstellen"



Sie erreichen dieses Dialogfeld über das Dialogfeld Histogramm bearbeiten.

Daten automatisch über Layer generieren

Wählen Sie diese Option, wenn die Daten automatisch über Layer generiert werden sollen. Beim Aufsummieren der Vorgänge zu einer Kurve werden für jeden Vorgang die Anfangs- und Enddaten des gewählten Layertyps (z.B. des Layers "Start-Ende") übernommen. Legen Sie dann Folgendes fest:

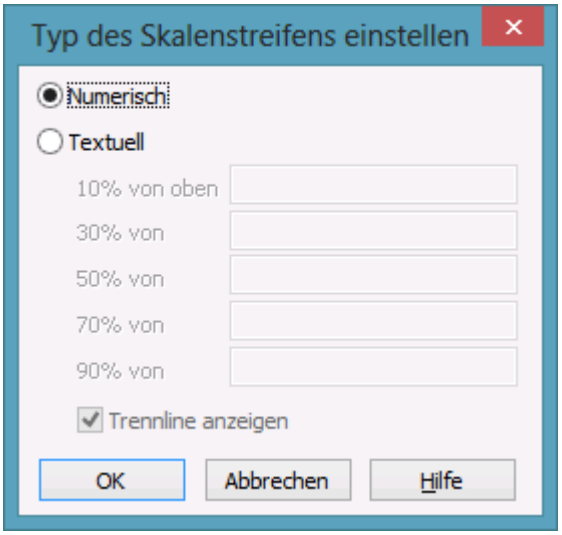
- **Layer**
- **Wertigkeitsfeld:** Datenfeld, aus dem pro Vorgang die Wertigkeit für die Addition der Kapazitäten entnommen wird.

Daten manuell eingeben

Wählen Sie diese Option, wenn die Daten manuell (per API) eingegeben werden sollen. Für diese Option können Sie außerdem die Option **Kurvenpunkte äquidistant** wählen. Andernfalls werden Kurvenpunkte nur an den Stellen erzeugt, an denen sich der y-Wert ändert.

Weitere Informationen finden Sie im Kapitel "Wichtige Begriffe: Histogramme".

4.48 Dialogfeld "Typ des Skalenstreifens einstellen"



Sie erreichen dieses Dialogfeld über das Dialogfeld **Histogramm bearbeiten**.

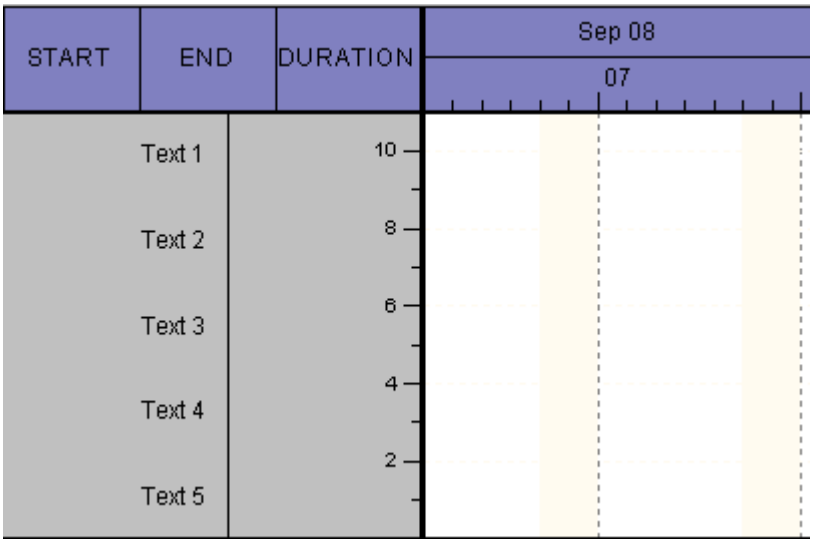
Numerisch

Wählen Sie diese Option, wenn der gewählte Skalenstreifen numerisch beschriftet werden soll.

Textuell

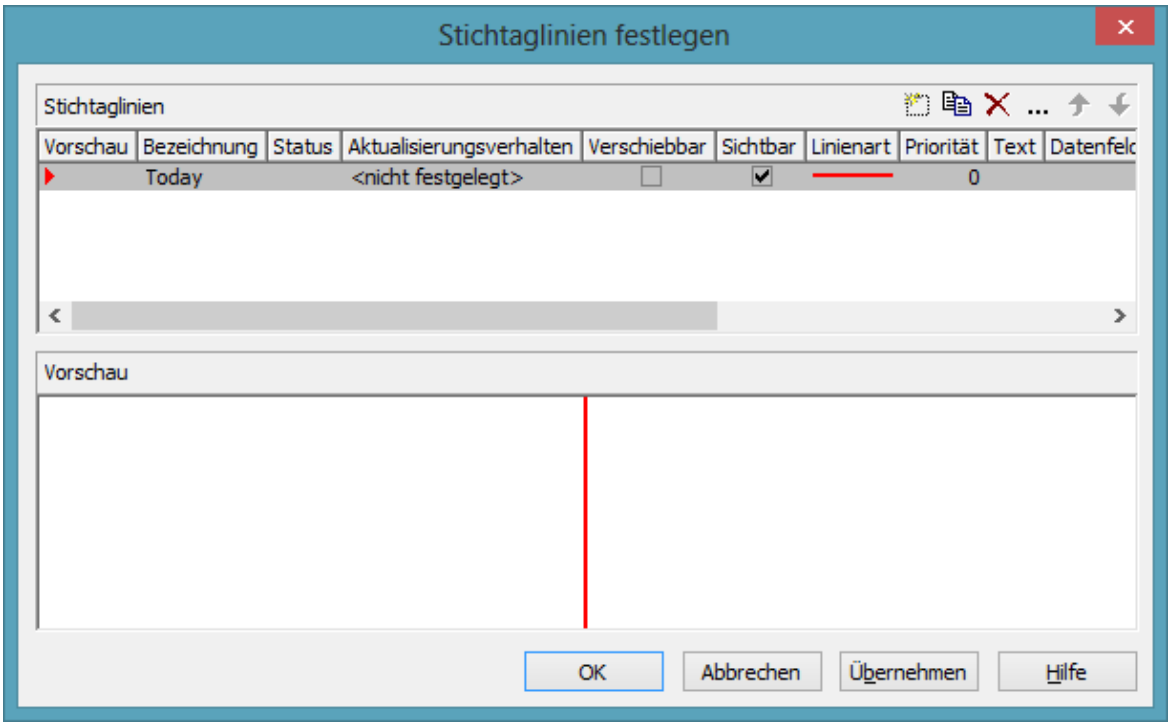
Wählen Sie diese Option, wenn der gewählte Skalenstreifen mit Texten Ihrer Wahl beschriftet werden soll. Sie können Text an insgesamt fünf vorgegebenen Positionen ausgeben (10%, 30%, 50%, 70 %, 90 % von oben).

Wenn Sie im Dialog **Histogramme bearbeiten** mehrere Skalenstreifen definiert haben, können Sie durch Anklicken der Option **Trennlinie anzeigen** bestimmen, ob rechts von dem jeweiligen Streifen eine senkrechte Trennlinie gezogen werden soll.



Textuelle Skala und numerische Skala

4.49 Dialogfeld "Stichtaglinien festlegen"



Durch Stichtaglinien (vertikale Linien im Diagramm) können Sie bestimmte Daten (das aktuelle Tagesdatum oder beliebig gesetzte Termine) besonders hervorheben. Alle Stichtaglinien, die in der Grafik dargestellt werden, werden in diesem Dialog verwaltet. Sie erreichen dieses Dialogfeld über die Eigenschaftenseite **Objekte**.

Vorschau

Die Stichtaglinie, die hier durch eine kleine rote Pfeilspitze gekennzeichnet ist, wird im Vorschaufenster angezeigt.

Bezeichnung

In dieser Spalte werden die Namen aller Stichtaglinien aufgeführt, die in der Grafik dargestellt werden. Die Bezeichnungen sind editierbar.

Status

In dieser Spalte wird jede Stichtaglinie 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 Stichtaglinien festgelegt wurde.

Verschiebbar

Aktivieren Sie dieses Kontrollkästchen, wenn die Stichtaglinie zur Laufzeit interaktiv verschiebbar sein soll.

Sichtbar

Aktivieren Sie dieses Kontrollkästchen, wenn die Stichtaglinie zur Laufzeit sichtbar sein soll.

Priorität

Hier können Sie die Priorität jeder Stichtaglinie festlegen (> 0: vor den Layern, < 0: hinter den Layern).

Text

Sie können hier einen Text eingeben, der an der Stichtaglinie ausgegeben werden soll.

Heute (dynamisch)

Aktivieren Sie dieses Kontrollkästchen, wenn die Stichtaglinie beim Start immer die aktuelle Systemzeit anzeigen soll. Das Feld **Datum** ist dann deaktiviert.

Datum

Sie können das Datum der Stichtaglinie editieren, indem Sie einen Teil des Datums markieren und dann mit Hilfe der Cursortasten einen neuen Wert auswählen. Oder klicken Sie auf die Pfeil-Schaltfläche (). Dann erscheint das Datum-Steuerelement. Darin ist das gewählte Datum markiert bzw. wenn kein Datum gewählt ist, das aktuelle Datum. Sie können per Mausklick einen Tag des dargestellten Monats auswählen und ggf. mit Hilfe der Pfeil-Schaltflächen von einem Monat zum anderen wechseln. Wenn Sie auf den Monatsnamen klicken, öffnet sich eine Kombobox mit den Monatsnamen.

Wenn Sie auf die Jahreszahl klicken, erscheinen Pfeil-Schaltflächen, mit denen Sie die Jahreszahl jeweils um ein Jahr erhöhen bzw. verringern können. Wenn Sie auf **Heute** klicken, wird das aktuelle Tagesdatum gewählt.

Datum

Aktivieren Sie dieses Kontrollkästchen, wenn die Stichtaglinie mittels der VcGantt-Methode **IdentifyObjectAt** identifizierbar sein soll.

Die Option kann auch mithilfe der Eigenschaft **VcDateLine.Identifiable** festgelegt werden.

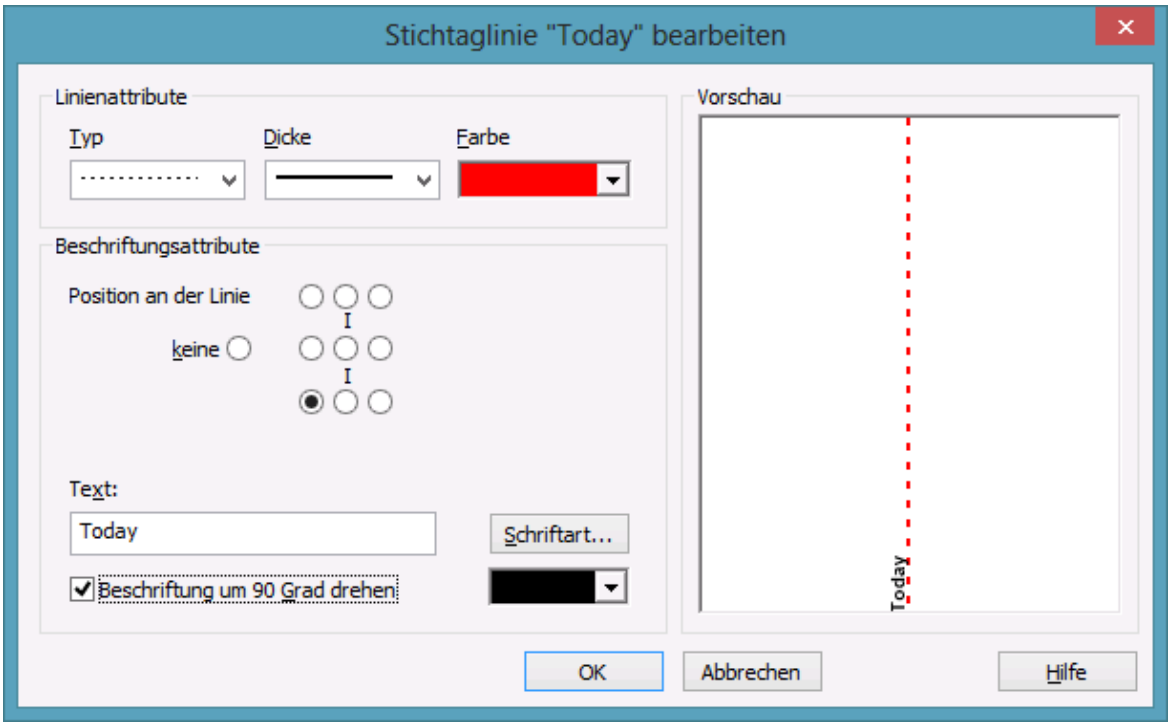
Einrastziel

Legen Sie hier fest, ob die Stichtaglinie ihre Position (und damit ihr Datum) als "Einrast"-Ziel für das Verschieben eines Knotens/Layers definiert.

Stichtaglinie hinzufügen/kopieren/löschen/eine Zeile nach oben/unten

 Mit Hilfe dieser Schaltflächen können Sie Stichtaglinien hinzufügen, kopieren, löschen oder in der Tabelle nach oben oder unten verschieben.

4.50 Dialogfeld "Stichtaglinie bearbeiten"



Linienattribute

Wählen Sie hier **Typ**, **Dicke** und **Farbe** der Stichtaglinie aus.

Position an der Linie

Wählen Sie hier aus, an welcher Position ggf. ein Text ausgegeben werden soll. Wenn kein Text ausgegeben werden soll, aktivieren Sie den Radiobutton **keine**. Er ist standardmäßig aktiviert, wenn kein Text für die Stichtaglinie eingegeben wurde. Wenn Sie einen Text für die Stichtaglinie eingegeben und das **Text**-Feld wieder verlassen haben, wird standardmäßig die Position oben rechts an der Linie gewählt. Sie können ggf. eine andere Position an der Linie wählen.

Text

Sie können hier einen Text eingeben, der an der Stichtaglinie ausgegeben werden soll. Standardmäßig ist das **Text**-Feld leer. Sobald Sie eine Position an der Linie wählen, wird der Name der Stichtaglinie in das **Text**-Feld gesetzt. Sie können den Text beliebig editieren.

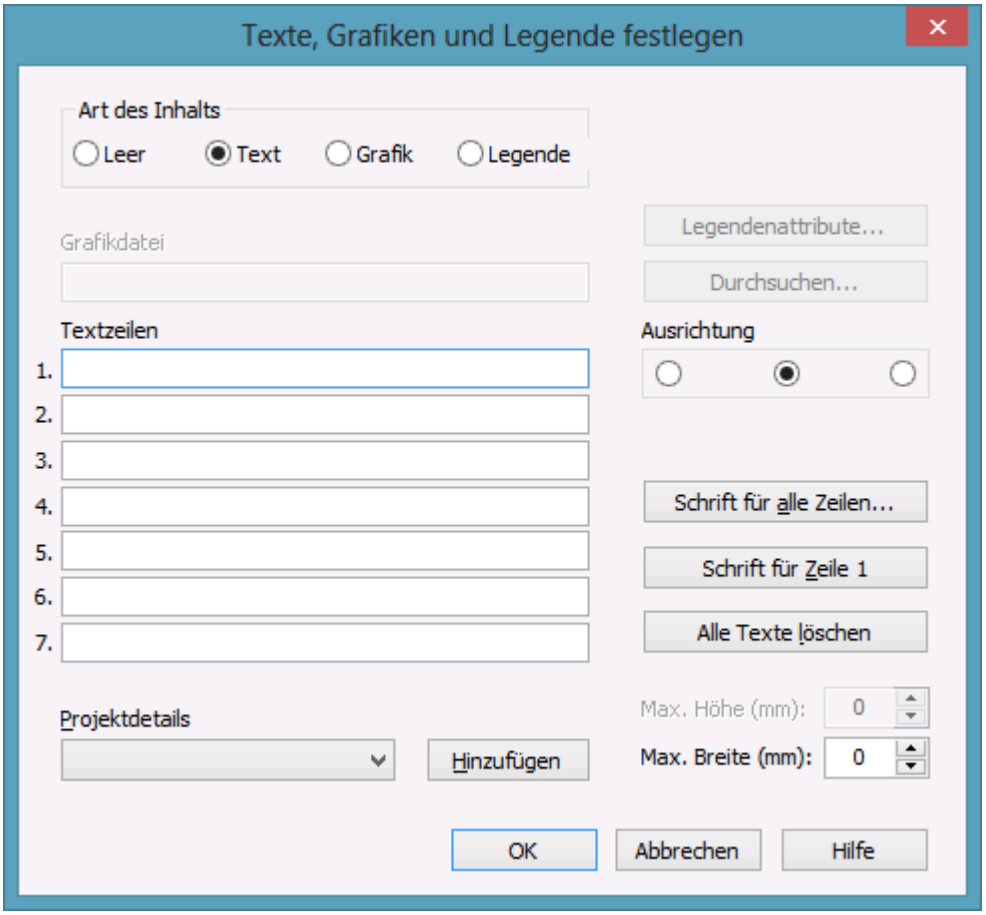
Schriftart

Über diese Schaltfläche gelangen Sie in den Windows-Dialog **Schriftart**, in dem Sie die Schriftart für die Beschriftung der Stichtaglinie festlegen können. Über die darunter liegende Schaltfläche gelangen Sie in den Windows-Farbselektor, über den Sie die Schriftfarbe wählen oder eine neue Farbe einrichten können.

Beschriftung um 90 Grad drehen

Aktivieren Sie dieses Kontrollkästchen, wenn die Beschriftung der Stichtaglinie in vertikaler Richtung ausgegeben werden soll.

4.51 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) plazieren. Grafiken werden immer mittig ausgerichtet.

- **Legende:** Eine Legende wird in dem gewählten Diagrammbereich ausgeben. 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.

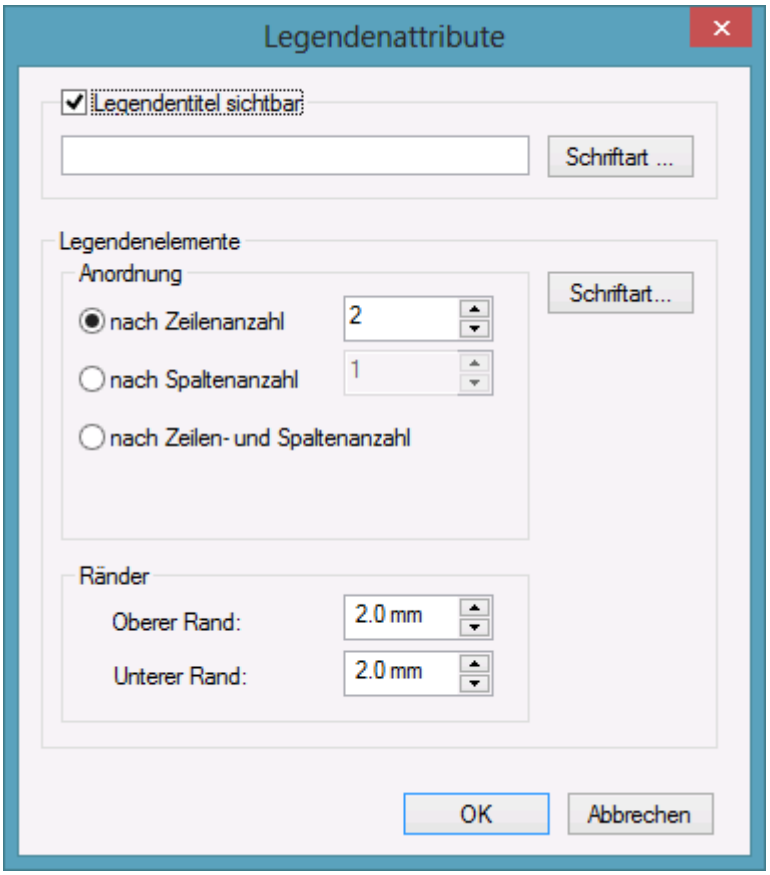
Max. Höhe (mm)

*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.

4.52 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 Layern, so werden die überzähligen Layer nicht dargestellt.

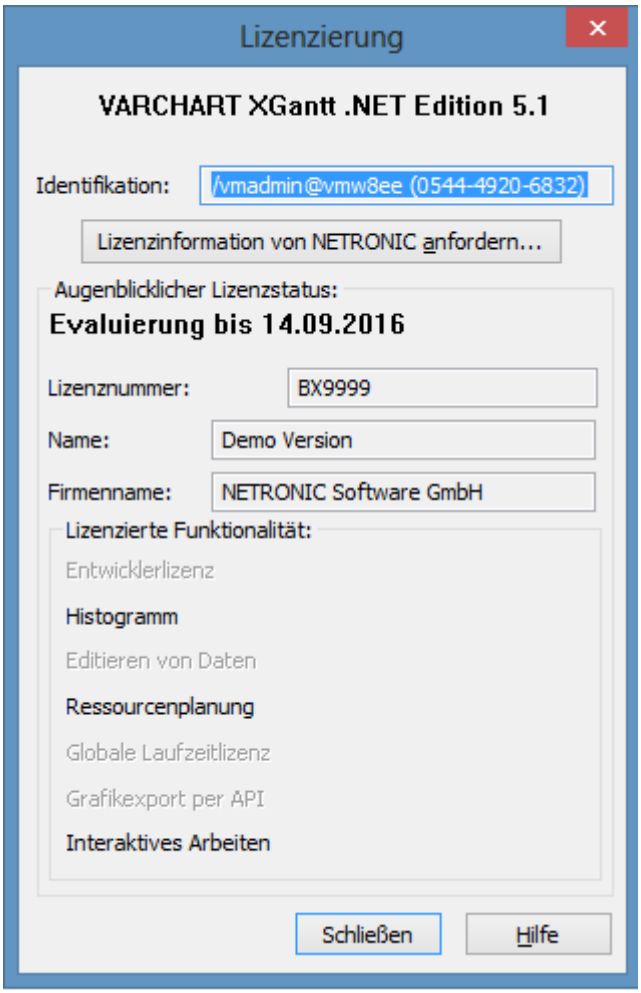
Ränder

- 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.

4.53 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

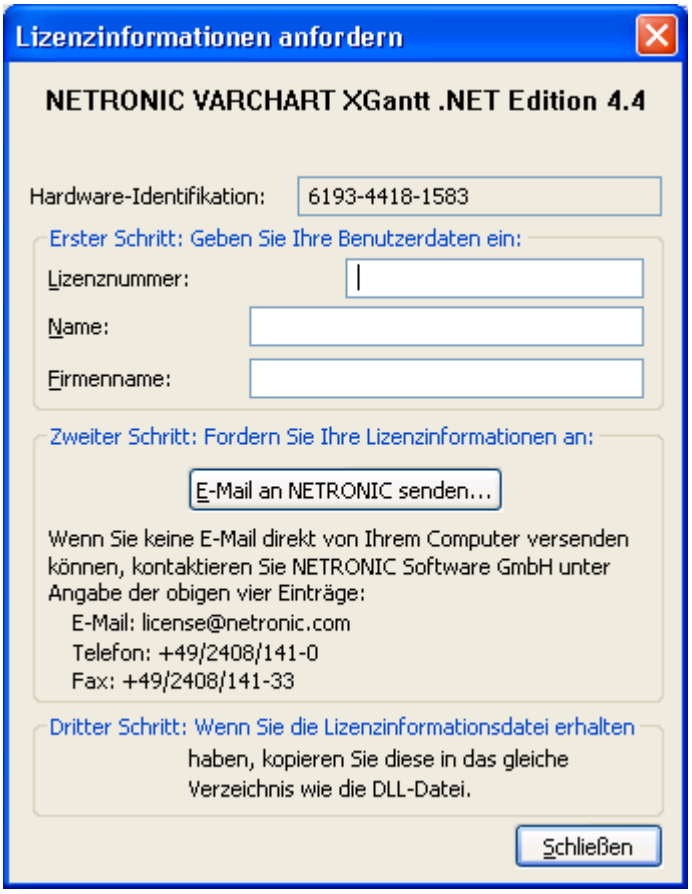
Hier wird angezeigt, welche Module für Sie freigegeben wurden. Wenn Sie die Lizenzierung vorgenommen haben, sind die freigegebenen Module aktiviert.

- **Entwicklerlizenz**
- **Histogramm**
- **Editieren von Daten** (ermöglicht sämtliche Funktionen zum Bearbeiten der Anwendungsdaten)
- **Ressourcenplanung** (benötigt alle übrigen Module und ermöglicht sämtliche Funktionen zur Ressourcenplanung)
- **Globale Laufzeitlizenz** (VARCHART XGantt läuft im Runtime-Modus auf jedem anderen Rechner.)
- **Einzelplatz-Laufzeitlizenzen** (VARCHART XGantt 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.

4.54 Dialogfeld "Lizenzinformationen anfordern"



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 **(NETRONIC.XGantt.VcGantt.lic)** generieren und sie Ihnen zusenden. Bitte kopieren Sie dann diese Datei in das Verzeichnis, in dem die Datei **NETRONIC.XGantt.dll** 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 in Diagramm und Tabelle
- Zoomen
- Knoten bzw. Layer markieren
- Knoten erzeugen
- Knoten verschieben
- Layer verschieben
- Anfangs-/Enddatum verändern
- Knoten löschen, ausschneiden, kopieren und einfügen
- Knotendaten bearbeiten
- Verbindungen bearbeiten
- Boxen an Knoten verankern
- Gruppendaten bearbeiten
- Gruppen expandieren/kollabieren
- Gruppen verschschieben
- Breitenverhältnis Tabelle/Diagramm bearbeiten
- Breite der Tabellenspalte bearbeiten
- Bearbeiten von Feldinhalten in der Tabelle
- Einfügen von Tabellenzeilen
- Zeitskala bearbeiten
- Skalierung und Grenzen von Zeitskalenabschnitten bearbeiten
- Stichtaglinien verschieben
- Legende bearbeiten
- Seite einrichten

- Druckvorschau verwenden

Kontextmenüs (rechte Maustaste):

- für das Diagramm
- für Knoten
- für Verbindungen
- für Gruppen
- für die Zeitskala
- für das Histogramm
- für die Legende
- für Boxen

Weitere Informationen zu Interaktionsmöglichkeiten bei gruppierter oder hierarchischer Anordnung finden Sie im Kapitel "Wichtige Begriffe" unter "Gruppierung" bzw. "Hierarchie".

Bei allen Interaktionen wird ein Ereignis ausgelöst, sodass Sie im Programm darüber informiert werden und ggf. darauf reagieren können.

5.2 Navigation in Diagramm und Tabelle

Folgende Tasten und Tastenkombinationen stehen für das Navigieren in Diagramm und Tabelle zur Verfügung:

- die Pfeiltasten verschieben die Markierung von Knoten oder Tabellenfeldern in die entsprechende Richtung (für weitere Informationen dazu, insbesondere auch zum Markieren innerhalb von Gruppen s. Kapitel 5.4 "Knoten bzw. Layer markieren")
- **Pos1:** an den linken Diagrammrand scrollen
- **Strg** + **Pos1:** an die linke obere Ecke des Diagramms scrollen
- **Ende:** an den rechten Diagrammrand scrollen
- **Strg** + **Ende:** an die rechte untere Ecke des Diagramms scrollen
- **Bild rauf/runter:** eine Bildschirmseite rauf/runter scrollen
- **Strg** + **Shift J:** zur nächsten Stichtaglinie scrollen
- **Strg** + ***** (Nummernblock): der Bildschirmausschnitt wird ggf. so verschoben, dass der Anfang des Knotens sichtbar ist

Auch mit der Maus kann navigiert werden:

- Drehen Sie das Mausrad, um vertikal im Diagramm bzw. im Histogramm (abhängig von der Cursorposition) zu scrollen
- Drücken Sie das Mausrad (bzw. die mittlere Maustaste) und bewegen die Maus entsprechend, um in sämtliche Richtungen zu scrollen.

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 Mausrad während Sie die Strg-Taste gedrückt halten. Dazu muss das Zoomen per Mausrad zugelassen sein. Dies geschieht entweder über die Option **Zoomen per Mausrad zulassen** auf der Eigenschaftenseite **Allgemeines** oder über die API-Eigenschaft **VcGantt1.-ZoomingPerMouseWheelAllowed**. (Diese Eigenschaft st standardmäßig deaktiviert.)

Weitere Information zu den Zoommöglicheiten für den Druck finden Sie in Kapitel 5.21 "Seite einrichten".

5.4 Knoten bzw. Layer markieren

Einen einzelnen Knoten markieren Sie durch Anklicken des Knotens mit der linken Maustaste. Dabei wird auch das jeweils erste Feld in der entsprechenden Tabellenzeile markiert.

Sie können jedoch auch direkt auf ein bestimmtes Feld in der Tabelle klicken - damit wird gleichzeitig der entsprechende Vorgang im Diagrammbereich markiert.

Mehrere Knoten, die sich im Diagrammbereich, unter- oder übereinander befinden, werden markiert, indem Sie bei gedrückter Umschalt-Taste diese Knoten im Diagrammbereich bzw. die entsprechenden Zeilen im Tabellenbereich anklicken, oder indem Sie mit der Maus ein Rechteck um die Knoten bzw. Zeilen aufziehen.

Mehrere Knoten die sich nicht unter- oder übereinander befinden, markieren Sie, indem Sie sie bei gedrückter Strg-Taste im Diagrammbereich oder im Tabellenbereich die entsprechenden Zeilen anklicken.

Die Markierungen können sowohl im Diagramm- als auch im Tabellenbereich mit Hilfe der Cursortasten oder der Enter-Taste in die entsprechende Richtung verschoben werden.

Bei Gruppen im Modus **Knoten in einer Zeile** und **optimiert** gilt Folgendes: Von oben kommend wird zuerst der erste Knoten der Gruppe markiert; von unten kommend zuerst der letzte Knoten der Gruppe. Innerhalb dieser Art von Gruppen kann mit den Pfeiltasten nach rechts bzw. links navigiert werden.

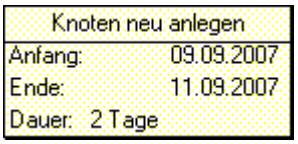
Hinweis: Durch erneutes Anklicken der Knoten bzw. Tabellenfelder/-zeilen oder mit Hilfe der Taste ESC werden die Markierungen wieder aufgehoben.

5.5 Knoten erzeugen

Um Knoten erzeugen zu können, müssen Sie erst den Erzeugemodus wählen, i. d. R. über das Standard-Kontextmenü für das Diagramm.

Der Modus **Knoten erzeugen** kann nur eingeschaltet werden, wenn auf der Eigenschaftenseite **Knoten** die Option **Erzeugung neuer Knoten zulassen** gesetzt ist.

Im Modus **Knoten erzeugen** nimmt der Mauszeiger die Form eines Kreuzchens an. Durch das Ziehen der Maus mit gedrückter linker Maustaste lassen sich in diesem Modus neue Knoten erzeugen. An der Position der Maus erscheint zunächst ein Informationsfenster, die den aktuellen Anfangs- und Endtermin sowie die Dauer des neuen Knotens anzeigt.



Wenn Sie bei mehrstufiger Gruppierung in einer kollabierten Gruppe einen Knoten anlegen, erscheint ein kleiner Pfeil an dem Kreuzchen. Dieser Pfeil zeigt an, ob der neu anzulegende Knoten als erster Knoten (Pfeil nach oben) bzw. als letzter Knoten (Pfeil nach unten) in die Gruppe eingefügt wird.

Bei expandierten Gruppen wird der neue Knoten immer als erster Knoten eingefügt, wenn sich der Mauszeiger in einer Gruppentitelzeile befindet.

Bei hierarchischer Gruppierung können Sie den neuen Knoten je nach Richtung des kleinen Pfeils immer vor oder hinter dem Bezugsknoten einfügen.

Wenn die Option **Neuen Knoten bearbeiten** auf der Eigenschaftenseite **Knoten** 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 nichts anderes vereinbaren, erscheint der neu angelegte Vorgang an der durch die Maus bestimmten Position.

Der **Modus: Knoten erzeugen** lässt sich auch über das Setzen der Eigenschaft **InteractionMode** auf den Wert **VcCreateNode** aktivieren.

Das Ereignis **VcNodeCreating** tritt ein, wenn der Anwender interaktiv einen Knoten erzeugt hat. Das Knotenobjekt wird als Parameter übergeben, sodass eine Datenvalidierung (der Dialog **Vorgänge bearbeiten** ist ggf. vorher aktiviert) vorgenommen werden kann. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird der Knoten wieder gelöscht.

5.6 Knoten verschieben mit der Maus

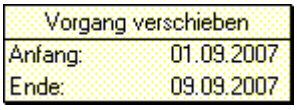
Verschieben von Knoten im Diagramm

Das Verschieben von Knoten im Diagramm führt je nach den Einstellungen auf der Eigenschaftenseite **Knoten** zu unterschiedlichen Ergebnissen. Im Folgenden werden die Möglichkeiten beschrieben, die Ihnen bei den Standardeinstellungen zur Verfügung stehen (für Informationen zu den davon abweichenden Einstellungen lesen Sie bitte das Kapitel 4.4: "Eigenschaftenseite Knoten"):

- Markierten Knoten als Ganzes verschieben
- Layer bei gedrückter Shift-Taste als Knoten verschieben

Wenn Sie mit der Maus auf einen Knoten zeigen, wird der Mauszeiger zu einem kleinen Quadrat mit je einem Pfeil rechts und links bzw. vier Pfeilen, falls der Knoten nur durch einen einzigen Layer dargestellt wird. Nun können Sie den gewünschten Layer mit gedrückter linker Maustaste verschieben.

Wenn Sie den kompletten Knoten (mit allen Layern) verschieben möchten, zeigen Sie mit gedrückter Umschalt-Taste auf den gewünschten Knoten. Der Cursor wird zu einem kleinen Quadrat mit vier Pfeilen. Halten Sie die Umschalt-Taste weiter gedrückt und ziehen den Knoten mit Hilfe der Maus an eine andere Position. Es erscheint eine Infobox, die Ihnen das aktuelle Anfangs- und Enddatum des Knotens angibt. Sobald Sie die Maustaste loslassen, bleibt der Knoten an der zuletzt gewählten Position, und die Infobox schließt sich wieder.



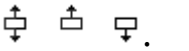
Hinweis: Die Umschalt-Taste muss nur gedrückt werden, wenn Sie einen Knoten verschieben möchten, der aus mehreren Layern besteht.

Wenn auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Verändern der Knotenreihenfolge über das Diagramm zulassen** aktiviert ist, können Knoten im Diagramm auch in **vertikaler** Richtung verschoben werden.

Wenn Sie einen Knoten vertikal verschieben, zeigt ein Cursor mit entsprechenden Pfeilen an, wie der Knoten relativ zu den anderen angelagert wird: .

Verschieben von Knoten in der Tabelle

Auch in der Tabelle lassen sich Knoten verschieben, wenn Sie auf der Eigenschaftenseite **Knoten** das Kontrollkästchen **Verändern der Knotenreihenfolge über die Tabelle zulassen** aktiviert haben. Allerdings lassen sich bislang in der Tabelle nur komplette Knoten nur vertikal verschieben.

Wenn Sie einen Knoten vertikal verschieben, zeigt ein Cursor mit entsprechenden Pfeilen an, wie der Knoten relativ zu den anderen angelagert wird:  .

Wenn ein Knoten interaktiv verändert worden ist (Verschieben, Veränderung des Anfangs- und/oder Enddatums des Knotens oder 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 der Knoten nicht verändert.

5.7 Knoten verschieben und Dauer ändern mit der Tastatur

Standardmäßig sind die Pfeiltasten <links> und <rechts> mit navigierenden Funktionen wie Scrollen im Diagramm oder Verschieben der Markierung von Knoten, Layern oder eines Tabellenfeldes belegt. Sie können den Pfeiltasten aber über die Eigenschaft **VcGantt.ArrowKeyMode** einen Modus zuweisen, in dem Knoten verändert werden können.

In diesem Modus kann der Benutzer einen Knoten verschieben, vergrößern oder verkleinern:

- **Verschieben**

  Der Knoten wird beim Tastendruck direkt verschoben. Die kleinste Schrittweite entspricht der des Verschiebens mit der Maus. Um den Knoten schneller zu verschieben, lässt sich die Schrittweite über die Eigenschaft **VcGantt.ArowKeyStepMultiplier** vergrößern und durch zusätzliches Drücken der <Strg>-Taste aktivieren.

  Tastenfunktionen:

  Pfeiltaste links/rechts: Knoten verschieben

  <Strg> + Pfeiltaste <links/rechts>: Schrittweite modifizieren

- **Ändern der Dauer**

  Die Dauer kann nur für alle **sichtbaren** Layer und nur, wenn nur ein Knoten markiert ist, über die Pfeiltasten verändert werden. Auch hier kann man mit dem oben beschriebenen Multiplikator arbeiten, der durch <Strg> aktiviert wird.

  Tastenfunktionen:

  <Umschalttaste > + Pfeiltaste <links/rechts>: Dauer ändern

  <Umschalttaste > + <Strg> + Pfeiltaste <links/rechts>: Schrittweite modifizieren

Ein Info-Fenster mit Informationen zur Position bleibt noch kurze Zeit nach Beendigung der Interaktion sichtbar, damit die Information gelesen werden kann.

Nähere Informationen zu den entsprechenden API-Eigenschschaften finden Sie in der API-Referenz.

Wenn ein Knoten interaktiv verändert worden ist (Verschieben, Veränderung des Anfangs- und/oder Enddatums des Knotens oder 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 der Knoten nicht verändert.

5.8 Layer verschieben

Wenn Sie mit dem Mauszeiger auf einen Layer zeigen, erscheint der Verschiebecursor. Nun können Sie den Layer bei gedrückter linker Maustaste mit der Maus horizontal verschieben. Sobald Sie die Maustaste wieder loslassen, wird der Layer an der aktuellen Position positioniert. Während Sie den Layer verschieben, zeigt Ihnen die Infobox **Layer verschieben** die aktuellen Start- und Enddaten des Layers an. Die Dauer bleibt dabei erhalten. Layer können nur innerhalb einer Zeile, aber nicht von einer Zeile in eine andere verschoben werden.

Wenn Sie einen Symbol-Layer verschieben, sieht die Infobox **Layer verschieben** folgendermaßen aus:

5.9 Anfangs-/Enddatum verändern

Sie können auf dieselbe Weise auch nur das Anfangs- bzw. nur das Enddatum eines Layers verändern, wenn Sie den Cursor am äußeren rechten oder linken Rand des Layers positioniert haben. Dann erscheint eine Infobox, die Ihnen den aktuellen Anfangs- bzw. Endtermin anzeigt. Die Dauer wird dabei verändert.

Das Verschieben des ganzen Layers ist nur möglich, wenn für den entsprechenden Layer in der Designphase im Dialogfeld **Layer bearbeiten** unter **Veränderbar** die entsprechende Schaltfläche aktiviert worden ist. Wenn ein Knoten interaktiv verändert worden ist (Veränderung des Anfangs- und/oder Enddatums des Knotens oder 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** können Sie die Veränderung unterdrücken.

5.10 Knoten löschen, ausschneiden, kopieren und einfügen

Mit der Entf-Taste können Sie markierte Knoten löschen.

Mit Strg-X können Sie markierte Knoten ausschneiden, mit Strg-C kopieren.

Wenn Sie die Umschalt-Taste gedrückt halten, können Sie mit den Pfeiltasten (oben bzw. unten) mehrere Knoten markieren.

Liegt innerhalb des gewählten Bereichs eine Gruppe im Modus **Knoten in einer Zeile** und **optimiert**, werden alle Knoten dieser Gruppe markiert.

Ist der erste Knoten einer solchen Gruppe markiert und geht man mit gedrückter Umschalt-Taste in eine andere Zeile, werden alle Knoten sowohl in der Ziel- als auch in der Startzeile markiert.

Sie können kopierte oder ausgeschnittene Knoten mit Strg-V oberhalb der Zielzeile (der Zeile, in der ein Knoten markiert ist) einfügen.

Sollen die Knoten unterhalb der Zielzeile eingefügt werden, ist Umschalt-Strg-V zu verwenden.

Die Einfügeposition relativ zum Bezugsknoten wird durch entsprechende Pfeile am Mauszeigersymbol angezeigt.

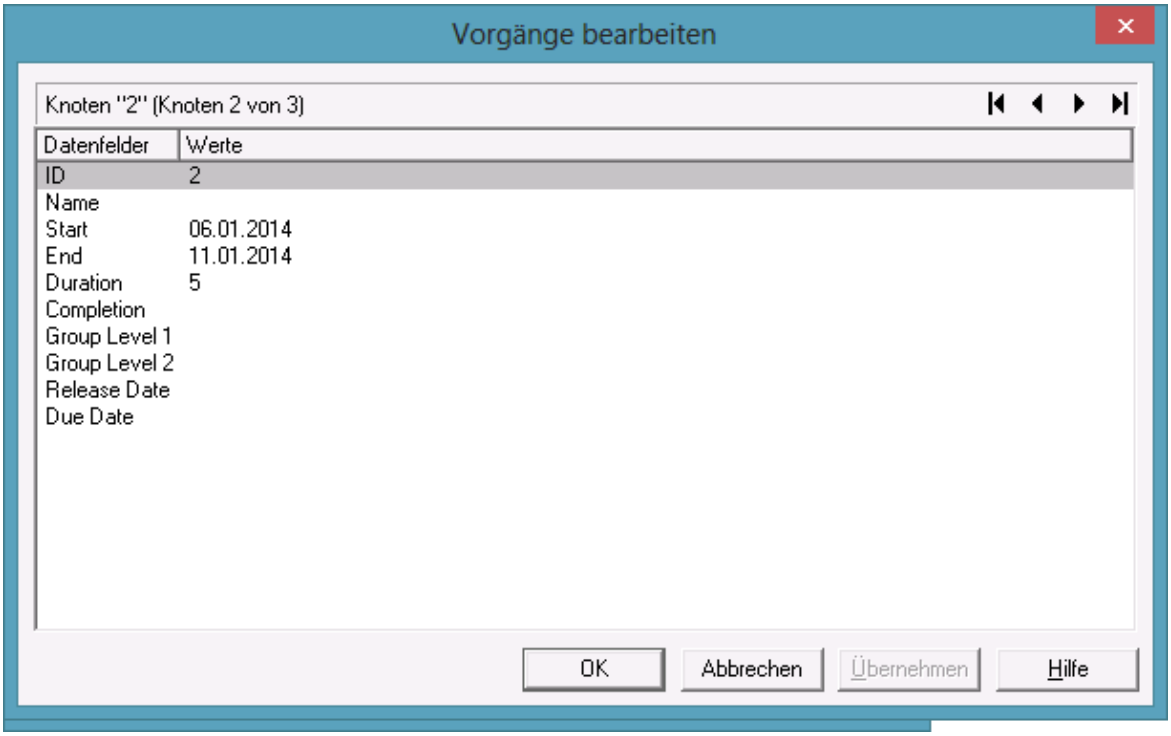
Beim Einfügen von Knoten wird die Reihenfolge von gruppierten Knoten beibehalten.

Knoten können nicht in eine leere Gruppe im Modus **Knoten in einer Zeile** und **optimiert** eingefügt werden.

5.11 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

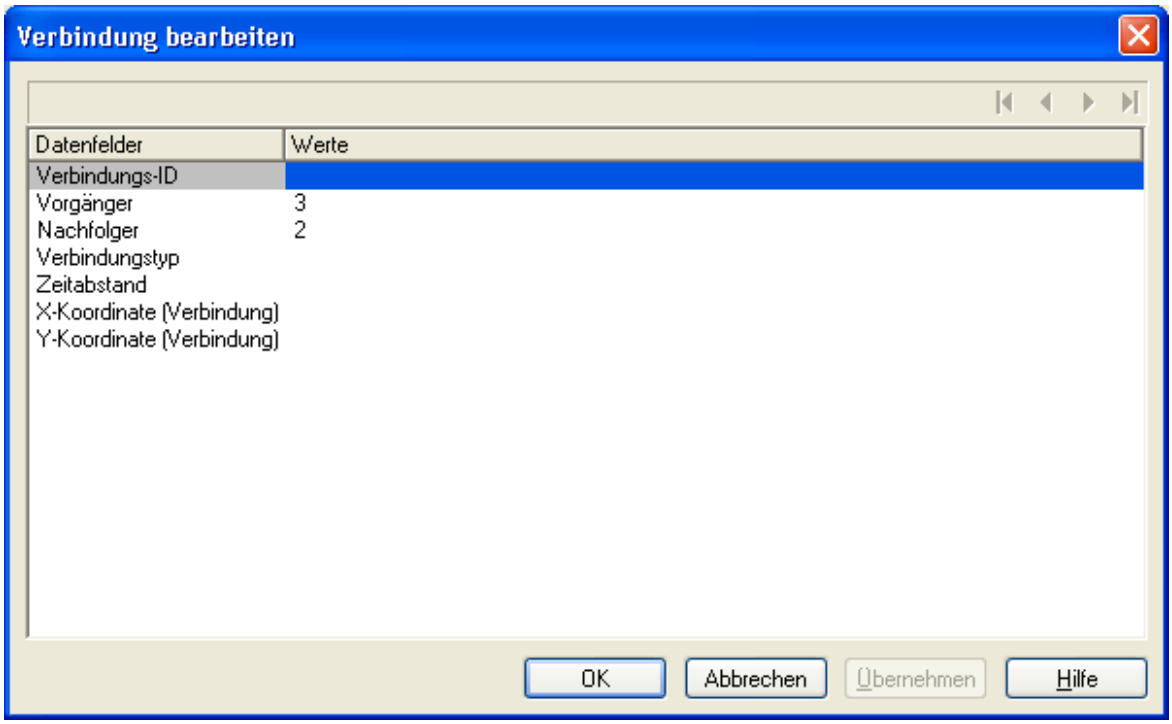
Sie können in dieser Spalte alle Werte des markierten Knotens direkt bearbeiten, sofern sie im Dialog **Datentabellen verwalten** als **editierbar** definiert worden sind.

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.

5.12 Verbindungen bearbeiten



Dieses Dialogfeld können Sie mit der Methode **VcGantt.EditLink** aufrufen. Hier können Sie die Daten einer markierten Verbindung ansehen und bearbeiten.

An erster Stelle wird die Identifikation (ID) der markierten Verbindung angezeigt.

5.13 Box an Knoten verankern

Sie können eine Box entweder interaktiv (mit der Maus + gedrückter Umschalt-Taste oder über das Kontextmenü) oder über die API mit einem Knoten verankern.

- **Verankern mit der Maus:** Zeigen Sie mit der Maus auf die gewünschte Box und drücken Sie die Umschalt-Taste. Ein kleiner Anker erscheint. Nun ziehen Sie mit gedrückter linker Maustaste eine Linie von der Box bis zu dem gewünschten Knoten und lassen die Maustaste wieder los. Die Box ist nun mit dem Knoten verankert und bei aktivierter Option **Ankerlinie sichtbar** im Dialog **Boxen verwalten** erscheint eine Verbindungslinie. Um die Verankerung wieder zu lösen, verfahren sie genauso.
- **Verankern über das Kontextmenü:** Markieren Sie den Knoten, mit dem die Box verankert werden soll und wählen dann aus dem Kontextmenü der Box den Befehl **Box am markierten Knoten verankern**. Sollte das Kontextmenü nicht erscheinen, muss auf der Eigenschaftenseite **Allgemeines** die Option **Kontextmenü für Boxen anzeigen** aktiviert werden.

Zum Lösen der Box von dem Knoten wählen Sie aus dem Kontextmenü den entsprechenden Befehl.

Wenn Sie die Box mit einem anderen Knoten verankern möchten, verfahren Sie genauso wie oben beschrieben

- **Verankern über API:** Bitte lesen Sie dazu in der API-Referenz die Beschreibung der Eigenschaft **AnchoringInteractionsAllowed** sowie der Methode **AnchorToNode** des Objektes **VcBox**

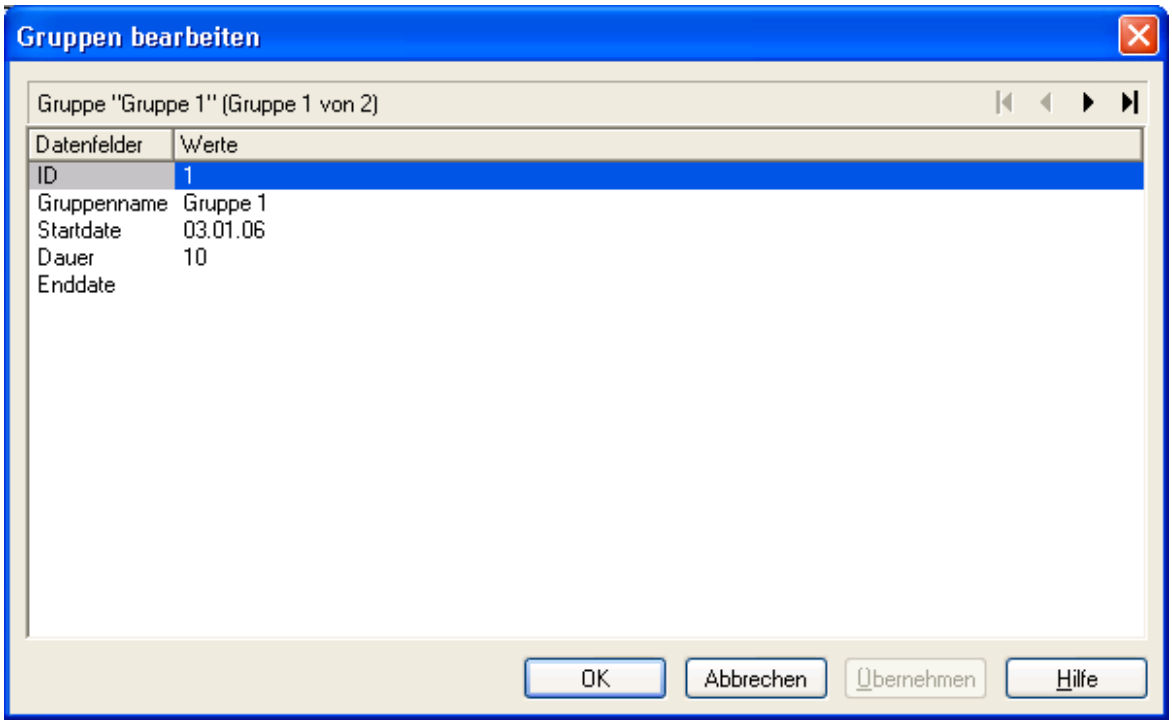
Eine verankerte Box kann weiterhin interaktiv verschoben werden (Voraussetzung: die Option **Verschiebbar** ist aktiviert).

Wird ein Knoten mit einer verankerten Box verschoben, so wird die Box entsprechend mitverschoben. Wird der Knoten kollabiert, wird auch die Box kollabiert d.h. sie ist nicht mehr sichtbar. Sobald der Knoten expandiert wird, ist auch die Box wieder sichtbar.

Bei interaktiver Ver/Entankerung bleibt die Position der Box auf dem Bildschirm erhalten - die zugrunde liegenden Offset-Werte werden entsprechend der Referenzpunkte (Origin, ReferencePoint) umgerechnet. Wenn sich also eine Textbox mit einem bestimmten Offset z.B. auf das Diagramm oben links (Origin) bezieht und dann mit einem Knoten verankert wird, wird automatisch ein Offset zum Knoten oben links berechnet und dieser Offset führt dazu, dass die Position auf dem Bildschirm beibehalten wird. Wird die Box vom Knoten gelöst, erfolgt eine Rückberechnung.

Bei Verwendung der API-Eigenschaft **AnchorToNode** wird genauso verfahren, nicht aber bei der Eigenschaft **NodeID**.

5.14 Gruppendaten bearbeiten



Dieses Dialogfeld erreichen Sie entweder über das Kontextmenü für Gruppen oder durch einen Doppelklick auf einen Gruppenbalken (der nur angezeigt wird, wenn im Dialog **Gruppierung** die Option **Gruppenknoten sichtbar** aktiviert wurde).

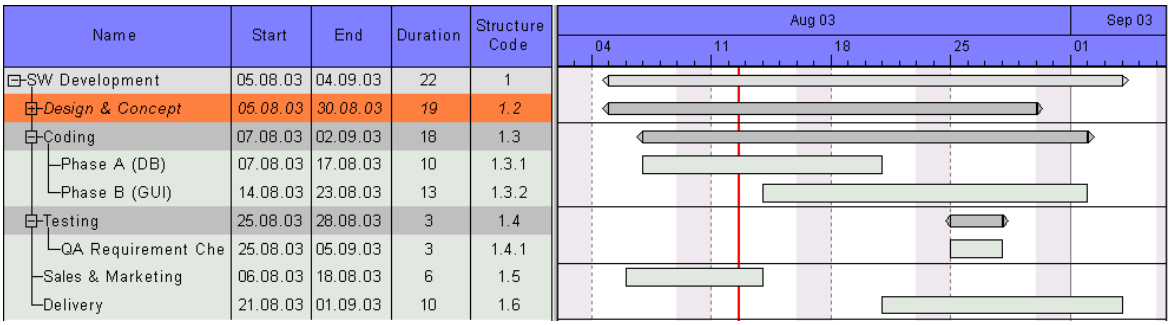
Hier können die Daten einer Gruppe, bzw. wenn mehrere Gruppen markiert sind, sukzessive die Daten aller markierter Gruppen bearbeitet werden.

Oberhalb der Tabelle wird angezeigt um die wievielte Gruppe der markierten Knoten es sich handelt.

Mit Hilfe der Pfeil-Schaltflächen können Sie zwischen den Gruppen navigieren.

5.15 Gruppen kollabieren bzw. expandieren

Wenn bei gruppierter Darstellung im Dialog **Gruppierung** das Kontrollkästchen **Änderungen erlaubt** aktiviert ist, können Sie durch einen Doppelklick auf die Gruppenüberschrift oder durch Anklicken des **Plus**- oder **Minus**-Symbols neben der Gruppenüberschrift eine kollabierte Gruppe expandieren bzw. eine expandierte Gruppe kollabieren.



Das Ereignis **VcGroupModifying** tritt ein, wenn der Anwender eine Gruppe interaktiv verändert. Das getroffene Group-Objekt, die Art der Veränderung und der Rückgabestatus werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Veränderung unterdrückt.

5.16 Gruppen verschieben

Gruppen lassen sich vertikal sowohl im Tabellen- als auch im Diagrammteil (dort über den Summenbalken) verschieben, wenn im Dialog **Gruppierung** die Optionen **Vertikal Verschieben über den Tabellenbereich** und/oder **Vertikal Verschieben über den Diagrammbereich** ausgewählt wurden. Beim Verschieben wird durch einen entsprechenden Cursor angezeigt, wo die Gruppe eingefügt wird  .

Hinweis. Eine Gruppe lässt sich nur innerhalb einer Parentgruppe verschieben.

5.17 Bearbeiten von Feldinhalten in der Tabelle

Zum Bearbeiten eines Feldinhaltes klicken Sie das gewünschte Feld an und geben entweder einen neuen Inhalt ein oder ändernd den bestehenden ab.

Weitere Möglichkeiten, den Feldinhalt zu bearbeiten, bieten sich Ihnen, wenn Sie die Option **Erweitertes Editieren in Tabelle zulassen** auf der Eigenschaftenseite **Allgemeines** aktiviert haben.

Dann erscheint bei Datumsfeldern ein Drop-Down-Pfeil, den Sie anklicken können, um das Datum-Steuerelement anzuzeigen. Für weitere Informationen zum Gebrauch des Datum-Steuerelementes s. Kapitel 4.40 Dialogfeld "Stichtaglinien festlegen".

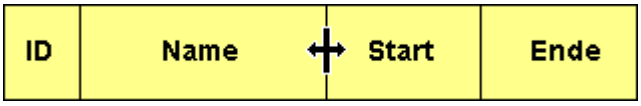
Bei numerischen Feldern finden sich Pfeil-Schaltflächen, mit deren Hilfe Sie den Wert jeweils erhöhen bzw. verringern können.

Eine ausführliche Beschreibung der erweiterten Editiermöglichkeiten in der Tabelle finden Sie in Kapitel 4.2 "Eigenschaftenseite Allgemeines".

Hinweis: Mit Hilfe der Taste ESC verlassen Sie die editierten Felder, ohne die Änderungen zu übernehmen.

5.18 Breite der Tabellenspalte bearbeiten

Die Breite einer Tabellenspalte können Sie interaktiv verändern, indem Sie im Bereich der Tabellenüberschrift mit dem Mauszeiger die Trennlinie zwischen zwei Spalten verschieben.



Das Verändern der Breite einer Tabellenspalte kann nur im Bereich der Tabellentitel erfolgen.

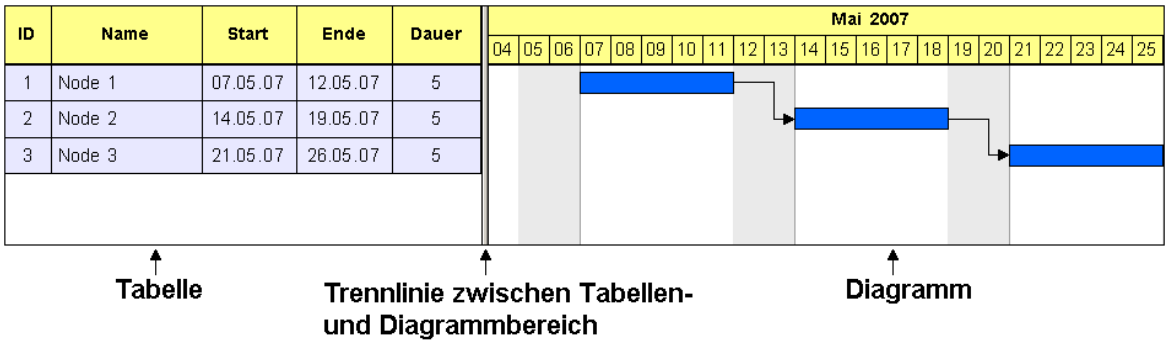
Das Ereignis **VcTableWidthChanging** tritt ein, wenn die Tabellenbreite interaktiv worden ist. Die Tabelle und das neue Diagramm-Breitenverhältnis werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Änderung rückgängig gemacht.

Das Ereignis **VcTableColumnWidthChanging** tritt ein, wenn der Anwender die Breite einer Tabellenspalte interaktiv verändert hat. Die Tabelle, der Index der geänderten Spalte und die neue Spaltenbreite (in 1/100 mm) werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Änderung rückgängig gemacht.

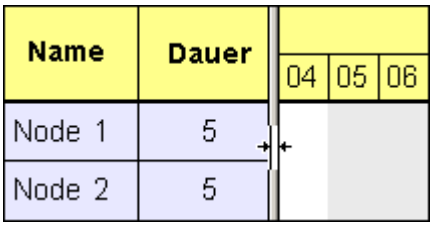
Sie können die Spaltenbreite auch automatisch berechnen lassen. Die Voraussetzung dafür ist, dass auf der Eigenschaftenseite **Allgemeines** die Option **Tabellenspalten optimierbar** aktiviert ist. Dann kann der Anwender durch einen Doppelklick auf die Trennlinie zwischen zwei Spalten erreichen, dass die Breite der linken Spalte automatisch an die Länge der darin enthaltenen Texte angepasst wird. Dabei wird das Ereignis **VcTableColumnWidthOptimizing** ausgelöst. Wenn die Optimierung erfolgt ist, wird das Ereignis **VcTableColumnWidthChanging** ausgelöst.

5.19 Breitenverhältnis Tabelle/ Diagramm bearbeiten

Die Tabellentitel und das Diagramm werden durch eine vertikale Trennlinie (Splitterbar) voneinander getrennt.



Wenn Sie im Bereich der Tabellenzeilen oder des Diagramms mit dem Mauszeiger darüber fahren, wird der Mauszeiger zu einer vertikalen Doppellinie mit je einem Pfeil von links und rechts.



Der Mauszeiger muss im Bereich von Tabellenzeilen und Diagramm, aber nicht im Bereich der Tabellentitel und der Zeitskala stehen.

Nun können Sie das Breitenverhältnis von Tabellen und Diagramm durch Ziehen mit der Maus verändern. (Der Tabellenbereich kann maximal so breit werden, wie im Dialog **Tabelle bearbeiten** als Summe der einzelnen Tabellenspaltenbreiten definiert wurde.)

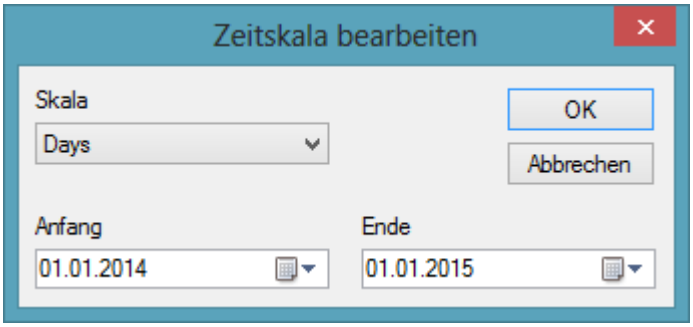
5.20 Einfügen von neuen Tabellenzeilen

Wenn auf der Eigenschaftenseite **Allgemeines** die Option **Erweitertes Editieren in Tabelle zulassen** aktiviert ist, lässt sich mithilfe der Taste EINFG oberhalb der jeweils markierten Zeile eine neue Tabellenzeile einfügen. Wenn nichts markiert ist, wird die neue Zeile am Ende der Tabelle eingefügt.

5.21 Zeitskala bearbeiten

Im Dialogfeld **Zeitskala bearbeiten** können Sie eine vordefinierte Zeitskala (Minuten, Stunden, Tage, Wochen, Monate) auswählen sowie deren Anfang und Ende festlegen. Wenn mehrere Zeitskalenabschnitte definiert sind, darf hier der Anfang der Zeitskala nicht in den zweiten Zeitskalenabschnitt gelegt werden.

Sie erreichen das Dialogfeld durch einen Doppelklick auf die Zeitskala oder über ihr Kontextmenü.



Wenn mehrere Zeitskalenabschnitte definiert sind, darf hier der Anfang der Zeitskala nicht in den zweiten Zeitskalenabschnitt gelegt werden.

Das Ereignis **VcTimeScaleLeftDoubleClicking** tritt ein, wenn der Anwender mit der linken Maustaste auf die Zeitskala doppelt klickt. Das getroffene TimeScale-Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. Das integrierte **Zeitskala bearbeiten**-Dialogfeld kann durch Setzen des Rückgabestatus auf **vcRetStatFalse** unterdrückt werden.

Das Dialogfeld "Zeitskala bearbeiten"

Skala

Hier können Sie die gewünschte Zeitskala auswählen. Zur Auswahl stehen Minuten, Stunden, Tage, Wochen und Monate.

Anfang

In diesem Feld legen Sie den Anfangstermin der Zeitskala fest.

Wenn Sie auf die Pfeil-Schaltfläche klicken, erscheint ein kleiner 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.

Ende

In diesem Feld legen Sie den Endtermin der Zeitskala fest.

Wenn Sie auf die Pfeil-Schaltfläche klicken, erscheint ein kleiner 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.

5.22 Skalierung und Grenzen von Zeitskalenabschnitten bearbeiten

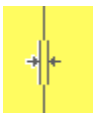
Skalierung von Zeitskalenabschnitten bearbeiten

Sie können einen Zeitskalenabschnitt interaktiv skalieren, indem Sie den Mauszeiger auf diesem Zeitskalenabschnitt positionieren und dann bei gedrückter linker Maustaste die Maus nach links oder rechts ziehen. Dabei ändert der Mauszeiger seine Form zu einer vertikalen Linie mit je einem Pfeil nach links und rechts. Ziehen Sie den Mauszeiger nun nach links, wird die Einheitenbreite des Zeitskalenabschnitts verkleinert; ziehen Sie den Mauszeiger nach rechts, wird sie vergrössert. Während des Ziehens mit der Maus zeigt Ihnen eine Infobox, um welchen Prozentfaktor die Einheitenbreite des Zeitskalenabschnitts (Sektion) verändert wird.

Hinweis: Wenn der Mauszeiger im Bereich der Zeitskala in der Nähe des Sash steht und Sie den Mauszeiger dann nach rechts bzw. links ziehen, wird die Zeitskala stark gedehnt bzw. gestaucht. Diese Reskalierung können Sie wieder rückgängig machen, indem Sie mit dem Cursor in der Zeitskala in die entgegengesetzte Richtung ziehen.

Das Ereignis **VcTimeScaleSectionRescaling** tritt ein, wenn der Anwender einen Abschnitt der Zeitskala interaktiv skaliert. Die Zeitskala, die Nummer des Zeitskalenabschnitts und die neue Grundeinheitsbreite (**BasicUnitWidth**) werden als Parameter übergeben, sodass Sie beispielsweise die Zulässigkeit der Skalierung prüfen können. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Änderung rückgängig gemacht.

Grenzen von Zeitskalenabschnitten bearbeiten



Die Grenze zwischen zwei Zeitskalenabschnitten können Sie interaktiv verändern, indem Sie mit der Maus die Linie zwischen diesen beiden Zeitskalenabschnitten verschieben. Dabei ändert der Mauszeiger seine Form zu einer vertikalen Doppellinie mit je einem Pfeil nach links und rechts.

Das Ereignis **VcTimeScaleSectionStartModifying** tritt ein, wenn der Anwender den Anfang eines Zeitskalenabschnitts interaktiv verändert. Die Zeitskala, die Nummer des Zeitskalenabschnitts und das neue Anfangsdatum

werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Änderung rückgängig gemacht.

5.23 Stichtaglinie verschieben

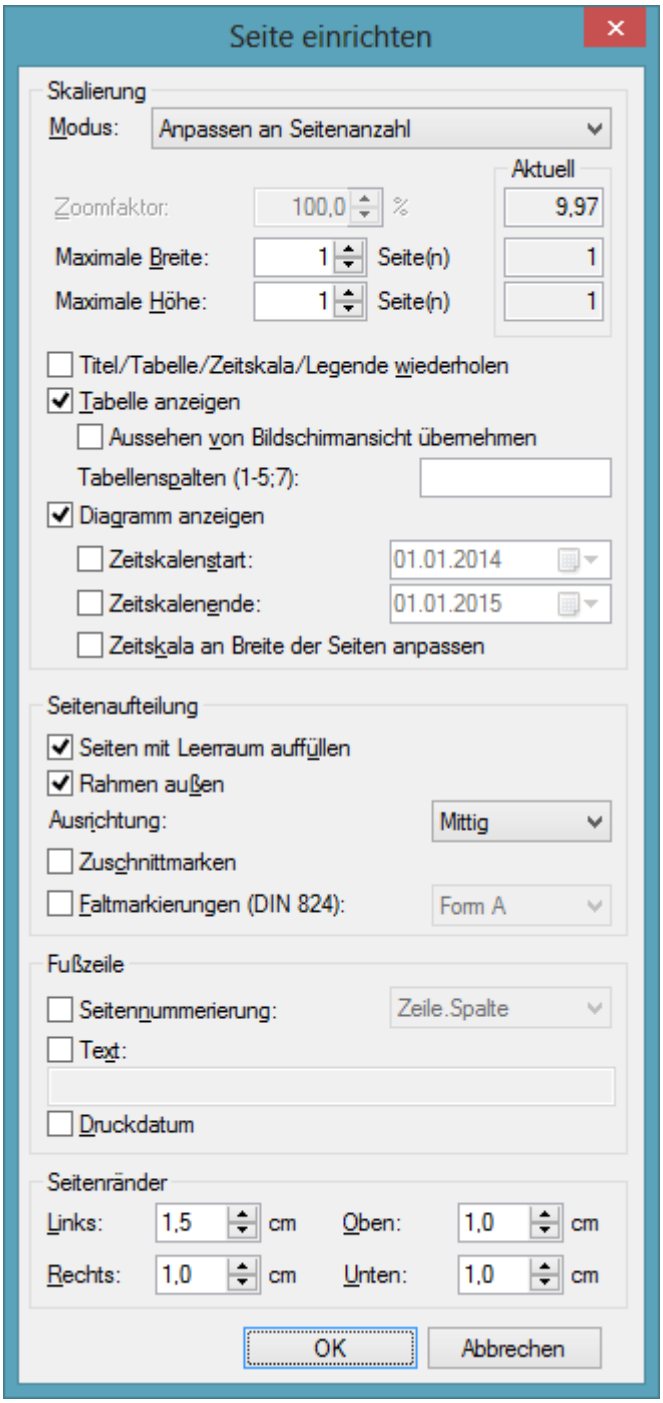
Das Stichtagdatum können Sie interaktiv verändern, indem Sie mit der Maus die Stichtaglinie verschieben. Die Voraussetzung dafür ist, dass im Dialog **Stichtaglinien festlegen** das Kontrollkästchen **Frei verschiebbar** für die betreffende Stichtaglinie aktiviert ist.

Stichtaglinien können auch per API gesetzt werden.

Das Ereignis **VcDateLineModifying** tritt ein, wenn eine Stichtaglinie durch den Anwender interaktiv verschoben wurde. Die veränderte Stichtaglinie wird als Parameter zurückgegeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Änderung rückgängig gemacht.

5.24 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.

Zoomen mit horizontaler Anpassung

Wählen Sie einen Zoomfaktor sowie eine feste Anzahl an Druckseiten in der Breite, die beim Druck durch Stauchen oder Strecken der Zeitskala eingehalten wird.

Titel/Tabelle/Zeitskala/Legende wiederholen

Aktivieren Sie dieses Kontrollkästchen, damit bei der Ausgabe des Diagramms auf mehreren Seiten auf jeder Seite Titel, Tabelle, Zeitskala und Legende ausgegeben werden.

Tabelle anzeigen

Bestimmen sie, ob die Tabelle gedruckt wird oder nicht. Wenn die Option nicht ausgewählt ist, wird die Tabelle nicht gedruckt.

Aussehen von Bildschirmansicht übernehmen

Hier können Sie festlegen, ob bei der Druckvorschau und beim Druck die aktuell auf dem Bildschirm sichtbare Breite der Tabelle übernommen werden soll.

Diese Option kann auch über die Eigenschaft **VcPrinter.TableWidth-AdoptionFromViewOnScreen** festgelegt werden.

Tabellenspalten

Hier können Sie festlegten, welche Tabellenspalten gedruckt werden sollen. Geben Sie dazu die gewünschten Spalten einzeln oder in Bereichen durch Komma oder Semikolon getrennt an. Beispiel: "1;5-7;3" selektiert die Spalten 1, 3 und 5 bis 7.

Diagramm anzeigen

Bestimmen sie, ob das Diagramm (Zeitskala und Balken) mit ausgedruckt werden soll oder nicht.

Zeitskalenstart

Mithilfe dieser Option kann der Starttermin des zu druckenden Zeitraums festgelegt werden. Dabei ist es nur möglich, den Zeitraum gegenüber dem am Bildschirm gezeigten einzuschränken, d.h. nur ein Startdatum, das nach dem über die Eigenschaft **VcGantt.TimeScaleStart** gesetzten liegt, führt zu einer Änderung des Ausdrucks.

Diese Option kann auch über die Eigenschaft **VcPrinter.TimeColumnStart-Date** festgelegt werden.

Zeitskalenende

Mithilfe dieser Option kann der Starttermin des zu druckenden Zeitraums festgelegt werden. Dabei ist es nur möglich, den Zeitraum gegenüber dem am Bildschirm gezeigten einzuschränken, d.h. nur ein Enddatum, das vor dem über die Eigenschaft **VcGantt.TimeScaleEnd** gesetzten liegt, führt zu einer Änderung des Ausdrucks.

Diese Option kann auch über die Eigenschaft **VcPrinter.TimeColumnEnd-Date** festgelegt werden.

Zeitskala an Breite der Seiten anpassen

Diese Option führt dazu, dass der Platz auf den Druckseiten besser ausgenutzt wird:

- Bei einer Skalierung über eine feste Anzahl Seiten: Der Zoomfaktor wird so berechnet, dass die eingestellte Anzahl Seiten in der Höhe voll

bedruckt wird. Gleichzeitig wird die Zeiskala so gestaucht oder gestreckt, dass die eingestellte Anzahl Seiten in der Breite voll ausgenutzt wird.

- Bei einer Skalierung über einen Zoomfaktor: Die Zeitskala wird so gestaucht oder gestreckt, dass die eingestellte Anzahl Seiten in der Breite voll ausgenutzt wird.

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 **Titel/Tabelle/Zeitskala 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 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

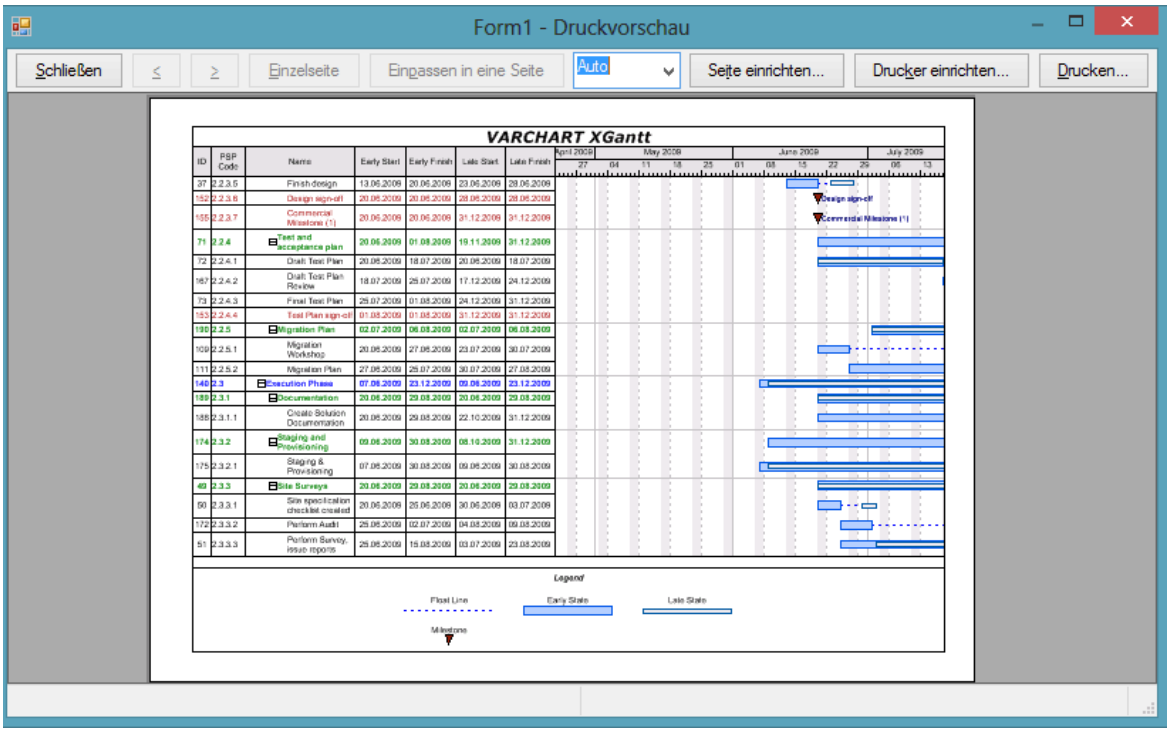
Druckdatum

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. Mindesträndern, 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.

5.25 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 einzeln 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 Mausrades 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 Mausrad 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 Druckereinstellungen vornehmen.

Drucken/Ausschnitt drucken

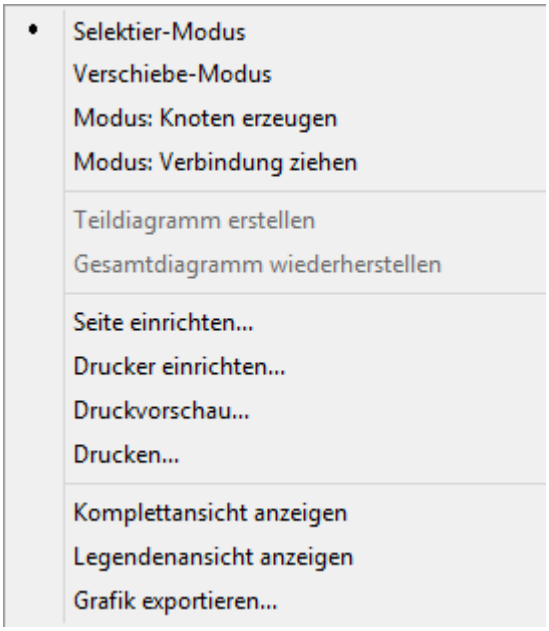
Über diese Schaltfläche gelangen Sie in das Windows-Dialogfeld **Drucken** und können von dort aus den Druckvorgang einleiten.

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.

5.26 Kontextmenü für das Diagramm

Wenn Sie die rechte Maustaste drücken, wenn der Mauszeiger im Diagrammbereich (jedoch nicht auf einem Knoten) steht, öffnet sich das folgende Kontextmenü:



Das Ereignis **VcDiagramRightClicking** tritt ein, wenn der Anwender mit der rechten Maustaste im Diagrammbereich klickt und dabei keinen Knoten trifft. Die Position (x,y-Koordinaten) 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.

Selektier-Modus

Der Selektier-Modus ist der Standardmodus, in dem alle Interaktionen außer dem Erzeugen von Knoten und Verbindungen möglich sind.

Verschiebe-Modus

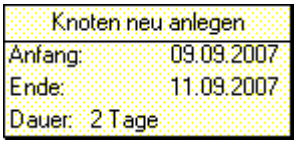
Im Verschiebemodus lässt sich mithilfe eines Cursors in Handform der darunter befindliche Bildschirmausschnitt verschieben.

Dieser Modus muss auf der Eigenschaftenseite **Allgemeines** aktiviert werden.

Modus: Knoten erzeugen

Dieser Modus kann nur eingeschaltet werden, wenn auf der Eigenschaftenseite **Knoten** die Option **Neue Knoten zulassen** gesetzt ist.

Der Mauszeiger nimmt die Form eines Kreuzchens an. Durch das Ziehen der Maus mit gedrückter linker Maustaste lassen sich in diesem Modus neue Knoten erzeugen. An der Position der Maus erscheint zunächst eine Box, die den aktuellen Anfangs- und Endtermin sowie die Dauer des neuen Knotens anzeigt.



Wenn Sie im Modus **Knoten erzeugen** in einer kollabierten Gruppe einen Knoten anlegen, erscheint ein kleiner Pfeil an dem Kreuzchen. Dieser Pfeil zeigt an, ob der neu anzulegende Knoten als erster Knoten (Pfeil nach oben) bzw. als letzter Knoten (Pfeil nach unten) in die Gruppe eingefügt wird.

Wenn die Option **Neuen Knoten bearbeiten** auf der Eigenschaftenseite **Knoten** 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 nichts anderes vereinbaren, erscheint der neu angelegte Vorgang an der durch die Maus bestimmten Position.

Der **Modus: Knoten erzeugen** lässt sich auch über das Setzen der Eigenschaft **InteractionMode** auf den Wert **VcCreateNode** aktivieren.

Das Ereignis **VcNodeCreating** tritt ein, wenn der Anwender interaktiv einen Knoten erzeugt hat. Das Knotenobjekt wird als Parameter übergeben, sodass eine Datenvalidierung (der Dialog **Vorgänge bearbeiten** ist ggf. vorher aktiviert) vorgenommen werden kann. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird der Knoten wieder gelöscht.

Modus: Verbindung ziehen

Dieser Modus kann nur eingeschaltet werden, wenn auf der Eigenschaftenseite **Verbindungen** die Option **Verbindungen anzeigen** gesetzt ist.

Der Mauszeiger nimmt die Form eines Bleistiftes an. Verbinden Sie nun mit der Maus zwei Knoten, so wird eine Ende-Anfang-Verbindung zwischen ihnen erstellt.

Das Ereignis **VcLinkCreating** tritt ein, wenn der Anwender interaktiv eine Verbindung zwischen zwei Knoten erzeugt hat. Das neu erzeugte Link-Objekt wird als Parameter zurückgegeben, sodass eine Validierung und ggf. ein Datenbank-Eintrag vorgenommen werden können. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Verbindung wieder gelöscht.

Modus: Box erzeugen

Dieser Modus kann nur eingeschaltet werden, wenn auf der Eigenschaften-seite **Allgemeines** die Option **Neue Boxen zulassen** gesetzt ist.

Durch das Ziehen der Maus mit gedrückter linker Maustaste lassen sich in diesem Modus neue Boxen erzeugen.

Der **Modus: Box erzeugen** lässt sich auch über das Setzen der Eigenschaft **InteractionMode** auf den Wert **VcCreateBox** aktivieren.

(siehe auch Ereignisse **VcBoxCreating** und **VcBoxCreated**).

Teildiagramm erstellen

(nur aktiv, wenn Knoten markiert sind) Wählen Sie diese Option, um ein Teildiagramm aus den markierten Knoten darzustellen.

Gesamtdiagramm wiederherstellen

*(nur aktiv, wenn zuvor die Option **Teildiagramm erstellen** gewählt wurde)* Wählen Sie diese Option, um das Gesamtdiagramm wiederherzustellen.

Seite einrichten

Sie gelangen in das Dialogfeld **Seite einrichten**.

Sie können dieses Dialogfeld auch mit der API-Methode **ShowPageSetupDialog** aufrufen.

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**. Sie können dieses Dialogfeld auch mit der VcGantt-Methode **PrinterSetup** aufrufen.

Druckvorschau

Sie gelangen in das Dialogfeld **Druckvorschau**. Sie können dieses Dialogfeld auch mit der VcGantt-Methode **ShowPrintPreviewDialog** aufrufen.

Drucken

Wenn Sie diese Option wählen, gelangen Sie in das Windows-Dialogfeld **Drucken**. Sie können dieses Dialogfeld auch mit der VcGantt-Methode **ShowPrintDialog** aufrufen.

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.

Grafik exportieren

Wenn Sie diese Option wählen, gelangen Sie in das Windows-Dialogfeld **Speichern unter**, in dem Sie die dargestellte Grafik als Grafikdatei speichern können.

Sie können diesen Dialog auch mit der VcGantt-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 <= -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 <= -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".

5.27 Kontextmenü für Knoten

Wenn Sie mit der rechten Maustaste einen oder mehrere markierte Knoten anklicken, erscheint das folgende Menü:

Wenn der Anwender auf einen Knoten (location = vcInDiagram) oder auf einen vorgangsbezogenen Tabelleneintrag (location = vcInTable) mit der rechten Maustaste klickt, tritt das Ereignis **VcNodeRightClicking** ein. Das getroffene Knotenobjekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

Daten bearbeiten

Wenn Sie diese Option wählen, öffnet sich das Dialogfeld **Vorgänge bearbeiten**. Wenn mehrere Knoten markiert sind, können Sie darin sukzessive die Daten aller markierter Knoten bearbeiten.

Knoten löschen

Wenn Sie diese Option wählen, werden die markierten Knoten gelöscht.

Teildiagramm erstellen

Wählen Sie diese Option, um ein Teildiagramm aus den markierten Knoten darzustellen.

Gesamtdiagramm wiederherstellen

*(nur aktiv, wenn zuvor die Option **Teildiagramm erstellen** gewählt wurde)*
Wählen Sie diese Option, um das Gesamtdiagramm wiederherzustellen.

Höher stufen

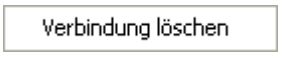
(nur für Hierarchie) Der markierte Knoten wird in der Hierarchie eine Ebene höher eingestuft.

Tiefer stufen

(nur für Hierarchie) Der markierte Knoten wird in der Hierarchie eine Ebene tiefer eingestuft.

5.28 Kontextmenü für Verbindungen

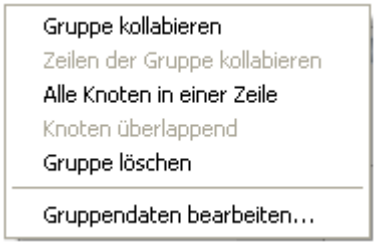
Wenn Sie eine Verbindung mit der rechten Maustaste anklicken, erscheint das Kontextmenü **Verbindung löschen**. Wenn Sie die markierte Verbindung löschen möchten, bestätigen Sie dies durch einen Klick mit der linken Maustaste.



Das Ereignis **VcLinksRightClicking** 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 Mauszeigers 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.

5.29 Kontextmenü für Gruppen

Wenn Sie eine Gruppenüberschrift in der Tabelle oder einen Gruppenbalken im Diagramm (der nur angezeigt wird, wenn im Dialog **Gruppierung** die Option **Gruppenknoten sichtbar** aktiviert wurde) mit der rechten Maustaste anklicken, erscheint ein Kontextmenü, das die wichtigsten Optionen für Gruppen enthält:



Das Ereignis **VcGroupRightClicking** tritt ein, wenn der Anwender in der Tabelle auf eine Gruppenüberschrift mit der rechten Maustaste klickt. Das getroffene Group-Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

Gruppe kollabieren/Gruppe expandieren

Mit dieser Option können Sie eine kollabierte Gruppe expandieren bzw. eine expandierte Gruppe kollabieren.

Zeilen der Gruppe expandieren/kollabieren

Mit dieser Option können Sie die Zeilen einer kollabierten Gruppe expandieren bzw. die Zeilen einer expandierten Gruppe kollabieren.

Wenn für die Gruppe alle Knoten in einer Zeile dargestellt werden, bewirkt diese Option, dass nur die vorhandenen Untergruppen in der gewählten Gruppe ausgeblendet werden.

Alle Knoten in einer Zeile/Jeder Knoten in eigener Zeile

Wenn Sie die Option **Alle Knoten in einer Zeile** wählen, werden alle Vorgänge einer Gruppe in einer Zeile ausgegeben. Überschneiden sich Vorgänge einer Gruppe, so werden sie im expandierten Status der Gruppe automatisch untereinander angeordnet, um Überlagerungen zu vermeiden.

Wird die Gruppe kollabiert, so sind Überschneidungen von Vorgängen möglich. Bei dieser Art der Anordnung entfällt der Tabellenteil für die Vorgänge. Man muss also die Layerbeschriftung oder den Tooltip verwenden, um die Vorgänge für den Anwender zu identifizieren.

Mit der Option **Jeder Knoten in eigener Zeile** erreichen Sie, dass jeder Knoten in einer eigenen Zeile dargestellt wird.

Knoten überlappend/Knoten optimiert

*nur aktiv, wenn **Alle Knoten in einer Zeile** gewählt wurde.* Wenn Sie die Option **Knoten überlappend** wählen, werden die Knoten einer Gruppe in einer Zeile dargestellt, auch wenn es dabei zu Überschneidungen kommt. Wenn Sie die Option **Knoten optimiert** wählen, werden die Knoten einer Gruppe ohne Überschneidungen dargestellt, auch wenn sie dann u. U. nicht in einer Zeile dargestellt werden können.

Gruppe löschen

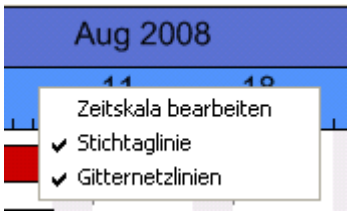
Mit dieser Option können Sie die markierte Gruppe löschen, vorausgesetzt sie ist leer.

Gruppendaten bearbeiten

Mit dieser Option rufen Sie den gleichnamigen Dialog auf.

5.30 Kontextmenü für die Zeitskala

Wenn Sie mit der rechten Maustaste auf die Zeitskala klicken, erscheint das folgende Menü:



Das Ereignis **VcTimeScaleRightClicking** tritt ein, wenn der Anwender mit der rechten Maustaste auf die Zeitskala klickt. Das getroffene TimeScale-Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. So können Sie an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt werden.

Zeitskala bearbeiten

Wenn Sie diese Option wählen, erscheint das Dialogfeld **Zeitskala bearbeiten**.

Stichtaglinie

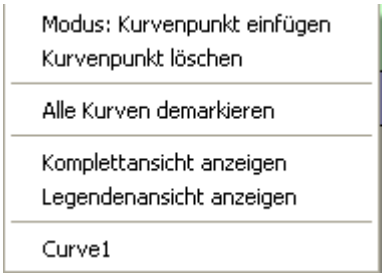
Wählen Sie, ob eine Stichtaglinie in Ihrer Darstellung angezeigt werden soll.

Gitternetzlinien

Wählen Sie, ob Gitternetzlinien in Ihrer Darstellung dargestellt werden sollen.

5.31 Kontextmenü für die Kurve

Wenn Sie mit der rechten Maustaste in einen leeren Histogrammbereich oder auf eine Kurve klicken, erscheint das folgende Kontextmenü:



Das Ereignis **VcHistogramRightClicking** oder **VcCurveRightClicking** wird ausgelöst, wenn der Anwender mit der rechten Maustaste auf das Histogramm oder auf die Kurve klickt. Das getroffene Histogramm- oder Kurven-Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. Das integrierte Kontextmenü kann an der gegebenen Position durch Setzen des Rückgabestatus auf **vcRetStatNoPopup** unterdrückt und ein eigenes Kontextmenü anzeigen werden.

Modus: Kurvenpunkt einfügen

In diesem Modus können Sie mit jedem Linksklick auf die Verfügbarkeitskurve einen neuen Stützpunkt für diese definieren.

Kurvenpunkt löschen

Um einen Stützpunkt zu löschen, klicken Sie diesen mit der rechten Maustaste an und wählen Sie im Kontextmenü den Befehl **Kurvenpunkt löschen**.

Alle Kurven demarkieren

Alle Kurven werden demarkiert.

Komplettansicht anzeigen

Über diesen Menüpunkt können Sie die Komplettansicht ein- oder ausschalten. Die Komplettansicht ist ein zusätzliches Fenster, in dem das komplette Diagramm und/oder Histogramm angezeigt wird. Ein Rahmen zeigt an, welchen Ausschnitt das Hauptfenster gerade anzeigt.

Legendenansicht anzeigen

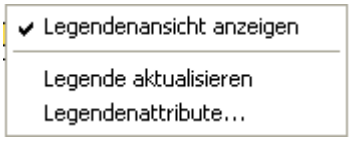
Über diesen Menüpunkt können Sie die Legendenansicht ein- oder ausschalten. Die Legendenansicht ist ein zusätzliches Fenster, in dem die Legende angezeigt wird.

Kurven

Falls vorhanden, werden die API-Kurven am Ende des Kontextmenüs angezeigt. Über den entsprechenden Menüeintrag können sie markiert werden.

5.32 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".

5.33 Kontextmenü für Boxen

Wenn Sie mit der rechten Maustaste auf eine Box klicken, erscheint das folgende Menü:

Sollte das Kontextmenü nicht erscheinen, muss auf der Eigenschaftenseite **Allgemeines** die Option **Kontextmenü für Boxen anzeigen** aktiviert werden.

Box am markierten Knoten verankern

Mit diesem Befehl können Sie eine Box am markierten Knoten verankern. Dies ist nur möglich, wenn zuvor im Dialogfeld **Boxen verwalten** die Option **Verankerungsinteraktionen erlaubt** ausgewählt wurde.

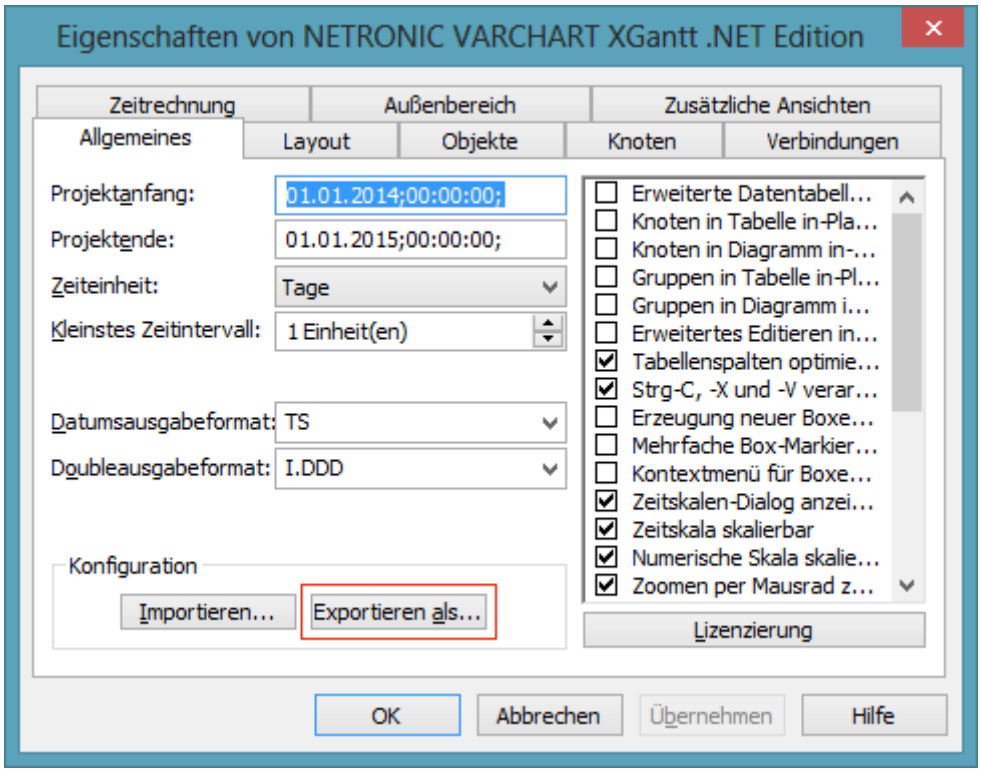
Box vom Knoten lösen

Mit diesem Befehl können Sie die Verankerung wieder lösen.

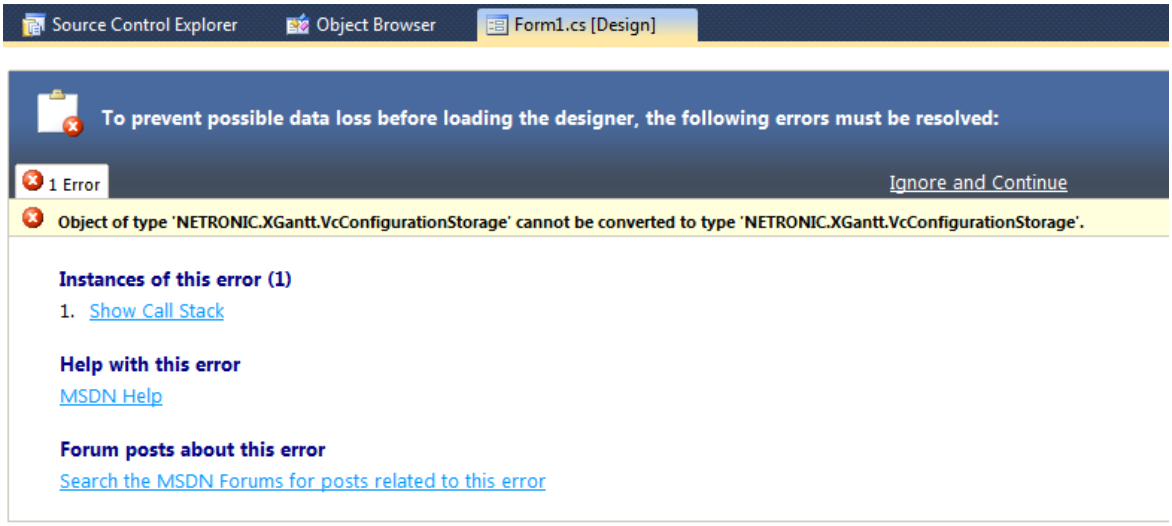
6 Häufig gestellte Fragen

6.1 Was muss ich tun, wenn ich von VARCHART XGantt 4.4 für .NET nach XGantt 5.0 umsteigen möchte?

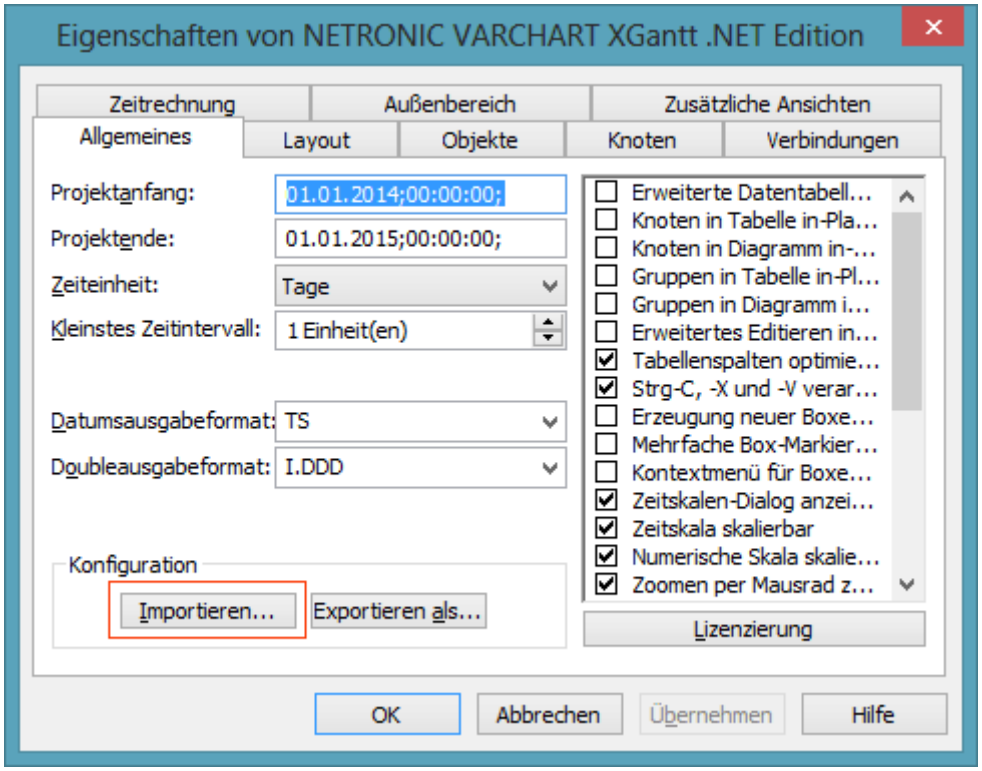
1. Bevor Sie VARCHART XGantt 5.0 für .NET installieren, öffnen Sie im Formdesigner von Visual Studio das Formular, das XGantt 4.4 verwendet, und sichern Sie die aktuelle Konfiguration von XGantt über die Schaltfläche **Exportieren** auf der Eigenschaftenseite **Allgemeines**:



2. Schließen Sie zuerst das Formular und dann Visual Studio.
3. Installieren Sie VARCHART XGantt 5.0 für .NET.
4. Öffnen Sie erneut Ihr Projekt in Visual Studio, löschen Sie im Solution Explorer die Referenz NETRONIC.XGantt (die ja noch auf XGantt 4.4 zeigt) und fügen Sie eine neue Referenz NETRONIC.XGantt ein, die aber auf XGantt 5.0 zeigen muss.
5. Öffnen Sie im Formdesigner das Formular, das XGantt enthält. Folgende Fehlermeldung erscheint:



6. Klicken Sie auf **Ignore and Continue**. Damit wird das Formular im Formdesigner wieder korrekt angezeigt, XGantt wurde jedoch auf seine Standardkonfiguration zurückgesetzt.
7. Importieren Sie Ihre zuvor gesicherte Konfiguration über die Schaltfläche **Importieren** auf der Eigenschaftenseite **Allgemeines**:

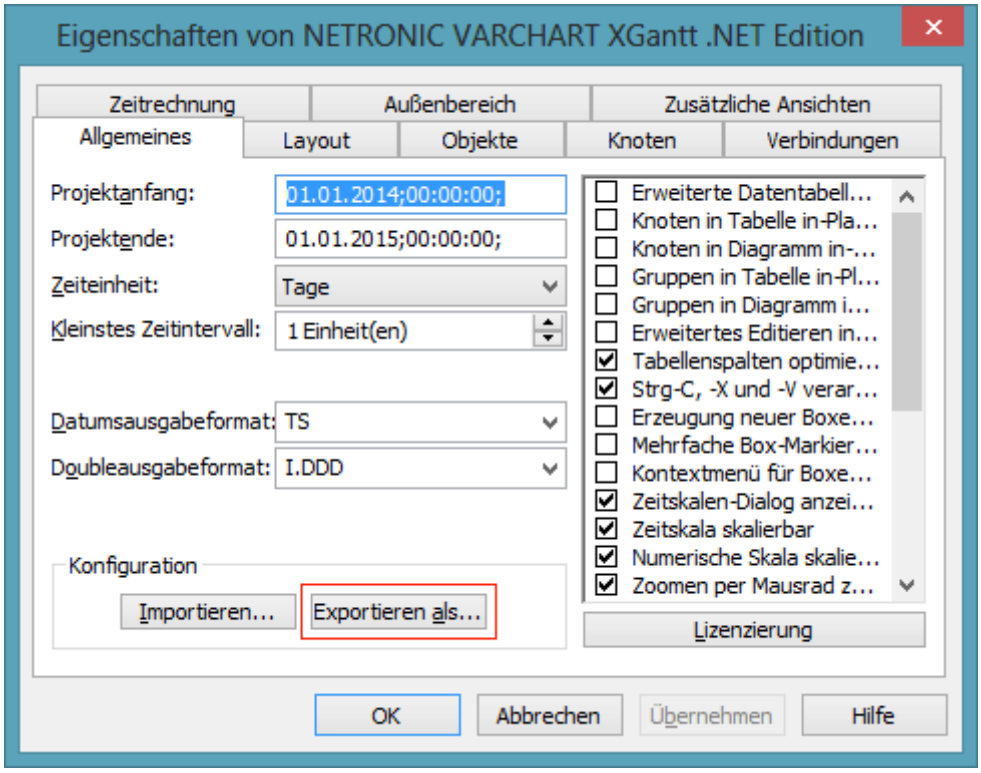


8.

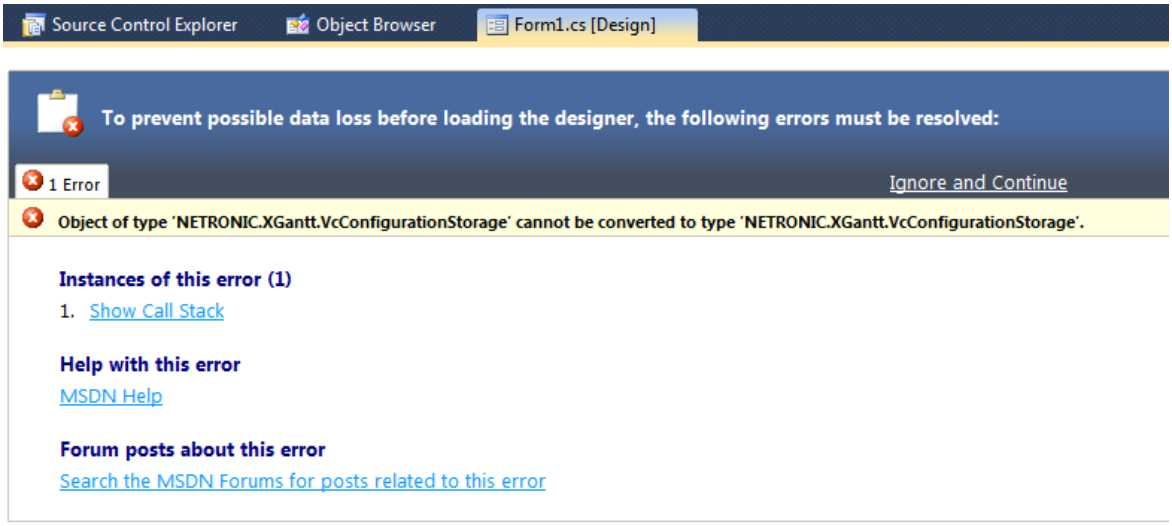
Damit hat VARCHART XGantt dann wieder die von Ihnen eingestellte Konfiguration.

6.2 Was muss ich tun, wenn ich im Rahmen eines Service Releases auf einen neuen Build von VARCHART XGantt umsteigen möchte?

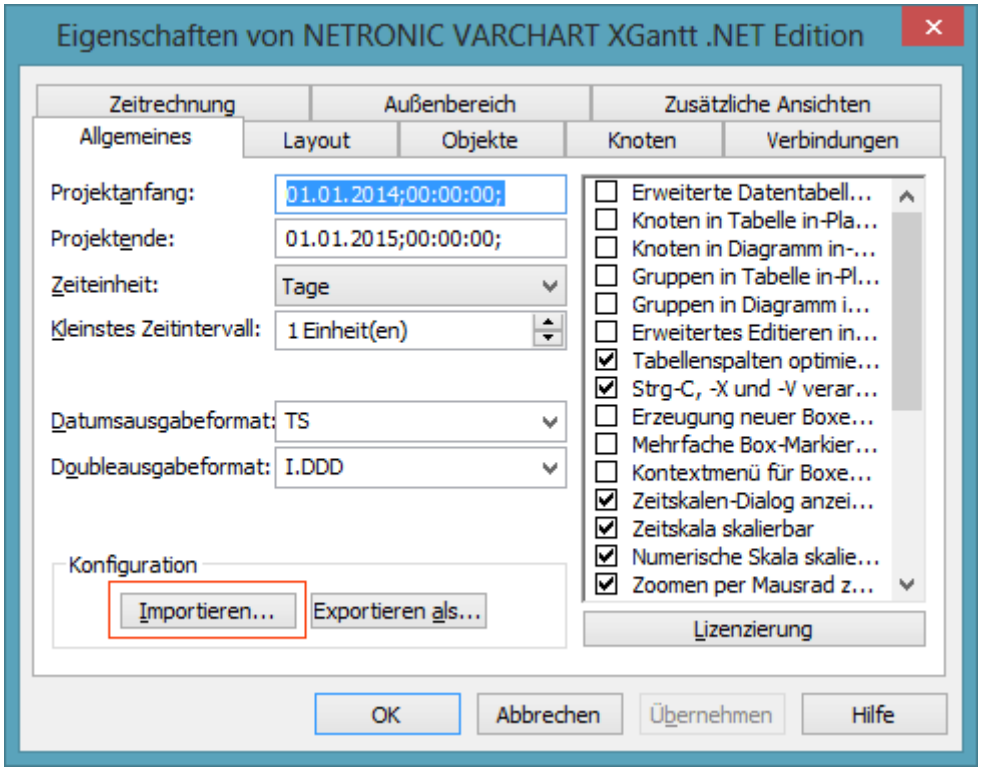
1. Bevor Sie VARCHART XGantt 5.0 für .NET installieren, öffnen Sie im Formdesigner von Visual Studio das Formular, das XGantt 4.4 verwendet, und sichern Sie die aktuelle Konfiguration von XGantt über die Schaltfläche **Exportieren** auf der Eigenschaftenseite **Allgemeines**:



2. Schließen Sie zuerst das Formular und dann Visual Studio.
3. nstallieren 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:



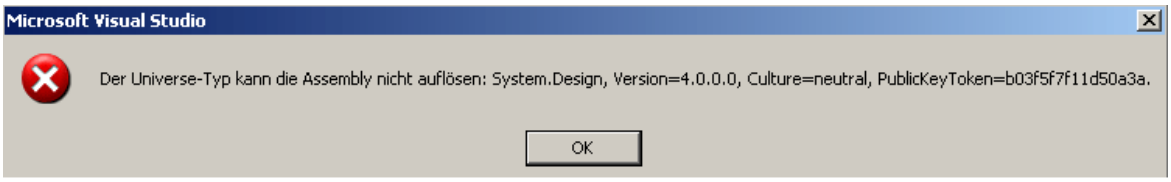
5. Klicken Sie auf **Ignore and Continue**. Damit wird das Formular im Formdesigner wieder korrekt angezeigt, XGantt wurde jedoch auf seine Standardkonfiguration zurückgesetzt.
6. Importieren Sie Ihre zuvor gesicherte Konfiguration über die Schaltfläche **Importieren** auf der Eigenschaftenseite **Allgemeines**:



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.XGantt.VcGantt.lic in das Installationsverzeichnis von VARCHART XGantt.NET kopieren
3. Umgebung wieder starten und Projekt erneut übersetzen

6.5 Wie kann die Spreizung der Zeitskala begrenzt werden?

Wenn die Zeitskala sehr weit links im sichtbaren Bereich mit gedrückter linker Maustaste angepackt wird, um die Zeitskala zu spreizen, kann leicht ein Faktor weit über 1.000% erreicht werden. Das lässt sich im Ereignis **VcTimeScaleSectionRescaling** leicht abfangen. Im folgenden Beispiel wird jeweils maximal eine Verdopplung der Auflösung zugelassen.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTimeScaleSectionRescaling(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcTimeScaleSectionRescalingEventArgs) Handles
VcGantt1.VcTimeScaleSectionRescaling

   Dim nOldUnitWidth As Long
   Dim returnStatus As VariantType

   nOldUnitWidth = e.TimeScale.Section(0).UnitWidth

   If (e.NewBasicUnitWidth > (2 * nOldUnitWidth)) Then
      nOldUnitWidth = 2 * nOldUnitWidth
      returnStatus = e.ReturnStatus.vcRetStatFalse
   End If

End Sub
```

Code-Beispiel C#

```
private void VcGantt1_VcTimeScaleSectionRescaling(object sender,
NETRONIC.XGantt.VcTimeScaleSectionRescalingEventArgs e)
   {
   long nOldUnitWidth = e.TimeScale.get_Section(0).UnitWidth;
   object returnStatus = e.ReturnStatus;
   if (e.NewBasicUnitWidth > (2 * nOldUnitWidth))
      {
      nOldUnitWidth = 2 * nOldUnitWidth;
      }
   }
```

6.6 Wie kann beim Klick in die Tabelle der zugehörige Balken in den sichtbaren Bereich gebracht werden?

Beim Ereignis **VcNodeLeftClicking** wird neben dem Knoten auch die Information **InTable** oder **InDiagram** übergeben. Fand der Klick in der Tabelle statt (**InTable**), wird aus dem Knoten das relevante Datum ausgefragt und mit der Methode **ScrollToDate** an das VcGantt-Objekt übergeben.

Code-Beispiel VB.NET

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

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

If (location = VcLocation.vcInTable) Then
      ' falls der Index des Feldes "Start" nicht bekannt ist
      myDataDef = VcGantt1.DataDefinition
      myDataDefTable =
myDataDef.DefinitionTable(VcDataTableType.vcMaindata)
      myDataField = myDataDefTable.DataDefinitionFieldByName("Start")
      myIndex = myDataField.ID
      VcGantt1.ScrollToDate(e.Node.DataField(myIndex),
VcHorizontalAlignment.vcLeftAligned, 2)
   End If

End Sub
```

Code-Beispiel C#

```
private void VcGantt1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   VcDataDefinition myDataDef = VcGantt1.DataDefinition;
   VcDataDefinitionTable myDataDefTable =
myDataDef.get_DefinitionTable(VcDataTableType.vcMaindata);
   VcDataDefinitionField myDataField =
myDataDefTable.DataDefinitionFieldByName("Start");
   int myIndex = myDataField.ID;
   VcLocation location = VcLocation.vcInTable;

   if (Location.ToString().Length > 0)
      VcGantt1.ScrollToDate(Convert.ToDateTime(e.Node.get_DataField(2)),
VcHorizontalAlignment.vcLeftAligned, 2);
   }
```

6.7 Wie werden Überschneidungen von Vorgängen einer Gruppe sichtbar?

Um Engpässe durch die Überschneidung des Urlaubs mehrerer Mitarbeiter oder durch die Mehrfachbelegung von Maschinen zu verhindern, können die Überschneidungen in einer Gruppe von Vorgängen sichtbar gemacht werden. Überschneidungen von Vorgängen kann es grundsätzlich nur geben, wenn eine Gruppierung verwendet wird und im Dialog **Gruppierung** die Option **Jeder Knoten in eigener Zeile** nicht gewählt wurde. Bei dieser Einstellung können Gruppen kollabiert oder expandiert dargestellt werden. Bei kollabierten Gruppen sind Überschneidungen nicht zu erkennen, während sie bei expandierten Gruppen durch eine versetzte Anordnung sichtbar gemacht werden.

Um Überschneidungen sichtbar zu machen, deaktivieren Sie ggf. am Dialog **Gruppierung** die Option **Jeder Knoten in eigener Zeile**, um die Vorgänge einer Gruppe in einer Zeile auszugeben. Überschneiden sich Vorgänge einer Gruppe, so werden sie - auch wenn diese Option nicht gewählt ist - untereinander angeordnet. Dadurch erkennen Sie die Überschneidungen auf einen Blick.

Wenn die Vorgänge kollabiert werden, sind Überschneidungen von Vorgängen jedoch nicht zu erkennen. Um sicherzustellen, dass der Anwender die Überschneidung von Vorgängen nicht übersieht, sollten Sie die Option **Änderungen erlaubt** abschalten, um das Umschalten zwischen diesen beiden Darstellungmodi zu verhindern. Deaktivieren Sie außerdem die Option **Zu Anfang kollabiert**. Dann erscheinen alle Gruppen expandiert, und Überschneidungen fallen sofort ins Auge, da die Vorgänge an den entsprechenden Stellen untereinander dargestellt werden.

6.8 Wie lässt sich die Reihenfolge der Vorgänge sichern und laden?

Die Voraussetzung für das Sichern und erneute Laden von Vorgängen ist, dass die Vorgänge aus einer Datei geladen werden.

Um die Reihenfolge der Vorgänge zu sichern und zu laden, wählen Sie auf der Eigenschaftenseite **Knoten** unter **Zeilennummer-Feld** ein Datenfeld aus. In dieses Datenfeld schreibt XGantt dann die Identifikation.

Wenn die Knotenreihenfolge interaktiv verändert wurde, lässt sie sich mit der Methode **UpdateRowNumberField** aktualisieren. (Gruppierung und Hierarchie dürfen dabei nicht eingeschaltet sein.)

Ergänzen Sie schließlich die folgenden Code-Zeilen:

Code-Beispiel VB.NET

```
Private Sub Form_Unload ()
   VcGantt1.UpdateRowNumberField
   VcGantt1.SaveAs ("<Dateiname.csv>")
End Sub
```

Code-Beispiel C#

```
private void Form_Unload()
    {
     VcGantt1.UpdateRowNumberFields;
     VcGantt1.SaveAs("<Dateiname.csv ");
     }
```

6.9 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 **Knoten** das Kontrollkästchen **Neue Knoten zulassen** aktiviert ist.

Wenn außerdem das Kontrollkästchen **Neue Knoten durch Doppelklick** aktiviert ist, können Sie zur Laufzeit Knoten durch Doppelklick erzeugen.

Außerdem lassen sich keine Knoten in arbeitsfreien Bereichen anlegen, wenn ein Kalender aktiviert ist.

Kontrollieren Sie außerdem, ob in Ihrem Programmcode die Eigenschaft **NodeCreationAllowed** auf **False** gesetzt wurde; dann können ebenfalls keine neuen Knoten angelegt werden.

6.10 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 VcGantt1_VcDiagramRightClicking(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramRightClicking
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub

' Kontextmenü für Verbindungen abschalten
Private Sub VcGantt1_VcLinksRightClicking(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcLinksClickingEventArgs) Handles
VcGantt1.VcLinksRightClicking
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub

' Kontextmenü für Knoten abschalten
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcNodeClickingEventArgs) Handles
VcGantt1.VcNodeRightClicking
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
End Sub
```

Code-Beispiel C#

```
' Kontextmenü für das Diagramm abschalten
private void VcGantt1_VcDiagramRightClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
    {
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
    }

' Kontextmenü für Verbindungen abschalten
private void VcGantt1_VcLinksRightClicking(object sender,
NETRONIC.XGantt.VcLinksClickingEventArgs e)
     {
     e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
     }

' Kontextmenü für Knoten abschalten
private void VcGantt1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
    {
    e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
    }
```

6.11 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 VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcGantt1.SuspendUpdate(True)

For Each dataRecord In dataRecordCltn
   dataRecord.DataField(4) = "10"
   dataRecord.Update()
Next

VcGantt1.SuspendUpdate(False)
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcGantt1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
   dataRecord.set_DataField(4, "10");
   dataRecord.Update();
}

vcGantt1.SuspendUpdate(false);
```

Die Performance lässt sich analog auch beim Update von vielen Verbindungen verbessern.

Auch beim Modifizieren von Tabellenformaten lässt sich die Performance steigern.

Code-Beispiel VB.NET

```
Private Sub ModifyTable_Click()

   Dim formatCltn As VcTableFormatCollection
   Dim aFormat As VcTableFormat
   Dim index As Integer

    VcGantt1.SuspendUpdate(True)

   formatCltn = VcGantt1.LeftTable.TableFormatCollection
   For Each aFormat In formatCltn
      For index = 1 To aFormat.FormatFieldCount
         aFormat.FormatField(index).BackgroundColor = Color.Green
         aFormat.FormatField(index).TextFontColor = Color.Red
         aFormat.FormatField(index).Alignment =
VcFormatFieldAlignment.vcFFACenter
      Next
   Next

   VcGantt1.SuspendUpdate(False)

End Sub
```

Code-Beispiel C#

```
private void ModifyTable_Click()
   {
   VcTableFormatCollection formatCltn =
vcGantt1.LeftTable.TableFormatCollection;

   vcGantt1.SuspendUpdate(true);

   foreach (VcTableFormat aFormat in formatCltn)
   {
      for (int index=1; index <= aFormat.FormatFieldCount; index++)
         {
         aFormat.get_FormatField(index).BackgroundColor = Color.Green;
         aFormat.get_FormatField(index).TextFontColor = Color.Red;
         aFormat.get_FormatField(index).Alignment =
VcFormatFieldAlignment.vcFFACenter;
         }
   }

   vcGantt1.SuspendUpdate(false);
   }
```

Auch bei nicht äquidistanten Histogrammkurven erreichen Sie mit dieser Methode eine signifikante Performancesteigerung.

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim index As Integer
```

```
Dim aDate As Date

histogram = VcGantt1.HistogramCollection.FirstHistogram
curve = histogram.CurveCollection.CurveByName("Curve1")

' aktuelles Datum
aDate = Date.Today()

VcGantt1.SuspendUpdate(True)

For index = 1 To 3000
   ' verschieben um 2h
   aDate = aDate.AddHours(2)
   curve.SetValues(aDate, index)
Next
VcGantt1.SuspendUpdate(False)
```

Code-Beispiel C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

// aktuelles Datum
DateTime aDate = DateTime.Today;

vcGantt1.SuspendUpdate(true);

for (int index=1; index < 3000; index++)
   {
   // verschieben um 2h
   aDate = aDate.AddHours(2);
   curve.SetValues(aDate, Convert.ToString(index));
   }

vcGantt1.SuspendUpdate(false);
```

Auch bei der Verwendung von Kalendern können Sie durch diese Methode das Update beschleunigen, denn wenn die Knoten schon geladen sind, erfordern Mehrfachmodifikationen an Kalendern einige Zeit, weil alle Knoten auf Kalenderabhängigkeiten geprüft werden.

> 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.

6.12 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.

6.13 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-Steuerelement 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

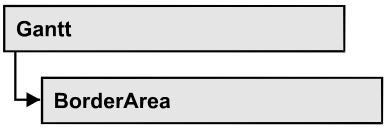
7.1 Objekttypen

- VcBorderArea
- VcBorderBox
- VcBox
- VcBoxCollection
- VcBoxFormat
- VcBoxFormatCollection
- VcBoxFormatField
- VcCalendar
- VcCalendarCollection
- VcCalendarGrid
- VcCalendarGridCollection
- VcCalendarProfile
- VcCalendarProfileCollection
- VcCurve
- VcCurveCollection
- VcDataDefinitionField
- VcDataDefinitionTable
- VcDataRecord
- VcDataRecordCollection
- VcDataTable
- VcDataTableCollection
- VcDataTableField
- VcDataTableFieldCollection
- VcDateLine
- VcDateLineCollection
- VcDateLineGrid
- VcDateLineGridCollection
- VcFilter
- VcFilterCollection
- VcFilterSubCondition
- VcGantt
- VcGroup

- VcGroupCollection
- VcGroupLevelLayout
- VcGroupLevelLayoutCollection
- VcHierarchyLevelLayout
- VcHistogram
- VcHistogramCollection
- VcInfoWindow
- VcInterval
- VcIntervalCollection
- VcLayer
- VcLayerCollection
- VcLayerFormat
- VcLayerFormatField
- VcLegendView
- VcLineFormat
- VcLineFormatCollection
- VcLineFormatField
- VcLink
- VcLinkAppearance
- VcLinkAppearanceCollection
- VcLinkCollection
- VcMap
- VcMapCollection
- VcMapEntry
- VcNode
- VcNodeCollection
- VcNodeLevelLayout
- VcNumericScale
- VcNumericScaleCollection
- VcPrinter
- VcRect
- VcResourceScheduler2
- VcRibbon
- VcScheduler
- VcSection
- VcTable
- VcTableCollection
- VcTableFormat
- VcTableFormatCollection
- VcTableFormatField

- VcTimeScale
- VcTimeScaleCollection
- VcUpdateBehavior
- VcUpdateBehaviorCollection
- VcUpdateBehaviorContext
- 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**
Parameter:		
boxPosition	VcBorderBoxPosition **Mögliche Werte:** .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
Rückgabewert	VcBorderBox	Box des Titel- und Legendenbereichs

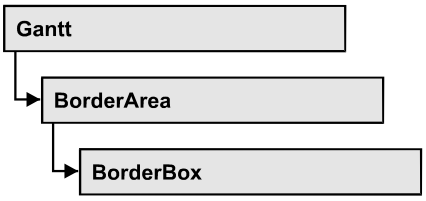
Code-Beispiel VB.NET

```
Dim boardArea As VcBorderArea
Dim bBoxBBL As VcBorderBox

boardArea = VcGantt1.BorderArea
bBoxBBL = boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
bBoxBBL.LegendTitle = "Explanation"
```

Code-Beispiel C#

```
VcBorderArea boardArea = vcGantt1.BorderArea;

VcBorderBox bBoxBBL =
boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
bBoxBBL.LegendTitle = "Explanation";
```

7.3 VcBorderBox



Ein Objekt vom Typ **VcBorderBox** 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 VcBorderBox

Mit dieser Eigenschaft können Sie die Ausrichtung dieses BorderBox-Objektes festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcBorderBoxAlignment	Ausrichtung der BorderBox
	Mögliche Werte:	
	.vcBBXACentered -1	unten mittig
	.vcBBXALeft -3	links

.vcBBXARight -2	rechts

GraphicsFileName

Eigenschaft von VcBorderBox

Mit dieser Eigenschaft können Sie den Namen der Grafikdatei angeben oder erfragen, die in dem aktuellen VcBorderBox-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 VcBorderBox

borderArea = VcGantt1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomTopRight)
borderBox.Type = VcBorderBoxType.vcBBXTGraphics
borderBox.GraphicsFileName = "C:\Asterix.jpg"
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.BorderArea;
VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomTopRight);
borderBox.Type = VcBorderBoxType.vcBBXTGraphics;
borderBox.GraphicsFileName = @"C:\Asterix.jpg";
```

LegendElementsArrangement

Eigenschaft von VcBorderBox

Mit dieser Eigenschaft können Sie die Anordnung der Elemente der Legende setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLegendElementsArrangement	Typ der Anordnung der Legendenelemente
	Mögliche Werte:	
	.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 VcBorderBox

Mit dieser Eigenschaft können Sie den Abstand der Elemente zum unteren Rand festlegen oder erfragen (Einheit: mm).

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite des unteren Randes

LegendElementsMaximumColumnCount

Eigenschaft von VcBorderBox

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 VcBorderBox

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 VcBorderBox

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 VcBorderBox

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 = VcGantt1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendFont.Name)
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.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
Eigenschaftswert	System.String	Legendentitel

Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGantt1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitle = "Explanation"
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.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
Eigenschaftswert	System.DrawingFont	Schriftattribute des Legendentitels

Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGantt1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTLegend
MsgBox(borderBox.LegendTitleFont.Name)
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.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
Eigenschaftswert	System.Boolean	Legendentitel sichtbar (True)/nicht sichtbar (False)

Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGantt1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.LegendTitleVisible = False
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.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**
Parameter:		
rowIndex	System.Int16	Zeilenindex {0...6}
Eigenschaftswert	System.String	Text in Textfeldern

Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim borderBox As VcBorderBox

borderArea = VcGantt1.BorderArea
borderBox = borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft)
borderBox.Type = VcBorderBoxType.vcBBXTText
borderBox.Text(index) = "Department A"
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.BorderArea;

VcBorderBox borderBox =
borderArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
borderBox.Type = VcBorderBoxType.vcBBXTText;
borderBox.set_Text(index, "DepartmentA");
```

TextFont

Eigenschaft von VcBorderBox

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**
Parameter:		
rowIndex	System.Int16	Zeilenindex {0...6}
Eigenschaftswert	System.DrawingFont	Schriftattribute des Textes

Code-Beispiel VB.NET

```
Dim borderArea As VcBorderArea
Dim bBoxTL As VcBorderBox

Set borderArea = VcGantt1.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
VcBorderBox borderBox =
VcGantt1.BorderArea.BorderBox(VcBorderBoxPosition.vcBBXPTopCentered);
borderBox.Type = VcBorderBoxType.vcBBXTText;

Font titleFont1 = new Font("Arial", 20, FontStyle.Bold);

borderBox.set_Text(1, "Time Scheduler");
borderBox.set_TextFont(1, titleFont1);
```

Type

Eigenschaft von VcBorderBox

Mit dieser Eigenschaft können Sie den Typ des BorderBox-Objekts angeben bzw. erfragen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcBorderBoxType	Typ der Box
	Mögliche Werte:	
	.vcBBXTGraphics 3	Grafik
	.vcBBXTLegend 4	Legende
	.vcBBXTNothing 0	Leer
	.vcBBXTText 1	Text
	.vcBBXTTextWithGraphics 2	Text und Grafik

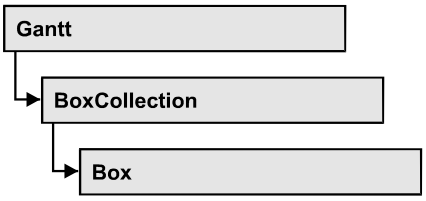
Code-Beispiel VB.NET

```
Dim bBoxBBL As VcBorderBox

bBoxBBL = boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomLeft)
bBoxBBL.Type = VcBorderBoxType.vcBBXTGraphics
```

Code-Beispiel C#

```
VcBorderArea boardArea = vcGantt1.BorderArea;

VcBorderBox bBoxBBL =
boardArea.BorderBox(VcBorderBoxPosition.vcBBXPBottomBottomLeft);
bBoxBBL.Type = VcBorderBoxType.vcBBXTGraphics;
```

7.4 VcBox



Ein Objekt vom Typ **VcBox** beschreibt eine Box, in der Texte und Grafiken ausgegeben werden können.

Eigenschaften

- AnchoringInteractionsAllowed
- AnchoringLineVisible
- FieldText
- FormatName
- LineColor
- LineThickness
- LineType
- Marked
- Moveable
- Name
- NodeID
- Origin
- Priority
- ReferencePoint
- Resizing
- Specification
- UpdateBehaviorName
- Visible

Methoden

- AnchorToNode
- GetActualExtent
- GetTopLeftPixel
- GetXYOffset
- IdentifyFormatField
- SetXYOffset
- SetXYOffsetByTopLeftPixel

Eigenschaften

AnchoringInteractionsAllowed

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie festlegen oder abfragen, ob eine Box interaktiv mit einem Knoten verankert werden kann.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Box kann/kann nicht interaktiv mit einem Knoten verankert werden **Standardwert:** False

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.AnchoringInteractionsAllowed = False
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.AnchoringInteractionsAllowed = false;
```

AnchoringLineVisible

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie festlegen oder abfragen, ob bei einer Verankerung die eingestellten Referenzpunkt bei Knoten und Box durch eine Linie verbunden werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Verankerungslinie zwischen Knoten und Box wird/wird nicht angezeigt **Standardwert:** False

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.AnchoringLineVisible = False
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.AnchoringLineVisible = false;
```

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 (field-Index) angesprochen wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ fieldIndex	System.Int16	Feldindex
Eigenschaftswert	System.String	Feldinhalt

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.FieldText(0) = "User: "
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	VcBoxFormat	BoxFormat-Objekt oder **Nothing**

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.FormatName = "Standard"
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.LineColor = System.Drawing.Color.Blue
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the dialog

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.LineThickness = 2
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.LineThickness = 2;
```

LineType

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie den Typ der Randlinie der Box festlegen oder erfragen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcLineType	Linientyp **Standardwert:** vcSolid
	Mögliche Werte:	
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.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
	.vcSolid 2	Linientyp **durchgezogen**
	.vcSolid 2	Linientyp **durchgezogen**

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.LineType = VcLineType.vcDotted
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	System.Boolean	**True**: Box markiert; **false**: Box nicht markiert

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.Marked = True
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	System.Boolean	Verschiebbar (True)/ nicht verschiebbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.Moveable = False
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	System.String	Name der Box

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim boxName As String

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Name)
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();

MessageBox.Show(box.Name);
```

NodeID

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie die ID des Knotens mit dem die Box verankert ist, festlegen oder abfragen. Diese Eigenschaft können Sie auch im Dialog **Boxen verwalten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	ID des Knotens, mit dem die Box verankert ist

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox
Dim box.NodeID As String

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
boxNodeID = 3
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();

boxNodeID = 3;
```

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**
Eigenschaftswert	VcBoxOrigin **Mögliche Werte:** .vcBOBottomCenter 28 .vcBOBottomLeft 27 .vcBOBottomRight 29 .vcBOCenterCenter 25 .vcBOCenterLeft 24 .vcBOCenterRight 26 .vcBOTopCenter 22 .vcBOTopLeft 21 .vcBOTopRight 23	Ursprungspunkt der Box unten mittig unten links unten rechts mittig mittig mittig links mittig rechts oben mittig oben links oben rechts

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.Origin = VcBoxOrigin.vcBOTopCenter
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	System.Int16	Prioritätsstufe

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.Priority = 3
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Eigenschaftswert	VcBoxReferencePoint **Mögliche Werte:** .vcBRPBottomCenter 28 .vcBRPBottomLeft 27 .vcBRPBottomRight 29 .vcBRPCenterCenter 25 .vcBRPCenterLeft 24 .vcBRPCenterRight 26 .vcBRPTopCenter 22 .vcBRPTopLeft 21 .vcBRPTopRight 23	Referenzpunkt der Box unten mittig unten links unten rechts mittig mittig mittig links mittig rechts oben mittig oben links oben rechts

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.ReferencePoint = VcBoxReferencePoint.vcBRPCenterRight;
```

Resizing

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob und wie die Größe einer Box verändert werden kann.

	Datentyp	Beschreibung
Eigenschaftswert	VcBoxResizing	Interaktive Größenänderung der Box
	Mögliche Werte:	
	.vcBRHeight 23	Die Höhe der Box kann interaktiv verändert werden.
	.vcBRNo 0	Die Größe der Box kann nicht interaktiv verändert werden.
	.vcBRWidth 24	Die Breite der Box kann interaktiv verändert werden.
	.vcBRWidth/Height 1050	Breite und Höhe der Box können interaktiv verändert werden.

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.Resizing = VcBoxRResizing.vcBRWidth
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Resizing = VcBoxRResizing.vcBRWidth;
```

Specification

Nur-Lese-Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie die Spezifikation dieser Box 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 Box mit der Methode **VcBox-Collection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Boxspezifikation

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn =VcGantt1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
MessageBox.Show(box.Specification);
```

UpdateBehaviorName

Eigenschaft von VcBox

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	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
Eigenschaftswert	System.Boolean	Box sichtbar/unsichtbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
box.Visible = False
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
VcBox box = boxCltn.FirstBox();
box.Visible = false;
```

Methoden

AnchorToNode

Methode von VcBox

Mit dieser Methode können Sie Boxen an Knoten verankern bzw. sie wieder lösen.

Eine verankerte Box kann weiterhin interaktiv verschoben werden (Voraussetzung: die Eigenschaft **Moveable** ist gesetzt). Um eine Box wieder vom Knoten zu lösen, müssen Sie als Parameter "NULL" übergeben.

Wird ein Knoten mit einer verankerten Box verschoben, so wird die Box entsprechend mitverschoben. Wird der Knoten kollabiert, wird auch die Box kollabiert d.h. sie ist nicht mehr sichtbar. Sobald der Knoten expandiert wird, ist auch die Box wieder sichtbar.

Beim Ver-/Entankern bleibt die Position der Box auf dem Bildschirm erhalten - die zu Grunde liegenden Offset-Werte werden entsprechend der Referenzpunkte (Origin, ReferencePoint) umgerechnet. Wenn sich also eine Textbox mit einem bestimmten Offset z.B. auf das Diagramm oben links (Origin) bezieht und dann mit einem Knoten verankert wird, wird automatisch ein Offset zum Knoten oben links berechnet und dieser Offset führt dazu, dass die Position auf dem Bildschirm beibehalten wird. Wird die Box vom Knoten gelöst, erfolgt eine Rückberechnung

	Datentyp	**Beschreibung**
Parameter: node	VcNode	Knotenobjekt, mit dem die Box verankert ist
Rückgabewert	System.Boolean	Box wird am Knoten verankert/vom Knoten gelöst

Code-Beispiel VB.NET

```
VcNode node = VcGantt1.NodeCollection.FirstNode()

VcGantt1.BoxCollection.FirstBox().AnchorToNode(node)
```

Code-Beispiel C#

```
VcNode node = vcGantt1.NodeCollection.FirstNode();

vcGantt1.BoxCollection.FirstBox().AnchorToNode(node);
```

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
Parameter:		
⇦ width	System.Int32	Breite der Box
⇦ height	System.Int32	Höhe der Box
Rückgabewert	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
Parameter:		
⇦ x	System.Int32	x-Wert des Offsets
⇦ y	System.Int32	y-Wert des Offsets

Rückgabewert	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**
Parameter:		
⇦ xOffset	System.Int32	x-Wert des Offsets
⇦ yOffset	System.Int32	y-Wert des Offsets
Rückgabewert	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**
Parameter:		
⇨ x	System.Int32	X-Koordinate der Position
⇨ y	System.Int32	Y-Koordinate der Position
⇦ format	VcBoxFormat	Identifiziertes Format
⇦ formatFieldIndex	System.Int16	Format-Feldindex
Rückgabewert	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
Parameter:		
⇨ xOffset	System.Int32	x-Wert des Offsets
⇨ yOffset	System.Int32	y-Wert des Offsets
Rückgabewert	System.Boolean	Offset wird gesetzt (True)/nicht gesetzt (False)

Code-Beispiel VB.NET

```
Dim offSet As Boolean
offSet = VcGantt1.BoxCollection.FirstBox.SetXYOffset(100, 100)
```

Code-Beispiel C#

```
bool offSet = vcGantt1.BoxCollection.FirstBox().SetXYOffset(100, 100);
```

SetXYOffsetByTopLeftPixel

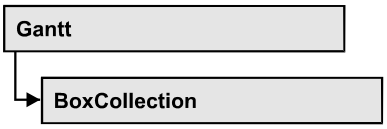
Methode von VcBox

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
Parameter:		
⇨ x	System.Int32	x-Wert des Offsets
⇨ y	System.Int32	y-Wert des Offsets
Rückgabewert	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 = VcGantt1.BoxCollection
numberOfBoxes = boxCltn.Count
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Parameter:		
⇨ boxName	System.String	Name der Box
Rückgabewert	VcBox	Neues Boxobjekt

Code-Beispiel VB.NET

```
newBox = VcGantt1.BoxCollection.Add("box1")
```

Code-Beispiel C#

```
newBox = vcGantt1.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 Hlife 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
Parameter:		
⇨ specification	System.String	Boxspezifikation
Rückgabewert	VcBox	Neues Boxobjekt

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGantt1.BoxCollection
boxCltn.AddBySpecification(textSpecification)
boxCltn.Update()
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Parameter:		
⇨ index	System.Int16	Index der Box
Rückgabewert	VcBox	Ermitteltes Box-Objekt

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGantt1.BoxCollection
box = boxCltn.BoxByIndex(0)
box.LineThickness = 2
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Parameter:		
⇨ boxName	System.String	Name der Box
Rückgabewert	VcBox	Box

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGantt1.BoxCollection
box = boxCltn.BoxByName("BoxOne")
box.LineThickness = 3
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Parameter:		
⇨ boxName	System.String	Name der zu kopierenden Box
⇨ newBoxName	System.String	Name der neuen Box
Rückgabewert	VcBox	Boxobjekt

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection

boxCltn = VcGantt1.BoxCollection
boxCltn.Copy("BoxOne", "NewBox")
boxCltn.Update()
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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 = VcGantt1.BoxCollection
box = boxCltn.FirstBox
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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 VcGantt1.BoxCollection
   ListBox1.Items.Add(box.FormatName)
Next
```

Code-Beispiel C#

```
foreach (VcBox box in vcGantt1.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 = VcGantt1.BoxCollection
box = boxCltn.FirstBox

While Not box Is Nothing
   ListBox1.Items.Add(box.Name)
   box = boxCltn.NextBox
End While
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Parameter:		
⇨ boxName	System.String	Name der Box
Rückgabewert	System.Boolean	Box gelöscht (True)/nicht gelöscht (False)

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

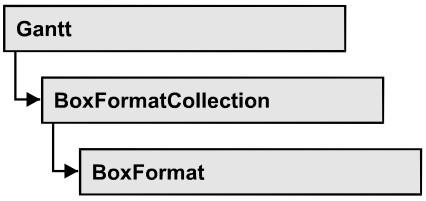
Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.BoxByIndex(0)
boxCltn.Remove(box.Name)
boxCltn.Update()
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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 = VcGantt1.BoxFormatCollection.FormatByIndex(0)
boxFormat.FieldsSeparatedByLines = True
```

Code-Beispiel C#

```
VcBoxFormat boxFormat = vcGantt1.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
Parameter:		
index	System.Int16	Index des Boxformatfeldes
Eigenschaftswert	VcBoxFormatField	Boxformatfeld

Code-Beispiel VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcGantt1.BoxFormatCollection.FirstFormat
formatField = boxFormat.FormatField(0)
MsgBox(formatField.FormatName)
```

Code-Beispiel C#

```
VcBoxFormat boxFormat = vcGantt1.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 = VcGantt1.BoxFormatCollection.FirstFormat
MsgBox(boxFormat.FormatFieldCount)
```

Code-Beispiel C#

```
VcBoxFormat boxFormat = vcGantt1.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 VcGantt1.BoxFormatCollection
   ListBox1.Items.Add(boxFormat.Name)
Next
```

Code-Beispiel C#

```
foreach (VcBoxFormat boxFormat in vcGantt1.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
Eigenschaftswert	System.String	Spezifikation des Boxformats

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormat = boxFormatCltn.FirstBoxFormat
MsgBox(boxFormat.Specification)
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Parameter:		
⇨ position	VcFormatFieldInnerPosition	Position des neuen Boxformatfeldes
	Mögliche Werte:	
	.vcInnerAbove 1	oberhalb
	.vcInnerBelow 3	unterhalb
	.vcInnerLeftOf 0	links von
	.vcInnerRightOf 4	rechts von
⇨ refIndex	System.Int16	Index des Referenz-Boxformatfeldes
Rückgabewert	VcBoxFormatField	Boxformatfeld-Objekt

Code-Beispiel VB.NET

```
Dim boxFormat As VcBoxFormat
Dim formatField As VcBoxFormatField

boxFormat = VcGantt1.BoxFormatCollection.FormatByIndex(2)
boxFormat.CopyFormatField(VcFormatFieldInnerPosition.vcInnerRightOf, 0)
```

Code-Beispiel C#

```
VcBoxFormat boxFormat = vcGantt1.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 = VcGantt1.BoxFormatCollection.FirstFormat
For Each formatField In boxFormat
   ListBox1.Items.Add(formatField.FormatName)
Next
```

Code-Beispiel C#

```
VcBoxFormat boxFormat = vcGantt1.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 numeriert sind.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des zu löschenden Boxformatfeldes

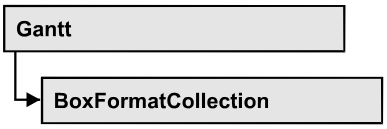
Code-Beispiel VB.NET

```
Dim boxFormat As VcBoxFormat
Dim i As Integer

boxFormat = VcGantt1.BoxFormatCollection.FirstFormat

For i = 0 To boxFormat.FormatFieldCount - 1
   boxFormat.RemoveFormatField(i)
Next
```

Code-Beispiel C#

```
VcBoxFormat boxFormat = vcGantt1.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 = VcGantt1.BoxFormatCollection
numberOfBoxformats = boxFormatCltn.Count
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Parameter:		
⇨ formatName	System.String	Name des Boxformats
Rückgabewert	VcBoxFormat	Neues Boxformatobjekt

Code-Beispiel VB.NET

```
Dim newBoxFormat = VcGantt1.BoxFormatCollection.Add("boxFormat1")
```

Code-Beispiel C#

```
newBoxFormat = vcGantt1.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
Parameter:		
⇨ formatSpecification	System.String	Boxformatspezifikation
Rückgabewert	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
Parameter:		
⇨ FormatName	System.String	Name des zu kopierenden Boxformats
⇨ newFormatName	System.String	Name des neuen Boxformats
Rückgabewert	VcBoxFormat	Boxformatobjekt

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormat = boxFormatCltn.Copy("CurrentBoxFormat", "NewBoxFormat")
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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 = VcGantt1.BoxFormatCollection.FirstFormat
```

Code-Beispiel C#

```
VcBoxFormat format = vcGantt1.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**
Parameter:		
⇨ index	System.Int16	Index des Boxformats
Rückgabewert	VcBoxFormat	Ermitteltes Boxformat-Objekt

Code-Beispiel VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcGantt1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByIndex(2)
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Parameter:		
⇨ formatName	System.String	Name des Boxformats
Rückgabewert	VcBoxFormat	Boxformat

Code-Beispiel VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcGantt1.BoxFormatCollection
formatBox = formatBoxCltn.FormatByName("Standard")
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGantt1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
   ListBox1.Items.Add(boxFormat.Name)
Next
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Rückgabewert	VcBoxFormat	Folgendes Boxformat

Code-Beispiel VB.NET

```
Dim formatBoxCltn As VcBoxFormatCollection
Dim formatBox As VcBoxFormat

formatBoxCltn = VcGantt1.BoxFormatCollection
formatBox = formatBoxCltn.FirstFormat

While Not formatBox Is Nothing
   ListBox1.Items.Add(formatBox.Name)
   formatBox = formatBoxCltn.NextFormat
End While
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Parameter:		
⇨ FormatName	System.String	Name des Boxformats
Rückgabewert	System.Boolean	Boxformat gelöscht (True)/nicht gelöscht (False)

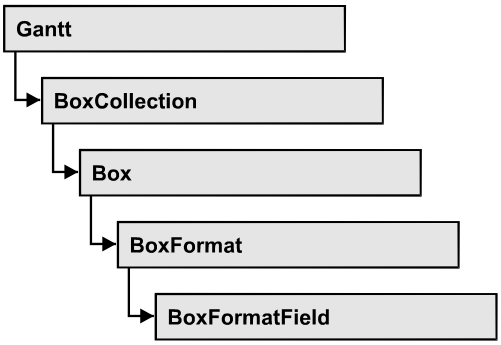
Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormat = boxFormatCltn.FormatByIndex(1)
boxFormatCltn.Remove(boxFormat.Name)
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
- TextFont
- TextFontColor
- Type

Eigenschaften

Alignment

Eigenschaft von VcBoxFormatField

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Boxformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFormatFieldAlignment **Mögliche Werte:** .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Ausrichtung des Feldinhalts 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 = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Alignment = VcFormatFieldAlignment.vcFFACenter
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Eigenschaftswert	System.String	Name des Boxformats

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.FormatName)
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Eigenschaftswert	System.Int16	Höhe der Grafik in mm 0...200

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Eigenschaftswert	System.Int16	Index des Boxformatfeldes

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.Index)
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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 **vcFFTText** ist. Bitte sehen Sie auch die Eigenschaft **MinimumTextLine-Count**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Maximale Zeilenzahl

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTText
boxFormatField.MaximumTextLineCount = 5
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTText;
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 **vcFFTText** 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	Minimale Zeilenzahl 0...20

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTText
boxFormatField.MinimumTextLineCount = 3
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.BoxFormatCollection;
VcBoxFormatField boxFormatField =
boxFormatCltn.FirstFormat().get_FormatField(0);
boxFormatField.Type = VcFormatFieldType.vcFFTText;
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
Eigenschaftswert	System.Int16	Minimale Breite des Boxformatfeldes 0...200

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.MinimumWidth = 100
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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 = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.BackgroundColor = Color.Red
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** -1

Code-Beispiel VB.NET

```
boxFormatField.PatternColor = RGB(0, 255, 0)
```

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
Eigenschaftswert	System.DrawingFont	Schriftart des Boxformatfelds

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
MsgBox(boxFormatField.TextFont.FontFamily.ToString())
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Eigenschaftswert	System.Drawing.Color	Schriftfarbe des Boxformatfelds **Standardwert:** Color.Black

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.TextFontColor = Color.Red
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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
Eigenschaftswert	VcFormatFieldType **Mögliche Werte:** .vcFFTGraphics 64 .vcFFTText 36	Typ des Boxformatfeldes Grafik Text

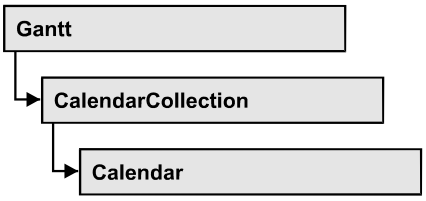
Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormatField As VcBoxFormatField

boxFormatCltn = VcGantt1.BoxFormatCollection
boxFormatField = boxFormatCltn.FirstFormat.FormatField(0)
boxFormatField.Type = VcFormatFieldType.vcFFTGraphics
boxFormatField.GraphicsHeight = 150
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.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. Durch den Kalender übernehmen die verschiedenen Balken und Layer die im Kalender festgelegten Zeitintervalle.

Sie können einen Kalender für Zeitberechnungen verwenden, z. B. um den Zeitunterschied zwischen zwei verschiedenen Terminen in Arbeitstagen zu ermitteln.

Sie können Kalender außerdem verwenden, um Knoten durch arbeitsfreie Intervalle unterbrechen zu lassen.

Außerdem werden Kalender für die Darstellung von Kalendergittern verwendet.

Eigenschaften

- CalendarProfileCollection
- IntervalCollection
- Name
- SecondsPerWorkday
- Specification
- Type

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 Intervallauflistung, 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 = VcGantt1.CalendarCollection.FirstCalendar
calendarName = calendar.Name
```

Code-Beispiel C#

```
VcCalendar calendar = vcGantt1.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 = VcGantt1.CalendarCollection
calendar = calendarCltn.FirstCalendar
MsgBox(calendar.Specification)
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.CalendarCollection;
VcCalendar calendar = calendarCltn.FirstCalendar();
MessageBox.Show(calendar.Specification);
```

Type

Eigenschaft von VcCalendar

Mit dieser Eigenschaft können Sie den Kalendertyp setzen bzw. erfragen. Wenn Sie den Typ ändern, gehen alle für den Kalender gesetzten Eigenschaften verloren.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendarType **Mögliche Werte:** .vcNormalCalendar 139 .vcShiftCalendar 12	Kalendertyp

Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcGantt1.CalendarCollection
calendar = calendarCltn.CalendarByIndex(0)
calendar.Type = VcCalendarType.vcNormalCalendar
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.CalendarCollection;
VcCalendar calendar = calendarCltn.CalendarByIndex(0);
calendar.Type = VcCalendarType.vcNormalCalendar;
```

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
Parameter:		
⇨ date	System.DateTime	Datum, an dem die Dauer eingefügt werden soll
⇨ duration	System.Int32	Anzahl der Zeiteinheiten (z.B. Tage) der Dauer
Rückgabewert	System.DateTime	Datum, an dem die Dauer eingefügt wurde

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim newDate As Date

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
newDate = calendar.AddDuration("16.06.2017", 3)
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter:		
⇨ 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
Rückgabewert	System.Int32	Anzahl der Zeiteinheiten (z.B. Tage) der Dauer

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim duration As Integer

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
duration = calendar.CalcDuration("01.01.2014", "31.12.2014")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter: ⇨ date	System.DateTime	Referenzdatum, zu dem die vorangehende Arbeitszeit ermittelt werden soll
Rückgabewert	System.DateTime	Enddatum der vorangegangenen Arbeitszeit

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim endOfWork As Date

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
endOfWork = calendar.GetEndOfPreviousWorktime("18.06.2014")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter: ⇨ date	System.DateTime	Referenzdatum, zu dem das Anfangsdatum der folgenden Intervallgrenze ermittelt werden soll
Rückgabewert	System.DateTime	Anfangsdatum der folgenden Intervallgrenze

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim nextIntervalBorder As Date

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
nextIntervalBorder = calendar.GetNextIntervalBorder("18.06.2014")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter: ⇨ date	System.DateTime	Referenzdatum, zu dem das Enddatum der vorausgehenden Intervallgrenze ermittelt werden soll
Rückgabewert	System.DateTime	Enddatum der vorausgehenden Intervallgrenze

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim previousIntervalBorder As Date

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
previousIntervalBorder = calendar.GetPreviousIntervalBorder("18.06.2014")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter:		
⇨ date	System.DateTime	Referenzdatum des Intervalls, zu dem das Anfangsdatum ermittelt werden soll
Rückgabewert	System.DateTime	Anfangsdatum des Intervalls

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim startOfInterval As Date

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
startOfInterval = calendar.GetStartOfInterval("18.06.2014")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter:		
⇨ date	System.DateTime	Referenzdatum, zu dem das Anfangsdatum der folgenden Arbeitszeit ermittelt werden soll
Rückgabewert	System.DateTime	Anfangsdatum der folgenden Arbeitszeit

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim startOfNextWorktime As Date

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
startOfNextWorktime = calendar.GetStartOfNextWorktime("18.06.2017")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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
Parameter: ⇨ date	System.DateTime	Datum, das darauf geprüft werden soll, ob es in einer Arbeitszeit liegt
Rückgabewert	System.Boolean	Übergebenes Datum fällt/fällt nicht in eine Arbeitszeit

Code-Beispiel VB.NET

```
Dim calendar As VcCalendar
Dim isWorktime As Boolean

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
isWorktime = calendar.IsWorktime ("18.06.2014")
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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	

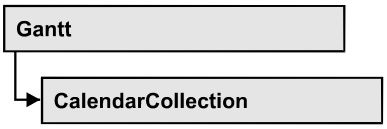
Code-Beispiel VB.NET

```
Dim calendar As VcCalendar

calendar = VcGantt1.CalendarCollection.CalendarByName("WeekCalendar")
calendar.Update()
```

Code-Beispiel C#

```
VcCalendar calendar =
vcGantt1.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 = VcGantt1.WorkdayCollection.CreateWorkday("Work day")
workday.AddNonWorkInterval("00:00:00", "00:00:00")
workday.AddWorkInterval("08:00:00", "16:30:00")
freeday = VcGantt1.WorkdayCollection.CreateWorkday("Workfree day")
freeday.AddNonWorkInterval("00:00:00", "00:00:00")
calendarCltn = VcGantt1.CalendarCollection
calendar = calendarCltn.CreateCalendar("New calendar")
workweek = VcGantt1.WorkweekCollection.CreateWorkweek("Work week")
workweek.AddWorkday(workday, VcWeekday.vcMonday, VcWeekday.vcFriday)
workweek.AddWorkday(freeday, VcWeekday.vcSaturday, VcWeekday.vcSunday)
calendar.AddWorkweek(workweek, "01.01.13", "31.12.14")
calendar.Update()
calendarCltn.Active = calendar
```

Code-Beispiel C#

```
VcWorkday workday = VcGantt1.WorkdayCollection.CreateWorkday("Work day");
workday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));
workday.AddWorkInterval(Convert.ToDateTime("08:00:00"),
Convert.ToDateTime("16:30:00"));
VcWorkday freeday = VcGantt1.WorkdayCollection.CreateWorkday("Workfree day");
freeday.AddNonWorkInterval(Convert.ToDateTime("00:00:00"),
Convert.ToDateTime("00:00:00"));
VcCalendarCollection calendarCltn = VcGantt1.CalendarCollection;
VcCalendar calendar = calendarCltn.CreateCalendar("New calendar");
VcWorkweek workweek = VcGantt1.WorkweekCollection.CreateWorkweek("Work week");
workweek.AddWorkday(workday, VcWeekday.vcMonday, VcWeekday.vcFriday);
workweek.AddWorkday(freeday, VcWeekday.vcSaturday, VcWeekday.vcSunday);
calendar.AddWorkweek(workweek, Convert.ToDateTime("01.01.13"),
Convert.ToDateTime("31.12.14"));
calendar.Update();
calendarCltn.Active = calendar;
```

Count

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 = VcGantt1.CalendarCollection
numberOfCalendar = calendarCltn.Count
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.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
Parameter:		
⇨ calendarName	System.String	Name des Kalenders
Rückgabewert	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
Parameter:		
⇨ Specification	System.String	Kalenderspezifikation
Rückgabewert	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
Parameter:		
⇨ index	System.Int16	Index des Kalenders
Rückgabewert	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
Parameter:		
⇨ calendarName	System.String	Name des Kalenders
Rückgabewert	VcCalendar	Kalender

Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection

calendarCltn = VcGantt1.CalendarCollection
calendarCltn.Active = calendarCltn.CalendarByName("Calendar_1")
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.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
Parameter:		
⇨ calendarName	System.String	Name des zu kopierenden Kalenders
⇨ newCalendarName	System.String	Name des neuen Kalenders
Rückgabewert	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
Rückgabewert	VcCalendar	Erster Kalender

Code-Beispiel VB.NET

```
Dim calendarCltn As VcCalendarCollection
Dim calendar As VcCalendar

calendarCltn = VcGantt1.CalendarCollection
calendar = calendarCltn.FirstCalendar
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.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 VcGantt1.CalendarCollection
   MsgBox(calendar.Name)
Next
```

Code-Beispiel C#

```
foreach (VcCalendar calendar in vcGantt1.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 = VcGantt1.CalendarCollection
calendar = calendarCltn.FirstCalendar

While Not calendar Is Nothing
   ListBox1.Items.Add(calendar.Name)
   calendar = calendarCltn.NextCalendar
End While
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.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)

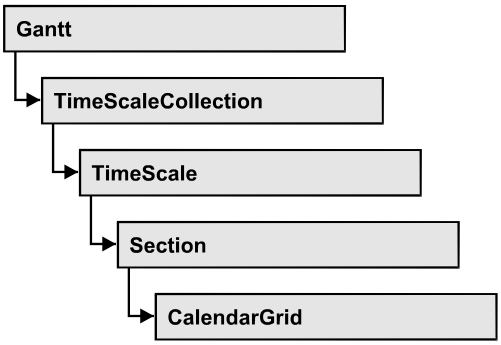
Update

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 VcCalendarGrid



Ein Objekt vom Typ **VcCalendarGrid** ist ein Kalendergitter, durch das arbeitsfreie Tage durch farbige Flächen besonders gekennzeichnet werden.

Eigenschaften

- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- CalendarName
- CalendarNameDataFieldIndex
- CalendarNameMapName
- EndSnapTarget
- Identifiable
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- Name
- Pattern
- PatternColor
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- Priority
- SnapTarget
- Specification
- StartSnapTarget
- UseGraphicalAttributesOfIntervals

- Visible
- VisibleDataFieldIndex
- VisibleMapName

Methoden

- IdentifyInterval

Eigenschaften

BackgroundColor

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie die Farbe der Flächen des Kalendergitters 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.

S. Auch **set/getBackgroundColor** und **set/getPattern**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB Farbwerte **Standardwert:** 14 211 288. Visual Basic: RGB (216, 216, 216)

Code-Beispiel VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Color = Color.Blue
```

Code-Beispiel C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Color = Color.LightSteelBlue;
```

BackgroundColorDataFieldIndex

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **BackColorMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

BackgroundColorMapName

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle 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** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

CalendarName

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie dem Kalendergitter einen Kalender zuweisen um dessen arbeitsfreie Zeiten farblich hervorzuheben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Kalendernamen übergibt

CalendarNameDataFieldIndex

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das den Namen des zu verwendenden Kalenders für das Kalendergitter der Gruppierungsebene enthält. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Kalendergitter** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes, das den Namen des zu verwendenden Kalenders enthält

CalendarNameMapName

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Namen einer Kalenderzuordnungstabelle (Typ vcTextMap) setzen oder erfragen. Wird hier "" angegeben, wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Kalenderzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **CalendarNameDataFieldIndex** angegeben ist, wird der Kalender aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird der Kalender verwendet, der dem Kalendergitter der Gruppierungsebene zugewiesen ist.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Kalenderzuordnungstabelle

EndSnapTarget

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Endtermin dieses Kalendergitters als Einrastziel definiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Endtermin des Kalendergitters wird/wird nicht als Einrastziel definiert.

Identifiable

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Kalendergitter identifizierbar ist. Wenn die Eigenschaft auf **True** gesetzt wurde, kann das Kalendergitter mittels der VcGantt-Methode **IdentifyObjectAt** identifiziert werden. Auch ein Tooltip-Text wird über **OnTooltipText** nur bei einem identifizierbaren Gitter angefordert. Ebenso wird das Ereignis **VcCalendarGridRightClicking** nur dann ausgelöst, wenn das Kalendergitter identifizierbar ist.

Für einen passenden Tooltip-Text muss zudem das entsprechende Intervall identifiziert werden: s. VcGantt-Methode **IdentifyInterval**.

Diese Eigenschaft kann ebenfalls im **Kalendergitter**-Bereich des Dialogs **Zeitskalenabschnitt bearbeiten** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kalendergitter identifizierbar / nicht identifizierbar **Standardwert:** False

Code-Beispiel VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Identifiable = True
```

Code-Beispiel C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Identifiable = true;
```

LineColor

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie die Farbe eines Kalendergitters erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Linienattribute des Kalendergitters** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingColor	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** As defined in the dialog

LineColorDataFieldIndex

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Farbzuordnungstabelle in der Eigenschaft **LineColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

LineColorMapName

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle 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 Farbzuordnungstabelle

LineThickness

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie die Linienstärke eines Kalendergitters 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 Dialog **Linienattribute des Kalendergitters** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the dialog

LineType

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Linientyp eines Kalendergitters erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Linienattribute des Kalendergitters** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLineType **Mögliche Werte:** .vcDashed 4 .vcDashed 4 .vcDashedDotted 5 .vcDashedDotted 5 .vcDotted 3 .vcDotted 3	Linientyp Linientyp **gestrichelt** Linientyp **gestrichelt** Linientyp **gestrichelt-gepunktet** Linientyp **gestrichelt-gepunktet** Linientyp **gepunktet** Linientyp **gepunktet**

.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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

Name

Nur-Lese-Eigenschaft von VcCalendarGrid

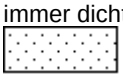
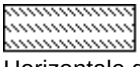
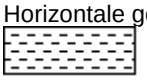
Mit dieser Eigenschaft können Sie den Namen eines Kalendergitters erfragen oder festlegen.

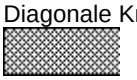
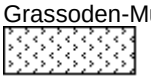
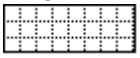
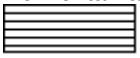
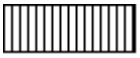
	Datentyp	**Beschreibung**
Eigenschaftswert	System.String	Name des Kalendergitters

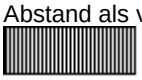
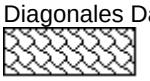
Pattern

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie das Muster des Kalendergitters festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte:	
	.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 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 

PatternColor

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie die Musterfarbe des Kalendergitters 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.

S. auch **set/getBackgroundColor** und **set/getPattern**

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

PatternColorDataFieldIndex

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	**Beschreibung**
Parameter: ⇨ Rückgabewert	System.Int16	Datenfeldindex
Eigenschaftswert	System.Int16	Datenfeldindex

PatternColorMapName

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle 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 Farbzuordnungstabelle

PatternDataFieldIndex

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

PatternMapName

Eigenschaft von VcCalendarGrid

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 Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **PatternDataFieldIndex** angegeben sind, wird das Muster des Kalendergitters aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird das Muster aus der Eigenschaft **Pattern** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Musterzuordnungstabelle

Priority

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie die Priorität des Kalendergitters erfragen oder festlegen. Wenn zwei Objekte dieselbe Position im Diagramm haben, liegt das Objekt mit der höheren Priorität über dem Objekt mit der niedrigeren Priorität. Kalendergitter haben standardmäßig die niedrigste Priorität. Knoten besitzen mit dem Wert 0 die höchste Priorität von allen Objekten. Soll ein Kalendergitter vor den Knoten liegen, müssen Sie seine Priorität auf einen positiven Wert setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Prioritätsstufe {-100 ... 100} **Standardwert:** -20

Code-Beispiel VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Priority = 3
```

Code-Beispiel C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Priority = 3;
```

SnapTarget

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob für dieses Kalendergitter ein Einrastziel am Datum definiert wird.

	Datentyp	Beschreibung

Specification

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie die Spezifikation dieses Kalendergitters 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 Kalendergitters mit der Methode **VcCalendarGridCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Kalendergitters

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGrid = calendarGridCltn.FirstCalendarGrid
MsgBox(calendarGrid.Specification)
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
VcCalendarGrid calendarGrid = calendarGridCltn.FirstCalendarGrid();
MessageBox.Show(calendarGrid.Specification);
```

StartSnapTarget

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Starttermin dieses Kalendergitters als Einrastziel definiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Starttermin des Kalendergitters wird/wird nicht als Einrastziel definiert.

UseGraphicalAttributesOfIntervals

Nur-Lese-Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob die für die Intervalle eingestellten grafischen Attribute dargestellt werden. Diese Option kann auch im Dialog **Intervalle verwalten** (zu erreichen über den Dialog **Kalender festlegen**) eingestellt werden. Wenn die Eigenschaft auf **False**

gesetzt wurde, bleiben die Einstellungen der Eigenschaft **VcInterval.UseGraphicalAttributes** wirkungslos.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Grafische Attribute der Intervalle werden(True) / werden nicht dargestellt (False)

Visible

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Kalendergitter sichtbar ist.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kalendergitter sichtbar / unsichtbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim section As VcSection
Dim calendarGrid As VcCalendarGrid

section = VcGantt1.TimeScaleCollection.Active.Section(0)
calendarGrid = section.CalendarGrid(0)
calendarGrid.Visible = False
```

Code-Beispiel C#

```
VcSection section = vcGantt1.TimeScaleCollection.Active.get_Section(0);
VcCalendarGrid calendarGrid = section.get_CalendarGrid(0);
calendarGrid.Visible = true;
```

VisibleDataFieldIndex

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das den Wert "1" (für "sichtbar") oder 0 (für "unsichtbar") enthält. Diese Eigenschaft kann auch im Dialog **Kalendergitter** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes, das den Sichtbarkeitsmodus enthält

VisibleMapName

Eigenschaft von VcCalendarGrid

Mit dieser Eigenschaft können Sie den Namen einer Zuordnungstabelle (Typ vcTextMap) für den Sichtbarkeitsmodus setzen oder erfragen. Wird hier "" angegeben, wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **VisibleDataFieldIndex** angegeben ist, wird der Sichtbarkeitsstatus aus der Zuordnungstabelle ausgewählt. Diese Eigenschaft kann auch im Dialog **Kalendergitter** festgelegt werden. Trifft kein Datenfeldeintrag in der Zuordnungstabelle zu, wird der Wert aus dem Dialog verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle für den Sichtbarkeitsmodus

Methoden

IdentifyInterval

Methode von VcCalendarGrid

Mit dieser Methode können Sie an den übergebenen Koordinaten ein Intervall-Objekt des Kalenders identifizieren, das dem Kalendergitter zugewiesen wurde. Da Intervalle wiederholt werden, sind meist nicht eindeutig (z.B. kann sich dasselbe Wochenendintervall in einem Zeitbereich von einem Jahr 52 mal wiederholen). Daher liefert die Methode zusätzlich die Anfangs- und Endtermine des betroffenen Intervalls.

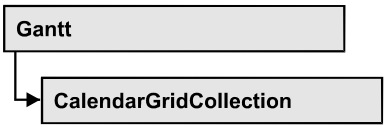
Die Methode kann z.B. innerhalb des TooltipText-Ereignisses aufgerufen werden, um das Intervall unter dem Mauszeiger zu ermitteln.

Falls sich an der bezeichneten Position ein Intervall befindet, wird **True** zurückgegeben, andernfalls **False**.

	Datentyp	Beschreibung
Parameter:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇦ identifiedIntervalParam	VcInterval	Gefundener Intervall

⇦ startDateParam	System.DateTime	Anfangsdatum des identifizierten Intervalls
⇦ endDateParam	System.DateTime	Enddatum des identifizierten Intervalls
Rückgabewert	Boolean	Ein Interval wurde gefunden (True) / nicht gefunden (False)

7.12 VcCalendarGridCollection



In einem Objekt des Typs VcCalendarGridCollection sind alle verfügbaren Kalendergitter zusammengefasst. Über **For Each calendarGrid In CalendarGridCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Kalendergitter zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **CalendarGridByName** und **CalendarGridByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Kalendergitter kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add...**, **Copy...** und **Remove...** ermöglichen das Hinzufügen, Kopieren und Löschen von Kalendergittern.

Eigenschaften

- Count

Methoden

- Add
- AddBySpecification
- CalendarGridByIndex
- CalendarGridByName
- Copy
- FirstCalendarGrid
- GetEnumerator
- NextCalendarGrid
- Remove
- Update

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcCalendarGridCollection

Mit dieser Eigenschaft kann die Anzahl der Kalenderliniengitter in der Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der CalendarGrids

Code-Beispiel VB.NET

```
Dim numberOfDateLine As Integer

numberOfDateLine = VcGantt1.DateLineCollection.Count
```

Code-Beispiel C#

```
int numberOfDateLines = vcGantt1.DateLineCollection.Count;
```

Methoden

Add

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie ein neues Kalenderliniengitter in der CalendarGridCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Kalenderliniengitterobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ calendarGridName	System.String	Name des CalendarGrids
Rückgabewert	VcCalendarGrid	Neues CalendarGrid-Objekt

Code-Beispiel VB.NET

```
newCalendarGrid = VcGantt1.CalendarGridCollection.Add("calendarGrid1")
```

Code-Beispiel C#

```
newCalendarGrid = vcGantt1.CalendarGridCollection.Add("calendarGrid1");
```

AddBySpecification

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie ein Kalenderliniengitter über eine Kalenderliniengitter-Spezifikation erzeugen. Dies dient der Persistenz von Kalenderliniengitter-Objekten. Die Spezifikation eines Kalenderliniengitters kann erfragt und gespeichert werden. Bei einer neuen Sitzung kann das

gleiche Kalenderliniengitter mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ specification	System.String	CalendarGrid-Spezifikation
Rückgabewert	VcCalendarGrid	Neues CalendarGrid-Objekt

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGridCltn.AddBySpecification(textSpecification)
calendarGridCltn.Update()
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
calendarGridCltn.AddBySpecification(textSpecification);
calendarGridCltn.Update();
```

CalendarGridByIndex

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie auf eine einzelne Kalenderliniengitterüber ihren Index zugreifen. Existiert kein Kalenderliniengitter unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des CalendarGrids
Rückgabewert	VcCalendarGrid	Ermitteltes CalendarGridobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByIndex(0)
MsgBox(dateLine.Name)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByIndex(0);
MessageBox.Show(dateLine.Name);
```

CalendarGridByName

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie unter Verwendung des Namens des Kalenderliniengitters auf ein bestimmtes Kalenderliniengitter zugreifen. Existiert kein CalendarGrid-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ calendarGridName	System.String	Name des CalendarGrids
Rückgabewert	VcCalendarGrid	CalendarGrid

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByName("DateLineOne")
MsgBox(dateLine.Name)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByName("DateLineOne");
MessageBox.Show(dateLine.Name);
```

Copy

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie ein Kalenderliniengitter kopieren. Wenn das Kalenderliniengitter mit dem angegebenen Namen existiert und der Name des neuen Kalenderliniengitters noch nicht verwendet wird, wird das neue Kalenderliniengitterobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ calendarGridName	System.String	Name des zu kopierenden CalendarGrids
⇨ newCalendarGridName	System.String	Name des neuen CalendarGrids
Rückgabewert	VcCalendarGrid	CalendarGridobjekt

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGridCltn.Copy("CalendarGridOne", "NewCalendarGrid")
calendarGridCltn.Update()
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
calendarGridCltn.Copy("CalendarGridOne", "NewCalendarGrid");
calendarGridCltn.Update();
```

FirstCalendarGrid

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie auf das erste Kalenderliniengitter der Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextCalendarGrid** über die nachfolgenden Kalenderliniengitter zu iterieren. Existiert kein Kalenderliniengitter in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcCalendarGrid	Erstes CalendarGrid

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGrid = calendarGridCltn.FirstCalendarGrid
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
VcCalendarGrid calendarGrid = calendarGridCltn.FirstCalendarGrid();
```

GetEnumerator

Methode von VcCalendarGridCollection

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 Terminlinien-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
For Each dateLine In dateLineCltn
   ListBox1.Items.Add(dateLine.Name)
Next
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
foreach (VcDateLine dateLine in dateLineCltn)
   listBox1.Items.Add(dateLine.Name);
```

NextCalendarGrid

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Kalenderliniengitter der Auflistung zugreifen, nachdem Sie mit der Methode **FirstCalendarGrid** den Initialwert erfasst haben. Sind alle Levels durchlaufen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcCalendarGrid	Nachfolgendes CalendarGrid

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGrid = calendarGridCltn.FirstCalendarGrid

While Not calendarGrid Is Nothing
   ListBox1.Items.Add(calendarGrid.Name)
   calendarGrid = calendarGridCltn.NextCalendarGrid
End While
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
VcCalendarGrid calendarGrid = calendarGridCltn.FirstCalendarGrid();

while (calendarGrid != null)
   {
   ListBox.Items.Add(calendarGrid.Name);
   calendarGrid = calendarGridCltn.NextCalendarGrid();
   }
```

Remove

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie ein Kalenderliniengitter löschen. Wenn das Terminliniengitter noch in irgendeinem anderen Objekt benutzt wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
Parameter:		
⇨ calendarGridName	System.String	Name des CalendarGrids
Rückgabewert	System.Boolean	CalendarGrid gelöscht (True)/nicht gelöscht (False)

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGrid = calendarGridCltn.CalendarGridByIndex(0)
calendarGridCltn.Remove(calendarGrid.Name)
calendarGridCltn.Update()
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
VcCalendarGrid calendarGrid = calendarGridCltn.CalendarGridByIndex(0);
calendarGridCltn.Remove(calendarGrid.Name);
calendarGridCltn.Update();
```

Update

Methode von VcCalendarGridCollection

Mit dieser Methode können Sie die Darstellung aller Objekte, die durch die verwendeten Terminliniengitter bestimmt werden, aktualisieren. Wenn Sie diese Methode nicht aufrufen, werden die Änderungen der Kalenderliniengitter zur Laufzeit nicht ausgeführt. Sie sollten diese Methode erst am Ende des Codes zur Festlegung der Kalenderliniengitter und der Level-Auflistung verwenden, damit die Aktualisierung nicht schon ausgeführt wird, bevor alle Festlegungen der Kalenderliniengitter ausgeführt worden sind.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

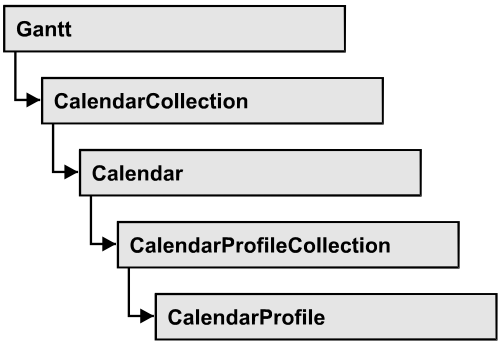
Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection
Dim calendarGrid As VcCalendarGrid

calendarGridCltn = VcGantt1.CalendarGridCollection
calendarGrid = calendarGridCltn.CalendarGridByIndex(0)
calendarGridCltn.Remove(calendarGrid.Name)
calendarGridCltn.Update()
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
VcCalendarGrid calendarGrid = calendarGridCltn.CalendarGridByIndex(0);
calendarGridCltn.Remove(calendarGrid.Name);
calendarGridCltn.Update();
```

7.13 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 = VcGantt1.CalendarProfileCollection
calendarProfile = calendarProfileCltn.FirstCalendarProfile
MsgBox(calendarProfile.Specification)
```

Code-Beispiel C#

```
VcCalendarProfileCollection calendarProfileCltn =
vcGantt1.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 **Mögliche Werte:** .vcDayProfile 4 .vcShiftProfile 5 .vcWeekProfile 3 .vcYearProfile 2	Typ desKalenderprofils

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
Parameter:		
refNameParam	String	Name des Kalenderprofils, hinter das das aktuelle Kalenderprofil gesetzt werden soll

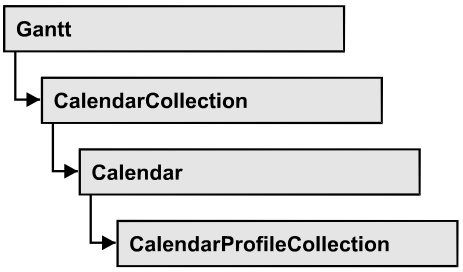
Code-Beispiel VB.NET

```
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();
```

7.14 VcCalendarProfileCollection



In einem Objekt vom Typ VcCalendarProfileCollection sind automatisch alle verfügbaren Kalenderprofile zusammengefasst. Über **For Each calendarProfile 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
Parameter:		
⇨ profileName	System.String	Name des Kalenderprofils
Rückgabewert	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
Parameter:		
⇨ Specification	System.String	Kalenderprofilspezifikation
Rückgabewert	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
Parameter:		
⇨ index	System.Int16	Index des Kalenderprofils
Rückgabewert	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
Parameter:		
⇨ profileName	System.String	Name des CalendarProfile-Objekts
Rückgabewert	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
Parameter:		
⇨ profileName	System.String	Name des zu kopierenden Kalenderprofils
⇨ newProfileName	System.String	Name des neuen Kalenderprofils
Rückgabewert	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 **NextCalendarProfilel** über die nachfolgenden Kalenderprofile zu iterieren. Existiert kein Kalenderprofil in der Auflistung, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	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
Rückgabewert	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
Parameter:		
⇨ profileName	System.String	Name des Kalenderprofils
Rückgabewert	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
Parameter:		
⇨ selectionType	CalendarProfileTypeEnum	Auszuwählender Kalenderprofiltyp
Rückgabewert	System.Int32	Anzahl der ausgewählten Kalenderprofile

Code-Beispiel VB.NET

```
Dim calendarProfileCltn As VcCalendarProfileCollection

Set calendarProfileCltn = VcGantt1.CalendarProfileCollection
calendarProfileCltn.SelectCalendarProfile (vcSelected)
```

Update

Methode von VcCalendarProfileCollection

Mit dieser Methode können Sie eine Kalenderprofil-Collection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

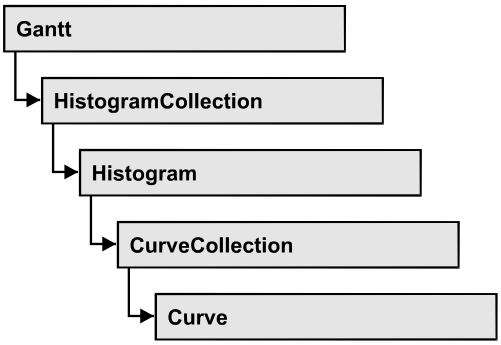
Update

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)

7.15 VcCurve



Ein Objekt vom Typ VcCurve stellt eine Kurve dar, mit der Sie beispielsweise die Ressourcenauslastung und -verfügbarkeit in einem Histogramm darstellen können. Die Werte von Histogrammkurven können Sie entweder direkt setzen oder aus einzelnen Layern ableiten lassen.

Um die Werte direkt zu setzen, wählen Sie im Dialog **Datenquelle der Kurve einstellen** die Option **Daten manuell eingeben** und weisen Sie in Ihrer Anwendung der Kurve mit der Methode **SetValues** Werte zu.

Um die Werte aus den Vorgängen berechnen zu lassen, wählen Sie im Dialog **Datenquelle der Kurve einstellen** die Option **Daten über Layer generieren** und wählen Sie dann einen der angebotenen Layer aus.

Eigenschaften

- Addend
- FillReference1BackgroundColor
- FillReference1Name
- FillReference1Pattern
- FillReference1PatternColor
- FillReference2Color
- FillReference2Name
- FillReference2Pattern
- FillReference2PatternColor
- FilterName
- Histogram
- LayerName
- LineColor
- LineThickness
- LineType
- Marked
- Name

- OverloadResultsCalendarName
- PointsEquidistant
- Source
- Specification
- StackReferenceName
- TimeUnit
- Type
- UnitsPerStep
- UpdateBehaviorName
- ValencyDataFieldIndex
- Visible

Methoden

- Clear
- DeletePoint
- GetFirstOverload
- GetFirstOverloadEx
- GetNextOverload
- GetNextOverloadEx
- GetValues
- GetValuesEx
- SetValues

Eigenschaften

Addend

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie alle y-Werte einer per API generierten Histogrammkurve um den übergebenen Wert erhöhen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Wert, um den die y-Werte der Histogrammkurve erhöht werden sollen

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Addend = 1
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("Availability");

fixCurve.Addend = 1;
```

FillReference1BackgroundColor

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Farbe des Bereichs zwischen einer Histogrammkurve und dem gewählten Bezugsobjekt erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** unter **Muster** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte **Standardwert:** As defined in the **Edit histogram** dialog

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference1BackgroundColor = Color.Blue
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1BackgroundColor = Color.LightSteelBlue;
```

FillReference1Name

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Namen des Bezugsobjekts (x-Achse oder eine Kurve) einer Histogrammkurve für eine Flächenfüllung erfragen.

Zusammen mit der Histogrammkurve begrenzt das Bezugsobjekt eine Fläche, die anschließend farbig und/oder mit Mustern gefüllt werden kann. Diese Eigenschaft wird im Dialog **Histogramm bearbeiten** festgelegt.

Hinweis: Der Name der x-Achse als Bezugskurve muss mit "VC_AXIS" angegeben werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Bezugskurve

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")
curve.FillReference1Name = "VC_AXIS"
```

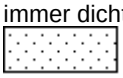
Code-Beispiel C#

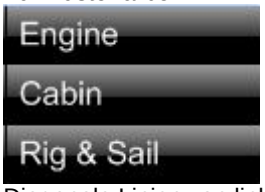
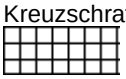
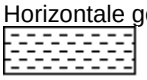
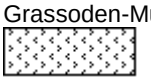
```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1Name = "VC_AXIS";
```

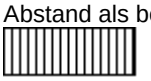
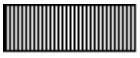
FillReference1Pattern

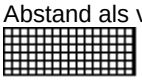
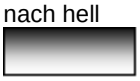
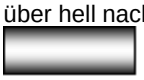
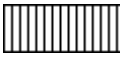
Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie das Füllmuster des Bereichs zwischen einer Histogrammkurve und dem gewählten Bezugsobjekt erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp **Standardwert:** As defined in the **Edit histogram** dialog
	Mögliche Werte: .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 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 histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference1Pattern = VcFillPattern.vcCrossPattern
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1Pattern = VcFillPattern.vcDiagCrossPattern;
```

FillReference1PatternColor

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Vordergrundfarbe des Musters des Bereichs zwischen einer Histogrammkurve und dem eingestellten Bezugsobjekt erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** As defined in the Edit histogram dialog

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference1PatternColor = Color.Blue
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference1PatternColor = Color.LightSteelBlue;
```

FillReference2Color

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Musters oberhalb der zweiten Bezugskurve erfragen oder festlegen.

Die Füllung zur 2. Bezugskurve wird nur dargestellt, wenn die y-Werte der aktuellen Kurve größer sind als die der 2. Bezugskurve.

Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** unter **Muster 2** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte **Standardwert:** As defined in the Edit histogram dialog

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference2BackgroundColor = Color.Blue
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference2BackgroundColor = Color.LightSteelBlue;
```

FillReference2Name

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Namen der 2. Bezugskurve einer Histogrammkurve erfragen oder setzen. Zusammen mit der Histogrammkurve begrenzt diese eine Fläche, die mit einem Muster gefüllt werden kann. Diese Eigenschaft wird im Dialog **Histogramm bearbeiten** festgelegt.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der 2. Bezugskurve

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fillRef As Object

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")
fillRef = histogram.CurveCollection.CurveByName(curve.FillReference2Name)
```

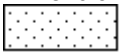
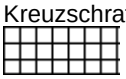
Code-Beispiel C#

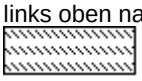
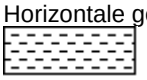
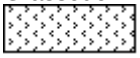
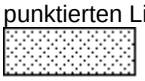
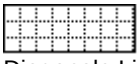
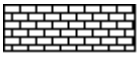
```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

object fillRef =
histogram.CurveCollection.CurveByName(curve.FillReference2Name);
```

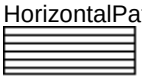
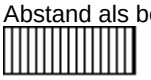
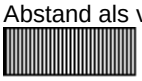
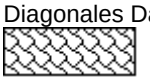
FillReference2Pattern

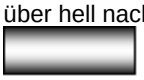
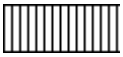
Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie das Füllmuster des Bereichs zwischen einer Histogrammkurve und der 2. Bezugskurve erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcFillPattern	Mustertyp **Standardwert:** As defined in the Edit histogram dialog
	Mögliche Werte: .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 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 histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference2Pattern = VcFillPattern.vcCrossPattern
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference2Pattern = VcFillPattern.vcDiagCrossPattern;
```

FillReference2PatternColor

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Vordergrundfarbe des Musters des Bereichs über der zweiten Bezugskurve erfragen oder festlegen. Die Füllung zur 2. Bezugskurve wird nur dargestellt, wenn die y-Werte der aktuellen Kurve größer sind als die der 2. Bezugskurve.

Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** As defined in the Edit histogram dialog

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FillReference2PatternColor = Color.Blue
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FillReference2PatternColor = Color.LightSteelBlue;
```

FilterName

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie der Kurve einen Filter zuweisen oder den vorhandenen erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Filtername

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.FilterName = "Critical"
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.FilterName = "Critical";
```

Histogram

Nur-Lese-Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie erfragen, zu welchem Histogramm eine Kurve gehört.

	Datentyp	Beschreibung
Eigenschaftswert	VcHistogram	Histogrammobjekt

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

curve =
VcGantt1.HistogramCollection.FirstHistogram.CurveCollection.CurveByName("Curve1"
)
histogram = curve.Histogram
```

Code-Beispiel C#

```
VcCurve curve =
vcGantt1.HistogramCollection.FirstHistogram().CurveCollection.CurveByName("Curve
1");
VcHistogram histogram = curve.Histogram;
```

LayerName

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie der Kurve einen Layer zuweisen oder den vorhandenen erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Layers

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LayerName = "Start-Ende"
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.LayerName = "Start-End";
```

LineColor

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Farbe einer Histogrammkurve erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte **Standardwert:** As defined in the **Edit histogram** dialog

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineColor = Color.Blue
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.LineColor = Color.LightSteelBlue;
```

LineThickness

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Linienstärke einer Histogrammkurve 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 Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the **Edit histogram** dialog

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineType = VcLineType.vcSolid
curve.LineThickness = 3

'or
curve.LineType = VcLineType.vcLineType5
curve.LineThickness = 20
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.LineType = VcLineType.vcSolid;
curve.LineThickness = 3;

//or:
curve.LineType = VcLineType.vcLineType5;
curve.LineThickness = 20;
```

LineType

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Linientyp einer Histogrammkurve erfragen oder festlegen. Soll z. B. bei gestapelten Kurven die Linie nicht dargestellt werden, wählen Sie **vcNone**. Sie können diese Eigenschaft auch im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcLineType	Linientyp **Standardwert:** vcSolid
	Mögliche Werte:	
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.LineType = VcLineType.vcSolid
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.LineType = VcLineType.vcSolid;
```

Marked

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Markierungszustand einer per API generierten Histogrammkurve erfragen oder festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Boolean	Kurve markiert/nicht markiert

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Marked = True
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("Availability");

fixCurve.Marked = true;
```

Name

Nur-Lese-Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Namen einer Histogrammkurve erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Kurvenname

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim curveName As String

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curveName = curve.Name
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

string curveName = curve.Name;
```

OverloadResultsCalendarName

Eigenschaft von VcCurve

Mithilfe dieser Eigenschaft können Sie einen Kalender setzen oder erfragen, in den die Intervalle, die sich aus den berechneten Overload-Terminen ergeben, geschrieben werden. Dieser Kalender kann z.B. für die Darstellung eines Kalendergitters in der Gruppe verwendet werden

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalenderobjektes für die Overload-Ergebnisse

PointsEquidistant

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Punkte der Kurve äquidistant sein sollen. Bei **False** werden Kurvenpunkte nur an den Stellen erzeugt, an denen sich der y-Wert ändert. Sie können diese Eigenschaft auch im Dialog **Datenquelle der Kurve einstellen** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kurvenpunkte äquidistant (True)/nicht äquidistant (False)

Source

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Quelle, aus der die Daten einer Histogrammkurve entnommen werden, setzen oder erfragen. Setzen können Sie diese Eigenschaft im Dialog **Datenquelle der Kurve einstellen**. Wenn Sie **vcSetCurve** (erzeugt durch die Option **Daten über Layer generieren** im Dialog **Datenquelle der Kurve einstellen**) zurück erhalten, können Sie die Daten in Ihrer Anwendung mit der Methode **SetValues** setzen. Wenn Sie **vcCalculateFromLayer** zurück erhalten (Option **Daten automatisch über Layer generieren**), werden die Daten aus den Layern berechnet.

	Datentyp	Beschreibung
Eigenschaftswert	VcCurveSource	Art der Werteberechnung, Kurvenwerte setzen
	Mögliche Werte:	
	.vcCalculateFromLayer 1	Werteberechnung aus Layer
	.vcSetCurve 3	Die Werte werden manuell (per API) gesetzt.

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim source As VcSource

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

source = curve.Source
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");
VcSource source = curve.Source;
```

Specification

Nur-Lese-Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Spezifikation dieser Kurve 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 Kurve mit der Methode **VcCurveCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Kurvenspezifikation

Code-Beispiel VB.NET

```
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

curveCltn = VcGantt1.CurveCollection
curve = curveCltn.FirstCurve
MsgBox(curve.Specification)
```

Code-Beispiel C#

```
VcCurveCollection curveCltn = vcGantt1.CurveCollection;
VcCurve curve = curveCltn.FirstCurve();
MessageBox.Show(curve.Specification);
```

StackReferenceName

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Namen der Stapelbezug-Kurve einer Histogrammkurve setzen oder erfragen. Er legt für jede Kurve fest, auf welche andere Kurve sie gestapelt werden soll. Er muss angegeben werden, wenn Kurven aufeinander gestapelt werden sollen. Diese Eigenschaft kann auch im Dialog **Histogramm bearbeiten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Stapelbezugkurve

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim referenceCurve As Object

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

referenceCurve = histogram.CurveCollection.CurveByName(curve.StackReferenceName)
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

object referencecurve =
histogram.CurveCollection.CurveByName(curve.StackReferenceName);
```

TimeUnit

Nur-Lese-Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Zeiteinheit einer Histogrammkurve erfragen. Sie ist nur auf äquidistante Kurven anwendbar, die per API generiert wurden. Wenn Sie die Eigenschaft auf eine aus Layerwerten generierte Kurve anwenden, wird das Ergebnis -1 zurückgeliefert. Die Zeiteinheit können Sie auf der Eigenschaftenseite **Allgemeines** festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcTimeUnit	Zeiteinheit **Standardwert:** -1
	Mögliche Werte:	
	.vcDay 5	Zeiteinheit **Tag**
	.vcHour 6	Zeiteinheit **Stunde**
	.vcMinute 7	Zeiteinheit **Minute**
	.vcSecond 8	Zeiteinheit **Sekunde**

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim timeUnit As VcTimeUnit

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

timeUnit = curve.TimeUnit
```

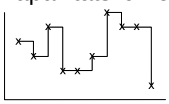
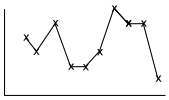
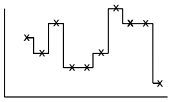
Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");
VcTimeUnit timeUnit = curve.TimeUnit;
```

Type

Nur-Lese-Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Typ einer Histogrammkurve erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcCurveType	Kapazitätskurve **Standardwert:** vcCapacityCurve
	Mögliche Werte: .vcCapacityCurve 215	Kapazitätskurve 
	.vcLineCurve 214	Linienkurve 
	.vcStepCurve 216	Stufenkurve 

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim type As VcType

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")
type = curve.Type
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");
VcType type = curve.Type;
```

UnitsPerStep

Nur-Lese-Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie die Anzahl der Einheiten pro Schritt für eine Histogrammkurve erfragen. Sie ist nur auf äquidistante Kurven anwendbar, die per API generiert wurden. Die Anzahl kann auf der Eigenschaftenseite **Allgemeines** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl Einheiten **Standardwert:** -1

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim unitsPerStep As Integer

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

unitsPerStep = curve.UnitsPerStep
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

int unitsPerStep = curve.UnitsPerStep;
```

UpdateBehaviorName

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

ValencyDataFieldIndex

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie das Wertigkeitsfeld einer automatisch generierten Kurve erfragen oder setzen, d. h. das Datenfeld, aus dem pro Vorgang die Wertigkeit für die Addition der Kapazitäten entnommen wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Wertigkeitsfeldes

Visible

Eigenschaft von VcCurve

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Kurve sichtbar sein soll oder nicht. Diese Eigenschaft kann auch im Dialog **Histogramme verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kurve sichtbar/unsichtbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

curve.Visible = True
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

curve.Visible = true;
```

Methoden

Clear

Methode von VcCurve

Mit dieser Methode werden die y-Werte aller Punkte der Kurve auf Null gesetzt. Diese Methode kann nur auf Kurven angewendet werden, deren Werte per Automation definiert worden sind.

	Datentyp	Beschreibung
Rückgabewert	Void	

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim fixCurve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
fixCurve = histogram.CurveCollection.CurveByName("Availability")

fixCurve.Clear()
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("Availability");

fixCurve.Clear();
```

DeletePoint

Methode von VcCurve

Mit dieser Methode können Sie bei einer per API erzeugten Kurve den Kurvenpunkt, der der angegebenen x-Koordinate am nächsten ist, löschen.

	Datentyp	Beschreibung
Parameter:		
⇨ x	System.Int32	x-Wert des zu löschenden Kurvenpunkts
⇨ y	System.Int32	y-Wert des zu löschenden Kurvenpunkts
⇦ pointDate	System.DateTime	Datum des Kurvenpunktes, der gelöscht worden ist
Rückgabewert	System.Boolean	Kurvenpunkt erfolgreich/nicht erfolgreich gelöscht

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurveRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurveClickingEventArgs) Handles VcGantt1.VcCurveRightClicking

   Dim pointDate As Date
   Dim deleted As Boolean

   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup
   deleted = e.Curve.DeletePoint(e.X, e.Y, pointDate)
   If deleted = True Then
      Call MsgBox(pointDate)
   End If

End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurveRightClicking(object sender,
NETRONIC.XGantt.VcCurveClickingEventArgs e)
   {
   DateTime pointDate = new DateTime();
   bool deleted;
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   deleted = e.Curve.DeletePoint(e.X, e.Y, ref pointDate);
   if (deleted == true)
      MessageBox.Show(pointDate.ToString());
}
```

GetFirstOverload

Methode von VcCurve

Ein Overload ist ein Zeitbereich, in dem die aktuelle Kurve höhere Werte aufweist als die Bezugskurve. Die Bezugskurve ist die Kurve, die im Dialog **Histogramm bearbeiten** als **2. Füllbezug** zur aktuellen Kurve definiert ist.

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Overload zugreifen, um anschließend in einer Schleife mit der Methode **GetNextOverload** über die nachfolgenden Overloads zu iterieren.

Hinweis:Für Fließkommazahlen in den Parametern **fromValue** und **toValue** verwenden Sie bitte die Methode **GetFirstOverloadEx**.

	Datentyp	Beschreibung
Parameter:		
⇦ fromDate	System.DateTime	Anfangsdatum des Overload-Bereiches
⇦ fromValue	System.Int32	Y-Wert am Anfangsdatum des Overload-Bereiches
⇦ toDate	System.DateTime	Enddatum des Overload Bereiches
⇦ toValue	System.Int32	y-Wert am Enddatum des Overload-Bereiches
Rückgabewert	System.Boolean	Overload erfolgreich/nicht erfolgreich erfragt

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fixCurve As VcCurve
Dim fromDate As Date
Dim toDate As Date
Dim fromValue As Integer
Dim toValue As Integer
Dim yValues As String
Dim bOk As Boolean

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
fixCurve = histogram.CurveCollection.CurveByName("FixCurve")
yValues = "6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6"
fixCurve.SetValues("31.08.14", yValues)
bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("FixCurve");
DateTime fromDate = new   DateTime();
DateTime toDate = new DateTime();
int fromValue = 0;
int toValue = 0;
string yValues =
"6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6";
fixCurve.SetValues(Convert.ToDateTime("31.08.14"),yValues);
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
```

GetFirstOverloadEx

Methode von VcCurve

Ein Overload ist ein Zeitbereich, in dem die aktuelle Kurve höhere Werte aufweist als die Bezugskurve. Die Bezugskurve ist die Kurve, die im Dialog **Histogramm bearbeiten** als **2. Füllbezug** zur aktuellen Kurve definiert ist.

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Overload zugreifen, um anschließend in einer Schleife mit der Methode **GetNextOverloadEx** über die nachfolgenden Overloads zu iterieren.

Hinweis: Im Vergleich zu der Methode **GetFirstOverload** erlaubt diese Methode Fließkommazahlen in den Parametern **fromValue** und **toValue**.

	Datentyp	**Beschreibung**
Parameter:		
⇦ fromDate	System.DateTime	Anfangsdatum des Overload-Bereiches
⇦ fromValue	System.Double	Y-Wert am Anfangsdatum des Overload-Bereiches
⇦ toDate	System.DateTime	Enddatum des Overload Bereiches
⇦ toValue	System.Double	y-Wert am Enddatum des Overload-Bereiches
Rückgabewert	System.Boolean	Overload erfolgreich/nicht erfolgreich erfragt

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim fixCurve As VcCurve
Dim fromDate As Date
Dim toDate As Date
Dim fromValue As Integer
Dim toValue As Integer
Dim yValues As String
Dim bOk As Boolean

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
fixCurve = histogram.CurveCollection.CurveByName("FixCurve")
yValues = "6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6"
fixCurve.SetValues("31.08.14", yValues)
bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
VcCurve fixCurve = histogram.CurveCollection.CurveByName("FixCurve");
DateTime fromDate = new   DateTime();
DateTime toDate = new DateTime();
int fromValue = 0;
int toValue = 0;
string yValues =
"6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6;0;0;6;6;6;6;6";
fixCurve.SetValues(Convert.ToDateTime("31.08.14"),yValues);
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
```

GetNextOverload

Methode von VcCurve

Ein Overload ist ein Zeitbereich, in dem die aktuelle Kurve höhere Werte aufweist als die Bezugskurve. Die Bezugskurve ist die Kurve, die im Dialog **Histogramm bearbeiten** als **2. Füllbezug** zur aktuellen Kurve definiert ist.

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Overloads zugreifen, nachdem Sie mit der Methode **GetFirstOverload** den Initialwert erfasst haben.

Hinweis:Für Fließkommazahlen in den Parametern **fromValue** und **toValue** verwenden Sie bitte die Methode **GetNextOverloadEx**.

	Datentyp	**Beschreibung**
Parameter:		
⇦ fromDate	System.DateTime	Anfangsdatum des Overload-Bereiches
⇦ fromValue	System.Int32	Y-Wert am Anfangsdatum des Overload-Bereiches
⇦ toDate	System.DateTime	Enddatum des Overload Bereiches
⇦ toValue	System.Int32	y-Wert am Enddatum des Overload-Bereiches
Rückgabewert	System.Boolean	Overload erfolgreich/nicht erfolgreich erfragt.

Code-Beispiel VB.NET

```
...
Dim bOk As Boolean

bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " ) - " +
toDate.ToString() + " ( " + toValues.ToString() + " ) ")
While bOk
            bOk = curve.GetNextOverload(fromDate, fromValue, toDate, tovalue)
            If bOk Then
                MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " )
- " + toDate.ToString() + " ( " + toValues.ToString() + " ) ")
            End If
        End While
```

Code-Beispiel C#

```
...
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
MessageBox.Show(fromDate.ToString()  + " ( " + fromValue.ToString()  + " ) - " +
toDate.ToString()  + " ( " + toValue.ToString()  + " )");
while (bOk == true)
   {
   bOk = curve.GetNextOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
   if (bOk == true)
      MessageBox.Show(fromDate.ToString() + " ( " + fromValue.ToString() + " ) -
" + toDate.ToString() + " ( " + toValue.ToString() + " )");
   }
```

GetNextOverloadEx

Methode von VcCurve

Ein Overload ist ein Zeitbereich, in dem die aktuelle Kurve höhere Werte aufweist als die Bezugskurve. Die Bezugskurve ist die Kurve, die im Dialog **Histogramm bearbeiten** als **2. Füllbezug** zur aktuellen Kurve definiert ist.

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Overloads zugreifen, nachdem Sie mit der Methode **GetFirstOverloadEx** den Initialwert erfasst haben.

Hinweis: Im Vergleich zu der Methode **GetNextOverload** erlaubt diese Methode Fließkommazahlen in den Parametern **fromValue** und **toValue**.

	Datentyp	**Beschreibung**
Parameter:		
⇦ fromDate	System.DateTime	Anfangsdatum des Overload-Bereiches
⇦ fromValue	System.Double	Y-Wert am Anfangsdatum des Overload-Bereiches
⇦ toDate	System.DateTime	Enddatum des Overload Bereiches
⇦ toValue	System.Double	y-Wert am Enddatum des Overload-Bereiches

Rückgabewert	System.Boolean	Overload erfolgreich/nicht erfolgreich erfragt.

Code-Beispiel VB.NET

```
...
Dim bOk As Boolean

bOk = curve.GetFirstOverload(fromDate, fromValue, toDate, toValue)
MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " ) - " +
toDate.ToString() + " ( " + toValues.ToString() + " ) ")
While bOk
            bOk = curve.GetNextOverload(fromDate, fromValue, toDate, tovalue)
            If bOk Then
                MsgBox(fromDate.ToString() + " ( " + fromValues.ToString() + " )
- " + toDate.ToString() + " ( " + toValues.ToString() + " ) ")
            End If
        End While
```

Code-Beispiel C#

```
...
bool bOk = curve.GetFirstOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
MessageBox.Show(fromDate.ToString()  + " ( " + fromValue.ToString()  + " ) - " +
toDate.ToString()  + " ( " + toValue.ToString()  + " )");
while (bOk == true)
   {
   bOk = curve.GetNextOverload(ref fromDate, ref fromValue, ref toDate, ref
toValue);
   if (bOk == true)
      MessageBox.Show(fromDate.ToString() + " ( " + fromValue.ToString() + " ) -
" + toDate.ToString() + " ( " + toValue.ToString() + " )");
   }
```

GetValues

Methode von VcCurve

Mit dieser Methode können Sie den Wert einer Histogrammkurve zu einem bestimmten Datum erfragen. Da nicht sichergestellt ist, dass die Kurve an diesem bestimmten Datum auch einen Stützwert besitzt, werden jeweils das Datum und der Wert des nächsten Stützpunktes vor bzw. nach dem angegebenen Datum ausgegeben. Wird ein Stützpunkt genau getroffen, so wird der zugehörige Wert zwei Mal, d.h. als voriger und nächster Wert ausgegeben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ inputDate	System.DateTime	Datum, an dem der Wert der Histogrammkurve erfragt werden soll
⇦ leftDate	System.DateTime	Datum des letzten Stützpunktes vor dem gewünschten Datum
⇦ leftValue	System.Int32	Wert des letzten Stützpunktes vor dem gewünschten Datum

⇦ rightDate	System.DateTime	Datum des nächsten Stützpunktes nach dem gewünschten Datum
⇦ rightValue	System.Int32	Wert des nächsten Stützpunktes nach dem gewünschten Datum
Rückgabewert	void	

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim inputDate As String
Dim leftDate As Date
Dim rightDate As Date
Dim leftValue As Integer
Dim rightValue As Integer

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
inputDate = InputBox("Date: ")
curve.GetValues(inputDate, leftDate, leftValue, rightDate, rightValue)
MsgBox(leftDate.ToString() & " ( " & leftValue.ToString() & " ) " &
rightDate.ToString() & " ( " & rightValue.ToString() & " ) ")
```

Code-Beispiel C#

```
DateTime leftDate = new DateTime();
DateTime rightDate = new DateTime();
int leftValue = 0;
int rightValue = 0;

VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
curve.GetValues(Convert.ToDateTime("01.05.2014"), ref leftDate, ref leftValue,
ref rightDate, ref rightValue;
MessageBox.Show(leftDate.ToString() + " ( " + leftValue.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValue.ToString() + " ) ");
```

GetValuesEx

Methode von VcCurve

Mit dieser Methode können Sie den Wert einer Histogrammkurve zu einem bestimmten Datum erfragen. Im Vergleich zur Methode **GetValues** können Sie hier auch Fließkomma-Zahlen verwenden. Da nicht sichergestellt ist, dass die Kurve an diesem bestimmten Datum auch einen Stützwert besitzt, werden jeweils das Datum und der Wert des nächsten Stützpunktes vor bzw. nach dem angegebenen Datum ausgegeben. Wird ein Stützpunkt genau getroffen, so wird der zugehörige Wert zwei Mal, das heißt, als voriger und nächster Wert ausgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ inputDate	System.DateTime	Datum, an dem der Wert der Histogrammkurve erfragt werden soll
⇦ leftDate	System.DateTime	Datum des letzten Stützpunktes vor dem gewünschten Datum
⇦ leftValue	System.Double	Wert des letzten Stützpunktes vor dem gewünschten Datum
⇦ rightDate	System.DateTime	Datum des nächsten Stützpunktes nach dem gewünschten Datum
⇦ rightValue	System.Double	Wert des nächsten Stützpunktes nach dem gewünschten Datum
Rückgabewert	void	

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim inputDate As String
Dim leftDate As Date
Dim rightDate As Date
Dim leftValues As Double
Dim rightValues As Double

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("LayerCurve")
inputDate = InputBox("Date: ")
curve.GetValuesEx(inputDate, leftDate, leftValues, rightDate, rightValues)
MsgBox(leftDate.ToString() + " ( " + leftValues.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValues.ToString() + " ) ")
```

Code-Beispiel C#

```
DateTime leftDate = new DateTime();
DateTime rightDate = new DateTime();
double leftValue = 0;
double rightValue = 0;

VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("LayerCurve");
curve.GetValuesEx(Convert.ToDateTime("01.05.2009"), ref leftDate, ref
leftValues, ref rightDate, ref rightValues);
MessageBox.Show(leftDate.ToString() + " ( " + leftValues.ToString() + " ) " +
rightDate.ToString() + " ( " + rightValues.ToString() + " ) ");
```

SetValues

Methode von VcCurve

Mit dieser Methode können Sie die Werte einer per API generierten Histogrammkurve setzen. Eine mittels **SetValues** gesetzte Kurve kann beispielsweise zur Anzeige der Maschinenkapazität verwendet werden oder als Bezugskurve dienen.

Von dem Kontrollkästchen **Kurvenpunkte äquidistant** im Dialog **Datenquelle der Kurve einstellen** hängt ab, wie diese Methode anzuwenden ist:

Kurvenpunkte äquidistant: Man kann mit der Methode einen Startwert (**startValue**) und einen String, der durch Semikola getrennt die y-Werte enthält, übergeben. Aus dem Startwert für x und dem String mit den y-Werten werden zusammen mit der **Zeiteinheit** und dem Wert für das **Kleinste Zeitintervall** (Eigenschaftenseite **Allgemeines**) die Wertepaare der Histogrammkurve ermittelt. So gesetzte Kurven können nicht interaktiv verändert werden.

Kurvenpunkte nicht äquidistant: Man muss die Methode für jedes Wertepaar einzeln aufrufen und jeweils Paare von x-y-Werten übergeben. Die **Zeiteinheit** und das **Kleinste Zeitintervall** sind hier irrelevant. Die Kurve kann hier interaktiv verändert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ startDate	System.DateTime	Startdatum
⇨ values	System.String	y-Werte als Sting
Rückgabewert	System.Boolean	Werte erfolgreich/nicht erfolgreich gesetzt

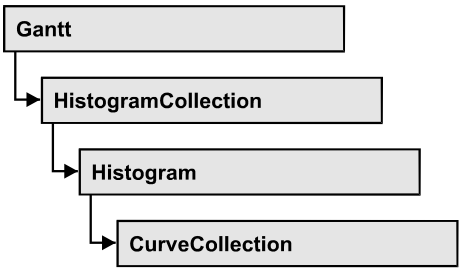
Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve
Dim yValues As String

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curve = histogram.CurveCollection.CurveByName("Curve1")

' If the option Curve points equidistant is checked for the curve:
yValues = "5;1;1;2;2;2;4;5;5;3;2;1;"
curve.SetValues("01.05.2014", yValues)

' If the option Curve points equidistant is not checked for the curve:
curve.SetValues("01.05.2014", 5)
curve.SetValues("03.05.2014", 1)
curve.SetValues("07.05.2014", 1)
curve.SetValues("16.05.2014", 2)
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HiSTOGRAM_1");
VcCurve curve = histogram.CurveCollection.CurveByName("Curve1");

//If the option Curve points equidistant is checked for the curve:
string yValues = "5;1;1;2;2;2;4;5;5;3;2;1;";
curve.SetValues(Convert.ToDateTime("01.05.2014"), yValues);

//If the option Curve points equidistant is not checked for the curve:
curve.SetValues(Convert.ToDateTime("01.05.2014"), "5");
curve.SetValues(Convert.ToDateTime("03.05.2014"), "1");
curve.SetValues(Convert.ToDateTime("07.05.2014"), "1");
curve.SetValues(Convert.ToDateTime("16.05.2014"), "2");
```

7.16 VcCurveCollection



In einem Objekt vom Typ VcCurveCollection sind alle Kurven eines Histogramms zusammengefasst. Über **For Each curve In CurveCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Kurven zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **CurveByName** und **CurveByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Kurven kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add...**, **Copy...** und **Remove...** ermöglichen das Hinzufügen, Kopieren und Löschen von Kurven.

Eigenschaften

- Count

Methoden

- Add
- AddBySpecification
- Copy
- CurveByIndex
- CurveByName
- FirstCurve
- GetEnumerator
- NextCurve
- Remove

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcCurveCollection

Mit dieser Eigenschaft können Sie die Anzahl der Kurven in der Kurven-Auflistung abfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl Kurven

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim numberOfCurves As Integer

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

numberOfCurves = curveCltn.Count
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;

int numberOfCurves = curveCltn.Count;
```

Methoden

Add

Methode von VcCurveCollection

Mit dieser Methode können Sie eine neue Kurve in der CurveCollection anlegen. Wenn der Name noch nicht verwendet wird, wird das neue Kurvenobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ curveName	System.String	Name der Kurve
Rückgabewert	VcCurve	Neues Kurvenobjekt

Code-Beispiel VB.NET

```
newCurve =
VcGantt1.HistogramCollection.HistogramByName("a").CurveCollection.Add("test1")
```

Code-Beispiel C#

```
newCurve =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").CurveCollection.Add(
"test1");
```

AddBySpecification

Methode von VcCurveCollection

Mit dieser Methode können Sie eine Kurve über eine Kurven-Spezifikation erzeugen. Dies dient der Persistenz von Kurvenobjekten. Die Spezifikation einer Kurve kann erfragt (siehe VcCurve-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann die gleiche Kurve mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ specification	System.String	Kurvenspezifikation
Rückgabewert	VcCurve	Neues Kurvenobjekt

Copy

Methode von VcCurveCollection

Mit dieser Methode können Sie eine Kurve kopieren. Wenn die Kurve mit dem angegebenen Namen existiert und der Name der neuen Kurve noch nicht verwendet wird, wird das neue Kurvenobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	**Beschreibung**
Parameter:		
⇨ curveName	System.String	Name der zu kopierenden Kurve
⇨ newCurveName	System.String	Name der neuen Kurve
Rückgabewert	VcCurve	Kurvenobjekt

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

histogram = VcGantt1.HistogramCollection.FirstHistogram
curveCltn = histogram.CurveCollection
curveCltn.Copy("CurrentCurve", "NewCurve")
```

Code-Beispiel C#

```
VcHistogram histogram  = vcGantt1.HistogramCollection.FirstHistogram();
VcCurveCollection curveCltn = histogram.CurveCollection;
curveCltn.Copy("CurrentCurve", "NewCurve");
```

CurveByIndex

Methode von VcCurveCollection

Mit dieser Methode können Sie auf eine einzelne Kurve über ihren Index zugreifen. Existiert keine Kurve an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index der Kurve
Rückgabewert	VcCurve	Ermitteltes Kurvenobjekt

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.CurveByIndex(2)
```

Code-Beispiel C#

```
VcHistogram histogram =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
VcCurve curve = curveCltn.CurveByIndex(2);
```

CurveByName

Methode von VcCurveCollection

Mit dieser Methode können Sie unter Verwendung des Kurvennamens auf eine bestimmte Kurve zugreifen. Existiert keine Kurve unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ curveName	System.String	Name der Kurve
Rückgabewert	VcCurve	Kurve

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.CurveByName("Curve1")
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;

VcCurve curve = curveCltn.CurveByName("Curve1");
```

FirstCurve

Methode von VcCurveCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Kurve der Kurven-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextCurve** über die nachfolgenden Kurven zu iterieren. Existiert keine Kurve in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcCurve	Erste Kurve

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.FirstCurve
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
VcCurve curve = curveCltn.FirstCurve;
```

GetEnumerator

Methode von VcCurveCollection

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 Kurven-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.FirstHistogram
For Each curve In histogram.CurveCollection
   ListBox1.Items.Add(curve.Name)
Next
```

Code-Beispiel C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
foreach (VcCurve curve in histogram.CurveCollection)
   listBox1.Items.Add(curve.Name);
```

NextCurve

Methode von VcCurveCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Kurven des CurveCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstCurve** den Initialwert erfasst haben. Sind alle Kurven durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcCurve	Folgekurve

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection
Dim curve As VcCurve

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection

curve = curveCltn.FirstCurve

While Not newCurve Is Nothing
   newCurve = curveCltn.NextCurve
End While
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
VcCurve curve = curveCltn.FirstCurve();

while (curve == null)
   {
   curve = curveCltn.NextCurve();
   }
```

Remove

Methode von VcCurveCollection

Mit dieser Methode können Sie eine Kurve löschen. Wenn die Kurve noch in irgendeinem anderen Objekt benutzt wird, kann sie nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	**Beschreibung**
Parameter:		
⇨ curveName	System.String	Name der Kurve
Rückgabewert	System.Boolean	Kurve gelöscht (True)/nicht gelöscht (False)

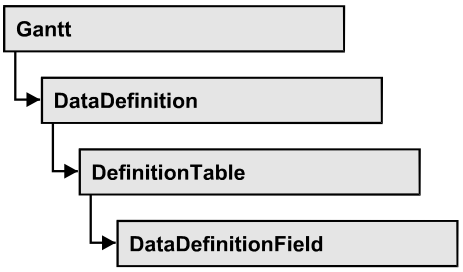
Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

histogram = VcGantt1.HistogramCollection.FirstHistogram
curveCltn = histogram.CurveCollection
curveCltn.Remove("CurrentCurve")
```

Code-Beispiel C#

```
VcHistogram histogram  = vcGantt1.HistogramCollection.FirstHistogram();
VcCurveCollection curveCltn = histogram.CurveCollection;
curveCltn.Remove("CurrentCurve");
```

7.17 VcDataDefinitionField



Ein Objekt vom Typ VcDefinitionField definiert ein Feld der Datendefinitionstabelle. Die Definition besteht im Kern aus einem Namen und einer Festlegung des Datentyps.

Eigenschaften

- DateFormat
- Editable
- Hidden
- Index
- Name
- Type

Eigenschaften

DateFormat

Eigenschaft von VcDataDefinitionField

Mit dieser Eigenschaft können Sie das Datumsformat des Feldes in einer Datendefinitionstabelle festlegen oder erfragen. Diese Eigenschaft ist nur wirksam, wenn der Datentyp des Feldes auf **vcDefFieldDateTimeType** eingestellt ist.

Die DateFormat-Einstellung wird nur beim Lesen und Schreiben von CSV-Dateien verwendet und wenn der Formattyp **String** beim Hinzufügen eines Datensatzes über die Methoden **InsertNodeRecord** oder **InsertLinkRecord** verwendet wird.

Das Format der Datumsausgabe in der Grafik wird über die Eigenschaft **DateOutputFormat** gesteuert.

Hinweis: Es sollte zuerst die Eigenschaft Type gesetzt werden, bevor die Eigenschaft DateFormat vereinbart wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Datumsformat {DMYhms:;./} **Standardwert:** bei vcDefFieldDateTime DD.MM.YYYY hh:mm:ss

Code-Beispiel VB.NET

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGantt1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType
'DateFormat = "01.12.2014"
dataDefField.DateFormat = "DD.MM.YYYY"
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefTable =
vcGantt1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName("Start");
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType;
//DateFormat = "01.12.2014"
dataDefField.DateFormat = "DD.MM.YYYY";
vcGantt1.DataTableCollection.Update();
```

Editable

Eigenschaft von VcDataDefinitionField

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
Eigenschaftswert	System.Boolean	Definitionsfeld editierbar/nicht editierbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGantt1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")
dataDefField.Editable = False
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefTable =
vcGantt1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName("Start");
dataDefField.Editable = false;
vcGantt1.DataTableCollection.Update();
```

Hidden

Eigenschaft von VcDataDefinitionField

Mit dieser Eigenschaft können Sie festlegen bzw. erfragen, ob ein Datenfeld zur Laufzeit versteckt ist.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Definitionsfeld versteckt/nicht versteckt **Standardwert:** False

Code-Beispiel VB.NET

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGantt1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")
dataDefField.Hidden = True
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefTable =
vcGantt1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName("Start");
dataDefField.Hidden = true;
vcGantt1.DataTableCollection.Update();
```

Index

Nur-Lese-Eigenschaft von VcDataDefinitionField

Mit dieser Eigenschaft können Sie den Index des Feldes einer Datendefinitionstabelle abfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Definitionsfeldes

Code-Beispiel VB.NET

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGantt1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")
MsgBox(dataDefField.Index.ToString())
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefTable =
vcGantt1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName("Start");
MessageBox.Show(dataDefField.Index.ToString());
```

Name

Eigenschaft von VcDataDefinitionField

Mit dieser Eigenschaft können Sie den Namen des Feldes einer Datendefinitionstabelle setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Definitionsfeldes

Code-Beispiel VB.NET

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGantt1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.CreateDataDefinitionField("Start")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefTable =
vcGantt1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.CreateDataDefinitionField("Start");
vcGantt1.DataTableCollection.Update();
```

Type

Eigenschaft von VcDataDefinitionField

Mit dieser Eigenschaft können Sie den Typ des Feldes einer Datendefinitionstabelle erfragen oder setzen.

Hinweis: Durch Setzen der Eigenschaft **Type** wird die Eigenschaft **DateFormat** geändert!

vcDefFieldAlphanumericType: DateFormat = ""

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

vcDefFieldIntegerType: DateFormat = ""

	Datentyp	Beschreibung
Eigenschaftswert	VcDataDefinitionFieldType	Typ des Definitionsfeldes **Standardwert:** vcDefFieldIntegerType
	Mögliche Werte: .vcDefFieldAlphanumericType 1 .vcDefFieldDateTimeType 4 .vcDefFieldIntegerType 2	 Datentyp **Alphanumerisch** Datentyp **Datum** Datentyp **Integer** (32 Bit)

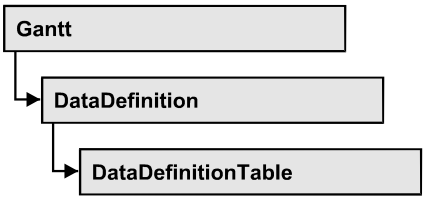
Code-Beispiel VB.NET

```
Dim dataDefTable As VcDataDefinitionTable
Dim dataDefField As VcDataDefinitionField

dataDefTable =
VcGantt1.DataDefinition.DataDefinitionTable(VcDataTableType.vcMaindata)
dataDefField = dataDefTable.DataDefinitionFieldByName("Start")
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefTable =
vcGantt1.DataDefinition.get_DataDefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefField =
dataDefTable.DataDefinitionFieldByName("Start");
dataDefField.Type = VcDataDefinitionFieldType.vcDefFieldDateTimeType;
vcGantt1.DataTableCollection.Update();
```

7.18 VcDataDefinitionTable



Ein Objekt vom Typ VcDataDefinitionTable ist ein Element der Datendefinition und stellt eine Tabelle aus Datendefinitionsfeldern dar. Auf diese Felder können Sie einzeln über die Methoden **DataDefinitionFieldByIndex** oder **DataDefinitionFieldByName** zugreifen oder über eine Schleife mit **FirstDataDefinitionField** und **NextDataDefinitionField** alle Felder abfragen. Über die Eigenschaft **Count** erhalten Sie die Anzahl der Felder. Die Datendefinitionstabelle wird auf der Eigenschaftenseite **Datentabellen verwalten** voreingestellt.

Eigenschaften

- Count

Methoden

- CreateDataDefinitionField
- DataDefinitionFieldByIndex
- DataDefinitionFieldByName
- FirstDataDefinitionField
- GetEnumerator
- NextDataDefinitionField

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcDataDefinitionTable

Mit dieser Eigenschaft kann die Anzahl der Felder in der Datendefinitionstabelle abgefragt werden. Anlegen können Sie Datenfelder im Dialog **Datentabellen verwalten** oder zur Laufzeit mit Hilfe der Methode **CreateDataDefinitionField**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl Felder

Code-Beispiel VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim numberOfFields As Integer

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable = dataDefinition.DefinitionTable(VcDataTableType.vcMaindata)

numberOfFields = dataDefinitionTable.Count
```

Code-Beispiel C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
dataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);

int numberOfFields = dataDefinitionTable.Count;
```

Methoden

CreateDataDefinitionField

Methode von VcDataDefinitionTable

Mit dieser Methode kann zur Laufzeit ein neues Datenfeld an das Ende der Datendefinitionstabelle angefügt werden. Das neue Datenfeld hat standardmäßig den Datentyp Integer; er kann aber mit Hilfe der Eigenschaft **Type** von **VcDataDefinitionField** geändert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ newfieldName	System.String	Name des neuen Feldes
Rückgabewert	VcDataDefinitionField	Datendefinitionsfeld

Code-Beispiel VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable = dataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
dataDefinitionTable.CreateDataDefinitionField("New data field 1")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataDefinition dataDefinition = VcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
dataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
dataDefinitionTable.CreateDataDefinitionField("New data field 1");
vcGantt1.DataTableCollection.Update();
```

DataDefinitionFieldByIndex

Methode von VcDataDefinitionTable

Mit dieser Methode können Sie über den Index auf ein beliebiges Feld der Datendefinitionstabelle zugreifen. Jedes Feld kann über seinen Namen oder über seinen Index angesprochen werden. Im Dialog **Datentabellen verwalten** können Sie die Datendefinitionen bearbeiten.

	Datentyp	**Beschreibung**
Parameter:		
⇨ fieldIndex	System.Int16	Index des Feldes
Rückgabewert	VcDataDefinitionField	Datendefinitionsfeld

Code-Beispiel VB.NET

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDataDefinitionField

dataDefinitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
definitionField = dataDefinitionTable.DataDefinitionFieldByIndex(2)
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefinitionTable =
VcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);VcDataDe
finitionField dataDefinitionField =
dataDefinitionTable.DataDefinitionFieldByIndex(2);
```

DataDefinitionFieldByName

Methode von VcDataDefinitionTable

Mit dieser Methode können Sie unter Verwendung des Feldnamens auf ein bestimmtes Feld der Datendefinitionstabelle zugreifen. Existiert kein Feld unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**). Jedes Feld kann über seinen Namen oder seinen Index angesprochen werden. Im Dialog **Datentabellen verwalten** können Sie die Datendefinitionen bearbeiten.

	Datentyp	Beschreibung
Parameter:		
⇨ fieldName	System.String	Feldname
Rückgabewert	VcDataDefinitionField	Datendefinitionsfeld

Code-Beispiel VB.NET

```
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDataDefinitionField

dataDefinitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
definitionField = dataDefinitionTable.DataDefinitionFieldByName("Start")
```

Code-Beispiel C#

```
VcDataDefinitionTable dataDefinitionTable =
VcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);VcDataDe
finitionField dataDefinitionField =
dataDefinitionTable.DataDefinitionFieldByName("Start");
```

FirstDataDefinitionField

Methode von VcDataDefinitionTable

Mit dieser Methode können Sie auf das erste Feld der Datendefinitionstabelle zugreifen, um anschließend in einer Schleife mit der Methode **NextDataDefinitionField** über die nachfolgenden Felder zu iterieren. Existiert kein Feld in der Datendefinitionstabelle, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataDefinitionField	Erstes Datendefinitionsfeld

Code-Beispiel VB.NET

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

Set dataDefinition = VcGantt1.DataDefinition
Set dataDefinitionTable = dataDefinition.DefinitionTable(vcMaindata)

Set dataDefinitionField = dataDefinitionTable.FirstDataDefinitionField
```

Code-Beispiel C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
vcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefinitionField =
dataDefinitionTable.FirstDataDefinitionField();
```

GetEnumerator

Methode von VcDataDefinitionTable

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 Felder des DataDefintionTable-Objektes iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Enumerator-Objekt

Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
For Each dataTable In dataTableCltn
   ListBox1.Items.Add(dataTable.Name)
Next
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.DataTableCollection;
foreach (VcDataTable dataTable in dataTableCltn)
   listBox1.Items.Add(dataTable.Name);
```

NextDataDefinitionField

Methode von VcDataDefinitionTable

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Felder der Datendefinitionstabelle zugreifen, nachdem Sie die Methode **FirstDataDefinitionField** aufgerufen haben.

Wenn kein weiteres Feld mehr existiert, wird ein Leerobjekt zurückgegeben (in Visual Basic: Nothing).

	Datentyp	Beschreibung
Rückgabewert	VcDataDefinitionField	Nachfolgendes Datendefinitionsfeld

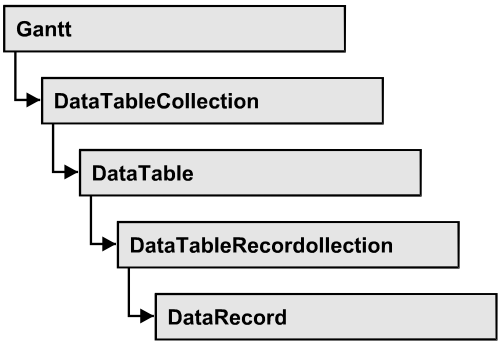
Code-Beispiel VB.NET

```
Dim dataDefinition As VcDataDefinition
Dim dataDefinitionTable As VcDataDefinitionTable
Dim definitionField As VcDataDefinitionField

dataDefinition = VcGantt1.DataDefinition
dataDefinitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
definitionField = dataDefinitionTable.FirstDataDefinitionField

While Not definitionField Is Nothing
   ListBox1.Items.Add(definitionField.Name)
   definitionField = dataDefinitionTable.NextField
End While
```

Code-Beispiel C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
VcDataDefinitionTable dataDefinitionTable =
vcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
VcDataDefinitionField dataDefinitionField =
dataDefinitionTable.FirstDataDefinitionField();

while (dataDefinitionField != null)
   {
   ListBox.Items.Add(dataDefinitionField.Name);
   dataDefinitionField = dataDefinitionTable. NextDataDefinitionField;
   }
```

7.19 VcDataRecord



Ein Datensatz ist das logische Grundelement eines Objektes in einem Gantt-Diagram, 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
- Update

Eigenschaften

AllData

Eigenschaft von VcDataRecord

Mit dieser Eigenschaft können alle Daten eines Datansatzes 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 = VcGantt1.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 =
vcGantt1.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**
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes
Eigenschaftswert	System.Object	Inhalt des Datenfeldes

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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**
Eigenschaftswert	System.String	Name der zughörigen Tabelle

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

MsgBox(dataRecord.DataTableName)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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 = VcGantt1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)
MsgBox(dataRecord.ID)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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 = VcGantt1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.Delete()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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**
Parameter:		
⇨ establishedObject Param	System.Object	Erkanntes Objekt
establishedObjectTypeParam	VcObjectType	Typ des erkannten Objekts
	Mögliche Werte:	
	.vcObjTypeBox 15	Objekttyp **Box**
	.vcObjTypeCalendarGrid 18	Objekttyp **Kalendergitter**
	.vcObjTypeCurve 12	Objekttyp **Kurve**
	.vcObjTypeDateLine 9	Objekttyp **Stichtaglinie**
	.vcObjTypeGroup 7	Objekttyp **Gruppe**
	.vcObjTypeGroupInDiagram 11	Objekttyp **Gruppe im Knotenbereich**
	.vcObjTypeGroupInTable 7	Objekttyp **Gruppe im Tabellenbereich**
	.vcObjTypeHistogram 13	Objekttyp **Histogramm**
	.vcObjTypeLayer 8	Objekttyp **Layer**
	.vcObjTypeLinkCollection 3	Objekttyp **LinkCollection**
	.vcObjTypeNodeInDiagram 2	Objekttyp **Knoten im Knotenbereich**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNodeInTable 1	Objekttyp **Knoten im Tabellenbereich**
	.vcObjTypeNone 0	kein Objekt
	.vcObjTypeNumericScale 10	Objekttyp **Werteskala**
	.vcObjTypeSummaryNode 14	Objekttyp **Summenbalken**
	.vcObjTypeTable 4	Objekttyp **Tabelle**
	.vcObjTypeTableCaption 5	Objekttyp **Tabellenüberschrift**
	.vcObjTypeTimeScale 6	Objekttyp **Zeitskala**

Rückgabewert	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**
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes
Rückgabewert	VcDataRecord	Zugeordneter Datensatz

Code-Beispiel VB.NET

```
   Private Sub VcGantt1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.VcNodeLeftClicking
      Dim dataTable As VcDataTable
      Dim dataRecordCltn As VcDataRecordCollection
      Dim firstDataRecord As VcDataRecord
      Dim secondDataRecord As VcDataRecord

      dataTable = VcGantt1.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 vcGantt1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   VcDataTable dataTable = vcGantt1.DataTableCollection.DataTableByIndex(0);
   VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
   VcDataRecord firstDataRecord =
dataRecordCltn.DataRecordByID(e.Node.get_DataField(0));
   VcDataRecord secondDataRecord = firstDataRecord.RelatedDataRecord(2);

   MessageBox.Show(secondDataRecord.AllData.ToString());
   }
```

Update

Methode von VcDataRecord

Nachdem Sie ein oder mehrere Datenfelder eines Datensatzes mit der Eigenschaft **DataField** verändert haben, aktualisieren Sie die grafische Darstellung im Diagramm mit **UpdateDataRecord**.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Datensatz erfolgreich (true) / nicht erfolgreich (false) aktualisiert

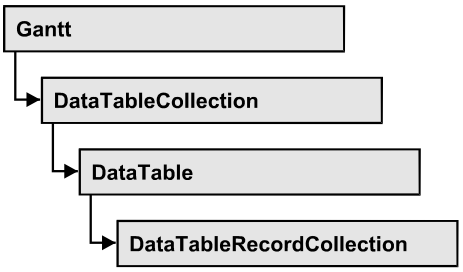
Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.FirstDataTable
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(1)

dataRecord.DataField(1) = "Node Two"
dataRecord.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.DataTableCollection.FirstDataTable();
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;
VcDataRecord dataRecord = dataRecordCltn.DataRecordByID(1);

dataRecord.set_DataField(1, "Node Two");
dataRecord.Update();
```

7.20 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 **NextData-Record** 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
- GetNewUniqueID
- NextDataRecord
- Remove
- Update

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcDataRecordCollection

Mit dieser Eigenschaft können Sie die Anzahl der Datensätze in der Data-Record-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Datensätze im Collection-Objekt

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
MsgBox("Number of DataRecords: " & dataRecordCltn.Count)
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.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 **VcPrimaryKeyNotUniqueException** erzeugt. Nach dem Anlegen des Datensatzes muss die Methode **VcGantt.EndLoading** aufgerufen werden, damit die Änderung wirksam wird.

	Datentyp	Beschreibung
Parameter:		
⇨ dataRecordContent	VcObject	Inhalt des Datensatzes (als Array oder String)
Rückgabewert	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 = VcGantt1.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)
VcGantt1.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 =
vcGantt1.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);
VcGantt1.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
Parameter:		
⇨ dataRecordID	System.String	ID des Datensatzes
Rückgabewert	VcDataRecord	Datensatzobjekt

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.DataRecordByID(0)
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.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 Datensatzauflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDataRecord	Erster Datensatz

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecord = dataRecordCltn.FirstDataRecord
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.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 = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcGantt1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.Update()
   dataRecord = dataRecordCltn.NextDataRecord
End While

VcGantt1.SuspendUpdate(False)
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcGantt1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
   dataRecord.set_DataField(4, "10");
   dataRecord.Update();
   dataRecordCltn.NextDataRecord();
}

vcGantt1.SuspendUpdate(false);
```

GetNewUniqueID

Methode von VcDataRecordCollection

Mit dieser Methode können Sie eine eindeutige ID für einen Datensatz generieren lassen. Die Methode ist nützlich, wenn Sie z.B. mit **Add** einen neuen Datensatz generieren und die ID nicht manuell vergeben.

	Datentyp	Beschreibung
Rückgabewert	System.Int32	Neue Datensatz-ID

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 = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection

VcGantt1.SuspendUpdate(True)

dataRecord = dataRecordCltn.FirstDataRecord
While Not dataRecord Is Nothing
   dataRecord.DataField(4) = "10"
   dataRecord.Update()
   dataRecord = dataRecordCltn.NextDataRecord
End While

VcGantt1.SuspendUpdate(False)
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecordCltn = dataTable.DataRecordCollection;

vcGantt1.SuspendUpdate(true);

foreach (VcDataRecord dataRecord in dataRecordCltn)
{
   dataRecord.set_DataField(4, "10");
   dataRecord.Update();
   dataRecordCltn.NextDataRecord();
}

vcGantt1.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**
Parameter:		
⇨ dataRecordContent	VcObject	Inhalt des Datensatzes (als Array oder String)
Rückgabewert	System.Boolean	true

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Remove("1;1Activity; Y;Z;18.01.14;;5")
VcGantt1.EndLoading()

' equivalent
' dataRecord = dataRecordCltn.DataRecordByID(1)
' dataRecord.Delete()
' dataRecord.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

dataRecCltn .Remove("1;1Activity Y;Z;18.01.14;;5");
VcGantt1.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 **VcGantt.EndLoading** aufgerufen werden, damit die Änderung wirksam wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ dataRecordContent	VcObject	Inhalt des Datensatzes (als Array oder String)
Rückgabewert	System.Boolean	Aktualisierung erfolgt (true) / nicht erfolgt (false)

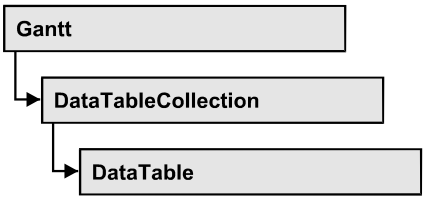
Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecordCltn As VcDataRecordCollection
Dim dataRecord As VcDataRecord

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecordCltn = dataTable.DataRecordCollection
dataRecordCltn.Update("1;1.8.2017;;8")
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;
dataRecCltn.Update("1;1.8.2017;;8");
VcGantt1.EndLoading();
```

7.21 VcDataTable



Eine Datentabelle umfasst **Datensätze** (data records) mit ihren Datenfeldern und ihren Inhalten sowie die Beschreibungen der Datenfelder, die **Tabellen-datenfelder** (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 = VcGantt1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataRecordCollection.Count)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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 **DataTableField-Collection** 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 = VcGantt1.DataTableCollection.DataTableByIndex(0)
MsgBox(dataTable.DataTableFieldCollection.Count)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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
Eigenschaftswert	System.String	Beschreibung der Datentabelle **Standardwert:** Empty string

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataTable.Description = "This table contains data for nodes"
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.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
Eigenschaftswert	System.Boolean	Verwendung von zusammengesetzten Primärschlüsseln erlaubt (true)/nicht erlaubt (false) **Standardwert:** 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
Eigenschaftswert	System.String	Name der Datentabelle **Standardwert:** Empty string

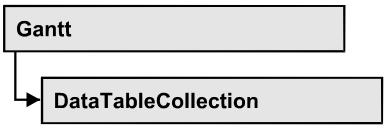
Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcGantt1.DataTableCollection.DataTableByIndex(0)
MsgBox(dataTable.Name)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.DataTableCollection.DataTableByIndex(0);
MessageBox.Show(dataTable.Name);
```

7.22 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 XGantt-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
Eigenschaftswert	System.Int32	Anzahl der Datentabellen im Collection-Objekt

Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection

dataTableCltn = VcGantt1.DataTableCollection
MsgBox(dataTableCltn.Count.ToString())
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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 Tabellenname 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
Parameter:		
⇨ dataTableName	System.String	Name der neuen Tabelle
Rückgabewert	VcDataTable	Neu angelegte Datentabelle

Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTableCltn.Update()
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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
Parameter:		
⇨ dataTableName	System.String	Name der zu kopierenden Datentabelle (Quelltabelle)
⇨ newDataTableName	System.String	Name der neu zu erstellenden Datentabelle (Zieltabelle)
Rückgabewert	VcDataTable	Neu erstelltes Datentabellen-Objekt

Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
dataTable = dataTableCltn.Copy("Resources", "NewResources")
dataTableCltn.Update()
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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
Parameter:		
⇨ index	System.Int16	Index der Datentabelle

Rückgabewert	VcDataTable	Ermitteltes Datentabellenobjekt

Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
dataTable = dataTableCltn.DataTableByIndex(2)
MsgBox(dataTable.Name)
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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**
Parameter:		
⇨ dataTableName	System.String	Name der Datentabelle
Rückgabewert	VcDataTable	Ermitteltes Datentabellenobjekt

Code-Beispiel VB.NET

```
Dim dataTablecltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTablecltn = VcGantt1.DataTableCollection
dataTable = dataTablecltn.DataTableByName("Resources")
MsgBox(dataTable.Description)
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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 = VcGantt1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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 = VcGantt1.DataTableCollection
For Each dataTable In dataTableCltn
   ListBox1.Items.Add(dataTable.Name)
Next
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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 = VcGantt1.DataTableCollection
dataTable = dataTableCltn.FirstDataTable
For i = 1 To dataTableCltn.Count
   ListBox1.Items.Add(dataTable.Name)
   dataTable = dataTableCltn.NextDataTable
Next
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.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 vermeiden.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (true) / nicht erfolgt (false)

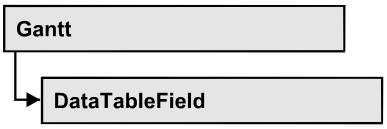
Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
dataTable = dataTableCltn.Add("Resources")
dataTable.DataTableFieldCollection.Add("Id")
dataTableCltn.Update()
```

Code-Beispiel C#

```
VcDataTableCollection dataTableCltn = vcGantt1.DataTableCollection;
VcDataTable dataTable = dataTableCltn.Add("Resources");
dataTable.DataTableFieldCollection.Add("Id");
dataTableCltn.Update();
```

7.23 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 = VcGantt1.DataTableCollection.FirstDataTable
MsgBox(dataTable.DataTableFieldCollection.FirstDataTableField.DataTableName)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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
Eigenschaftswert	System.String	Datumsformat {DMYhms:;./}

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
'DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY"
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
//DateFormat = "01.12.2014"
dataTableField.DateFormat = "DD.MM.YYYY";
vcGantt1.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
Eigenschaftswert	System.Boolean	Feld editierbar (True) / nicht editierbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Editable = False
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Editable = false;
VcGantt1.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
Eigenschaftswert	System.Boolean	Feld wird nicht angezeigt (True) / angezeigt (False) **Standardwert:** False

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Hidden = True
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Hidden = true;
vcGantt1.DataTableCollection.Update();
```

Index

Nur-Lese-Eigenschaft von VcDataTableField

Mit dieser Eigenschaft können Sie den Index des Datentabellenfelds in der zugehörigen Datentabelle erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	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
Eigenschaftswert	System.String	Name des Feldes **Standardwert:** Empty string

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.Add("Start")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Start");
vcGantt1.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
Eigenschaftswert	System.Boolean	Das Feld dient (True) / dient nicht (False) als Primärschlüssel **Standardwert:** False

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField
Dim isPrimaryKey As Boolean

dataTable = VcGantt1.DataTableCollection.DataTableByName("Operation")
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Id")
dataTableField.PrimaryKey = True
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Id");
dataTableField.PrimaryKey = true;
vcGantt1.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
Eigenschaftswert	System.Int32	Index des Datensatzfeldes, auf das sich die Datendefinition des Datentabellenfeldes bezieht. **Standardwert:** -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 = VcGantt1.DataTableCollection.Add("Task")
dataTaskFieldId = dataTableTask.DataTableFieldCollection.Add("Id")
dataTaskFieldId.PrimaryKey = True
dataTaskFieldName = dataTableTask.DataTableFieldCollection.Add("Name")
dataTaskFieldName.Type = VcDataTableFieldType.vcDataTableFieldStringType

'Create table Operation
dataTableOperation = VcGantt1.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 =
VcGantt1.DetectFieldIndex("Task", "Id")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
//Create table Task
VcDataTable dataTableTask = vcGantt1.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 = vcGantt1.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 =
vcGantt1.DetectFieldIndex("Task","Id");
vcGantt1.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
Eigenschaftswert	VcDataTableFieldType	Datentyp des Feldes, kann maximal 512 Zeichen enthalten **Standardwert:** vcDataTableFieldIntegerType
	Mögliche Werte:	
	.vcDataFieldAlphanumericType 1	Datentyp **Alphanumerisch**
	.vcDataFieldDateTimeType 3	Datentyp **Datum**
	.vcDataTableFieldIntegerType 2	Datentyp **Integer** (32 Bit)

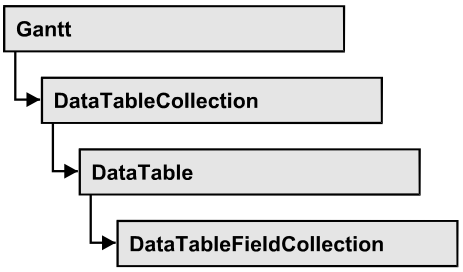
Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

VcGantt1.DataTableCollection.DataTableByName("Operation")
dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start")
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Operation");
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Start");
dataTableField.Type = VcDataTableFieldType.vcDataTableFieldDateTimeType;
vcGantt1.DataTableCollection.Update();
```

7.24 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 **FirstDataField** und **NextDataField** 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
Eigenschaftswert	System.Int32	Anzahl der Datentabellenfelder im Collection-Objekt

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable

dataTable = VcGantt1.DataTableCollection.FirstDataTable()
MsgBox(dataTable.DataTableFieldCollection.Count.ToString())
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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
Parameter:		
⇨ dataTableFieldName	System.String	Name des neuen Datentabellenfeldes
Rückgabewert	VcDataTableField	Neu angelegtes Datentabellenfeld

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Add("Priority")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Add("Priority");
vcGantt1.DataTableCollection.Update();
```

Copy

Methode von VcDataTableFieldCollection

Mit dieser Methode können Sie ein Datentabellenfeld kopieren. Die Identifizierung des Feldes erfolgt über den Namen.

	Datentyp	Beschreibung
Parameter:		
⇨ dataTableFieldName	System.String	Name des zu kopierenden Datentabellenfeldes (Quellfeld)
⇨ newDataTableFieldName	System.String	Name des neu zu erstellenden Datentabellenfeldes (Zielfeld)
Rückgabewert	VcDataTableField	Neu angelegtes Datentabellenfeld

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.Copy("Name", "NewName")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.Copy("Name", "NewName");
vcGantt1.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
Parameter:		
⇨ index	System.Int16	Index des Datentabellenfeldes
Rückgabewert	VcDataTableField	Ermitteltes Datentabellenfeld

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByIndex(1)
MsgBox(dataTableField.Name)
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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**
Parameter:		
⇨ dataTableFieldName	System.String	Name des Datentabellenfeldes
Rückgabewert	VcDataTableField	Ermitteltes Datentabellenfeld

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableField As VcDataTableField

dataTable = VcGantt1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.DataTableFieldByName("Name")
dataTableField.Editable = False
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.DataTableCollection.FirstDataTable();
VcDataTableField dataTableField =
dataTable.DataTableFieldCollection.DataTableFieldByName("Name");
dataTableField.Editable = false;
vcGantt1.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 = VcGantt1.DataTableCollection.FirstDataTable()
dataTableField = dataTable.DataTableFieldCollection.FirstDataTableField()
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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 = VcGantt1.DataTableCollection.FirstDataTable()
For Each dataTableField In dataTable.DataTableFieldCollection
   ListBox1.Items.Add(dataTableField.Name)
Next
```

Code-Beispiel C#

```
VcDataTable dataTable = vcGantt1.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

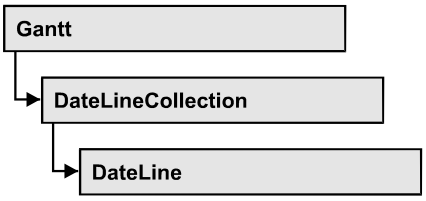
Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataTableFieldCltn As VcDataTableFieldCollection
Dim dataTableField As VcDataTableField
Dim i As Integer

dataTable = VcGantt1.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 = vcGantt1.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();
   }
```

7.25 VcDateLine



Ein Objekt vom Typ VcDateLine stellt eine Terminstichwertlinie dar. Eine Terminstichwertlinie ist eine vertikale Stichwertlinie im Balkendiagramm zur Kennzeichnung eines Termins.

Eigenschaften

- AlwaysCurrentDate
- Date
- DateDataFieldIndex
- Font
- FontColor
- Identifiable
- LabelPosition
- LineColor
- LineThickness
- LineType
- Movable
- Name
- Priority
- SnapTarget
- Specification
- Text
- TurningAnnotationEnabled
- UpdateBehaviorName
- Visible
- VisibleDataFieldIndex
- VisibleMapName

Methoden

- PutInOrderAfter

Eigenschaften

AlwaysCurrentDate

Nur-Lese-Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie erfragen, ob die Stichtaglinie immer das aktuelle Datum (mit Uhrzeit) zur Zeit des Aufrufs des VARCHART Steuerelements anzeigt. Diese Eigenschaft können Sie im Dialog **Stichtaglinien festlegen** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft/nicht in Kraft **Standardwert:** False

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine
Dim dateLineTimer As Timer

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
If dateLine.AlwaysAtCurrentDate = True Then
   dateLineTimer.Enabled = True
End If
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
Timer dateLineTimer;

if (dateLine.AlwaysAtCurrentDate)
   dateLineTimer.Enabled = true;
```

Date

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Position der Stichtaglinie setzen oder erfragen. Bitte beachten Sie, dass zwischen Datum und Uhrzeit ein Leerzeichen stehen muss. Sie können diese Eigenschaft auch im Dialog **Stichtaglinien festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Datum {1.1.1970...31.12.2035} **Standardwert:** none or current date

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Date = "30.09.14 12:00:00"
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Date = Convert.ToDateTime("30.09.14 12:00:00");
```

DateDataFieldIndex

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können sie den Index des Datenfeldes setzen oder erfragen, das das Datum für eine individuelle Stichtaglinie enthält.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das Datum enthält

Font

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Schriftattribute einer Stichtaglinie erfragen oder festlegen. Diese Eigenschaft können Sie auch im Dialog **Stichtaglinien bearbeiten** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftattribute der Stichtaglinientexte

FontColor

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Schriftfarbe einer Stichtaglinie erfragen oder festlegen. Mit dieser Eigenschaft können Sie die Schriftattribute einer Stichtaglinie erfragen oder festlegen. Diese Eigenschaft können Sie auch im Dialog **Stichtaglinien bearbeiten** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte

Identifiable

Nur-Lese-Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Stichtaglinie identifizierbar ist. Wenn die Eigenschaft auf **True** gesetzt wurde, kann die Stichtaglinie mittels der VcGantt-Methode **IdentifyObject-At** identifiziert werden.

Diese Eigenschaft kann auch im Dialog **Stichtaglinien festlegen** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Stichtaglinie identifizierbar / nicht identifizierbar **Standardwert:** False

LabelPosition

Nur-Lese-Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie festlegen oder erfragen, an welcher Position ggf. ein Text ausgegeben werden soll. Diese Eigenschaft können Sie auch im Dialog **Stichtaglinien bearbeiten** einstellen.

	Datentyp	Beschreibung

LineColor

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Farbe einer Stichtaglinie erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Stichtaglinie bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** 255. Visual Basic: RGB (255, 0, 0)

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineColor = Color.Blue
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.LineColor = Color.LightSteelBlue;
```

LineThickness

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Linienstärke einer Stichtaglinie 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 Dialog **Stichtaglinie bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the dialog

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineType = VcLineType.vcSolid
dateLine.LineThickness = 3
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.LineType = VcLineType.vcSolid;
dateLine.LineThickness = 3;
```

LineType

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie den Linientyp einer Stichtaglinie erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Stichtaglinie bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLineType	Linientyp **Standardwert:** vcSolid
	Mögliche Werte:	
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.LineType = VcLineType.vcSolid
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.LineType = VcLineType.vcSolid;
```

Movable

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Stichtaglinie interaktiv verschiebbar sein soll. Sie können diese Eigenschaft auch im Dialog **Stichtaglinien festlegen** setzen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Boolean	Verschiebbar (True)/ nicht verschiebbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateLine.Movable = False
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Movable = false;
```

Name

Nur-Lese-Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie den Namen der Stichtaglinie erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Terminstichlinie

Code-Beispiel VB.NET

```
Dim datelineCltn As VcDateLineCollection
Dim dateline As VcDateLine

datelineCltn = VcGantt1.DateLineCollection
For Each dateline In datelineCltn
   ListBox1.Items.Add(dateline.Name)
Next
```

Code-Beispiel C#

```
VcDateLineCollection datelineCltn = vcGantt1.DateLineCollection;

foreach (VcDateLine dateline in datelineCltn)
   {
   ListBox.Items.Add(dateline.Name);
   }
```

Priority

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Priorität einer Stichtaglinie setzen oder erfragen. Wenn zwei Objekte dieselbe Position im Diagramm haben, liegt das Objekt mit der höheren Priorität über den Objekten mit einer niedrigeren Priorität. Gitter haben standardmäßig die niedrigste Priorität; Knoten besitzen mit dem Wert 0 die höchste Priorität von allen Objekten. Stichwertlinien liegen standardmäßig vor Kalender- und Terminliniengittern, aber hinter den Knoten. Soll eine Stichwertlinie vor den Knoten liegen, müssen Sie ihre Priorität auf einen positiven Wert setzen. Sie können diese Eigenschaft auch im Dialog **Stichtaglinien festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Prioritätsstufe **Standardwert:** 0

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateline.Priority = 10
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Priority = 10;
```

SnapTarget

Nur-Lese-Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob für diese Stichtaglinie ein Einrastziel am Datum definiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Einrastziel wird/wird nicht am Datum der Stichtaglinie definiert.

Specification

Nur-Lese-Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Spezifikation dieser Stichtaglinie 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 einer Stichtaglinie mit der Methode **VcDateLineCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation der Stichtaglinie

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.FirstDateLine
MsgBox(dateLine.Specification)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.FirstDateLine();
MessageBox.Show(dateLine.Specification);
```

Text

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie einen Erläuterungstext zu der Stichtaglinie setzen oder erfragen. Sie können diese Eigenschaft auch im Dialog **Stichtaglinien festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Beschriftungstext

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateline.Text = "Stichtag"
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Text = "Stichtag";
```

TurningAnnotationEnabled

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie einstellen oder erfragen ob die Beschriftung der Stichtaglinie um 90 Grad gedreht wird. Diese Eigenschaft kann auch im Dialog **Stichtaglinie bearbeiten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Beschriftung der Stichtaglinie ist/ist nicht um 90 Grad gedreht

UpdateBehaviorName

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

Visible

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie die Sichtbarkeit einer Stichtaglinie setzen oder erfragen. Sie können diese Eigenschaft auch im Dialog **Stichtaglinien festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Stichtaglinie sichtbar/unsichtbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim dateLine As VcDateLine

dateLine = VcGantt1.DateLineCollection.DateLineByName("DateLine1")
dateline.Visible = False
```

Code-Beispiel C#

```
VcDateLine dateLine = vcGantt1.DateLineCollection.DateLineByName("DateLine1");
dateLine.Visible = false;
```

VisibleDataFieldIndex

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können sie den Index des Datenfeldes für die individuelle Stichtaglinie setzen oder erfragen, das den Wert "1" (für "sichtbar") oder 0 (für "unsichtbar") enthält. Sie können diese Eigenschaft auch im Dialog **Stichtaglinien festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes, das den Sichtbarkeitsmodus enthält

VisibleMapName

Eigenschaft von VcDateLine

Mit dieser Eigenschaft können Sie den Namen einer Zuordnungstabelle (Typ vcTextMap) für den Sichtbarkeitsmodus setzen oder erfragen. Wird hier "" angegeben, wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **VisibleDataFieldIndex** angegeben ist, wird der Sichtbarkeitsstatus aus der Zuordnungstabelle ausgewählt. Diese Eigenschaft kann auch im Dialog **Stichtaglinien festlegen** festgelegt werden. Trifft kein Datenfeldeintrag in der Zuordnungstabelle zu, wird der Wert aus dem Dialog verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle für den Sichtbarkeitsmodus

Methoden

PutInOrderAfter

Methode von VcDateLine

Mit dieser Methode können Sie die Terminline in der Auflistung aller Terminlinien hinter den durch den Namen angegebenen setzen. Wenn als Name "" übergeben wird, wird die Terminlinie an die erste Stelle gesetzt. Die Reihenfolge der Terminlinien in der Auflistung entscheidet darüber, in welcher Reihenfolge sie dargestellt werden.

	Datentyp	Beschreibung
Parameter:		
⇦ refName	System.String	Name der Stichtaglinie, hinter den die aktuelle Stichtaglinie eingeordnet werden soll.
Rückgabewert	Void	

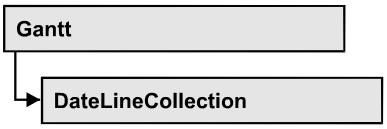
Code-Beispiel VB.NET

```
Dim datLinCltn As VcDateLineCollection
Dim datLin1 As VcDateLine
Dim datLin2 As VcDateLine

datLinCltn = VcGantt1.DateLineCollection()
datLin1 = datLinCltn.Add("datLin1")
datLin2 = datLinCltn.Add("datLin2")
datLin1.PutInOrderAfter("datLin2")
datLinCltn.Update()
```

Code-Beispiel C#

```
VcDateLineCollection datLinCltn = vcGantt1.DateLineCollection;
VcDateLine datLin1 = datLinCltn.Add("datLin1");
VcDateLine datLin2 = datLinCltn.Add("datLin2");
datLin1.PutInOrderAfter("datLin2");
datLinCltn.Update();
```

7.26 VcDateLineCollection



In einem Objekt vom Typ **VcDateLineCollection** sind alle verfügbaren Terminlinien automatisch enthalten. Über **For Each dateLine In DateLine-Collection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Terminlinien zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **DateLineByName** und **DateLineFormat-ByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Terminlinien kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add...**, **Copy...** und **Remove...** ermöglichen das Hinzufügen, Kopieren und Löschen von Terminlinien.

Eigenschaften

- Count

Methoden

- Add
- AddBySpecification
- Copy
- DateLineByIndex
- DateLineByName
- FirstDateLine
- GetEnumerator
- NextDateLine
- Remove
- Update

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcDateLineCollection

Mit dieser Eigenschaft können Sie die Anzahl von Stichtaglinien in der DateLine-Auflistung erfragen.

	Datentyp	Beschreibung
Parameter:		
⇨ Rückgabewert	System.Int32	Anzahl der Stichtaglinien
Eigenschaftswert	System.Int32	Anzahl der Stichtaglinien

Code-Beispiel VB.NET

```
Dim numberOfDateLine As Integer

numberOfDateLine = VcGantt1.DateLineCollection.Count
```

Code-Beispiel C#

```
int numberOfDateLines = vcGantt1.DateLineCollection.Count;
```

Methoden

Add

Methode von VcDateLineCollection

Mit dieser Methode können Sie eine neue Stichtaglinie in der DateLineCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Stichtaglinienobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Um die neu angelegte Stichtaglinie im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ dateLineName	System.String	Name der Stichtaglinie
Rückgabewert	VcDateLine	Neues Stichtaglinienobjekt

Code-Beispiel VB.NET

```
newDateLine = VcGantt1.DateLineCollection.Add("dateLine1")
```

Code-Beispiel C#

```
newDateLine = vcGantt1.DateLineCollection.Add("dateLine1");
```

AddBySpecification

Methode von VcDateLineCollection

Mit dieser Methode können Sie eine Stichtaglinie über eine Stichtaglinien-Spezifikation erzeugen. Dies dient der Persistenz von Stichtaglinien-Objekten. Die Spezifikation einer Stichtaglinie kann erfragt (siehe VcDateLine-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann die gleiche Stichtaglinie mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden. Um die neu angelegte Stichtaglinie im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ specification	System.String	Stichtaglinien-Spezifikation
Rückgabewert	VcDateLine	Neues Stichtaglinienobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection

dateLineCltn = VcGantt1.DateLineCollection
dateLineCltn.AddBySpecification(textSpecification)
dateLineCltn.Update()
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
dateLineCltn.AddBySpecification(textSpecification);
dateLineCltn.Update();
```

Copy

Methode von VcDateLineCollection

Mit dieser Methode können Sie eine Stichtaglinie kopieren. Wenn die Stichtaglinie mit dem angegebenen Namen existiert und der Name der neuen Stichtaglinie noch nicht verwendet wird, wird das neue Stichtaglinienobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB). Um die kopierte Stichtaglinie im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ dateLineName	System.String	Name der zu kopierenden Stichtaglinie
⇨ newDateLineName	System.String	Name der neuen Stichtaglinie

Rückgabewert	VcDateLine	Stichtaglinienobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection

dateLineCltn = VcGantt1.DateLineCollection
dateLineCltn.Copy("DateLineOne", "NewDateLine")
dateLineCltn.Update()
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
dateLineCltn.Copy("DateLineOne", "NewDateLine");
dateLineCltn.Update();
```

DateLineByIndex

Methode von VcDateLineCollection

Mit dieser Methode können Sie auf eine einzelne Terminlinie über ihren Index zugreifen. Existiert keine Terminlinie an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	**Beschreibung**
Parameter:		
⇨ index	System.Int16	Index der Stichtaglinie
Rückgabewert	VcDateLine	Ermitteltes Stichtaglinienobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByIndex(0)
MsgBox(dateLine.Name)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByIndex(0);
MessageBox.Show(dateLine.Name);
```

DateLineByName

Methode von VcDateLineCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf eine bestimmte Stichtaglinie zugreifen. Existiert keine Stichtaglinie unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ dateLineName	System.String	Name der Stichtaglinie
Rückgabewert	VcDateLine	Stichtaglinie

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByName("DateLineOne")
MsgBox(dateLine.Name)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByName("DateLineOne");
MessageBox.Show(dateLine.Name);
```

FirstDateLine

Methode von VcDateLineCollection

Mit dieser Methode können Sie auf den Initialwert, d. h. die erste Stichtaglinie der Dateline-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextDateLine** über die nachfolgenden Stichtaglinien zu iterieren. Existiert keine Stichtaglinie in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDateLine	Erste Stichtaglinie

Code-Beispiel VB.NET

```
Dim datelineCltn As VcDateLineCollection
Dim dateline As VcDateLine

datelineCltn = VcGantt1.DateLineCollection
dateline = datelinecltn.FirstDateLine

While Not dateline Is Nothing
   ListBox1.Items.Add(dateline.Name)
   dateline = datelineCltn.NextDateLine
End While
```

Code-Beispiel C#

```
VcDateLineCollection datelineCltn = vcGantt1.DateLineCollection;
VcDateLine dateline = datelineCltn.FirstDateLine();

while (dateline != null)
    {
   ListBox.Items.Add(dateline.Name);
   dateline = datelineCltn.NextDateLine();
   }
```

GetEnumerator

Methode von VcDateLineCollection

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 Terminlinien-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
For Each dateLine In dateLineCltn
   ListBox1.Items.Add(dateLine.Name)
Next
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
foreach (VcDateLine dateLine in dateLineCltn)
   listBox1.Items.Add(dateLine.Name);
```

NextDateLine

Methode von VcDateLineCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Stichtaglinien des VcDateLineCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstDateLine** den Initialwert erfasst haben. Sind alle Stichtaglinien durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDateLine	Nachfolgende Stichtaglinie

Code-Beispiel VB.NET

```
Dim datelineCltn As VcDateLineCollection
Dim dateline As VcDateLine

datelineCltn = VcGantt1.DateLineCollection
dateline = datelinecltn.FirstDateLine

While Not dateline Is Nothing
   ListBox1.Items.Add(dateline.Name)
   dateline = datelineCltn.NextDateLine
End While
```

Code-Beispiel C#

```
VcDateLineCollection datelineCltn = vcGantt1.DateLineCollection;
VcDateLine dateline = datelineCltn.FirstDateLine();

while (dateline != null)
   {
   ListBox.Items.Add(dateline.Name);
   dateline = datelineCltn.NextDateLine();
   }
```

Remove

Methode von VcDateLineCollection

Mit dieser Methode können Sie eine Stichtaglinie löschen. Um das Löschen der Stichtaglinie im Diagramm sichtbar werden zu lassen, muss die Auflistung (Collection) mit **Update** aktualisiert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ dateLineName	System.String	Name der Stichtaglinie
Rückgabewert	System.Boolean	Stichtaglinie gelöscht (True)/nicht gelöscht (False)

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByIndex(0)
dateLineCltn.Remove(dateLine.Name)
dateLineCltn.Update()
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByIndex(0);
dateLineCltn.Remove(dateLine.Name);
dateLineCltn.Update();
```

Update

Methode von VcDateLineCollection

Mit dieser Methode können Sie eine DateLineCollection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

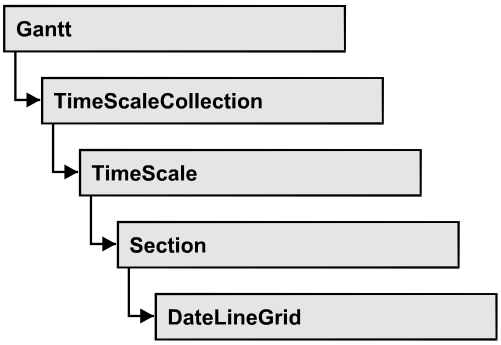
Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByIndex(0)
dateLineCltn.Remove(dateLine.Name)
dateLineCltn.Update()
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByIndex(0);
dateLineCltn.Remove(dateLine.Name);
dateLineCltn.Update();
```

7.27 VcDateLineGrid



Ein Objekt vom Typ **VcDateLineGrid** ist ein vordefiniertes Liniengitter, in dem Zeitabschnitte (Tage, Wochen, Monate, etc.) durch vertikale Linien markiert werden.

Eigenschaften

- AdjustToReferenceDate
- AnnotationAtBottom
- AnnotationAtCenter
- AnnotationAtTop
- FormatName
- HorAlignment
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- LineThickness
- LineType
- ObserveDST
- Period
- Priority
- ReferenceDate
- SnapTarget
- TurningAnnotationEnabled
- Unit
- UseReferenceDate
- Visible
- VisibleDataFieldIndex
- VisibleMapName

Eigenschaften

AdjustToReferenceDate

Eigenschaft von VcDateLineGrid

Die Linien eines Liniengitters werden standardmäßig am Anfang der jeweiligen Zeiteinheit positioniert, z.B. immer um 00:00 Uhr jedes Tages. Mit dieser Eigenschaft können die Linien des Liniengitters am Referenzdatum ausgerichtet werden, d.h. die Position kann vom Anfang einer Zeiteinheit abweichen, z.B. um 13:17 jedes Tages. Das Referenzdatum wird gesetzt über die Eigenschaft **set/getReferenceDate**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Liniengitter auf Referenzdatum positioniert (True) / nicht positioniert (False) **Standardwert:** False

AnnotationAtBottom

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Beschriftung der Linien im Liniengitter nach unten setzen oder erfragen, ob die Beschriftung am unteren Rand des Gantt Graphen ausgerichtet ist. S. auch **set/getAnnotationAtCenter** und **set/getAnnotationAtTop**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Liniengitterbeschriftung unten (True) / nicht unten (False) positioniert **Standardwert:** False

AnnotationAtCenter

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Beschriftung der Linien im Liniengitter in die Mitte des Gantt Graphen setzen oder erfragen, ob die Beschriftung mittig ausgerichtet ist. S. auch **set/getAnnotationAtBottom** und **set/getAnnotationAtTop**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Liniengitterbeschriftung mittig (True) / nicht mittig (False) positioniert **Standardwert:** False

AnnotationAtTop

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Beschriftung der Linien im Liniengitter nach oben setzen oder erfragen, ob die Beschriftung am oberen Rand des Gantt Graphen ausgerichtet ist. S. auch **set/getAnnotationAtCenter** und **set/getAnnotationAtBottom**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Liniengitterbeschriftung oben (True) / nicht oben (False) positioniert **Standardwert:** False

FormatName

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie den Namen des Linienformats dieses Terminliniengitters erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Linienformats

HorAlignment

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft wird für die Beschriftungen der Linien die horizontale Ausrichtung festgelegt oder erfragt.

	Datentyp	Beschreibung
Eigenschaftswert	VcHorizontalAlignment **Mögliche Werte:** .vcHorCenterAligned - 1 .vcLeftAligned -3	Horizontale Ausrichtung horizontal mittig linksbündig

.vcRightAligned -2	rechtsbündig

LineColor

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Farbe eines Terminliniengitters erfragen oder festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** 255. Visual Basic: RGB (255, 0, 0)

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.LineColor = Color.Blue
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.LineColor = Color.LightSteelBlue;
```

LineColorDataFieldIndex

Nur-Lese-Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Farbzuordnungstabelle in der Eigenschaft **LineColorMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int32	Datenfeldindex

LineColorMapName

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle 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 Farbzuordnungstabelle

LineThickness

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Linienstärke eines Terminliniengitters 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 Dialog **Stichtaglinie bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the dialog

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.LineThickness = 2
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.LineThickness = 2;
```

LineType

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie den Linientyp eines Terminliniengitters erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLineType	Linientyp **Standardwert:** vcDashed
	Mögliche Werte:	
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
datelinegrid.LineType = VcLineType.vcSolid
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.LineType = VcLineType.vcSolid;
```

ObserveDST

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob für dieses Liniengitter die Sommerzeit in der Darstellung berücksichtigt wird oder nicht.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcDateLineGridObserveDST **Mögliche Werte:** .vcGODDefault 9999 .vcGODNo 0 .vcGODYes 1	Sommerzeit wird/wird nicht berücksichtigt. Standardeinstellung aus der .INI-Datei wird verwendet Sommerzeit wird nicht berücksichtigt Sommerzeit wird berücksichtigt

Period

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Zeiteinheiten (Unit) jeweils eine Linie des Terminliniengitters gezogen werden soll. Der Abstand zwischen den Rasterlinien ist gleich dem Produkt aus der Zeiteinheit des Gitters (Unit) und der Periode (Period).

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Periodenwert **Standardwert:** 1

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay
dateLineGrid.Period = 1
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay;
dateLineGrid.Period = 1;
```

Priority

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Priorität eines Terminliniengitters erfragen oder festlegen. Wenn zwei Objekte dieselbe Position im Diagramm haben, liegt das Objekt mit der höheren Priorität über den Objekten mit einer niedrigeren Priorität. Gitter haben standardmäßig die niedrigste Priorität; Knoten besitzen mit dem Wert 0 die höchste Priorität von allen Objekten. Liniengitter liegen standardmäßig vor Kalenderliniengittern, aber hinter Stichtaglinien und Knoten. Soll ein Liniengitter vor den Knoten liegen, müssen Sie seine Priorität auf einen positiven Wert setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Prioritätsstufe {-1000...+1000} **Standardwert:** -20

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
datelinegrid.Priority = 10
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Period = 10;
```

ReferenceDate

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie das Referenzdatum des Liniengitters erfragen oder festlegen. Für die tatsächliche Verwendung dieses Datums muss die Eigenschaft **UseReferenceDate** gesetzt werden. Um das Liniengitter an diesem Datum auszurichten, s. Eigenschaft **Adjust-ToReferenceDate**. Das Referenzdatum versetzt den Anfang des Gitters um den angegebenen Abstand zu dem standardmäßigen Beginn am Montag 00.00 Uhr. Dabei spielt das absolute Datum keine Rolle, sondern nur die Differenz zum Standardbeginn. Beispiel: Soll das Gitter an einem Dienstag beginnen, setzen Sie das Referenzdatum z.B. auf den 6.5.2014. Das gleiche Ergebnis können Sie erzielen, wenn Sie das Referenzdatum eine Woche früher setzen, also auf den 29.4.2014. Ausschlaggebend ist die Differenz zu Montag, sie entspricht in dem Beispiel jeweils 1 Tag. Der Abstand muss nicht in Tagen angegeben werden, auch Uhrzeiten sind möglich, z.B. 29.4.2014 8:15 Uhr. Wenn ein Stundengitter vorliegt, sind nur die Minuten relevant und bei letzterem Beispiel wird das Gitter um 15 Minuten versetzt.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	

SnapTarget

Nur-Lese-Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob für dieses Liniengitter ein Einrastziel am Datum definiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Einrastziel wird/wird nicht am Datum des Liniengitters definiert.

TurningAnnotationEnabled

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob die Beschriftungen an den Linien des Liniengitters um 90 Grad (vertikal) gedreht werden können

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Die Beschriftungen können gedreht werden (True) / wurden bereits gedreht (False) **Standardwert:** True

Unit

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie die Zeiteinheit des Liniengitters erfragen oder festlegen. Der Abstand zwischen den einzelnen Rasterlinien ist gleich dem Produkt aus der Zeiteinheit des Liniengitters (Unit) und der Periode (Period).

	Datentyp	Beschreibung
Eigenschaftswert	VcGridUnit	Zeiteinheit **Standardwert:** vcGridUnitWeek
	Mögliche Werte:	
	.vcGridUnitDay 5	Gittereinheit **Tag**
	.vcGridUnitHour 6	Gittereinheit **Stunde**
	.vcGridUnitMinute 7	Gittereinheit **Minute**
	.vcGridUnitMonth 3	Gittereinheit **Monat**
	.vcGridUnitQuarter 2	Gittereinheit **Quartal**
	.vcGridUnitSecond 8	Gittereinheit **Sekunde**
	.vcGridUnitWeek 4	Gittereinheit **Woche**
	.vcGridUnitYear 1	Gittereinheit **Jahr**

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Period = 1;
dateLineGrid.Unit = VcGridUnit.vcGridUnitDay;
```

UseReferenceDate

Nur-Lese-Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Liniengitter ein Referenzdatum verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Liniengitter verwendet (True)/verwendet kein (False) Referenzdatum **Standardwert:** False

Visible

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Liniengitter sichtbar sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Liniengitter sichtbar/unsichtbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim dateLineGrid As VcDateLineGrid

dateLineGrid = VcGantt1.TimeScaleCollection.Active.Section(0).DateLineGrid(0)
dateLineGrid.Visible = True
```

Code-Beispiel C#

```
VcDateLineGrid dateLineGrid =
vcGantt1.TimeScaleCollection.Active.get_Section(0).get_DateLineGrid(0);
dateLineGrid.Visible = true;
```

VisibleDataFieldIndex

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das den Wert "1" (für "sichtbar") oder 0 (für "unsichtbar") enthält. Diese Eigenschaft kann auch im Dialog **Terminiliniengitter** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes, das den Sichtbarkeitsmodus enthält

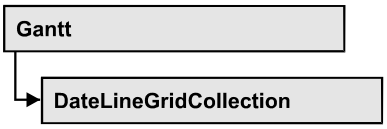
VisibleMapName

Eigenschaft von VcDateLineGrid

Mit dieser Eigenschaft können Sie den Namen einer Zuordnungstabelle (Typ vcTextMap) für den Sichtbarkeitsmodus setzen oder erfragen. Wird hier "" angegeben, wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **VisibleDataFieldIndex** angegeben ist, wird der Sichtbarkeitsstatus aus der Zuordnungstabelle ausgewählt. Diese Eigenschaft kann auch im Dialog **Terminiliniengitter** festgelegt werden. Trifft kein Datenfeldeintrag der Zuordnungstabelle zu, wird der Wert aus dem Dialog verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle für den Sichtbarkeitsmodus

7.28 VcDateLineGridCollection



In einem Objekt des Typs VcDateLineGridCollection sind alle verfügbaren Terminliniengitter zusammengefasst. Über **For Each dateLineGrid In DateLineGridCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Terminliniengitter zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **DateLineGridByName** und **DateLineGridByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Terminliniengitter kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add...**, **Copy...** und **Remove...** ermöglichen das Hinzufügen, Kopieren und Löschen von Terminliniengittern.

Eigenschaften

- Count

Methoden

- Add
- AddBySpecification
- Copy
- DateLineGridByIndex
- DateLineGridByName
- FirstDateLineGrid
- GetEnumerator
- NextDateLineGrid
- Remove
- Update

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcDateLineGridCollection

Mit dieser Eigenschaft kann die Anzahl der Terminliniengitter in der Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der DateLineGrids

Code-Beispiel VB.NET

```
Dim numberOfDateLine As Integer

numberOfDateLine = VcGantt1.DateLineCollection.Count
```

Code-Beispiel C#

```
int numberOfDateLines = vcGantt1.DateLineCollection.Count;
```

Methoden

Add

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie ein neues Terminliniengitter in der DateLineGridCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Terminliniengitterobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter: ⇨ DateLineGridName	System.String	Name des DateLineGrids
Rückgabewert	VcDateLineGrid	Neues DateLineGrid-Objekt

Code-Beispiel VB.NET

```
newDateLineGrid = VcGantt1.DateLineGridCollection.Add("dateLineGrid1")
```

Code-Beispiel C#

```
newDateLineGrid = vcGantt1.DateLineGridCollection.Add("dateLineGrid1");
```

AddBySpecification

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie ein Terminliniengitter über eine Terminliniengitter-Spezifikation erzeugen. Dies dient der Persistenz von Terminliniengitter-Objekten. Die Spezifikation eines Terminliniengitters kann erfragt und gespeichert werden. Bei einer neuen Sitzung kann das

gleiche Terminliniengitter mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ Specification	System.String	DateLineGrid-Spezifikation
Rückgabewert	VcDateLineGrid	Neues DateLineGrid-Objekt

Code-Beispiel VB.NET

```
Dim dateLineGridCltn As VcDateLineGridCollection

dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGridCltn.AddBySpecification(textSpecification)
dateLineGridCltn.Update()
```

Code-Beispiel C#

```
VcDateLineGridCollection dateLineGridCltn = vcGantt1.DateLineGridCollection;
dateLineGridCltn.AddBySpecification(textSpecification);
dateLineGridCltn.Update();
```

Copy

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie ein Terminliniengitter kopieren. Wenn das Terminliniengitter mit dem angegebenen Namen existiert und der Name des neuen Terminliniengitters noch nicht verwendet wird, wird das neue Terminliniengitterobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ DateLineGridName	System.String	Name des zu kopierenden DateLineGrids
⇨ newDateLineGridName	System.String	Name des neuen DateLineGrids
Rückgabewert	VcDateLineGrid	DateLineGrid-Objekt

Code-Beispiel VB.NET

```
Dim dateLineGridCltn As VcDateLineGridCollection

dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGridCltn.Copy("DateLineGridOne", "NewDateLineGrid")
dateLineGridCltn.Update()
```

Code-Beispiel C#

```
VcDateLineGridCollection dateLineGridCltn = vcGantt1.DateLineGridCollection;
dateLineGridCltn.Copy("DateLineGridOne", "NewDateLineGrid");
dateLineGridCltn.Update();
```

DateLineGridByIndex

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie auf eine einzelne Terminliniengitterüber ihren Index zugreifen. Existiert kein Terminliniengitter unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des DateLineGrids
Rückgabewert	VcDateLineGrid	Ermitteltes DateLineGridobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByIndex(0)
MsgBox(dateLine.Name)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByIndex(0);
MessageBox.Show(dateLine.Name);
```

DateLineGridByName

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie unter Verwendung des Namens des Terminliniengitters auf ein bestimmtes Terminliniengitter zugreifen. Existiert kein DateLineGrid-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ DateLineGridName	System.String	Name des DateLineGrids
Rückgabewert	VcDateLineGrid	DateLineGrid

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
dateLine = dateLineCltn.DateLineByName("DateLineOne")
MsgBox(dateLine.Name)
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
VcDateLine dateLine = dateLineCltn.DateLineByName("DateLineOne");
MessageBox.Show(dateLine.Name);
```

FirstDateLineGrid

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie auf das erste Terminliniengitter der Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextDateLineGrid** über die nachfolgenden Terminliniengitter zu iterieren. Existiert kein Terminliniengitter in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDateLineGrid	Erstes DateLineGrid

Code-Beispiel VB.NET

```
Dim dateLineGridCltn As VcDateLineGridCollection

dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGrid = dateLineGridCltn.FirstDateLineGrid
```

Code-Beispiel C#

```
VcDateLineGridCollection dateLineGridCltn = vcGantt1.DateLineGridCollection;
VcDateLineGrid dateLineGrid = dateLineGridCltn.FirstDateLineGrid();
```

GetEnumerator

Methode von VcDateLineGridCollection

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 Terminlinien-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	System.Object	Referenzobjekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
Dim dateLine As VcDateLine

dateLineCltn = VcGantt1.DateLineCollection
For Each dateLine In dateLineCltn
   ListBox1.Items.Add(dateLine.Name)
Next
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
foreach (VcDateLine dateLine in dateLineCltn)
   listBox1.Items.Add(dateLine.Name);
```

NextDateLineGrid

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Terminliniengitter der Auflistung zugreifen, nachdem Sie mit der Methode **FirstDateLineGrid** den Initialwert erfasst haben. Sind alle Levels durchlaufen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcDateLineGrid	Nachfolgendes DateLineGrid

Code-Beispiel VB.NET

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGrid = dateLineGridCltn.FirstDateLineGrid

While Not dateLineGrid Is Nothing
   ListBox1.Items.Add(dateLineGrid.Name)
   dateLineGrid = dateLineGridCltn.NextDateLineGrid
End While
```

Code-Beispiel C#

```
VcDateLineGridCollection dateLineGridCltn = vcGantt1.DateLineGridCollection;
VcDateLineGrid dateLineGrid = dateLineGridCltn.FirstDateLineGrid();

while (dateLineGrid != null)
   {
   ListBox.Items.Add(dateLineGrid.Name);
   dateLineGrid = dateLineGridCltn.NextDateLineGrid();
   }
```

Remove

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie ein Terminliniengitter löschen. Wenn das Terminliniengitter noch in irgendeinem anderen Objekt benutzt wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
Parameter:		
⇨ DateLineGridName	System.String	Name des DateLineGrids
Rückgabewert	System.Boolean	DateLineGrid gelöscht (True)/nicht gelöscht (False)

Code-Beispiel VB.NET

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGrid = dateLineGridCltn.DateLineGridByIndex(0)
dateLineGridCltn.Remove(dateLineGrid.Name)
dateLineGridCltn.Update()
```

Code-Beispiel C#

```
VcDateLineGridCollection dateLineGridCltn = vcGantt1.DateLineGridCollection;
VcDateLineGrid dateLineGrid = dateLineGridCltn.DateLineGridByIndex(0);
dateLineGridCltn.Remove(dateLineGrid.Name);
dateLineGridCltn.Update();
```

Update

Methode von VcDateLineGridCollection

Mit dieser Methode können Sie die Darstellung aller Objekte, die durch die verwendeten Terminliniengitter bestimmt werden, aktualisieren. Wenn Sie diese Methode nicht aufrufen, werden die Änderungen der Terminliniengitter zur Laufzeit nicht ausgeführt. Sie sollten diese Methode erst am Ende des Codes zur Festlegung der Terminliniengitter und der Level-Auflistung verwenden, damit die Aktualisierung nicht schon ausgeführt wird, bevor alle Festlegungen der Terminliniengitter ausgeführt worden sind.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

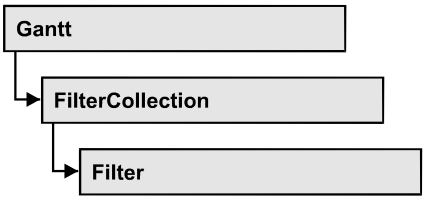
Code-Beispiel VB.NET

```
Dim dateLineGridCltn As VcDateLineGridCollection
Dim dateLineGrid As VcDateLineGrid

dateLineGridCltn = VcGantt1.DateLineGridCollection
dateLineGrid = dateLineGridCltn.DateLineGridByIndex(0)
dateLineGridCltn.Remove(dateLineGrid.Name)
dateLineGridCltn.Update()
```

Code-Beispiel C#

```
VcDateLineGridCollection dateLineGridCltn = vcGantt1.DateLineGridCollection;
VcDateLineGrid dateLineGrid = dateLineGridCltn.DateLineGridByIndex(0);
dateLineGridCltn.Remove(dateLineGrid.Name);
dateLineGridCltn.Update();
```

7.29 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**
Eigenschaftswert	VcDataTableType **Mögliche Werte:** .vcMainData 0 .vcMaindata 0 .vcRelations 1 .vcRelations 1	Typ der Datendefinitionstabelle Definition von Knotendaten Tabellentyp **vcMaindata** (für Knoten) Definition von Verbindungsdaten Tabellentyp **vcRelations** (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**
Eigenschaftswert	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**
Eigenschaftswert	System.String	Name des Filters

Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcGantt1.FilterCollection

For Each filter In filterCltn
   ListBox1.Items.Add(filter.Name)
Next
```

Code-Beispiel C#

```
VcFilterCollection filterCltn = vcGantt1.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 = VcGantt1.FilterCollection
filter = filterCltn.FirstFilter
MsgBox(filter.Specification)
```

Code-Beispiel C#

```
VcFilterCollection boxCltn = vcGantt1.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
Eigenschaftswert	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
Parameter:		
⇨ index	System.Int16	Index der Filterbedingung {0 ... VcFilter.SubConditionCount-1}
Eigenschaftswert	VcFilterSubCondition	Filterbedingungsobjekt

SubConditionCount

Nur-Lese-Eigenschaft von VcFilter

Mit dieser Eigenschaft können Sie die Anzahl der Filterbedingungen erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	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
Parameter:		
⇨ 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)}
Rückgabewert	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
Parameter:		
⇨ 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)}
Rückgabewert	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**
Parameter:		
⇨ dataObjectParam	Variant	Datensatzobjekt
Rückgabewert	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**
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim filter As VcFilter
Dim filterCond As VcFilterSubCondition

filter = VcGantt1.FilterCollection.FirstFilter

For Each filterCond In filter
   Debug.Write(filterCond.Index)
Next
```

Code-Beispiel C#

```
VcFilter filter = vcGantt1.FilterCollection.FirstFilter();
foreach(VcFilterSubCondition filterCond in filter)
   {
   Console.Write(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
Rückgabewert	System.Boolean	Filterbedingungen korrekt (True)/ nicht korrekt (False)

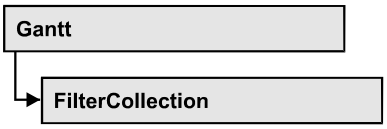
RemoveSubCondition

Methode von VcFilter

Mit dieser Methode können Sie eine Filterbedingung mit Hilfe der Indexangabe löschen.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index der zu löschenden Filterbedingung

7.30 VcFilterCollection



In einem Objekt vom Typ VcFilterCollection sind automatisch alle verfügbaren Filter zusammengefasst. Über **For Each filter In Filter-Collection** 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 = VcGantt1.FilterCollection
numberOfFilters = filterCltn.Count
```

Code-Beispiel C#

```
VcFilterCollection filterCltn = vcGantt1.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

```
VcGantt1.ActiveNodeFilter = VcGantt1.FilterCollection.MarkedNodesFilter
```

Code-Beispiel C#

```
vcGantt1.ActiveNodeFilter = vcGantt1.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
Parameter:		
⇨ newName	System.String	Name des Filters
Rückgabewert	VcFilter	Neues Filterobjekt

Code-Beispiel VB.NET

```
newFilter = VcGantt1.FilterCollection.Add("foo")
```

Code-Beispiel C#

```
newFilter = vcGantt1.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
Parameter:		
⇨ filterSpecification	System.String	Filterspezifikation
Rückgabewert	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
Parameter:		
⇨ fromName	System.String	Name des zu kopierenden Filters

⇨ newName	System.String	Name des neuen Filters
Rückgabewert	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**
Parameter:		
⇨ index	System.Int16	Index des Filters
Rückgabewert	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**
Parameter:		
⇨ filterName	System.String	Name des Filters
Rückgabewert	VcFilter	Filter

Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
Dim filter As VcFilter

filterCltn = VcGantt1.FilterCollection
filter = filterCltn.FilterByName("Department A")
```

Code-Beispiel C#

```
VcFilterCollection filterCltn = vcGantt1.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 = VcGantt1.FilterCollection
filter = filtercltn.FirstFilter
```

Code-Beispiel C#

```
VcFilterCollection filterCltn = vcGantt1.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 = VcGantt1.FilterCollection.FirstFilter

For Each filterCond In filter
   Debug.Write(filterCond.FilterName)
Next
```

Code-Beispiel C#

```
VcFilter filter = vcGantt1.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 = VcGantt1.FilterCollection
filter = filtercltn.FirstFilter

While Not filter Is Nothing
   ListBox1.Items.Add(filter.Name)
   filter = filterCltn.NextFilter
End While
```

Code-Beispiel C#

```
VcFilterCollection filterCltn = vcGantt1.FilterCollection;
VcFilter filter = filterCltn.FirstFilter();

while (filter != null)
   {
   ListBox.Items.Add(filter.Name);
   filter = filterCltn.NextFilter();
   }
```

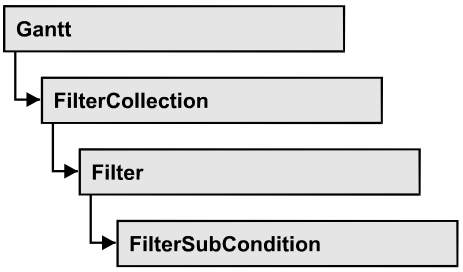
Remove

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
Parameter:		
⇨ name	System.String	Name des Filters
Rückgabewert	System.Boolean	Filter gelöscht (True)/nicht gelöscht (False)

7.31 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

- GetEnumerator
- 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 **Mögliche Werte:** .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.

Sonderwerte:

- -1: kein Datenfeld (ungültig)
- vcBarGroupLevel: Platzhalter für Gruppenebenennummer
- vcGroupCollapsed: für kollabierte Gruppen
- vcGroupNodeOrSummaryNode: Eintrag, um Filter für Sammelvorgänge definieren zu können
- vcNodesInSeparateRows: für eine Darstellung, bei der jeder Knoten in einer eigenen Zeile dargestellt wird
- vcNodesOverlaid: für eine Darstellung, bei der Knoten einander ggf. überlappen
- vcRowNumber: Eintrag, um Filter zeilenweise definieren zu können
- vcSumBarLevel: Platzhalter für Summenbalken-Ebenennummer

Diese Eigenschaft können Sie auch im **Filter bearbeiten**-Dialog setzen.

	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
Eigenschaftswert	VcOperator **Mögliche Werte:** .vcDateEarlier 27 .vcDateEarlierOrEqual 28 .vcDateEqual 25 .vcDateIn 31 .vcDateLater 29 .vcDateLaterOrEqual 30 .vcDateNotEqual 26 .vcDateNotIn 32 .vcIntEqual 9 .vcIntGreater 13 .vcIntGreaterOrEqual 14 .vcIntIn 15 .vcIntLess 11 .vcIntLessOrEqual 12 .vcIntNotEqual 10 .vcIntNotIn 16 .vcInvalidOp 0 .vcStringBeginsWith 3 .vcStringContains 5 .vcStringEqual 1 .vcStringIn 7 .vcStringNotBeginsWith 4 .vcStringNotContains 6 .vcStringNotEqual 2	Vergleichsoperator Datum früher als Datum früher als oder gleich Datum gleich Datum in Datum später als Datum später als oder gleich Datum ungleich Datum nicht in Integer gleich Integer größer Integer größer oder gleich Integer in Integer kleiner als Integer kleiner als oder größer Integer ungleich Integer nicht in ungültiger Operator String beginnt mit String enthält String gleich String enthält String beginnt nicht mit String enthält nicht String nicht gleich

	.vcStringNotIn 8	String nicht enthalten in

Methoden

GetEnumerator

Methode von VcFilterSubCondition

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 Felder des FilterSubCondition-Objektes iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

IsValid

Methode von VcFilterSubCondition

Diese Methode prüft, ob die Filterbedingung korrekt formuliert ist.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Filterbedingung korrekt (True)/ nicht korrekt (False)

7.32 VcGantt

Ein VcGantt-Objekt ist das VARCHART-XGantt-Steuerelement. Über Ereignisse können Sie dessen Interaktionen kontrollieren. Das VcGantt-Objekt kann durch eine Vielzahl von Eigenschaften und Methoden den Anforderungen entsprechend konfiguriert werden.

Eigenschaften

- ActiveNodeFilter
- AllLayersMovingTogether
- AllLayersMovingTogetherAlways
- Arrangement
- ArrowKeyMode
- ArrowKeyStepSizeMultiplier
- BorderArea
- BoxCollection
- BoxCreationAllowed
- BoxFormatCollection
- CalendarCollection
- CalendarGridCollection
- CalendarProfileCollection
- ConsiderLinkRelationTypesOnNodeDragging
- ContextMenuForBoxesEnabled
- CtrlCXVProcessingEnabled
- DataDefinition
- DataTableCollection
- DateLineCollection
- DateLineCollection
- DateOutputFormat
- DiagramAlternatingRowBackgroundColor
- DiagramBackgroundColor
- DiagramHistogramHeightRatio
- DiagramHistogramHeightRatioEx
- DiagramVisible
- DialogFont
- DirectDataWritingModeEnabled
- DoubleOutputFormat
- Enabled

- EndDateForAutomaticScheduling
- EventsSecurityCheck
- ExtendedDataTablesEnabled
- ExtendedEditingBehavior
- FilePath
- FilterCollection
- FontAntiAliasingEnabled
- GroupCollection
- GroupingDataFieldIndex
- GroupingModificationsAllowed
- GroupLevelLayoutCollection
- GroupOptimizationOnInteractionsEnabled
- GroupSortingDataFieldIndex
- GroupSortingOrder
- HierarchyDataFieldIndex
- HierarchyLevelLayout
- HistogramCollection
- HistogramSeparationLineColor
- HorizontalMovementWhileDraggingAllowed
- InbuiltMouseCursorWhileDraggingEnabled
- InfoWindow
- InitialRowCount
- InPlaceEditingOnGroupsInDiagramEnabled
- InPlaceEditingOnGroupsInTableEnabled
- InPlaceEditingOnNodesInDiagramEnabled
- InPlaceEditingOnNodesInTableEnabled
- InteractionMode
- KeepingNodesTogetherDataFieldIndex
- LayerCollection
- LayersWithNonWorkInterval
- LeavingControlWhileDraggingAllowed
- LeftTable
- LeftTableDiagramWidthRatio
- LeftTableDiagramWidthRatioEx
- LegendView
- LineFormatCollection
- LinkAppearanceCollection
- LinkCollection
- LinkPredecessorDataFieldIndex
- LinksDataTableName

- LinkSuccessorDataFieldIndex
- LinkTypeDataFieldIndex
- MapCollection
- MinimumRowHeight
- MouseProcessingEnabled
- MoveMode
- MovingLayersAsNodeWithShiftKeyAllowed
- MultipleBoxMarkingAllowed
- NodeCalendarNameDataFieldIndex
- NodeCollection
- NodeCreationAllowed
- NodeCreationAtDroppingEnabled
- NodeCreationViaDoubleclick
- NodeCreationWithDialog
- NodeDurationDataFieldIndex
- NodeEndDateDataFieldIndex
- NodeLevelLayout
- NodeRowNumberDataFieldIndex
- NodesDataTableName
- NodeSortingDataFieldIndex
- NodeSortingOrder
- NodeStartDateDataFieldIndex
- NodesUseCalendars
- NodeToolTipTextDataFieldIndex
- NumericScaleCollection
- NumericScaleRescalingAllowed
- OLEDragViaDiagram
- OLEDragViaTable
- OverlapLayerEnabled
- OverlapLayerName
- PanningModeAllowed
- PartialLoadThreshold
- PhantomDrawingWhileDraggingEnabled
- PhantomLayerHeight
- Printer
- ResourceScheduler2
- RightTable
- RightTableDiagramWidthRatio
- RightTableDiagramWidthRatioEx
- RoundedLinkSlantsEnabled

- RowHeightReductionEnabled
- RowMargins
- Sash3DStyleEnabled
- SashThickness
- Scheduler
- ScrollEventsEnabled
- SelectedNodesMovingTogether
- SelectedRowBackgroundColor
- SelectionViaRubberRectAllowed
- ShowSnapLines
- ShowSnapMarkings
- SnapTargetNodesSelectionMode
- StartDateForAutomaticScheduling
- SubRowMargins
- SummaryBarsVisible
- TableCollection
- TableColumnWidthOptimizationAllowed
- TextEntrySupplyingEventEnabled
- TimeScaleCollection
- TimeScaleDialogEnabled
- TimeScaleEnd
- TimeScaleRescalingAllowed
- TimeScaleStart
- TimeUnit
- TimeUnitsPerStep
- ToolTipChangeDuration
- ToolTipDuration
- ToolTipPointerDuration
- ToolTipShowAfterClick
- ToolTipTextSupplyingEventEnabled
- TrackingSpaceBackgroundColor
- TrackingSpacePattern
- TrackingSpacePatternColor
- UpdateBehaviorCollection
- UseHigherDiagramHistogramHeightRatioPrecision
- UseHigherTableDiagramWidthRatioPrecision
- UseSnapTargetsInInteractions
- UseTwinLineSashPhantom
- VerticalNodeMovementAllowed
- VerticalNodeMovementViaTableAllowed

- ViewComponentsBackgroundColor
- ViewComponentsBorderColor
- WaitCursorEnabled
- WorldView
- ZoomFactor
- ZoomingPerMouseWheelAllowed

Methoden

- ConvertDistance
- DeleteLinkRecord
- DeleteNodeRecord
- DetectDataTableFieldName
- DetectDataTableName
- DetectFieldIndex
- DumpConfiguration
- EndLoading
- ExportGraphicsToFileEx
- FitChartIntoView
- FitHistogramsIntoView
- FitRangeIntoView
- GetAValueFromARGB
- GetBValueFromARGB
- GetCurrentComponentStart
- GetCurrentViewDates
- GetDate
- GetDateAsString
- GetGValueFromARGB
- GetLinkByID
- GetLinkByNodeIDs
- GetNodeByID
- GetRValueFromARGB
- GetViewComponentSize
- GroupNodes
- IdentifyField
- IdentifyLayerAt
- IdentifyObject
- IdentifyObjectAt
- ImportConfiguration
- InitializeForWebService
- InsertLinkRecord

- InsertNodeRecord
- Load
- MakeARGB
- OptimizeTimeScaleStartEnd
- PrintEx
- PrintToFile
- RecalculateAllStructureCodes
- Reset
- SaveAsEx
- ScheduleProject
- ScrollComponentStartTo
- ScrollToDate
- ScrollToGroupLine
- ScrollToNode
- ScrollToNodeLine
- SetImageResource
- ShowAboutDialog
- ShowEditGroupDialog
- ShowExportGraphicsDialog
- ShowLinkEditDialog
- ShowNodeEditDialog
- ShowPageSetupDialog
- ShowPrintDialog
- ShowPrinterSetupDialog
- ShowPrintPreviewDialog
- SortGroups
- SortNodes
- SuspendUpdate
- UpdateLinkRecord
- UpdateNodeRecord
- UpdateRowNumberFields
- Zoom

Ereignisse

- KeyDown
- KeyPress
- KeyUp
- VcBoxCreated
- VcBoxCreating
- VcBoxLeftClicking

- VcBoxLeftDoubleClicking
- VcBoxModified
- VcBoxModifying
- VcBoxRightClicking
- VcCalendarGridRightClicking
- VcComponentScrolled
- VcComponentScrolling
- VcCurveLeftClicking
- VcCurveLeftDoubleClicking
- VcCurveModified
- VcCurveModifying
- VcCurveModifyingEx
- VcCurvePointDeleting
- VcCurvePointDeletingEx
- VcCurvePointInserting
- VcCurvePointInsertingEx
- VcCurveRightClicking
- VcDataModified
- VcDataRecordCreated
- VcDataRecordCreating
- VcDataRecordDeleted
- VcDataRecordDeleting
- VcDataRecordModified
- VcDataRecordModifying
- VcDataRecordNotFound
- VcDateLineModifying
- VcDateLineRightClicking
- VcDateShowing
- VcDiagramHorizontalScrolled
- VcDiagramHorizontalScrolling
- VcDiagramLeftClicking
- VcDiagramLeftDoubleClicking
- VcDiagramRightClicking
- VcDragCompleting
- VcDragOver
- VcDragStarting
- VcErrorOccurring
- VcFieldSelecting
- VcGroupDeleting
- VcGroupLeftClicking

- VcGroupLeftDoubleClicking
- VcGroupModified
- VcGroupModifying
- VcGroupRightClicking
- VcGroupsMarked
- VcGroupsMarking
- VcHelpRequested
- VcHistogramCurveNameShowingInMenu
- VcHistogramLeftClicking
- VcHistogramLeftDoubleClicking
- VcHistogramRightClicking
- VcHistogramsHeightChanged
- VcHistogramsHeightChanging
- VcHistogramsHeightChangingEx
- VcInPlaceEditorShowing
- VcInteractionEnded
- VcInteractionModeChanged
- VcInteractionModeChanging
- VcInteractionObjectChanged
- VcInteractionStarted
- VcLegendViewClosed
- VcLinkCreated
- VcLinkCreating
- VcLinkDeleted
- VcLinkDeleting
- VcLinksLeftClicking
- VcLinksLeftDoubleClicking
- VcLinksRightClicking
- VcNodeCreated
- VcNodeCreating
- VcNodeDeleted
- VcNodeDeleting
- VcNodeLeftClicking
- VcNodeLeftDoubleClicking
- VcNodeModified
- VcNodeModifiedEx
- VcNodeModifying
- VcNodeResizeStarting
- VcNodeRightClicking
- VcNodesMarked

- VcNodesMarking
- VcNumericScaleLeftClicking
- VcNumericScaleLeftDoubleClicking
- VcNumericScaleRescaling
- VcNumericScaleRightClicking
- VcObjectDrawing
- VcObjectDrawn
- VcResourceSchedulingProgressing
- VcResourceSchedulingWarning
- VcSashButtonClicked
- VcStatusLineTextShowing
- VcTableCaptionLeftClicking
- VcTableCaptionLeftDoubleClicking
- VcTableCaptionRightClicking
- VcTableColumnWidthChanged
- VcTableColumnWidthChanging
- VcTableColumnWidthOptimizing
- VcTableWidthChanging
- VcTableWidthChangingEx
- VcTextEntrySupplying
- VcTimeScaleEndModified
- VcTimeScaleLeftClicking
- VcTimeScaleLeftDoubleClicking
- VcTimeScaleModified
- VcTimeScaleRightClicking
- VcTimeScaleSectionRescaled
- VcTimeScaleSectionRescaledEx
- VcTimeScaleSectionRescaling
- VcTimeScaleSectionRescalingEx
- VcTimeScaleSectionStartModifying
- VcTimeScaleStartModified
- VcToolTipTextSupplying
- VcViewComponentsSizeModified
- VcWorldViewClosed
- VcZoomFactorModified

Eigenschaften

ActiveNodeFilter

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Filter festlegen oder erfragen, der die darzustellenden Knoten selektiert.

	Datentyp	Beschreibung
Eigenschaftswert	VcFilter	Filterobjekt **Standardwert:** Nothing

Code-Beispiel VB.NET

```
VcGantt1.ActiveNodeFilter = VcGantt1.FilterCollection.FilterByName("Milestone")
```

Code-Beispiel C#

```
vcGantt1.ActiveNodeFilter = vcGantt1.FilterCollection.FilterByName("Milestone");
```

AllLayersMovingTogether

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob ein markierter Knoten als ganzes mit der Maus verschoben werden kann (True) oder ob jeder Layer eines markierten Knotens nur einzeln verschoben werden kann (False). Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Markierter Knoten als ganzes mit der Maus verschiebbar (True)/jeder Layer eines markierten Knotens nur einzeln verschiebbar (False)

Code-Beispiel VB.NET

```
VcGantt1.AllLayersMovingTogether = True
```

Code-Beispiel C#

```
vcGantt1.AllLayersMovingTogether = true;
```

AllLayersMovingTogetherAlways

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob alle Layer eines Knotens gemeinsam verschoben werden können, ohne dass sie zuvor markiert wurden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Direktes Verschieben von kompletten Knoten ohne vorherige Markierung ist an- (true) oder abgeschaltet (false) **Standardwert:** false

Code-Beispiel VB.NET

```
VcGantt1.AllLayersMovingTogetherAlways = True
```

Code-Beispiel C#

```
vcGantt1.AllLayersMovingTogetherAlways = true;
```

Arrangement

Eigenschaft von VcGantt

Mit dieser Eigenschaft legen Sie fest, ob Vorgänge hierarchisch oder gruppenweise angeordnet werden sollen. Diese Eigenschaft kann auch im Dialog **Gruppierung**, Kontrollkästchen **Hierarchie** eingestellt werden. Diese Eigenschaft wird wirksam, wenn die Eigenschaft **HierarchyDataFieldIndex** bzw. **GroupDataFieldIndex** entsprechend gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	VcArrangementType	Anordung der Vorgänge gruppenweise oder hierarchisch **Standardwert:** vcArrangementTypeGroupwise
	Mögliche Werte:	
	.vcArrangementTypeGroupwise 1	Gruppenweise Anordnung der Vorgänge
	.vcArrangementTypeHierarchical 2	Hierarchische Anordnung der Vorgänge

Code-Beispiel VB.NET

```
VcGantt1.GroupLevelLayoutCollection.FirstGroupLevelLayout().GroupDataFieldIndex
= VcGantt1.DetectFieldIndex("Maindata", "Department")
VcGantt1.Arrangement = VcArrangementType.vcArrangementTypeGroupwise
VcGantt1.GroupNodes(True)

// alternativ:

VcGantt1.HierarchyDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"StructureCode")
VcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical
VcGantt1.GroupNodes(True)
```

Code-Beispiel C#

```
vcGantt1.GroupLevelLayoutCollection.FirstGroupLevelLayout().GroupDataFieldIndex
= vcGantt1.DetectFieldIndex("Maindata", "Department");
vcGantt1.Arrangement = VcArrangementType.vcArrangementTypeGroupwise;
vcGantt1.GroupNodes(true);

// alternativ:

vcGantt1.HierarchyDataFieldIndex = vcGantt1.DetectFieldIndex("Maindata",
"StructureCode");
vcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical;
vcGantt1.GroupNodes(true);
```

ArrowKeyMode

Eigenschaft von VcGantt

Mit dieser Eigenschaft wird der Modus der Pfeiltasten <links> und <rechts> festgelegt. Standardmäßig sind die Pfeiltasten mit navigierenden Funktionen wie Abrollen des Diagramms oder Bewegen eines markierten Feldes in einem Knoten oder in der Tabelle belegt. Sie können den Pfeiltasten durch diese Eigenschaft verändernde Funktionen zuweisen, so dass der Benutzer einen Knoten verschieben, vergrößern oder verkleinern kann. Das Info-Fenster mit Informationen zur Position verbleibt noch eine kurze Zeit nach Beendigung der Interaktion, damit die Information gelesen werden kann.

Kommt der Knoten beim Verschieben an den Rand des Views, rollte das Diagramm automatisch ab (Autoscroll).

Beim einfachen Drücken der Pfeiltasten wird der Knoten verschoben, beim gleichzeitigen Drücken der <Shift> -Taste wird er verlängert oder verkürzt.

Die Belegung der Pfeiltasten mit verändernden Funktionen ist besonders nützlich bei Diagrammen mit niedriger Auflösung, da das Positionieren von Knoten auf ein genaues Datum mit der Maus unpräzise sein kann. Die Postionierung mittels Pfeiltasten ist genauer, weil in dem Informationsfenster jeder Bewegungsschritt mit einem veränderten Datum angezeigt wird und dadurch eine höhere Auflösung angeboten wird als in der Zeitskala sichtbar

ist. Die Schrittweite wird dabei gesteuert über die Eigenschaften **VcGantt.TimeUnit** und **VcGantt.TimeUnitsPerStep** sowie **VcGantt.ArrowKeyStepSizeMultiplier**.

	Datentyp	Beschreibung
Eigenschaftswert	VcArrowKeyMode	Modus der Pfeiltasten <links> und <rechts>
	Mögliche Werte:	
	.vcnodeJumpToSnapTarget 512	Durch Drücken der von <STRG> + Pfeiltasten <links> und <rechts> wird ein zuvor markierter Knoten zum vorherigen/nächsten Einrastziel bewegt.
	.vcResizeOrMoveNode 384	Die Pfeiltasten <links> und <rechts> befinden sich in dem Modus, Knoten zu verändern
	.vcStandard 127	Die Pfeiltasten <links> und <rechts> befinden sich in ihrem Standard-Modus

Code-Beispiel VB.NET

```
'Assigning the function to an option button
Private Sub OptionEditNode_Click()
    If OptionStandard.Value = True Then
        VcGantt1.ArrowKeyMode = vcStandard
    Else
        VcGantt1.ArrowKeyMode = vcResizeOrMoveNode
    End If
End Sub
```

Code-Beispiel C#

```
//Assigning the function to an option button
private void OptionEditNode_Click(object sender, EventArgs e)
{
     if (OptionEditNode.Checked)
         vcGantt1.ArrowKeyMode = VcArrowKeyMode.vcStandard;
     else
         vcGantt1.ArrowKeyMode = VcArrowKeyMode.vcResizeOrMoveNode;
}
```

ArrowKeyStepSizeMultiplier

Eigenschaft von VcGantt

Über diese Eigenschaft können Sie dem Pfeiltasten-Schrittweitenmultiplikator einen Wert setzen oder diesen erfragen. Die Schrittweite beim Bewegen des Cursors mit der Maus oder mit den Pfeiltasten (s. Eigenschaft **VcGantt.ArrowKeyMode**) wird über die beiden Eigenschaften **VcGantt.TimeUnit** und **VcGantt.TimeUnitsPerStep** festgelegt. Die Multiplikation der dort gesetzten Werte ergibt die Schrittweite: Wenn die Zeiteinheit auf "Tag" gesetzt wurde und die Einheiten pro Schritt auf 2, beträgt die Schrittweite 2 Tage. Da bei der Maus eine größere Bewegung durch weiteres Ziehen möglich ist, bei Tasten aber nicht, gibt es den zusätzlichen Multiplikationsfaktor für die Pfeiltasten. Er wird vom Benutzer

durch das zusätzliche Drücken der <Strg>-Taste aktiviert. Wenn Sie diesen Wert auf 10 setzen, beträgt die Schrittweite im o.g. Beispiel 20 Tage pro Tastendruck.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Wert des Multiplikationsfaktors

Code-Beispiel VB.NET

```
'Reducing the time scale resolution and enlarging the step size
Private Sub CommandExtendScale_Click()
    'Filling up the available space for the Gantt graph by extending the time
scale
    VcGantt1.TimeScaleEnd = DateSerial(2017, 1, 1)
    ' Reducing the resolution of the time scale by the factor 10
     VcGantt1.TimeScaleCollection.Active.Section(0).UnitWidth =
VcGantt1.TimeScaleCollection.Active.Section(0).UnitWidth / 10
    ' Increasing the multiplier for the arrow keys to move in larger steps
    VcGantt1.ArrowKeyStepSizeMultiplier = 25
End Sub
```

Code-Beispiel C#

```
'Reducing the time scale resolution and enlarging the step size
private void CommandExtendScale_Click(object sender, EventArgs e)
{
    //Filling up the available space for the Gantt graph by extending the time
scale
    vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.01.2017");
    //Reducing the resolution of the time scale by the factor 10
    vcGantt1.TimeScaleCollection.Active.get_Section(0).UnitWidth =
vcGantt1.TimeScaleCollection.Active.get_Section(0).UnitWidth / 10;
     //Increasing the multiplier for the arrow keys to move in larger steps
     vcGantt1.ArrowKeyStepSizeMultiplier = 25;
}
```

BorderArea

Nur-Lese-Eigenschaft von VcGantt

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 = VcGantt1.BorderArea
```

Code-Beispiel C#

```
VcBorderArea borderArea = vcGantt1.BorderArea;
```

BoxCollection

Nur-Lese-Eigenschaft von VcGantt

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 = VcGantt1.BoxCollection
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.BoxCollection;
```

BoxCreationAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft wird dem Anwender das Anlegen neuer Boxen erlaubt (True) oder verboten (False). Wird diese Eigenschaft auf False gesetzt, ist der **Modus: Boxen erzeugen** nicht verfügbar. Außerdem kann der **InteractionMode** nicht auf **VcCreateBox** gesetzt werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft (True)/ nicht in Kraft (False)

Code-Beispiel VB.NET

```
VcGantt1.BoxCreationAllowed = False
```

Code-Beispiel C#

```
vcGantt1.BoxCreationAllowed = false;
```

BoxFormatCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die Boxformatauflistung, in der alle zur Verfügung stehenden Boxformate enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcBoxFormatCollection	BoxFormatCollection-Objekt

Code-Beispiel VB.NET

```
Dim boxFormatCltn As VcBoxFormatCollection
Dim boxFormat As VcBoxFormat

boxFormatCltn = VcGantt1.BoxFormatCollection
For Each boxFormat In boxFormatCltn
   ListBox1.Items.Add(boxFormat.Name)
Next
```

Code-Beispiel C#

```
VcBoxFormatCollection boxFormatCltn = vcGantt1.BoxFormatCollection;
foreach (VcBoxFormat boxFormat in boxFormatCltn)
{
   listBox1.Items.Add(boxFormat.Name);
}
```

CalendarCollection

Nur-Lese-Eigenschaft von VcGantt

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 = VcGantt1.CalendarCollection
```

Code-Beispiel C#

```
VcCalendarCollection calendarCltn = vcGantt1.CalendarCollection;
```

CalendarGridCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die CalendarGrid-Auflistung, in der alle zur Verfügung stehenden Kalenderliniengitter zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendarGridCollection	CalendarGridCollection-Objekt

Code-Beispiel VB.NET

```
Dim calendarGridCltn As VcCalendarGridCollection
calendarGridCltn = VcGantt1.CalendarGridCollection
```

Code-Beispiel C#

```
VcCalendarGridCollection calendarGridCltn = vcGantt1.CalendarGridCollection;
```

CalendarProfileCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die Kalenderprofilauflistung, in der alle zur Verfügung stehenden Kalenderprofile enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcCalendarProfileCollection	CalendarProfileCollection-Objekt

ConsiderLinkRelationTypesOnNodeDragging

Eigenschaft von VcGantt

Wenn diese Eigenschaft auf True gesetzt wird, werden beim Ziehen von Knoten und eingeschalteten Verbindungen die Phantomlinien, die die Verbindungen zu verschobenen Knoten und deren Anschlussknoten darstellen, typgerecht dargestellt, d.h. sie beginnen/enden nicht in der Mitte der Knotenphantome, sondern auf deren linken oder rechten Seite.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung

Code-Beispiel VB.NET

```
VcGantt1.ConsiderLinkRelationTypesOnNodeDragging = True
```

ContextMenuForBoxesEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen oder abfragen, ob das Kontextmenü für Boxen erscheinen kann. Diese Eigenschaft könne Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kontextmenü für Box wird/wird nicht angezeigt

Code-Beispiel VB.NET

```
VcGantt1.ContextMenuForBoxesEnabled = True
```

Code-Beispiel C#

```
vcGantt1.ContextMenuForBoxesEnabled = true;
```

CtrlCXVProcessingEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft werden die Tastenkombinationen Strg+C, Strg+X und Strg+V automatisch in die Zwischenablage-Operationen **CopyNodesTo-Clipboard**, **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
Eigenschaftswert	System.Boolean	Tastenkombinationen werden/werden nicht in Zwischenablage-Operationen übersetzt **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.CtrlCXVProcessingEnabled = False
```

Code-Beispiel C#

```
vcGantt1.CtrlCXVProcessingEnabled = false;
```

DataDefinition

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das aktuelle VcDataDefinition-Objekt, um darin z. B. Feldnamen oder Feldtypen abzufragen.

Die Datendefiniton des VcGantt hat zwei Datendefinitionstabellen: vcMaindata und vcRelations.

	Datentyp	Beschreibung
Eigenschaftswert	VcDataDefinition	Datendefinition

Code-Beispiel VB.NET

```
Dim dataDefinition As VcDataDefinition
dataDefinition = VcGantt1.DataDefinition
```

Code-Beispiel C#

```
VcDataDefinition dataDefinition = vcGantt1.DataDefinition;
```

DataTableCollection

Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf die DataTable-Auflistung und auf die darin verwendeten Datentabellen.

	Datentyp	Beschreibung
Eigenschaftswert	VcDataTableCollection	Ermitteltes Auflistungsobjekt der Datentabellen

Code-Beispiel VB.NET

```
Dim dataTableCltn As VcDataTableCollection
Dim dataTable As VcDataTable

dataTableCltn = VcGantt1.DataTableCollection
For Each dataTable In dataTableCltn
   ListBox1.Items.Add(dataTable.Name)
Next
```

Code-Beispiel C#

```
VcDataTableCollection dataTablecltn = vcGantt1.DataTableCollection;
foreach(VcDataTable dataTable in dataTablecltn)
   listBox1.Items.Add(dataTable.Name);
```

DateLineCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die DateLine-Auflistung, in der alle zur Verfügung stehenden Stichtaglinien zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcDateLineCollection	DateLineCollection-Objekt

Code-Beispiel VB.NET

```
Dim dateLineCltn As VcDateLineCollection
dateLineCltn = VcGantt1.DateLineCollection
```

Code-Beispiel C#

```
VcDateLineCollection dateLineCltn = vcGantt1.DateLineCollection;
```

DateLineCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die DateLine-Auflistung, in der alle zur Verfügung stehenden Stichtaglinien enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcDateLineCollection	DateLineCollection-Objekt

DateOutputFormat

Eigenschaft von VcGantt

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**
Eigenschaftswert	System.String	Datum {DMYhms:;/}

Code-Beispiel VB.NET

```
VcGantt1.DateOutputFormat = "DD.MM.YY"
```

Code-Beispiel C#

```
vcGantt1.DateOutputFormat = "DD.MM.YY";
```

DiagramAlternatingRowBackgroundColor

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie eine zweite Hintergrundfarbe für das Diagramms setzen oder erfragen, die mit der durch die Eigenschaft **Diagram-BackgroundColor** gesetzen Hintergrundfarbe ein zeilenweise alternierendes Farbmuster bildet. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** SystemDrawing.Color.White

Code-Beispiel VB.NET

```
VcGantt1.DiagramAlternatingRowBackgroundColor = System.Drawing.Color.Blue
```

Code-Beispiel C#

```
vcGantt1.DiagramAlternatingRowBackgroundColor = System.Drawing.Color.Blue;
```

DiagramBackgroundColor

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Diagramms setzen oder erfragen. Zusammen mit der Eigenschaft **DiagramAlternating-RowBackgroundColor** können Sie ein zeilenweise alternierendes Farbmuster einrichten. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** SystemDrawing.Color.White

Code-Beispiel VB.NET

```
VcGantt1.DiagramBackgroundColor = System.Drawing.Color.Blue
```

Code-Beispiel C#

```
vcGantt1.DiagramBackgroundColor = System.Drawing.Color.Blue;
```

DiagramHistogramHeightRatio

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das prozentuale Verhältnis von der Höhe des Diagrammbereichs (also des Diagramms ohne Histogramm) und der Höhe des Histogramms beim Start erfragen oder festlegen. Bei den Werten -1 und 0 wird das Histogramm beim Start komplett angezeigt. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	Integer {-1, 0, 1, ..., 1000}	Verhältnis von Diagrammhöhe und Histogrammhöhe

Code-Beispiel VB.NET

```
Dim ratio As Integer
ratio = VcGantt1.DiagramHistogramHeightRatio
```

Code-Beispiel C#

```
int ratio = vcGantt1.DiagramHistogramHeightRatio;
```

DiagramHistogramHeightRatioEx

Eigenschaft von VcGantt

Mit dieser Eigenschaft lässt sich das Verhältnis der Gesamthöhe des Diagramms (in %) zur Höhe des Histogramms erfragen oder festlegen.

Diese Eigenschaft unterscheidet sich von der Eigenschaft **DiagramHistogramHeightRatio** durch die Rückgabe des Datentyps "Double", mit dem eine höhere Genauigkeit erzielt wird. Die Verwendung dieser Eigenschaft muss zuvor über die Eigenschaft **UseHigherDiagramHistogramHeightRatioPrecision** oder über die Option **Höhere Genauigkeit für das Höhenverhältnis zwischen Diagramm und Histogramm(en)** auf der Eigenschaftenseite **Allgemeines** aktiviert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Höhenverhältnis

Code-Beispiel VB.NET

```
VcGantt1.DiagramHistogramHeightRatioEx = 40
```

Code-Beispiel C#

```
vcGantt1.DiagramHistogramHeightRatioEx = 40;
```

DiagramVisible

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob der Diagrammbereich (Tabelle und GanttGraph) sichtbar sein soll. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Layout** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Diagrammbereich sichtbar (True) / nicht sichtbar (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.DiagramVisible = False
```

Code-Beispiel C#

```
vcGantt1.DiagramVisible = true;
```

DialogFont

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Schriftgröße und den Schrifttyp der zur Laufzeit erscheinenden Dialogfelder im VARCHART XGantt erfragen oder festlegen. Als Objekt wird ein Font-Objekt Ihrer Programmierumgebung erwartet, z. B. bei Visual Basic ein Objekt der Klasse **StdFont**.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftattribute

Code-Beispiel VB.NET

```
Dim newFont As Font
newFont = New Font("Verdana", 14)
VcGantt1.DialogFont = newFont
```

Code-Beispiel C#

```
Font newFont = new Font("Verdana", 14);
vcGantt1.DialogFont = newFont;
```

DirectDataWritingModeEnabled

Eigenschaft von VcGantt

Ist diese Eigenschaft auf "True" gesetzt, werden Datenänderung, die über **VcNode/VcLink/VcDataRecord/.set_DataField** bzw **.AllData** ausgeführt werden, direkt in den Datenbestand übernommen, OHNE Auswertungen (z.B. Filterauswertungen, Mapping etc.) durchzuführen. Dies führt zu einer Verbesserung der Performance.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Datenänderungen ohne Auswertungen werden (True)/werden nicht (False) ausgeführt **Standardwert:** False

DoubleOutputFormat

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das Ausgabeformat von Zahlen als double-Wert im Gantt-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.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenfolge, die das Double-Format beschreibt, z.B. "I,DDDD ppm"

Code-Beispiel VB.NET

```
VcGantt1.DoubleOutputFormat = "I,DDDD ppm"
```

Code-Beispiel C#

```
vcGantt1.DoubleOutputFormat = "$I,III.DD";
```

Enabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das VARCHART-XGantt-Steuerelement so abschalten, dass es auf Maus- und Tastenbefehle nicht reagiert.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	VARCHART-Steuerelement angeschaltet/abgeschaltet

Code-Beispiel VB.NET

```
VcGantt1.Enabled = False
```

Code-Beispiel C#

```
vcGantt1.Enabled = false;
```

EndDateForAutomaticScheduling

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie für Autoschedule (Eigenschaftenseite **Zeitrechnung**) das Enddatum für die Zeitrechnung des aktuellen Projekts angeben oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Enddatum

EventsSecurityCheck

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Event-Sicherheitsprüfung an- oder abschalten. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Event-Sicherheitsprüfung an/aus

Code-Beispiel VB.NET

```
VcGantt1.EventsSecurityCheck = False
```

Code-Beispiel C#

```
vcGantt1.EventsSecurityCheck = false;
```

ExtendedDataTablesEnabled

Eigenschaft von VcGantt

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.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**false**: nur zwei Datentabellen (Maindata und Relations) **true**: bis zu 99 Datentabellen **Standardwert:** false

Code-Beispiel VB.NET

```
VcGantt1.ExtendedDataTablesEnabled = True
```

Code-Beispiel C#

```
vcGantt1.ExtendedDataTablesEnabled = true;
```

ExtendedEditingBehavior

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob zur Laufzeit die erweiterten Möglichkeiten zur Tabellenbearbeitung genutzt werden können. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Erweiterte Tabellenbearbeitung an/aus **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.ExtendedEditingBehavior = True
```

Code-Beispiel C#

```
vcGantt1.ExtendedEditingBehavior = true;
```

FilePath

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einen Verzeichnispfad setzen, damit auch Grafikdateien, 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 XGantt-Steuerelements gesucht.

Es empfiehlt sich, diese Eigenschaft beim Start der Applikation während des Initialisierens des VARCHART XGantt-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
Eigenschaftswert	System.String	Verzeichnispfad **Standardwert:** " "

Code-Beispiel VB.NET

```
Dim exeName As String
Dim exeDir As String

exeName = System.Environment.GetCommandLineArgs(0)
exeDir = System.IO.Path.GetDirectoryName(exeName)
VcGantt1.FilePath = exeDir + "\Bitmaps"
```

Code-Beispiel C#

```
String exeName = Environment.GetCommandLineArgs()[0];
vcGantt1.FilePath = System.IO.Path.GetDirectoryName(exeName)+ @"\..\Bitmaps";
```

FilterCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf die Filter-Auflistung und damit auf die verwendeten Filter.

	Datentyp	Beschreibung
Eigenschaftswert	VcFilterCollection	FilterCollection-Objekt

Code-Beispiel VB.NET

```
Dim filterCltn As VcFilterCollection
filterCltn = VcGantt1.FilterCollection
```

Code-Beispiel C#

```
VcFilterCollection filterCltn = vcGantt1.FilterCollection;
```

FontAntiAliasingEnabled

Nur-Lese-Eigenschaft von VcGantt

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 Tabellenfeld 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
Eigenschaftswert	System.Boolean	Schriftzeichen werden optisch geglättet/nicht geglättet. **Standardwert:** true

GroupCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht bei gruppierter Darstellung den Zugriff auf das GroupCollection-Objekt, in dem alle verfügbaren Gruppen zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcGroupCollection	GroupCollection-Objekt

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
groupCltn = VcGantt1.GroupCollection
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
```

GroupingDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einstellen oder erfragen, welches Feld aus der Datendefinitionstabelle als Gruppierkriterium auf einer bestimmten Ebene verwendet werden soll. Die Sortierung der Gruppen erfolgt standardmäßig in der Reihenfolge, in der jeweils der erste Vorgang einer Gruppe eingelesen wird. Dieses Verhalten kann über die Eigenschaft **GroupSorting-DataFieldIndex** verändert werden.

Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

Die Eigenschaft **GroupingDataFieldIndex** ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_GroupingDataFieldIndex (grouping-Level, pvn) und get_GroupingDataFieldIndex (groupingLevel) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ groupingLevel	System.Int16	Gruppierebene (beginnend mit 0)
Eigenschaftswert	System.Int32	Feld-ID aus der Datendefinitionstabelle

Code-Beispiel VB.NET

```
Dim definitionTable As VcDataDefinitionTable
definitionTable =
VcGantt1.DataDefinition.DefinitionTable(VcDataTableType.vcMaindata)
VcGantt1.GroupingDataFieldIndex(0) =
definitionTable.DataDefinitionFieldByName("Code 1").ID
VcGantt1.GroupNodes(True)
```

Code-Beispiel C#

```
VcDataDefinitionTable definitionTable =
vcGantt1.DataDefinition.get_DefinitionTable(VcDataTableType.vcMaindata);
vcGantt1.set_GroupingDataFieldIndex(0, "Code 1");
vcGantt1.GroupNodes(true);
```

GroupingModificationsAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft stellen Sie ein, ob der Anwender expandierte Gruppen kollabieren oder kollabierte Gruppen wieder expandieren kann. Der Anwender kann dann die Gruppen durch einen Doppelklick auf die Gruppenüberschrift im Tabellenteil oder durch einen Klick auf das Minus- oder Plus-Zeichen neben der Gruppenüberschrift oder über das Kontextmenü für Gruppen kollabieren bzw. expandieren.

Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

Die Eigenschaft GroupingModificationsAllowed ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_Grouping-ModificationsAllowed (groupingLevel, pvn) und get_Grouping-ModificationsAllowed (groupingLevel) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ groupingLevel	System.Int16	Gruppierungsebene
Eigenschaftswert	System.Boolean	Änderungen erlaubt (True)/ nicht erlaubt (False)

Code-Beispiel VB.NET

```
VcGantt1.GroupingModificationsAllowed(0) = False
```

Code-Beispiel C#

```
vcGantt1.set_GroupingModificationsAllowed(0, false);
```

GroupLevelLayoutCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die GroupLevelLayout-Auflistung, in der alle zur Verfügung stehenden Gruppierungsebenenlayouts enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcGroupLevelLayoutCollection	GroupLevelLayoutCollection-Objekt

GroupOptimizationOnInteractionsEnabled

Eigenschaft von VcGantt

Ist diese Eigenschaft eingeschaltet, dann werden bei Interaktionen wie Aufziehen von Knoten, Knotenverschieben horizontal oder vertikal und Ändern des Start- oder Endtermins eines Layers die Knoten der gesamten Zielgruppe automatisch erneut optimiert, wenn sie bisher bereits optimiert dargestellt wird. Wird diese Eigenschaft abgeschaltet, dann wird bei denselben Interaktionen der Knoten unter der Cursorposition platziert, wenn dies nicht zu einer Überlappung führt. Bei einer möglichen Überlappung wird der Knoten in die nächste Zeile zu anderen Knoten platziert, wenn dies dort keine Überlappung verursacht. Wenn doch, dann bekommt der Knoten eine eigene Zeile unter der, an der der Mauscursor sich befindet.

Die Eigenschaft kann zur Designzeit auch in der Eigenschaftenseite **Allgemeines** eingestellt werden.

Siehe auch Methode **VcGroup.ReOptimizeNodes**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Die optimierte Anordnung nach einer Interaktion erfolgt (True) / erfolgt nicht (False)

GroupSortingDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft stellen Sie ein, welches Feld der Datendefinitionstabelle zum Sortieren der Gruppen verwendet werden soll. Durch Verwendung von **GroupSortingDataFieldIndex** können Sie die Sortierung in alphabetischer Reihenfolge auf- oder absteigend nach diesem Feld erzielen. Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

Die Eigenschaft GroupSortingDataFieldIndex ist eine indizierte Eigenschaft, die in C# über die beiden Methoden

set_GroupSortingDataFieldIndex (groupingLevel, pvn)

und get_GroupSortingDataFieldIndex (groupingLevel)

angesprochen wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ groupingLevel	System.Int16	Gruppierungsebene
Eigenschaftswert	System.Int32	Feldindex aus der Datendefinitionstabelle

Code-Beispiel VB.NET

```
VcGantt1.GroupSortingDataFieldIndex(0) = 12
VcGantt1.GroupSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcGantt1.SortGroups()
```

Code-Beispiel C#

```
vcGantt1.set_GroupSortingDataFieldIndex(0, 12);
vcGantt1.set_GroupSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcGantt1.SortGroups();
```

GroupSortingOrder

Eigenschaft von VcGantt

Mit dieser Eigenschaft legen Sie fest, ob die Gruppen auf- oder absteigend sortiert werden. Das Feld, nach dem die Gruppen sortiert werden, legen Sie mit der Eigenschaft **GroupSortingDataFieldIndex** fest. Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeSortingOrder	Aufsteigende oder absteigende Sortierreihenfolge **Standardwert:** vcAscending
	Mögliche Werte: .vcAscending 1 .vcDescending 2	 aufsteigende Abfolge absteigende Abfolge

Code-Beispiel VB.NET

```
VcGantt1.GroupSortingDataFieldIndex(0) = 12
VcGantt1.GroupSortingOrder(0) = VcNodeSortingOrder.vcAscending
VcGantt1.SortGroups()
```

Code-Beispiel C#

```
vcGantt1.set_GroupSortingDataFieldIndex(0, 12);
vcGantt1.set_GroupSortingOrder(0, VcNodeSortingOrder.vcAscending);
vcGantt1.SortGroups();
```

HierarchyDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, nach dem Vorgänge hierarchisch angeordnet werden sollen. Dies kann auch erfolgen **nachdem** bereits Daten geladen wurden. Damit Änderungen wirksam werden, muss über die Eigenschaft **VcGantt.Arrangement** die hierarchische Anordnung der Vorgänge (**vcArrangementTypeHierarchical**) eingestellt, sowie eine Aktualisierung über die Methode **VcGantt.GroupNodes** vorgenommen werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, nach dem Vorgänge hierarchisch angeordnet werden sollen

Code-Beispiel VB.NET

```
 VcGantt1.HierarchyDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata",
"Hierarchy")
 VcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical
VcGantt1.GroupNodes(True)
```

Code-Beispiel C#

```
vcGantt1.HierarchyDataFieldIndex = vcGantt1.DetectFieldIndex("Maindata",
"Hierarchy");
vcGantt1.Arrangement = VcArrangementType.vcArrangementTypeHierarchical;
vcGantt1.GroupNodes(true);
```

HierarchyLevelLayout

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf das HierarchyLevelLayout-Objekt.

	Datentyp	Beschreibung
Eigenschaftswert	VcHierarchyLevelLayout	HierarchyLevelLayout-Objekt

HistogramCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf das HistogramCollection-Objekt, in dem alle zur Verfügung stehenden Histogramme zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcHistogramCollection	HistogramCollection-Objekt

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
histogramCltn = VcGantt1.HistogramCollection
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
```

HistogramSeparationLineColor

Eigenschaft von VcGantt

Mit dieser Eigenschaft setzen oder erfragen Sie die Farbe für die Trennlinien zwischen Histogrammen. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Layout** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Farbwert ({0...255},{0...255},{0...255})

HorizontalMovementWhileDraggingAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob ein Knoten auch dann horizontal verschoben werden kann, wenn das Steuerelement die Zielkomponente eines im Moment laufenden Drag&Drop-Vorgangs ist. Für Verschiebeoperationen, die nur innerhalb eines Gantt-Diagramms stattfinden, ist diese Eigenschaft nicht wirksam.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Drag&Drop Aktion erlaubt (True) / nicht erlaubt (False) **Standardwert:** false

InbuiltMouseCursorWhileDraggingEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie in der Zielkomponente festlegen, ob beim Ziehen bei einem Drag&Drop Vorgang die für das VARCHART-Steuerelement üblichen Mauszeiger angezeigt werden sollen. Andernfalls werden nur die für das Drag&Drop üblichen Mauszeiger angezeigt (Pfeil mit Rechteck bzw. Verbotszeichen).

Letzterer kann kann durch eigene Mauszeiger ersetzt werden, wenn Sie einen Event-Handler für das Ereignis **GiveFeedback** der Basisklasse **Control** implementieren.

S. auch die VcGantt-Eigenschaften **LeavingControlWhileDragging-Allowed**, **NodeCreationAtDroppingEnabled**, **PhantomDrawingWhile-DraggingEnabled** sowie die Eigenschaft **AllowDrop** der Basisklasse **Control**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Der VARCHART-Mauszeiger wird angezeigt (true) / wird nicht angezeigt (false). **Standardwert:** true

InfoWindow

Nur-Lese-Eigenschaft von VcGantt

Über diese Eigenschaft haben Sie Zugriff auf das VcInfoWindow-Objekt, das das Informationsfenster eines Knotens bezeichnet, das in einem Balkenplan beim Anlegen oder Verändern des Knotens erscheint.

	Datentyp	Beschreibung
Eigenschaftswert	VcInfoWindow	InfoWindow-Objekt

InitialRowCount

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Anzahl der Knotenzeilen beim Programmstart setzen bzw. erfragen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Knotenzeilen beim Programmstart

InPlaceEditingOnGroupsInDiagramEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob zur Laufzeit das direkte Editieren von Gruppendatenfeldern im Diagramm-Bereich möglich ist. Voraussetzung ist die Verwendung von eigenen Datentabellen für Gruppendaten. 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.

S. ebenfalls **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditing-OnNodesInTableEnabled** und **InPlaceEditingOnGroupsInTableEnabled**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Direktes Editieren möglich (True) / nicht möglich (False) **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = True
```

Code-Beispiel C#

```
vcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = true;
```

InPlaceEditingOnGroupsInTableEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob zur Laufzeit das direkte Editieren von Gruppendatenfeldern im Tabellenbereich möglich ist. Voraussetzung ist die Verwendung von eigenen Datentabellen für Gruppendaten. 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.

S. ebenfalls **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditing-OnNodesInTableEnabled** und **InPlaceEditingOnGroupsInTableEnabled**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Direktes Editieren möglich (True) / nicht möglich (False) **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = True
```

Code-Beispiel C#

```
vcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = true;
```

InPlaceEditingOnNodesInDiagramEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob zur Laufzeit das direkte Editieren von Knotendatenfeldern im Diagramm-Bereich 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.

S. ebenfalls **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditingOnNodesInTableEnabled** und **InPlaceEditingOnGroupsInTableEnabled**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Direktes Editieren möglich (True) / nicht möglich (False) **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = True
```

Code-Beispiel C#

```
vcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = true;
```

InPlaceEditingOnNodesInTableEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob zur Laufzeit das direkte Editieren von Knotendatenfeldern im Diagramm-Bereich 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.

S. ebenfalls **InPlaceEditingOnNodesInDiagramEnabled**, **InPlaceEditingOnGroupsInTableEnabled** und **InPlaceEditingOnGroupsInDiagramEnabled**.

	Datentyp	Beschreibung

Code-Beispiel VB.NET

```
VcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = True
```

Code-Beispiel C#

```
vcGantt1.InPlaceEditingOnGroupsInDiagramEnabled = true;
```

InteractionMode

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einen der verfügbaren Interaktionsmodi einstellen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcInteractionMode	Interaktionsmodus **Standardwert:** vcPointer
	Mögliche Werte: .vcCreateBox 36 .vcCreateLink 4 .vcCreateNode 2 .vcDeleteLink 5 .vcDeleteNode 3 .vcPanning 6 .vcPointer 0	 Erzeugemodus für Boxen Erzeugemodus für Verbindungen Erzeugemodus für Knoten Löschmodus für Verbindungen Löschmodus für Knoten Verschiebemodus Selektiermodus

Code-Beispiel VB.NET

```
VcGantt1.InteractionMode = VcInteractionMode.vcCreateNode
```

Code-Beispiel C#

```
vcGantt1.InteractionMode = VcInteractionMode.vcCreateNode;
```

KeepingNodesTogetherDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen bzw. erfragen, nach welchem Datenfeld Knoten in Gruppen ggf. separiert werden. Diese Eigenschaft kann nur angewendet werden, wenn die Knoten gruppiert werden und die Gruppierungsoptionen **In einer Zeile** und **Knoten optimiert anordnen** gewählt worden sind (Dialog **Gruppierung**). Dann können Sie für **KeepingNodesTogetherDataFieldIndex** ein Datenfeld auswählen. Alle Knoten einer Gruppe, die denselben Wert in diesem Datenfeld haben, werden

dann in derselben Zeile angeordnet, auch wenn es dabei zu Überlagerungen kommt.

Hinweis: Bitte beachten Sie, dass die Felder vom Datentyp **Integer** oder **Alphanumerisch** sein müssen und der Wertebereich zwischen 1 und long_MAX (2147483647) liegen muss, damit die Eigenschaft zufriedenstellende Ergebnisse liefert. Hat ein Feld den Wert 0, so wird der Knoten nicht mit anderen Knoten der Gruppe zusammengehalten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Nummer des Feldes, das für die Separierung von Knoten in Gruppen verwendet werden soll

Code-Beispiel VB.NET

```
VcGantt1.KeepingNodesTogetherDataFieldIndex = 3
```

Code-Beispiel C#

```
vcGantt1.KeepingNodesTogetherDataFieldIndex = 3;
```

LayerCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das LayerCollection-Objekt, in dem alle verfügbaren Layer zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcLayerCollection	LayerCollection-Objekt

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
layerCltn = VcGantt1.LayerCollection
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
```

LayersWithNonWorkInterval

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob arbeitsfreie Zeiten im Knoten dargestellt werden sollen oder nicht. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

Hinweis: NodesUseCalendars muss hierfür auf True gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Arbeitsfreie Intervalle anzeigen (True)/nicht anzeigen (False)

Code-Beispiel VB.NET

```
VcGantt1.LayersWithNonWorkInterval = True
```

Code-Beispiel C#

```
vcGantt1.LayersWithNonWorkInterval = true;
```

LeavingControlWhileDraggingAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie bestimmen, ob ein Ziehen von Vorgängen über die Grenzen der Quellkomponente hinaus erlaubt sein soll. Damit können Sie z.B. einen Knoten von einem VARCHART-Steuerelement

- zu einem andern VARCHART-Steuerelement
- in andere Steuerelemente der gleichen Anwendung oder sogar
- in andere Anwendungen

verschieben oder kopieren. Intern wird der Drag&Drop-Mechanismus in der Basisklasse **Control** ausgelöst, der das OLE-Drag&Drop des Windows-Betriebssystems auslöst.

S. auch die VcGantt-Eigenschaften **NodeCreationAtDroppingEnabled**, **InbuiltMouseCursorWhileDraggingEnabled**, **PhantomDrawingWhile-DraggingEnabled** und die **AllowDrop** Eigenschaft der Basisklasse **Control**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Das Verlassen der Quellkomponent ist erlaubt (true) / nicht erlaubt (false). **Standardwert:** false

LeftTable

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das Tabellenobjekt, um so auf die verwendeten Formate zugreifen und darin die Tabellenspalten oder ihre Überschriften verändern zu können.

	Datentyp	Beschreibung
Eigenschaftswert	VcTable	Tabelle

Code-Beispiel VB.NET

```
Dim table As VcTable
table = VcGantt1.LeftTable
```

Code-Beispiel C#

```
VcTable table = vcGantt1.LeftTable;
```

LeftTableDiagramWidthRatio

Eigenschaft von VcGantt

Mit dieser Eigenschaft lässt sich das Verhältnis der Breite der linken Tabelle zur Gesamtbreite des Diagramms (in %) erfragen oder festlegen. Beim Wert -1 wird die Tabelle immer komplett dargestellt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breitenverhältnis {-1, 1...100}

Code-Beispiel VB.NET

```
VcGantt1.LeftTableDiagramWidthRatio = 40
```

Code-Beispiel C#

```
vcGantt1.LeftTableDiagramWidthRatio = 40;
```

LeftTableDiagramWidthRatioEx

Eigenschaft von VcGantt

Mit dieser Eigenschaft lässt sich das Verhältnis der Breite der linken Tabelle zur Gesamtbreite des Diagramms (in %) erfragen oder festlegen. Beim Wert -1 wird die Tabelle immer komplett dargestellt.

Diese Eigenschaft unterscheidet sich von der Eigenschaft **LeftTable-DiagramWidthRatio** durch die Rückgabe des Datentyps "Double", mit dem eine höhere Genauigkeit erzielt wird. Die Verwendung dieser Eigenschaft muss zuvor über die Eigenschaft **UseHigherTableDiagramWidthRatio-Precision** oder über die Option **Höhere Genauigkeit für die Breiten-verhältnisse zwischenTabelle(n) und Diagramm** auf der Eigenschaftenseite **Allgemeines** aktiviert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Breitenverhältnis

Code-Beispiel VB.NET

```
VcGantt1.LeftTableDiagramWidthRatioEx = 40
```

Code-Beispiel C#

```
vcGantt1.LeftTableDiagramWidthRatioEx = 40;
```

LegendView

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das LegendView-Objekt, das die Legendenansicht des Diagramms definiert.

	Datentyp	Beschreibung
Eigenschaftswert	VcLegendView	LegendView-Objekt

Code-Beispiel VB.NET

```
Dim legendview As VcLegendView
legendview = VcGantt1.LegendView
legendview.Visible = True
```

Code-Beispiel C#

```
VcLegendView legendview = vcGantt1.LegendView;
legendview.Visible = true;
```

LineFormatCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die Linienformat-auflistung, in der alle zur Verfügung stehenden Linienformate enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcLineFormatCollection	LineFormatCollection-Objekt

Code-Beispiel VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection
Dim lineFormat As VcLineFormat

lineFormatCltn = VcGantt1.LineFormatCollection
For Each lineFormat In lineFormatCltn
   ListBox1.Items.Add(lineFormat.Name)
Next
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
foreach (VcLineFormat lineFormat in lineFormatCltn)
{
   listBox1.Items.Add(lineFormat.Name);
}
```

LinkAppearanceCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die LinkAppearanceCollection, in der alle zur Verfügung stehenden LinkAppearance-Objekte zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcLinkAppearanceCollection	LinkAppearanceCollection-Objekt

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.LinkAppearanceCollection;
```

LinkCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das LinkCollection-Objekt, in dem alle Verbindungen zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcLinkCollection	LinkCollection Objekt

Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
linkCltn = VcGantt1.LinkCollection
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.LinkCollection;
```

LinkPredecessorDataFieldIndex

Eigenschaft von VcGantt

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.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** gesetzt werden.

Die Eigenschaft LinkPredecessorDataFieldIndex ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_LinkPredecessorDataFieldIndex (identifierIndex, pvn) und get_LinkPredecessorDataFieldIndex (identifierIndex) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ identifierIndex	System.Int16	Index des Vorgängerknotens {0...2}
Eigenschaftswert	System.Int32	Feldindex aus der Datendefinitionstabelle

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcGantt1.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcGantt1.DataTableCollection.Update()

VcGantt1.LinkPredecessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")
VcGantt1.LinkSuccessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")

'Load Data
dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcGantt1.DataTableCollection.Add("LinkDataTable");
vcGantt1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcGantt1.DataTableCollection.Update();

vcGantt1.set_LinkPredecessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));
vcGantt1.set_LinkSuccessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcGantt1.EndLoading();
```

LinksDataTableName

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Namen der Datentabelle setzen oder erfragen, die die Datenfelder für die Verbindungen zur Verfügung stellt.

Damit die Verbindungen erscheinen können, müssen die Eigenschaften **LinkSuccessorDataFieldIndex** und **LinkPredecessor** ebenfalls gesetzt sein.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** gesetzt werden.

	Datentyp	**Beschreibung**
Eigenschaftswert	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 = VcGantt1.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcGantt1.DataTableCollection.Update()

VcGantt1.LinkPredecessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")
VcGantt1.LinkSuccessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")

'Load Data
dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcGantt1.DataTableCollection.Add("LinkDataTable");
vcGantt1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcGantt1.DataTableCollection.Update();

vcGantt1.set_LinkPredecessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));
vcGantt1.set_LinkSuccessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcGantt1.EndLoading();
```

LinkSuccessorDataFieldIndex

Eigenschaft von VcGantt

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.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** gesetzt werden.

Die Eigenschaft LinkSuccessorDataFieldIndex ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_LinkSuccessorData-

FieldIndex (identifierIndex, pvn) und get_LinkSuccessorDataFieldIndex (identifierIndex)angesprochen wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ identifierIndex	System.Int16	Index des Nachfolgerknotens {0...2}
Eigenschaftswert	System.Int32	Feldindex aus der Datendefinitionstabelle

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecord As VcDataRecord

'create Link DataTable
dataTable = VcGantt1.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
VcGantt1.DataTableCollection.Update()

VcGantt1.LinkPredecessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")
VcGantt1.LinkSuccessorDataFieldIndex(0) =
VcGantt1.DetectFieldIndex("LinkDataTable", "Id")

'Load Data
dataTable = VcGantt1.DataTableCollection.DataTableByName("LinkDataTable")
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;")
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Link DataTable
dataTable = vcGantt1.DataTableCollection.Add("LinkDataTable");
vcGantt1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
vcGantt1.DataTableCollection.Update();

vcGantt1.set_LinkPredecessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));
vcGantt1.set_LinkSuccessorDataFieldIndex(0,
vcGantt1.DetectFieldIndex("LinkDataTable", "Id"));

//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("LinkDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;1;2;");
vcGantt1.EndLoading();
```

LinkTypeDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können sie das Datenfeld setzen oder erfragen, das den Verbindungstyp enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das die Verbindungstypen enthält

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable

'create Link DataTable
dataTable = VcGantt1.DataTableCollection.Add("LinkDataTable")
VcGantt1.LinksDataTableName = dataTable.Name
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
dataTable.DataTableFieldCollection.Add("Predecessor")
dataTable.DataTableFieldCollection.Add("Successor")
dataTable.DataTableFieldCollection.Add("LinkType")
VcGantt1.DataTableCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable;

//create Link DataTable
dataTable = vcGantt1.DataTableCollection.Add("LinkDataTable");
vcGantt1.LinksDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
dataTable.DataTableFieldCollection.Add("Predecessor");
dataTable.DataTableFieldCollection.Add("Successor");
dataTable.DataTableFieldCollection.Add("LinkType");
vcGantt1.DataTableCollection.Update();
```

MapCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das MapCollection-Objekt, in dem eine bestimmte Menge von Zuordnungstabellen zusammengefasst ist. Die Menge der Maps wird durch die Methode **VcMapCollection.SelectMaps** definiert.

	Datentyp	Beschreibung
Eigenschaftswert	VcMapCollection	MapCollection-Objekt

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
```

MinimumRowHeight

Eigenschaft von VcGantt

Mit dieser Eigenschaft wird die minimale Höhe einer Zeile (Einheit: 1/100 mm) eingestellt. Die eingestellte Höhe sollte etwa der mittleren Höhe der Darstellung eines Vorgangs entsprechen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** 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	Minimale Zeilenhöhe

Code-Beispiel VB.NET

```
VcGantt1.MinimumRowHeight = 100
```

Code-Beispiel C#

```
vcGantt1.MinimumRowHeight = 100;
```

MouseProcessingEnabled

Eigenschaft von VcGantt

Diese Eigenschaft kann dazu genutzt werden, Ihre eigene Verarbeitung von Mausereignissen zu ermöglichen. Wenn Sie eine eigene Verarbeitung der .NET-Mausereignisse MouseDown/Up/Move durchführen möchten, setzen Sie die Eigenschaft **MouseProcessingEnabled** für diese Zeit auf False. Dann ignoriert VARCHART XGantt bis zum Zurücksetzen auf True alle Mausbewegungen und Klicks.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft (True)/ nicht in Kraft (False) **Standardwert:** True

MoveMode

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einstellen oder erfragen, in welche Richtung(en) ein Knoten interaktiv verschoben werden darf.

	Datentyp	Beschreibung

Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll)

For Each node In nodeCltn
  If node.DataField(2) < Now Then
     node.MoveMode = VcNodeMoveMode.vcNodeMoveModeNoMove
  End If
Next
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = VcGantt1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
foreach (VcNode node in nodeCltn)
   {
      if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
DateTime.Today).Equals(true))
         node.MoveMode = VcNodeMoveMode.vcNodeMoveModeNoMove;
   }
```

MovingLayersAsNodeWithShiftKeyAllowed

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob alle Layer eines markierten Knotens gleichzeitig mit der Maus verschoben werden können, wenn während des Verschiebens die Umschalttaste gedrückt wird (True) oder ob jeder Layer eines markierten Knotens nur einzeln verschoben werden kann (False). Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Verschieben von allen Layern eines Knotens mit Umschalttaste zugelassen/nicht zugelassen **Standardwert:** true

Code-Beispiel VB.NET

```
VcGantt1.MoveLayersAsNodeWithShiftKey = False
```

MultipleBoxMarkingAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Markieren von mehreren Boxen gleichzeitig zur Laufzeit möglich ist. Ist die Eigenschaft nicht gesetzt, muss man zum Markieren mehrerer Boxen die STRG-Taste gedrückt halten. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Mehrfachmarkierung von Boxen möglich/nicht möglich **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.MultipleBoxMarking = True
```

Code-Beispiel C#

```
vcGantt1.MultipleBoxMarking = true;
```

NodeCalendarNameDataFieldIndex

Eigenschaft von VcGantt

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.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das den Namen des für den Knoten zu verwendenden Kalenders enthält

NodeCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das NodeCollection-Objekt, in dem eine bestimmte Menge von Knoten zusammengefasst ist. Diese Menge der Knoten wird durch die Methode **VcNodeCollection.SelectNodes** definiert.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeCollection	NodeCollection-Objekt

Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll)
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
```

NodeCreationAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft wird dem Anwender das Anlegen neuer Vorgänge erlaubt (True) oder verboten (False). Wird diese Eigenschaft auf False gesetzt, kann der Interaktionsmodus **Knoten erzeugen** nicht eingeschaltet werden.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Neue Knoten zugelassen/nicht zugelassen **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.NodeCreationAllowed = False
```

Code-Beispiel C#

```
vcGantt1.NodeCreationAllowed = false;
```

NodeCreationAtDroppingEnabled

Eigenschaft von VcGantt

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

Wenn Sie diese Eigenschaft auf **false** setzen, muss ein Event-handler für das DragDrop-Ereignis geschrieben werden. Diese Eigenschaft und das erwähnte Ereignis sind nur dann aktiv, wenn vorher **AllowDrop** der Basisklasse **Control** auf **true** gesetzt wurde.

S. auch die VcGantt-Eigenschaften **LeavingControlWhileDragging-Allowed**, **InbuiltMouseCursorWhileDraggingEnabled**, **Phantom-DrawingWhileDraggingEnabled** und die **AllowDrop** Eigenschaft der Basisklasse **Control**.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Boolean	Ein Vorgang soll erzeugt (true) / nicht erzeugt (false) werden. **Standardwert:** false

NodeCreationViaDoubleclick

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie steuern, ob der Anwender neue Knoten per Doppelklick (im Diagrammbereich) anlegen kann oder nicht. Voraussetzung ist, dass die Eigenschaft **NodeCreationAllowed** auf True gesetzt ist. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** eingestellt werden.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Boolean	Anlegen neuer Knoten per Doppelklick zugelassen/nicht zugelassen **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.NodeCreationViaDoubleclick = True
```

Code-Beispiel C#

```
vcGantt1.NodeCreationViaDoubleclick = true;
```

NodeCreationWithDialog

Eigenschaft von VcGantt

Mit dieser Eigenschaft wird festgelegt, ob bei der Erzeugung eines neuen Knotens das Dialogfeld **Vorgänge bearbeiten** erscheinen soll oder nicht. Die Eigenschaft **NodeCreationAllowed** muss auf **True** gesetzt sein, damit überhaupt ein neuer Knoten erzeugt werden kann. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**Vorgänge bearbeiten**-Dialogfeld erscheint /erscheint nicht **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.NodeCreationWithDialog = False
```

Code-Beispiel C#

```
vcGantt1.NodeCreationWithDialog = false;
```

NodeDurationDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das die Dauer eines interaktiv angelegten Knotens enthält. Dies ist nur möglich, solange noch keine Daten geladen wurden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes,, das die Dauer eines interaktiv angelegten Knotens enthält

NodeEndDateDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen/erfragen, in das der Endtermin eines interaktiv angelegten Vorgangs geschrieben werden soll. Dies ist nur möglich, solange noch keine Daten geladen wurden. Diese Eigenschaft kann auch auf der Eigenschaften-seite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes,, das den Endtermin eines interaktiv angelegten Vorgangs enthält

NodeLevelLayout

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf das NodeLevelLayout-Objekt. Mit diesem Objekt können Sie die Eigenschaften einer hierarchischen Anordnung setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeLevelLayout	NodeLevelLayout-Objekt **Standardwert:** True

NodeRowNumberDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen/erfragen, in das die Zeilennummer jedes Vorgangs geschrieben werden soll. Dies ist nur möglich, solange noch keine Daten geladen wurden. Damit Änderungen wirksam werden, muss eine Aktualisierung über die Methode **VcGantt.UpdateRowNumberFields** vorgenommen werden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	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
   VcGantt1.NodeRowNumberDataFieldIndex =
VcGantt1.DetectFieldIndex("NodeDataTable", "SortNumber")
    'Load data
    LoadData()

    VcGantt1.UpdateRowNumberFields()
    VcGantt1.SaveAsEx("C:\ProjectData.txt",  VcEncoding.vcUnicodeEncoding)
End Sub
```

Code-Beispiel C#

```
private void Form1_Load(object sender, System.EventArgs e)
{
   vcGantt1.NodeRowNumberDataFieldIndex =
VcGantt1.DetectFieldIndex("NodeDataTable", "SortNumber");

   // Load data
   loadData();

    vcGantt1.UpdateRowNumberFields();
    vcGantt1.SaveAsEx(@"C:\ProjectData.txt", VcEncoding.vcUnicodeEncoding);
 }
```

NodesDataTableName

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Namen der Datentabelle setzen oder erfragen, die die Datenfelder für die Darstellung der Knoten zur Verfügung stellt. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	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 = VcGantt1.DataTableCollection.Add("NodeDataTable")
        VcGantt1.NodesDataTableName = dataTable.Name
        dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = True
```

Code-Beispiel C#

```
VcDataTable dataTable;
VcDataRecord dataRecord;

//create Node DataTable
dataTable = vcGantt1.DataTableCollection.Add("NodeDataTable");
vcGantt1.NodesDataTableName = dataTable.Name;
dataTable.DataTableFieldCollection.Add("Id").PrimaryKey = true;
//Load Data
dataTable = vcGantt1.DataTableCollection.DataTableByName("NodeDataTable");
dataRecord = dataTable.DataRecordCollection.Add("1;Node One;");
dataRecord = dataTable.DataRecordCollection.Add("2;Node Two;");
vcGantt1.EndLoading();
```

NodeSortingDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Sortierfelder für Ihre Knoten festlegen. Es gibt drei Sortierungsebenen, für die jeweils ein Feldindex der Datendefinitionstabelle angegeben werden kann. Die Sortierungsrichtung wird über die Eigenschaft **NodeSortingOrder** eingestellt. Die Sortierung wird durch die Methode **SortNodes** ausgelöst.

Die Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

Die Eigenschaft NodeSortingDataFieldIndex ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_NodeSortingDataFieldIndex (sort-Level, pvn) und get_NodeSortingDataFieldIndex (sortLevel) angesprochen wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sortLevel	System.Int16	Sortierebene {0...2}
Eigenschaftswert	System.Int32	Feldindex aus der Datendefinitionstabelle

Code-Beispiel VB.NET

```
VcGantt1.NodeSortingDataFieldIndex(0) = 11
VcGantt1.NodeSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcGantt1.SortNodes()
```

Code-Beispiel C#

```
vcGantt1.set_NodeSortingDataFieldIndex(0,11);
vcGantt1.set_NodeSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcGantt1.SortNodes();
```

NodeSortingOrder

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie für jede der drei Sortierebenen die Sortierrichtung festlegen. Die Sortierung wird durch die Methode **SortNodes** ausgelöst. Sie können diese Eigenschaft auch im Dialog **Gruppierung** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	Integer	Aufsteigende oder absteigende Sortierreihenfolge **Standardwert:** vcAscending
	Mögliche Werte: .vcAscending 1 .vcDescending 2	 aufsteigende Abfolge absteigende Abfolge

Code-Beispiel VB.NET

```
VcGantt1.NodeSortingDataFieldIndex(0) = 11
VcGantt1.NodeSortingOrder(0) = VcNodeSortingOrder.vcDescending
VcGantt1.SortNodes()
```

Code-Beispiel C#

```
vcGantt1.set_NodeSortingDataFieldIndex(0,11);
vcGantt1.set_NodeSortingOrder(0, VcNodeSortingOrder.vcDescending);
vcGantt1.SortNodes();
```

NodeStartDateDataFieldIndex

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das Datenfeld festlegen/erfragen, in das der Starttermin eines interaktiv angelegten Vorgangs geschrieben werden soll. Dies ist nur möglich, solange noch keine Daten geladen wurden. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das den Starttermin eines interaktiv angelegten Vorgangs enthält

NodesUseCalendars

Eigenschaft von VcGantt

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** festlegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft/nicht in Kraft

Code-Beispiel VB.NET

```
VcGantt1.NodesUseCalendars = False
```

Code-Beispiel C#

```
vcGantt1.NodesUseCalendars = false;
```

NodeToolTipTextDataFieldIndex

Eigenschaft von VcGantt

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
Eigenschaftswert	System.Int16	Index des Knoten-Datenfelds für Tooltiptexte **Standardwert:** 4

Code-Beispiel VB.NET

```
VcGantt1.NodeToolTipTextDataFieldIndex = 1
```

Code-Beispiel C#

```
vcGantt1.NodeToolTipTextDataFieldIndex = 1;
```

NumericScaleCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das NumericScaleCollection-Objekt, in dem alle zur Verfügung stehenden numerischen Skalen zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcNumericScaleCollection	NumericScaleCollection-Objekt

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim numericScaleCltn As VcNumericScaleCollection
histogramCltn = VcGantt1.HistogramCollection
numericScaleCltn = histogram.FirstHistogram.NumericScaleCollection
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcNumericScaleCollection numericScaleCltn =
histogramCltn.FirstHistogram().NumericScaleCollection;
```

NumericScaleRescalingAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen bzw. erfragen, ob die numerische Skala zur Laufzeit reskalierbar sein soll.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Numerische Skala reskalierbar (True)/ nicht reskalierbar (False)

Code-Beispiel VB.NET

```
VcGantt1.NumericScaleRescalingAllowed = True
```

Code-Beispiel C#

```
vcGantt1.NumericScaleRescalingAllowed = true;
```

OLEDragViaDiagram

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob OLE-DragDrop im Diagrammbereich erlaubt sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	OLE-DragDrop im Diagramm erlaubt/nicht erlaubt **Standardwert:** True

OLEDragViaTable

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob OLE-DragDrop im Tabellenbereich erlaubt sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	OLE-DragDrop in Tabelle erlaubt/nicht erlaubt **Standardwert:** True

OverlapLayerEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Überlappungs-Layer des Diagramms aktivieren. Bitte sehen Sie auch **OverlapLayerName** sowie beim Layer-Objekt **UsedAsOverlap Layer**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Überlappungslayer ein (True) / aus (False) **Standardwert:** False

OverlapLayerName

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Layer, der im Diagramm als Überlappungs-Layer dienen soll, über seinen Namen festlegen oder erfragen. Der Überlappungs-Layer muss vorher beim Layer-Objekt angelegt und mittels **UsedAsOverlap Layer** als Überlappungs-Layer gekennzeichnet werden sowie schließlich über **OverlapLayerEnabled** beim Gantt-Objekt aktiviert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Überlappungslayers **Standardwert:** " "

PanningModeAllowed

Eigenschaft von VcGantt

Durch diese Eigenschaft kann ein Bildschirmausschnitt unterhalb eines Handcursors verschoben werden.

Die Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Bildschirmausschnitt verschieben erlaubt (True)/nicht erlaubt (False) **Standardwert:** False

PartialLoadThreshold

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen, oder erfragen ab welcher Knotenzahl von einem optimierten Teilupdate auf ein komplettes Update der Datensätze umgeschaltet wird.

Standardmäßig wird beim Hinzufügen eines neuen Datensatzes ein Ladezyklus gestartet, der intern für das Laden großer Datenmengen optimiert ist: Strukturen wie Gruppierung und Sortierung, Summenbalkenberechnungen usw. werden verworfen und komplett neu erstellt. Dies ist beim Laden großer Datenmengen in einen leeren Plan vorteilhaft, beim Nachladen weniger Datensätze in eine schon bestehende Datenstruktur jedoch kann damit im Extremfall das Nachladen eines einzigen Knotens in einen vorhandenen Plan mit z.B. 4000 Vorgängen, 500 Gruppen, sortierten Knoten, berechneten Summenbalken, positionierten Links usw. einen kompletten Neuaufbau dieser Strukturen bewirken. Damit würde das Nachladen des einen Knotens genauso lange dauern wie das Laden der 4000 Vorgänge, da der Aufbau dieser Strukturen den Hauptteil der Performance ausmacht.

Die Eigenschaft **PartialLoadTreshold** bietet hier Abhilfe: Eine kleine Datenmenge wird durch ein Teilupdate in eine große vorhandene Datenmenge optimiert eingefügt. Der Wert, der hier angegeben wird, stellt den Schwellenwert für die Anzahl von Datensätzen dar, ab dem von einem"kleinem" auf ein "großes" Update umgeschaltet wird.

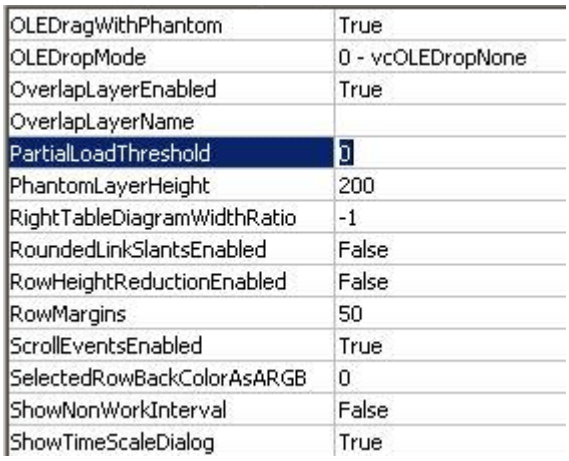
Der optimale Schwellenwert hängt von verschiedenen Faktoren in der jeweiligen Applikation ab und muss vom Anwender individuell ermittelt werden:

- Anzahl der vorhanden Knoten,

- Komplexität des Gantts (z.B. Gruppierung, Sortierung, Summenbalken, Links, Mapping usw.)

Die Eigenschaft sollte in erster Linie genutzt werden, wenn bereits viele Knoten vorhanden sind, und man zur Laufzeit noch einige wenige hinzufügen möchte.

Diese Eigenschaft kann auch bei den Eigenschaften des Steuerelementes gesetzt werden:



Hinweis. Die Optimierung beschränkt sich zur Zeit noch auf die **Maindata**-Tabelle. Werden also in einem Ladezyklus auch Daten anderer Tabellen oder Verbindungen geladen, wird die Einstellung ignoriert.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int32	Anzahl der Knoten, ab der von Teilupdate auf Vollupdate umgeschaltet wird **Standardwert:** 0

PhantomDrawingWhileDraggingEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie in der Zielkomponente festlegen, ob beim Ziehen das für das VARCHART-Steuerelement übliche Phantom angezeigt wirden soll. Dies ist nur möglich, wenn die beim Ziehen übergebenen Daten im CSV-Textformat vorliegen und inhaltlich der Datendefinition in der Zielkomponente entsprechen.

Die übergebenen Daten befinden sich im DataObject, das von den Ereignissen **DragEnter** und **DragOver** der Basisklasse **Control** übergeben wird.

S. auch die VcGantt-Eigenschaften **LeavingControlWhileDragging-Allowed**, **NodeCreationAtDroppingEnabled**, **InbuiltMouseCursorWhile-DraggingEnabled** sowie die Eigenschaft **AllowDrop** der Basisklasse **Control**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Das Phantom soll angezeigt (true) / nicht angezeigt (false) werden. **Standardwert:** false

PhantomLayerHeight

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Höhe eines Layer-Phantoms beim Aufziehen eines Knotens in 1/100 mm einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Höhe des Layerphantoms

Code-Beispiel VB.NET

```
Dim phantomLayerHeight As Integer
phantomLayerHeight = VcGantt1.PhantomLayerHeight
```

Code-Beispiel C#

```
int phantomLayerHeight = vcGantt1.PhantomLayerHeight;
```

Printer

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf das Printer-Objekt. Mit diesem Objekt können Sie die Eigenschaften des aktuell verwendeten Druckers erfragen oder setzen.

	Datentyp	Beschreibung
Eigenschaftswert	VcPrinter	Druckerobjekt

Code-Beispiel VB.NET

```
Dim printerZoomfactor As Integer
Dim printerCuttingMarks As String
printerZoomfactor = VcGantt1.Printer.ZoomFactor
printerCuttingMarks = VcGantt1.Printer.CuttingMarks
```

Code-Beispiel C#

```
int printerZoomfactor = vcGantt1.Printer.ZoomFactor;
bool printerCuttingMarks = vcGantt1.Printer.CuttingMarks;
```

ResourceScheduler2

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das ResourceScheduler2-Objekt für die Ressourcenplanung.

	Datentyp	Beschreibung
Eigenschaftswert	VcResourceScheduler2	ResourceScheduler2-Objekt

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskDataTableName = "Task"
VcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 1
VcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2
 '...
VcGantt1.ResourceScheduler2.Process()
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskDataTableName = "Task";
vcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 1;
vcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2;
//...
vcGantt1.ResourceScheduler2.Process();
```

RightTable

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das zweite, rechtsseitige Tabellenobjekt, um auf dessen verwendete Formate zugreifen oder die Tabellenspalten bzw. -überschriften verändern zu können.

	Datentyp	Beschreibung
Eigenschaftswert	VcTable	Zweite, rechtsseitige Tabelle

Code-Beispiel VB.NET

```
Dim rightTable As VcTable
rightTable = VcGantt1.RightTable
```

Code-Beispiel C#

```
VcTable rightTable = vcGantt1.RightTable;
```

RightTableDiagramWidthRatio

Eigenschaft von VcGantt

Mit dieser Eigenschaft lässt sich das Verhältnis der Breite der rechten Tabelle zur Gesamtbreite des Diagramms (in %) erfragen oder festlegen. Beim Wert -1 wird die Tabelle immer komplett dargestellt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breitenverhältnis {-1, 1...100}

Code-Beispiel VB.NET

```
VcGantt1.RightTableDiagramWidthRatio = 40
```

Code-Beispiel C#

```
vcGantt1.RightTableDiagramWidthRatio = 40;
```

RightTableDiagramWidthRatioEx

Eigenschaft von VcGantt

Mit dieser Eigenschaft lässt sich das Verhältnis der Breite der rechten Tabelle zur Gesamtbreite des Diagramms (in %) erfragen oder festlegen. Beim Wert -1 wird die Tabelle immer komplett dargestellt.

Diese Eigenschaft unterscheidet sich von der Eigenschaft **RightTableDiagramWidthRatio** durch die Rückgabe des Datentyps "Double", mit dem eine höhere Genauigkeit erzielt wird. Die Verwendung dieser Eigenschaft muss zuvor über die Eigenschaft **UseHigherTableDiagramWidthRatioPrecision** oder über die Option **Höhere Genauigkeit für die Breitenverhältnisse zwischenTabelle(n) und Diagramm** auf der Eigenschaftenseite **Allgemeines** aktiviert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Breitenverhältnis

Code-Beispiel VB.NET

```
VcGantt1.RightTableDiagramWidthRatioEx = 40
```

RoundedLinkSlantsEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob die Schrägen bei Verbindungen vom Routing-Typ **vcLRTOrthogonalDistinguishable** als Viertelkreise statt als gerade Linien dargestellt werden sollen. Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Schrägen bei Verbindungen werden/werden nicht als Viertelkreise dargestellt **Standardwert:** false

Code-Beispiel VB.NET

```
VcGantt1.RoundedLinkSlantsEnabled = True
```

Code-Beispiel C#

```
vcGantt1.RoundedLinkSlants.Enabled = true;
```

RowHeightReductionEnabled

Eigenschaft von VcGantt

Diese Eigenschaft beeinflusst das Verfahren zur Berechnung der Zeilenhöhe im Diagramm. Ist sie auf **false** gesetzt, werden die vertikalen Offsets der Layer ausgehend von einer gedachten Nulllinie in der vertikalen Mitte einer Knotenzeile wirksam. Damit diese Nulllinie stets in der Mitte einer Zeile bleibt, kann so entweder der obere oder der untere Rand einer Zeile sehr groß

erscheinen, wobei aber Layer mit vertikalem Offset von 0 immer vertikal zentriert bleiben.

Wenn die Eigenschaft auf **true** gesetzt ist, wird zwar weiterhin die gedachte Nulllinie verwendet, diese befindet sich jedoch nicht mehr zwingend in der Zeilenmitte, sondern wird so positioniert, dass für die Zeilenhöhe das kleinstmögliche Maß verwendet werden kann. Dadurch kann es vorkommen dass Layer mit einem vertikalen Offset von 0 nicht mehr auf gleicher Höhe liegen wie der vertikal zentrierte Text der zugehörigen Tabellenzeile.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Reduzierung der Zeilenhöhe erlaubt (true)/nicht erlaubt (false) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.RowHeightReductionEnabled = True
```

Code-Beispiel C#

```
vcGantt1.RowHeightReductionEnabled = true;
```

RowMargins

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Abstand zwischen dem oberen/ unteren Knotenrand und dem oberen/ unteren Rand der Knotenzeilen in 1/100 mm festlegen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Abstand zwischen dem oberen/ unteren Knotenrand und dem oberen/ unteren Rand der Knotenzeilen in 1/100 mm

Code-Beispiel VB.NET

```
VcGantt1.RowMargins = 100
```

Code-Beispiel C#

```
vcGantt1.RowMargins = 100
```

Sash3DStyleEnabled

Eigenschaft von VcGantt

Mithilfe dieser Eigenschaft kann gesetzt oder erfragt werden, ob der Sash im 3D-Modus angezeigt wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	3D-Ansicht Sash an/aus **Standardwert:** True

SashThickness

Eigenschaft von VcGantt

Mithilfe dieser Eigenschaft kann die Dicke des Sashs gesetzt oder erfragt werden. Wertebereich: 3 - 20 Pixe

Die Eigenschaft SashThickness ist eine indizierte Eigenschaft, die in C# über die Methode get_SashThickness (gridIndex) angesprochen wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Integer	Dicke des Sash ändern **Standardwert:** 4

Scheduler

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft gibt das VcScheduler-Objekt zurück.

	Datentyp	Beschreibung
Eigenschaftswert	VcScheduler	Gibt das VcScheduler-Objekt zurück

ScrollEventsEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Scroll Events **VcComponentScrolled, VcComponentScrolling, VcDiagramHorizontalScrolled** und **VcDiagram-**

HorizontalScrolling an- oder abschalten. Dies kann auch auf der Eigenschaftenseite **Allgemeines** geschehen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Scroll-Ereignisse ein- (True) oder ausgeschaltet (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.ScrollEventsEnabled = True
```

Code-Beispiel C#

```
vcGantt1.ScrollEventsEnabled = true;
```

SelectedNodesMovingTogether

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob der Anwender mehrere markierte Knoten gemeinsam verschieben können soll. Andernfalls können nur einzelne Layer oder Knoten (je nachdem, ob auf der Eigenschaftenseite **Knoten**die Option **Alle Layer gemeinsam verschieben** ausgewählt wurde oder nicht) bewegt werden, auch wenn mehrere Knoten markiert sind).

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Mehrere markierte Knoten gemeinsam verschiebbar (True)/nur einzelne Layer bzw. Knoten verschiebbar, auch wenn mehrere Knoten markiert sind (False)

Code-Beispiel VB.NET

```
VcGantt1.SelectedNodesMovingTogether = True
```

Code-Beispiel C#

```
vcGantt1.SelectedNodesMovingTogether = true;
```

SelectedRowBackgroundColor

Eigenschaft von VcGantt

Diese Eigenschaft gibt die Farbe an, mit der die gerade selektierte Zeile eingefärbt wird. Sie können bei dieser Farbe auch einen Alpha-Wert

verwenden, der die Transparenz für die Färbung einstellt, um z.B. einen Farbschleier über die normale Hintergrundfarbe der Zeile zu legen (s. Eigenschaften DiagramBackgroundColor und DiagramAlternatingRow-BackgroundColor). Standardmäßig ist das Einfärben abgeschaltet, indem ein volltransparenter Farbwert vorgegeben ist. Die Eigenschaft kann auch auf der Eigenschaftenseite **Layout** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255}) **Standardwert:** Color.FromArgb(0,0,0,0)

SelectionViaRubberRectAllowed

Eigenschaft von VcGantt

Mihilfer dieser Eigenschaft lässt sich bestimmen, ob im leeren Diagrammbereich zur Selektion von Knoten ein Gummirechteck erscheint, wenn die Maus mit gedrückter linker Maustaste bewegt wird.

Die Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Markieren per Gummirechteckt erlaubt (True)/nicht erlaubt (False) **Standardwert:** False

ShowSnapLines

Eigenschaft von VcGantt

Mit dieser Eigenschaft legen Sie fest, ob beim Verschieben mit definierten Einrastzielen entprechende Linien gezeigt werden, damit der/die definierten Einrastziele besser zu erkennen sind.

Diese Funktionalität kann auch auf der Eigenschaftenseite **Knoten** aktiviert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Einrastlinien werden/werden nicht angezeigt **Standardwert:** false

ShowSnapMarkings

Eigenschaft von VcGantt

Mit dieser Eigenschaft legen Sie fest, ob beim Verschieben mit definierten Einrastzielen an den als Ziel definierten Knoten eine Markierung gezeigt wird, damit der/die definierten Einrastziele besser zu erkennen sind.

Diese Funktionalität kann auch auf der Eigenschaftenseite **Knoten** aktiviert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Einrastmarkierungen werden/werden nicht angezeigt **Standardwert:** false

SnapTargetNodesSelectionMode

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft legen Sie fest, ob beim Verschieben mit definierten Einrastzielen Knoten automatisch oder manuell selektiert werden. Mit der Eigenschaft **VcNode.SnapTargetMode** werden die Knoten bei manueller Selektion als mögliche Einrastziele ausgewählt.

	Datentyp	Beschreibung
Eigenschaftswert	VcSnapTargetNodesSelectionMode	Selektionsmodus für Knoten beim Verschieben mit Einrastzielen **Standardwert:** vcAutomatically

StartDateForAutomaticScheduling

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie für Autoschedule (Eigenschaftenseite **Zeitrechnung**) das Anfangsdatum für die Zeitrechnung des aktuellen Projekts angeben bzw. erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Startdatum

SubRowMargins

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den vertikalen Abstand von Subzeilen in 1/100 mm festlegen. Diese Subzeilen existieren nur, wenn Gruppen optimiert dargestellt werden und daraus Knoten in einer jeweiligen Gruppenzeile in mehrere Subzeilen verteilt werden, um nicht zu überlappen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Abstand zwischen Subzeilen in 1/100 mm ({0...200} **Standardwert:** 50

Code-Beispiel VB.NET

```
VcGantt1.SubRowMargins = 100
```

SummaryBarsVisible

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Summenbalken sichtbar sind oder nicht.

Die Eigenschaft SummaryBarsVisible ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_SummaryBarsVisible (groupingLevel, pvn) und get_SummaryBarsVisible (groupingLevel) angesprochen wird.

	Datentyp	Beschreibung
Parameter: ⇨ groupingLevel	System.Int16	*(nicht für Hierarchie)* Gruppierungsebene (GroupingLevel = -1: beim Setzen: alle Ebenen, beim Lesen: mindestens eine Ebene)
Eigenschaftswert	System.Boolean	Summenbalken sichtbar (True)/ unsichtbar (False)

Code-Beispiel VB.NET

```
VcGantt1.SummaryBarsVisible(-1) = True
```

Code-Beispiel C#

```
vcGantt1.set_SummaryBarsVisible(-1, true);
```

TableCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf das TableCollection-Objekt, in dem alle zur Verfügung stehenden Tabellen enthalten sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcTableCollection	Ermitteltes Tabellen-Collection-Objekt

Code-Beispiel VB.NET

```
Dim tableCltn As VcTableCollection
Dim table As VcTable

tableCltn = VcGantt1.TableCollection
For Each table In tableCltn
   ListBox1.Items.Add(table.Name)
Next
```

Code-Beispiel C#

```
VcTableCollection tableCltn = vcGantt1.TableCollection;
foreach(VcTable table in tableCltn)
   listBox1.Items.Add(table.Name);
```

TableColumnWidthOptimizationAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie dem Anwender erlauben, die Breite der Tabellenspalten automatisch optimieren zu lassen. Die Optimierung einer Tabellenspalte wird ausgelöst, wenn der Anwender auf die Trennlinie zwischen der zu optimierenden Spalte und der Spalte rechts davon doppelklickt. Es wird anschließend das Ereignis **VcTableColumnWidth-Optimizing** ausgelöst. Wenn die Veränderung der Spaltenbreite erfolgt ist, wird noch das Ereignis **VcTableColumnWidthChanging** ausgelöst.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Optimierung der Spaltenbreite zugelassen/nicht zugelassen **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.TableColumnWidthOptimizationAllowed = False
```

Code-Beispiel C#

```
vcGantt1.TableColumnWidthOptimizationAllowed = false;
```

TextEntrySupplyingEventEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das Auftreten des Ereignisses **VcTextEntrySupplying** aktivieren bzw. deaktivieren. Mit Hilfe dieses Ereignisses können Sie die Texte aller Kontextmenüs, Dialogfelder, Infoboxen, Fehlermeldungen und Monats- und Tagesnamen, die zur Laufzeit erscheinen, verändern, beispielsweise um sie in unterschiedliche Sprachen zu übersetzen.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft/nicht in Kraft

Code-Beispiel VB.NET

```
VcGantt1.TextEntrySupplyingEventEnabled = True
```

Code-Beispiel C#

```
vcGantt1.TextEntrySupplyingEventEnabled = false;
```

TimeScaleCollection

Nur-Lese-Eigenschaft von VcGantt

Diese Eigenschaft ermöglicht den Zugriff auf das TimeScaleCollection-Objekt und damit auf die verfügbaren Zeitskalen.

	Datentyp	Beschreibung
Eigenschaftswert	VcTimeScaleCollection	TimeScaleCollection-Objekt

Code-Beispiel VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection
timeScaleCltn = VcGantt1.TimeScaleCollection
```

Code-Beispiel C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
```

TimeScaleDialogEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob für den Anwender bei einem Doppelklick auf die Zeitskala das Dialogfeld **Zeitskala bearbeiten** erscheinen soll. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft (True)/ nicht in Kraft (False) **Standardwert:** true

Code-Beispiel VB.NET

```
VcGantt1.TimeScaleDialogEnabled = False
```

Code-Beispiel C#

```
vcGantt1.TimeScaleDialogEnabled = false;
```

TimeScaleEnd

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das Ende der Zeitskala erfragen oder festlegen. Bei der Festlegung muss das Ende immer nach dem Anfang (s. Eigenschaft **TimeScaleStart**) liegen, die Eigenschaften müssen aber in umgekehrter Reihenfolge gesetzt werden. Es ist empfehlenswert, die vorgegebene Reihenfolge im Programmierbeispiel hier zu verwenden.

Hinweis: Das Enddatum ist nicht eingeschlossen. Geben Sie beispielsweise das Enddatum "31.12.02" an, ist der letzte Tag, der dargestellt wird, der 30.12.02.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Enddatum der Zeitskala {1.1.1980...31.12.2035}

Code-Beispiel VB.NET

```
'Timescale from 01.10.2014 to 30.11.2014
VcGantt1.TimeScaleEnd = "01.12.2014"
VcGantt1.TimeScaleStart = "01.10.2014"
VcGantt1.TimeScaleEnd = "01.12.2014"
```

Code-Beispiel C#

```
//Timescale from 01.10.2014 to 30.11.2014
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
vcGantt1.TimeScaleStart = Convert.ToDateTime("01.10.14");
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
```

TimeScaleRescalingAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie dem Anwender erlauben, die Zeitskala interaktiv zu skalieren. Wenn der Anwender die Zeitskala skalieren darf, wird nach dem Skalieren das Ereignis **VcTimeScaleSectionRescaling** ausgelöst.

Diese Eigenschaft können Sie auch auf der Eigenschaftenseite **Allgemeines** setzen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Boolean	Reskalierung zugelassen/nicht zugelassen **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.TableColumnWidthOptimizationAllowed = False
```

Code-Beispiel C#

```
vcGantt1.TableColumnWidthOptimizationAllowed = false;
```

TimeScaleStart

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie den Anfang der Zeitskala erfragen oder festlegen. Bei der Festlegung muss der Anfang immer vor dem Ende (s. Eigenschaft **TimeScaleEnd**) liegen, die Eigenschaften müssen aber in umgekehrter Reihenfolge gesetzt werden. Es ist empfehlenswert, die vorgegebene Reihenfolge im Programmierbeispiel hier zu verwenden.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.DateTime	Anfangsdatum der Zeitskala {1.1.1980...31.12.2035}

Code-Beispiel VB.NET

```
'Timescale from 01.10.2014 to 30.11.2014
VcGantt1.TimeScaleEnd = "01.12.2014"
VcGantt1.TimeScaleStart = "01.10.2014"
VcGantt1.TimeScaleEnd = "01.12.2014"
```

Code-Beispiel C#

```
//Timescale from 01.10.2014 to 30.11.2014
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
vcGantt1.TimeScaleStart = Convert.ToDateTime("01.10.14");
vcGantt1.TimeScaleEnd = Convert.ToDateTime("01.12.14");
```

TimeUnit

Eigenschaft von VcGantt

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**
Eigenschaftswert	VcTimeUnit	Zeiteinheit **Standardwert:** vcDay
	Mögliche Werte: .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	 Zeiteinheit **Tag** Zeiteinheit **Stunde** Zeiteinheit **Minute** Zeiteinheit **Sekunde**

Code-Beispiel VB.NET

```
Dim timeUnit As VcTimeUnit
timeUnit = VcGantt1.TimeUnit
```

Code-Beispiel C#

```
VcTimeUnit timeUnit = vcGantt1.TimeUnit;
```

TimeUnitsPerStep

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie festlegen, wie viele Zeiteinheiten die kleinstmögliche interaktive Veränderung eines Knotens beträgt. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** unter **Kleinstes Zeitintervall** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Zeiteinheiten pro Schritt **Standardwert:** 1

Code-Beispiel VB.NET

```
VcGantt1.TimeUnitsPerStep = 4
```

Code-Beispiel C#

```
vcGantt1.TimeUnitsPerStep = 4;
```

ToolTipChangeDuration

Eigenschaft von VcGantt

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 ein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Zeitdauer in Millisekunden. Maximalwert = 32767 ms **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ToolTipChangeDuration = 1000
```

Code-Beispiel C#

```
vcGantt1.ToolTipChangeDuration = 1000;
```

ToolTipDuration

Eigenschaft von VcGantt

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
Eigenschaftswert	System.Int32	Zeitdauer in Millisekunden. Maximalwert = 32767 ms **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ToolTipDuration = 1000
```

Code-Beispiel C#

```
vcGantt1.ToolTipDuration = 1000;
```

ToolTipPointerDuration

Eigenschaft von VcGantt

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 ein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Zeitdauer in Millisekunden **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ToolTipPointerDuration = 1000
```

Code-Beispiel C#

```
vcGantt1.ToolTipPointerDuration = 1000;
```

ToolTipShowAfterClick

Eigenschaft von VcGantt

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) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.ToolTipShowAfterClick = True
```

Code-Beispiel C#

```
vcGantt1.ToolTipShowAfterClick = true;
```

ToolTipTextSupplyingEventEnabled

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie das Ereignis **VcToolTipTextSupplying** aktivieren oder deaktivieren. Mit Hilfe dieses Ereignisses können Sie die Texte der Tooltips verändern. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** festgelegt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ Rückgabewert	System.Boolean	Eigenschaft in Kraft/nicht in Kraft
Eigenschaftswert	System.Boolean	ToolTipTextSupplying-Ereignis ein- (True) oder ausgeschaltet (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.ToolTipTextSupplyingEventEnabled = True
```

Code-Beispiel C#

```
vcGantt1.ToolTipTextSupplyingEventEnabled = true;
```

TrackingSpaceBackgroundColor

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Trackingbereichs setzen oder erfragen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

Code-Beispiel VB.NET

```
VcGantt1.TrackingSpaceBackgroundColor = System.Drawing.Color.Blue
```

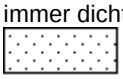
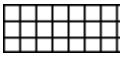
Code-Beispiel C#

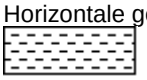
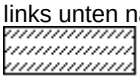
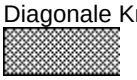
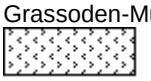
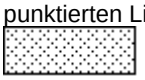
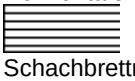
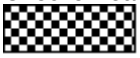
```
vcGantt1.DiagramTrackingSpaceColor = System.Drawing.Color.Blue;
```

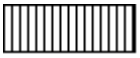
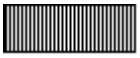
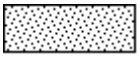
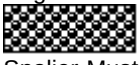
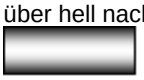
TrackingSpacePattern

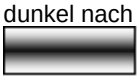
Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie für den Hintergrund desTrackingbereichs ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte: .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 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 

TrackingSpacePatternColor

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Musterfarbe des Trackingbereichs 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**
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255})

UpdateBehaviorCollection

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft haben Sie Zugriff auf die UpdateBehavior-Auflistung, in der alle zur Verfügung stehenden Aktualisierungsverhalten zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcUpdateBehaviorCollection	UpdateBehaviorCollection-Objekt

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
updBehCltn = VcGantt1.UpdateBehaviorCollection
```

Code-Beispiel C#

```
VcUpdateBehviorCollection updBehCltn = vcGantt1.UpdateBehviorCollection;
```

UseHigherDiagramHistogramHeightRatioPrecision

Eigenschaft von VcGantt

Ist diese Eigenschaft auf "True" gesetzt, werden zur Ermittlung des Höhenverhältnisses zwischen Diagramm und Histogramm die genauere Methode **DiagramHistogramHeightRatioEx** bzw. das Ereignis **VcHistogramHeightChangingEx** verwendet, die jeweils einen Wert vom Datentyp "Double" zurückgeben.

Beim Standardwert "False" wird die Methode **DiagramHistogramHeightRatio** bzw. das Ereignis **VcHistogramHeight** verwendet.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Genauere Methoden zur Ermittlung des Höhenverhältnisses Diagramm/Histogramm werden (True)/werden nicht (False) verwendet **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.UseHigherTableDiagramHeightRatioPrecision = False
```

Code-Beispiel C#

```
vcGantt1.UseHigherTableDiagramHeightRatioPrecision = true;
```

UseHigherTableDiagramWidthRatioPrecision

Eigenschaft von VcGantt

Ist diese Eigenschaft auf "True" gesetzt, werden zur Ermittlung des Breitenverhältnisses zwischen Tabelle und Diagramm die genaueren Methoden **LeftTableDiagramWidthRatioEx** und **RightTableDiagram-WidthRatioEx** bzw. das Ereignis **VcTableWidthChangingEx** verwendet, die jeweils einen Wert vom Datentyp "Double" zurückgeben.

Beim Standardwert "False" werden die Methoden **LeftTableDiagram-WidthRatio** und **RightTableDiagramWidthRatio** bzw. das Ereignis **VcTableWidthChanging** verwendet.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Allgemeines** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Genauere Methoden zur Ermittlung des Breitenverhältnisses Tabelle(n)/Diagramm werden (True)/werden nicht (False) verwendet **Standardwert:** False

UseSnapTargetsInInteractions

Nur-Lese-Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Einrastziele bei Knoten-/Layerinteraktionen verwendet werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Einrastziele werden/werden nicht bei Knoten-/Layerinteraktionen verwendet .

UseTwinLineSashPhantom

Eigenschaft von VcGantt

Mithilfe dieser Eigenschaft kann gesetzt oder erfragt werden, ob beim interaktiven Verschieben eines Sashes im **Standard**-Update-Behavior eine einzelne Phantomlinie oder eine Doppellinie erscheinen soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Doppelte Phantomlinie beim interaktiven Verschieben des Sash an/aus **Standardwert:** True

VerticalNodeMovementAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das vertikale Verschieben von Knoten im Diagramm zugelassen ist. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Knoten** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Vertikales Verschieben von Knoten im Diagramm zugelassen/nicht zugelassen **Standardwert:** false

VerticalNodeMovementViaTableAllowed

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das vertikale Verschieben von Knoten in der Tabelle zugelassen ist. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Knoten** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Vertikales Verschieben von Knoten in Tabelle zugelassen/nicht zugelassen **Standardwert:** false

ViewComponentsBackgroundColor

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Diagramms setzen oder erfragen. Zusammen mit der Eigenschaft **DiagramAlternating-RowBackgroundColor** können Sie ein zeilenweise alternierendes Farbmuster einrichten. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** SystemDrawing.Color.White

Code-Beispiel VB.NET

```
VcGantt1.ViewComponentsBackgroundColor = System.Drawing.Color.Blue
```

Code-Beispiel C#

```
vcGantt1.ViewComponentsBackgroundColor = System.Drawing.Color.Blue;
```

ViewComponentsBorderColor

Eigenschaft von VcGantt

Mit dieser Eigenschaft können Sie die Randlinienfarbe aller Fenster gemeinsam setzen oder erfragen. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Layout** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** SystemDrawing.Color.White

Code-Beispiel VB.NET

```
VcGantt1.ViewBorderColor = System.Drawing.Color.Blue
```

Code-Beispiel C#

```
vcGantt1.ViewBorderColor = System.Drawing.Color.Blue;
```

WaitCursorEnabled

Eigenschaft von VcGantt

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 **Standardwert:** False

WorldView

Nur-Lese-Eigenschaft von VcGantt

Über diese Eigenschaft erhalten Sie Zugriff auf das VcWorldView-Objekt, das die Komplettansicht des Diagramms definiert.

	Datentyp	Beschreibung
Eigenschaftswert	VcWorldView	Komplettansicht-Objekt

Code-Beispiel VB.NET

```
Dim worldview As VcWorldView
worldview = VcGantt1.WorldView
worldview.Visible = True
```

Code-Beispiel C#

```
VcWorldView worldview = vcGantt1.WorldView;
worldview.Visible = true;
```

ZoomFactor

Eigenschaft von VcGantt

Mit dieser Eigenschaft kann der absolute Zoomfaktor der Bildschirmdarstellung in % angegeben oder erfragt werden (Zoomfaktor = 100: Originalgröße, Zoomfaktor > 100: Vergrößerung, Zoomfaktor < 100: Verkleinerung).

Der absolute Zoomfaktor ist ein gerundeter Wert und kann daher zu Ungenauigkeiten führen.

Siehe auch die Gantt-Methoden **FitChartIntoView()** und **Zoom()**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	absoluter Zoomfaktor

Code-Beispiel VB.NET

```
VcGantt1.ZoomFactor = 150
```

Code-Beispiel C#

```
vcGantt1.ZoomFactor = 150;
```

ZoomingPerMouseWheelAllowed

Eigenschaft von VcGantt

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. Sie können diese Eigenschaft auch auf der Eigenschaftenseite **Allgemeines** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Zoomen erlaubt (True)/nicht erlaubt (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.ZoomingPerMouseWheelAllowed = True
```

Code-Beispiel C#

```
VcGantt1.ZoomingPerMouseWheelAllowed = true;
```

Methoden

ConvertDistance

Methode von VcGantt

Mit dieser Methode können Abstände, die in der Einheit "1/100 Millimeter" vorliegen, in die tatsächliche Anzahl Pixel in X- oder Y-Richtung gewandelt werden und umgekehrt. Die Umrechnung berücksichtigt den aktuell eingestellten Zoomfaktor (s. Eigenschaft **VcGantt.ZoomFactor**).

	Datentyp		Beschreibung
Parameter:			
⇨ conversionType	VcDistanceConversionType		Art der Umwandlung
	Mögliche Werte:		
	.vcXCentiMillimetersToPixels	1	Umwandlung eines Abstands in X-Richtung von 1/100 mm in Pixel.
	.vcXPixelsToCentiMillimeters	3	Umwandlung eines Abstands in X-Richtung von Pixeln in 1/100 Millimeter.
	.vcYCentiMillimetersToPixels	2	Umwandlung eines Abstands in Y-Richtung von 1/100 mm in Pixel.

	.vcYPixelsToCentiMillimeters 4	Umwandlung eines Abstands in Y-Richtung von Pixeln in 1/100 Millimeter.
⇨ value	System.Int32	Anzahl der Quellen-Einheiten (die umgewandelt werden sollen)
Rückgabewert	System.Int32	Anzahl der Ziel-Einheiten (in welche umgewandelt wurde)

DeleteLinkRecord

Methode von VcGantt

Mit dieser Methode können Sie eine Verbindung zwischen zwei Knoten löschen. Der Datensatz der Verbindung wird durch die im Dialog **Datentabellen verwalten** festgelegten Primärschlüssel identifiziert.

	Datentyp	**Beschreibung**
Parameter:		
⇨ linkRecordContent	System.Object	Inhalt des Verbindungsdatensatzes
Rückgabewert	System.Boolean	Datensatz der Verbindung erfolgreich/nicht erfolgreich gelöscht

Code-Beispiel VB.NET

```
VcGantt1.DeleteLinkRecord("A100;A105;;")
```

Code-Beispiel C#

```
vcGantt1.DeleteLinkRecord("A100;A105;;");
```

DeleteNodeRecord

Methode von VcGantt

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**
Parameter:		
⇨ nodeRecordContent	System.Object	Inhalt des Knotendatensatzes
Rückgabewert	System.Boolean	Datensatz des Knotens erfolgreich/nicht erfolgreich gelöscht

Code-Beispiel VB.NET

```
VcGantt1.DeleteNodeRecord("A100;;;;;;")
```

Code-Beispiel C#

```
vcGantt1.DeleteNodeRecord("A100;;;;;;");
```

DetectDataTableFieldName

Methode von VcGantt

Mit dieser Eigenschaft können Sie über den Index eines Tabellendatenfeldes seinen Namen erfragen.

	Datentyp	Beschreibung
Parameter: ⇨ fieldIndex	System.Int32	Index des Datentabellenfeldes, dessen Name ermittelt werden soll
Rückgabewert	System.String	Zurückgegebener Name des Datentabellenfeldes

Code-Beispiel VB.NET

```
'Find the name of a DataTableField
Dim fieldName As String

fieldName = VcGantt1.DetectDataTableFieldName(0)
```

Code-Beispiel C#

```
//Find the name of a DataTableField
string fieldName = vcGantt1.DetectDataTableFieldName(0);
```

DetectDataTableName

Methode von VcGantt

Mit dieser Eigenschaft können Sie über den Index einer Datentabelle ihren Namen erfragen.

	Datentyp	Beschreibung
Parameter: ⇨ fieldIndex	System.Int32	Index der Datentabelle, deren Name ermittelt werden soll
Rückgabewert	System.String	Zurückgegebener Name der Datentabelle

Code-Beispiel VB.NET

```
'Find the name of a DataTable
Dim tableName As String

tableName = VcGantt1.DetectDataTableName(0)
```

Code-Beispiel C#

```
//Find the name of a DataTable
string tableName = vcGantt1.DetectDataTableName(0);
```

DetectFieldIndex

Methode von VcGantt

Mit dieser Eigenschaft können Sie über den Namen einer Datentabelle und den Feldnamen den Index eines Tabellendatenfeldes erfragen.

	Datentyp	Beschreibung
Parameter:		
⇨ 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
Rückgabewert	System.Int32	Zurückgegebener Index des Datentabellenfeldes

Code-Beispiel VB.NET

```
'Find the index of a DataTableField
Dim fieldIndex As Integer

fieldIndex = VcGantt1.DetectFieldIndex("Maindata", "Name")
```

Code-Beispiel C#

```
//Find the index of a DataTableField
int fieldIndex = vcGantt1.DetectFieldIndex("Maindata", "Name");
```

DumpConfiguration

Methode von VcGantt

Mit dieser Methode können Sie die Konfiguration, bestehend aus .INI und .IFD-Datei, speichern.

Die Methode sollte nur zu Diagnosezwecken genutzt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ FileName	System.String	Dateiname, ggf. mit Pfad.
⇨ encoding	VcEncoding	Art der Kodierung
	Mögliche Werte:	

	.vcANSIEncoding 1 .vcUnicodeEncoding 2	Wird eine Datei in der ANSI-Kodierung gespeichert, so geschieht dies in Abhängigkeit von den lokalen Einstellungen des Windows-Betriebssystems, d.h. die Datei enthält Zeichen, die nur in der aktuell eingestellten Sprachversion auch wieder korrekt eingelesen werden können. 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 VARCHART-Steuerelement eingelesen werden soll.
Rückgabewert	System.Boolean	Datei erfolgreich (True)/nicht erfolgreich (False) abgespeichert.

EndLoading

Methode von VcGantt

Mit dieser Methode wird das Ende des Ladevorgangs bei **InsertNodeRecord** und **InsertLinkRecord** angezeigt. Dadurch wird eine Aktualisierung der Grafik ausgelöst.

	Datentyp	**Beschreibung**
Rückgabewert	System.Boolean	Ladevorgang erfolgreich beendet

Code-Beispiel VB.NET

```
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
vcGantt1.EndLoading();
```

ExportGraphicsToFileEx

Methode von VcGantt

Mit dieser Methode können Sie ein Gantt-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 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.

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 VARCHART XGantt wie folgt berechnet:

- PNG: Es wird eine Auflösung von 100 dpi bei einem Zoomfaktor von 100% angenommen. Wird alternativ im Parameter SizeX ein Wert <= -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 <= -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**
Parameter:		
⇨ fileName	System.String	Dateiname, ggf. mit Pfad.
⇨ printOutputFormat	PrintOutputFormat	Format der abzuspeichernden Datei
	Mögliche Werte:	
	.vcBMP 2	Datei wird im Format BMP geschrieben.
	.vcEMF 9	Datei wird im Format EMF geschrieben.
	.vcEMFPlus 12	Datei wird als *.EMF-Datei geschrieben, beinhaltet aber nur das EMF+-Format.
	.vcEMFWithEMFPlusIncluded 11	Datei wird als *.EMF-Datei geschrieben, beinhaltet aber zusätzlich das EMF+-Format.
	.vcEPS 3	Wird nicht mehr unterstützt.
	.vcGIF 4	Datei wird im Format GIF geschrieben.
	.vcJPG 5	Datei wird im Format JPG geschrieben.
	.vcPCX 6	Wird nicht mehr unterstützt.
	.vcPNG 7	Datei wird im Format PNG geschrieben.
	.vcTIF 8	Datei wird im Format TIF geschrieben.
	.vcVMF 0	Datei wird im Format VMF geschrieben.
	.vcWMF 1	Datei wird im Format WMF geschrieben.
	.vcWMFWithEMFIncluded 10	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.
Rückgabewert	System.Boolean	Datei erfolgreich (true) / nicht erfolgreich (false) abgespeichert.

Code-Beispiel VB.NET

```
VcGantt1.ExportGraphicsToFileEx "C:\Tmp\test1.vmf", vcVMF,0,0
```

Code-Beispiel C#

```
vcGantt1.ExportGraphicsToFileEx(@"c:\Tmp\test.vmf",
VcPrintOutputFormat.vcVMF,0,0);
```

FitChartIntoView

Methode von VcGantt

Mit dieser Methode haben Sie die Möglichkeit, das Diagramm unter Beibehaltung des Seitenverhältnisses so in die Größe des Steuerelementes einzupassen, dass es entweder in der Höhe oder der Breite komplett sichtbar ist. Es wird die relative Vergrößerung bzw. Verkleinerung in Prozent * 1000 zurückgegeben.

Siehe auch die Eigenschaft **ZoomFactor** und die Methode **Zoom()** von VcGantt.

	Datentyp	**Beschreibung**
Parameter:		
⇨ fitMode	VcFitMode	Auswahl des Zoomfaktors
	Mögliche Werte:	
	.vcFitHeight 23	Das Diagramm wird in der Höhe an die Größe des Steuerelements angepasst.
	.vcFitMaximumOfWidthAndHeight 1051	Das Diagramm wird mit seiner größten Ausdehnung an die Größe des Steuerelements angepasst.
	.vcFitMinimumOfWidthAndHeight 1052	Das Diagramm wird mit seiner kleinsten Ausdehnung an die Größe des Steuerelements angepasst.
	.vcFitWidth 24	Das Diagramm wird in der Breite an die Größe des Steuerelements angepasst.
	.vcUseLargerZoomFactor 1053	Der größere der beiden Zoomfaktoren wird verwendet. Damit ist das Diagramm in dieser entsprechenden Ausdehnung immer größer als das Steuerelement
	.vcUseSmallerZoomFactor 1054	Der kleinere der beiden Zoomfaktoren wird verwendet. Damit passt das Diagramm mit der entsprechenden Ausdehnung immer komplett in das Steuerelement.
Rückgabewert	System.Int32	Relativer Zoomfaktor

Code-Beispiel VB.NET

```
VcGantt1.(FitChartIntoView(VcFitMode.vcFitWidth)
```

Code-Beispiel C#

```
vcGantt1.FitChartIntoView(VcFitMode.vcFitWidth);
```

FitHistogramsIntoView

Methode von VcGantt

Mit dieser Methode werden alle sichtbaren Histogramm so skaliert, dass sie gemeinsam in einem Fenster sichtbar sind. Die Skalierung erfolgt proportional; ihre Größenverhältnisse werden also beibehalten.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Histogramme mussten (True) / mussten nicht (False) neu skaliert werden.

Code-Beispiel VB.NET

```
VcGantt1.FitHistogramsIntoView = True
```

Code-Beispiel C#

```
VcGantt1.FitHistogramsIntoView = true;
```

FitRangeIntoView

Methode von VcGantt

Mit dieser Methode wird ein bestimmter Teilbereich der gesamten Zeitskala in das Diagramm eingepasst und damit sichtbar gemacht. Die Skalierung der Zeitskala wird dabei entsprechend verändert. Mit dem Parameter **gapAsNoOfTimeUnits** können Sie festlegen, um wie viele Zeiteinheiten der sichtbare Bereich früher beginnen soll als der durch den Parameter **startValue** festgelegte Anfang des Bereiches und später enden soll als das durch den Parameter **endDate** festgelegte Ende des Bereiches. Die Zeiteinheit selbst wird auf der Eigenschaftenseite **Allgemeines** eingestellt.

	Datentyp	Beschreibung
Parameter:		
⇨ startDate	System.DateTime	Anfangsdatum des einzupassenden Bereiches.
⇨ endDate	System.DateTime	Enddatum des einzupassenden Bereiches.
⇨ gapAsNoOfTimeUnits	System.Int32	Anzahl der Zeiteinheiten zwischen startDate und dem Anfang des sichtbaren Bereiches bzw. zwischen endDate und dem Ende des sichtbaren Bereiches
Rückgabewert	System.Boolean	Bereich wurde erfolgreich/nicht erfolgreich eingepasst.

Code-Beispiel VB.NET

```
VcGantt1.FitRangeIntoView("14.09.2014", "21.09.2014", 1)
```

Code-Beispiel C#

```
vcGantt1.FitRangeIntoView(Convert.ToDateTime("14.09.2014"),
Convert.ToDateTime("21.09.2014"),1);
```

GetAValueFromARGB

Methode von VcGantt

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
Parameter:		
⇨ argb	System.Int32	ARGB- Wert, aus dem der Alpha-Wert ermittelt werden soll
Rückgabewert	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 = VcGantt1.MakeARGB(alpha,red,green,blue)
alpha = VcGantt1.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 = VcGantt1.MakeARGB(alpha,red,green,blue);
alpha = VcGantt1.GetAValueFromARGB(argb);
```

GetBValueFromARGB

Methode von VcGantt

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
Parameter:		
⇨ argb	System.Int32	ARGB- Wert, aus dem der Blau-Wert ermittelt werden soll
Rückgabewert	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 = VcGantt1.MakeARGB(alpha,red,green,blue)
blue = VcGantt1.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 = VcGantt1.MakeARGB(alpha,red,green,blue);
blue = VcGantt1.GetBValueFromARGB(argb);
```

GetCurrentComponentStart

Methode von VcGantt

Diese Methode liefert den aktuellen Scrollwert (die Startkoordinate, auf die gerade gescrollt ist) eines grafischen Grundelements des VARCHART-Windows-Forms-Steuerelements (Zeitskala, Diagramm, Histogramm, Tabelle, Tabellenüberschrift usw.) in beliebiger Richtung in 1/100 mm.

	Datentyp	Beschreibung
Parameter:		
⇦ component	VcComponentType	Typ des grafischen Elements
	Mögliche Werte:	
	.vcAdditionalListComponent 1	zusätzliche Tabelle
	.vcBottomListTitleComponent 14	untere Titelleiste
	.vcBottomRightListTitleComponent 17	Tabellenbereich unten rechts
	.vcBottomTimeScaleComponent 15	untere Zeitskala
	.vcDiagramComponent 4	Diagramm
	.vcHistogramComponent 8	Histogramm
	.vcHistogramVerScaleComponent 7	numerische Skala (vertikale Histogrammskala)

	.vcLegendComponent 10	Legende (zur Zeit ohne Funktion; Rückgabewerte 00)
	.vcListComponent 0	Tabelle
	.vcListTitleComponent 2	Tabellenüberschrift
	.vcRightListComponent 5	rechte Tabelle
	.vcRightListTitleComponent 16	Tabellenüberschrift der rechten Tabelle
	.vcTimeScaleComponent 3	obere Zeitskala
	.vcTopTitleComponent 11	obere Titelleiste
⇦ scrollOrientation	VcScrollOrientation	Richtung, in die gescrollt wird
	Mögliche Werte:	
	.vcHorizontal 1	horizontales Scrollen
	.vcVertical 2	vertikales Scrollen
Rückgabewert	System.Int32	Scrollwert in 1/100 mm

GetCurrentViewDates

Methode von VcGantt

Mit dieser Methode können Sie das Start- oder Enddatum des sichtbaren Bereichs der Zeitskala erfragen.

	Datentyp	**Beschreibung**
Parameter:		
⇦ leftDate	System.DateTime	Startdatum des sichtbaren Bereichs der Zeitskala
⇦ rightDate	System.DateTime	Enddatum des sichtbaren Bereichs der Zeitskala
Rückgabewert	System.Boolean	Start- bzw. Enddatum des sichtbaren Bereichs der Zeitskala werden zurückgegeben/nicht zurückgegeben.

Code-Beispiel VB.NET

```
Dim bGetCurrentViewDates As Boolean
Dim leftDate As Date
Dim rightDate As Date
GetCurrentViewDates = VcGantt1.GetCurrentViewDates(leftDate, rightDate)
```

Code-Beispiel C#

```
DateTime leftDate = new DateTime();
DateTime rightDate = new DateTime();
bool bGetCurrentViewDates = vcGantt1.GetCurrentViewDates(ref leftDate, ref
rightDate);
```

GetDate

Methode von VcGantt

Mit dieser Methode können Sie innerhalb des Diagrammbereichs das Datum zu einer beliebigen X-Koordinate erfragen.

	Datentyp	Beschreibung
Parameter:		
⇨ x	System.Int32	X-Koordinate im Gantt-Diagramm, zu der das Datum erfragt werden soll
Rückgabewert	System.DateTime	Erfragtes Datum

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramLeftClicking
   Label1.Name = VcGantt1.GetDate(e.X)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDiagramLeftClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
   {
   label1.Text = vcGantt1.GetDate(e.X).ToString();
   }
```

GetDateAsString

Methode von VcGantt

Mit dieser Methode können Sie innerhalb des Diagrammbereichs das Datum zu einer beliebigen X-Koordinate erfragen.

	Datentyp	Beschreibung
Parameter:		
⇨ x	System.Int32	X-Koordinate im Gantt-Diagramm, zu der das Datum erfragt werden soll
Rückgabewert	System.String	Erfragtes Datum

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramLeftClicking

    MsgBox(VcGantt1.GetDateAsString(e.X))

End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDiagramLeftClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)

{

    MessageBox.Show(vcGantt1.GetDateAsString(e.X));

}
```

GetGValueFromARGB

Methode von VcGantt

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
Parameter:		
⇨ argb	System.Int32	ARGB- Wert, aus dem der Grün-Wert ermittelt werden soll
Rückgabewert	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 = VcGantt1.MakeARGB(alpha,red,green,blue)
green = VcGantt1.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 = VcGantt1.MakeARGB(alpha,red,green,blue);
green = VcGantt1.GetGValueFromARGB(argb);
```

GetLinkByID

Methode von VcGantt

Mit dieser Methode können Sie auf einen einzelne Verbindung 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
Parameter:		
⇨ linkID	System.Object	Identifikation der Verbindung
Rückgabewert	VcLink	Verbindung

Code-Beispiel VB.NET

```
Dim link As VcLink
Dim successor As Integer
link = VcGantt1.GetLinkByID(" 1")
successor = link.DataField(2)
```

Code-Beispiel C#

```
VcLink link = vcGantt1.GetLinkByID(" 1");
int successor = Convert.ToInt32(link.get_DataField(2));
```

GetLinkByNodeIDs

Methode von VcGantt

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
Parameter:		
⇨ predecessorID	System.String	Identifikation des Vorgängerknotens
⇨ successorID	System.String	Identifikation des Nachfolgerknotens
Rückgabewert	VcLink	Verbindung

Code-Beispiel VB.NET

```
Dim link As VcLink
link = VcGantt1.GetLinkByNodeIDs(" 2", " 3")
```

Code-Beispiel C#

```
VcLink link = vcGantt1.GetLinkByNodeIDs(" 2", " 3");
```

GetNodeByID

Methode von VcGantt

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
Parameter:		
⇨ nodeID	System.Object	Identifikation des Knotens
Rückgabewert	VcNode	Knoten

Code-Beispiel VB.NET

```
Dim node As VcNode
node = VcGantt1.GetNodeByID("10")
```

Code-Beispiel C#

```
VcNode node = vcGantt1.GetNodeByID("10");
```

GetRValueFromARGB

Methode von VcGantt

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
Parameter:		
⇨ argb	System.Int32	ARGB- Wert, aus dem der Rot-Wert ermittelt werden soll
Rückgabewert	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 = VcGantt1.MakeARGB(alpha,red,green,blue)
red = VcGantt1.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 = VcGantt1.MakeARGB(alpha,red,green,blue);
red = VcGantt1.GetRValueFromARGB(argb);
```

GetViewComponentSize

Methode von VcGantt

Mit dieser Methode können Sie zur Laufzeit die Größe und Position eines grafischen Grundelements des VARCHART XGantt-Steuerelements (Zeitskala, Diagramm, Histogramm, Tabelle, Tabellenüberschrift usw.) erfragen (siehe Ereignis **VcViewComponentsSizeModified**).

Hinweise:

1. Die Position bezieht sich auf den Nullpunkt des VARCHART XGantt-Steuerelements.

2. Die Werte werden in Pixel zurückgegeben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ viewComponent	VcComponentType	Komponententyp
	Mögliche Werte:	
	.vcAdditionalListComponent 1	zusätzliche Tabelle
	.vcBottomListTitleComponent 14	untere Titelleiste
	.vcBottomRightListTitleComponent 17	Tabellenbereich unten rechts
	.vcBottomTimeScaleComponent 15	untere Zeitskala
	.vcDiagramComponent 4	Diagramm
	.vcHistogramComponent 8	Histogramm
	.vcHistogramVerScaleComponent 7	numerische Skala (vertikale Histogrammskala)

	.vcLegendComponent 10 .vcListComponent 0 .vcListTitleComponent 2 .vcRightListComponent 5 .vcRightListTitleComponent 16 .vcTimeScaleComponent 3 .vcTopTitleComponent 11	Legende (zur Zeit ohne Funktion; Rückgabewerte 00) Tabelle Tabellenüberschrift rechte Tabelle Tabellenüberschrift der rechten Tabelle obere Zeitskala obere Titelleiste
⇦ x	System.Int32	X-Koordinate der Komponente
⇦ y	System.Int32	Y-Koordinate der Komponente
⇦ width	System.Int32	Breite der Komponente
⇦ height	System.Int32	Höhe der Komponente
Rückgabewert	Void	

Code-Beispiel VB.NET

```
Private Sub handleHideHistogram()
   Dim x As Integer
   Dim y As Integer
   Dim width As Integer
   Dim height As Integer
   VcGantt1.GetViewComponentSize(VcComponentType.vcHistogramVerScaleComponent,
x, y, width, height)
   ' plus 6 because of the sash
   TextBox1.Top = VcGantt1.Top + y + 6
   TextBox1.Left = VcGantt1.Left + x
   ' minus 25 because of the numeric scale
   TextBox1.Width = width - 25
    ' minus 6 because of the sash
   TextBox1.Height = height - 6
End Sub
```

Code-Beispiel C#

```
private void handleHideHistogram()
   {
   int x;
   int y;
   int width;
   int height;
    vcGantt1.GetViewComponentSize(VcComponentType.vcHistogramVerScaleComponent,
ref x, ref y, ref width, ref height);
   // plus 6 because of the sash
   textBox1.Top = vcGantt1.Top + y + 6;
   textBox1.Left = vcGantt1.Left + x;
   // minus 25 because of the numeric scale
   textBox1.Width = width - 25;
   // minus 6 because of the sash
   textBox1.Height = height - 6;
   }
```

GroupNodes

Methode von VcGantt

Mit dieser Methode können Sie die Gruppierung an- oder abschalten. Wenn Sie mit der Eigenschaft **GroupingDataFieldIndex** ein Gruppierfeld und/oder mit der Eigenschaft **GroupSortingDataFieldIndex** eine Gruppierreihenfolge

eingestellt haben, so müssen Sie danach mit **GroupNodes** die Gruppierung aktivieren.

	Datentyp	Beschreibung
Parameter:		
⇨ onOff	System.Boolean	Gruppierung ein/aus
Rückgabewert	System.Boolean	Knoten erfolgreich/nicht erfolgreich gruppiert

Code-Beispiel VB.NET

```
VcGantt1.GroupingDataFieldIndex(0) = 11
VcGantt1.GroupSortingDataFieldIndex(0) = 12

VcGantt1.GroupNodes(True)
```

Code-Beispiel C#

```
vcGantt1.set_GroupingDataFieldIndex(0, 11);
vcGantt1.set_GroupSortingDataFieldIndex(0,12);

vcGantt1.GroupNodes(true);
```

IdentifyField

Methode von VcGantt

Mit dieser Methode können Sie den Index eines Datenfeldes identifizieren, dessen Inhalt im Tabellenfeld an der gegebenen Cursorposition dargestellt wird.

	Datentyp	Beschreibung
Parameter:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
Rückgabewert	System.Int32	Identifizierter Datenfeldindex
		-1, wenn an der Position kein Tabellenfeld liegt

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.VcNodeRightClicking
   Dim intField As Integer
   intField = VcGantt1.IdentifyField(e.X, e.Y)
   Label1.Text = e.Node.DataField(intField)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   int i = vcGantt1.IdentifyField(e.X, e.Y) ;

   if (i > 1) then

      Label1.Text =
Convert.ToString(e.Node.get_DataField(vcGantt1.IdentifyField(i)) ;
   }
```

IdentifyLayerAt

Methode von VcGantt

Mit dieser Methode können Sie einen Layer identifizieren. Wenn Sie mit der Methode **IdentifyObjectAt** einen Knoten identifiziert haben, können Sie ihn als Referenzobjekt benutzen, um mit **IdentifyLayerAt** den Layer dieses Knotens zu erfragen, der auf den angegebenen Koordinaten liegt.

	Datentyp	**Beschreibung**
Parameter:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇨ referenceNode	VcNode	Bezugsknoten
⇦ identifiedLayer	VcLayer	Identifizierter Layer
Rückgabewert	System.Boolean	Objekt identifiziert/Objekt nicht identifiziert

Code-Beispiel VB.NET

```
Dim identifiedObj As Object
Dim identifiedObjType As VcObjectType
Dim identifiedLayer As VcLayer
Dim node As VcNode

VcGantt1.IdentifyObjectAt(e.X, e.Y, identifiedObj, identifiedObjType)
If identifiedObjType Is VcObjectType.vcObjTypeNodeInDiagram Then
  node = identifiedObj
End If
Point mousePos=VcGantt1.PointToClient(new Point(){X=mousePos.X, Y=mousePos.Y}


Select Case identifiedObjType
   Case VcObjectType.vcObjTypeNodeInDiagram
     VcGantt1.IdentifyLayerAt(X, Y, identifiedLayer, identifiedLayerType)
      If Not identifiedLayer Is Nothing Then
         MsgBox("The Node " + node.DataField(0) + " , Layer " +
identifiedLayer.Name + ", was identified in the diagram area.")
      Else
         MsgBox("The Node" + node.DataField(0) + " was identified in diagram
area; no layer was identified")
      End If
   Case VcObjectType.vcObjTypeNodeInTable
      MsgBox("The Node" + node.DataField(0) + " was identified via table")
   Case Else
      MsgBox("No node was identified")
End Select
```

Code-Beispiel C#

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using NETRONIC.XGantt;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

       using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using NETRONIC.XGantt;

namespace WindowsFormsApplication1
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void vcGantt1_MouseDown(object sender, MouseEventArgs e)
        {
            object identifiedObj = null;
            VcObjectType identifiedObjType =
VcObjectType.vcObjTypeNodeInDiagram;
            VcLayer identifiedLayer = null;
            VcNode node;

            vcGantt1.IdentifyObjectAt(e.X, e.Y, ref identifiedObj, ref
identifiedObjType);
            //Please note: the .NET events DragOver, DragDrop and DragEnter will
return screen coordinates, not client coordinates.
            //In case you used those events before, you will need to convert
manually the coordinates returned into client coordinates:
            //Point mousePos=vcGantt1.PointToClient(new Point(){X=e.X, Y=e.Y});
            //vcGantt1.IdentifyObjectAt(mousePos.X, mousePos.Y, ref
identifiedObj, ref identifiedObjType);


            switch (identifiedObjType)
            {
                case VcObjectType.vcObjTypeNodeInDiagram:
                    node = (VcNode)identifiedObj;
                    vcGantt1.IdentifyLayerAt(e.X, e.Y, node, ref
identifiedLayer);
                    if (identifiedLayer != null)
```

```
                        MessageBox.Show("The Node " + node.get_DataField(0) + "
, Layer " + identifiedLayer.Name + ", was identified in the diagram area.");
                    break;
                default:
                    break;
            }
        }
    }
}
```

IdentifyObject

Methode von VcGantt

Mit dieser Methode können Sie ein beliebiges Objekt in VARCHART XGantt identifizieren. Der Typ des Objekts wird zurückgegeben. Wenn Sie mit dieser Methode einen Knoten identifiziert haben, können Sie ihn als Referenzobjekt benutzen, um mit einem weiteren Aufruf von **IdentifyObject** den Layer dieses Knotens an derselben Position zu erfragen.

Sollten Sie mit einer Entwicklungsumgebung arbeiten, die immer eine Referenz auf ein Objekt erfordert, benutzen Sie bitte die Methode **IdentifyObjectAt**, weil dort der Parameter **referenceObject** nicht erforderlich ist.

	Datentyp	**Beschreibung**
Parameter:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇨ referenceObject	VcObject	Bezugsobjekt, auf das sich die Identifizierung bezieht
⇦ identifiedObject	System.Object	Erkanntes Objekt
⇦ identifiedObjectType	VcObjectType	Typ des erkannten Objekts
	Mögliche Werte:	
	.vcObjTypeBox 15	Objekttyp **Box**
	.vcObjTypeCalendarGrid 18	Objekttyp **Kalendergitter**
	.vcObjTypeCurve 12	Objekttyp **Kurve**
	.vcObjTypeDateLine 9	Objekttyp **Stichtaglinie**
	.vcObjTypeGroup 7	Objekttyp **Gruppe**
	.vcObjTypeGroupInDiagram 11	Objekttyp **Gruppe im Knotenbereich**
	.vcObjTypeGroupInTable 7	Objekttyp **Gruppe im Tabellenbereich**
	.vcObjTypeHistogram 13	Objekttyp **Histogramm**
	.vcObjTypeLayer 8	Objekttyp **Layer**
	.vcObjTypeLinkCollection 3	Objekttyp **LinkCollection**
	.vcObjTypeNodeInDiagram 2	Objekttyp **Knoten im Knotenbereich**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNodeInTable 1	Objekttyp **Knoten im Tabellenbereich**
	.vcObjTypeNone 0	kein Objekt
	.vcObjTypeNumericScale 10	Objekttyp **Werteskala**
	.vcObjTypeSummaryNode 14	Objekttyp **Summenbalken**
	.vcObjTypeTable 4	Objekttyp **Tabelle**
	.vcObjTypeTableCaption 5	Objekttyp **Tabellenüberschrift**

	.vcObjTypeTimeScale 6	Objekttyp **Zeitskala**
Rückgabewert	System.Boolean	Objekt identifiziert/Objekt nicht identifiziert

IdentifyObjectAt

Methode von VcGantt

Mit dieser Methode können Sie ein beliebiges Objekt in VARCHART XGantt identifizieren. Der Typ des Objekts wird zurückgegeben. Wenn Sie mit dieser Methode einen Knoten identifiziert haben, können Sie ihn als Referenzobjekt benutzen, um mit einem Aufruf von **IdentifyLayerAt** den Layer dieses Knotens an derselben Position zu erfragen. Zum Identifizieren einer Kurve im Histogram muss die Methode **IdentifyObject** verwendet werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇦ identifiedObject	System.Object	Erkanntes Objekt
⇦ identifiedObjectType	VcObjectType	Typ des erkannten Objekts
	Mögliche Werte:	
	.vcObjTypeBox 15	Objekttyp **Box**
	.vcObjTypeCalendarGrid 18	Objekttyp **Kalendergitter**
	.vcObjTypeCurve 12	Objekttyp **Kurve**
	.vcObjTypeDateLine 9	Objekttyp **Stichtaglinie**
	.vcObjTypeGroup 7	Objekttyp **Gruppe**
	.vcObjTypeGroupInDiagram 11	Objekttyp **Gruppe im Knotenbereich**
	.vcObjTypeGroupInTable 7	Objekttyp **Gruppe im Tabellenbereich**
	.vcObjTypeHistogram 13	Objekttyp **Histogramm**
	.vcObjTypeLayer 8	Objekttyp **Layer**
	.vcObjTypeLinkCollection 3	Objekttyp **LinkCollection**
	.vcObjTypeNodeInDiagram 2	Objekttyp **Knoten im Knotenbereich**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNodeInTable 1	Objekttyp **Knoten im Tabellenbereich**
	.vcObjTypeNone 0	kein Objekt
	.vcObjTypeNumericScale 10	Objekttyp **Werteskala**
	.vcObjTypeSummaryNode 14	Objekttyp **Summenbalken**
	.vcObjTypeTable 4	Objekttyp **Tabelle**
	.vcObjTypeTableCaption 5	Objekttyp **Tabellenüberschrift**
	.vcObjTypeTimeScale 6	Objekttyp **Zeitskala**
Rückgabewert	System.Boolean	Objekt identifiziert/Objekt nicht identifiziert

Code-Beispiel VB.NET

```
 Private Sub VcGantt1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles VcGantt1.MouseMove

      Dim identifiedObject As Object = Nothing
      Dim identifiedObjectType As VcObjectType = VcObjectType.vcObjTypeNone
      Dim node As VcNode = Nothing
      Dim identifiedLayer As VcLayer = Nothing

      VcGantt1.IdentifyObjectAt(e.X, e.Y, identifiedObject,
identifiedObjectType)

      Select Case identifiedObjectType
         Case VcObjectType.vcObjTypeNodeInDiagram
            node = identifiedObject

            VcGantt1.IdentifyLayerAt(e.X, e.Y, node, identifiedLayer)

            If identifiedLayer IsNot Nothing Then
               Label1.Text = "X = " & e.X & "   Y = " & e.Y & vbCrLf & _
                             "Node ID = " & node.DataField(0) & vbCrLf & _
                             "Layer Name = " & identifiedLayer.Name
            End If

         Case Else
            Label1.Text = ""
      End Select

   End Sub
```

Code-Beispiel C#

```
private void VcGantt1_MouseMove(object sender, MouseEventArgs e)
      {
         object identifiedObject = null;
         VcObjectType identifiedObjectType = VcObjectType.vcObjTypeNone;
         VcNode node = null;
         VcLayer identifiedLayer = null;

         VcGantt1.IdentifyObjectAt(e.X, e.Y, ref identifiedObject, ref
identifiedObjectType);

         switch (identifiedObjectType)
         {
            case VcObjectType.vcObjTypeNodeInDiagram:
               {
                  node = (VcNode)identifiedObject;

                  VcGantt1.IdentifyLayerAt(e.X, e.Y,  node, ref
identifiedLayer);

                  if (identifiedLayer != null)
                     label1.Text = "X = " + e.X + "   Y = " + e.Y +
                                   "\nNode ID = " + node.get_DataField(0) +
                                   "\nLayer Name = " + identifiedLayer.Name;
                  break;
               }
            default:
               {
                  label1.Text = "";
                  break;
               }
         }
      }
```

ImportConfiguration

Methode von VcGantt

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
Parameter:		
⇨ fileName	System.String	Name der zu importierenden Datei
Rückgabewert	Void	

Code-Beispiel VB.NET

```
VcGantt1.ImportConfiguration ( "c:\VARCHART\XGantt\sample.ini")
'or
VcGantt1.ImportConfiguration
("http://members.tripod.de/netronic_te/xgantt_sample.ini)
```

Code-Beispiel C#

```
vcGantt1.ImportConfiguration (@"c:\VARCHART\XGantt\sample.ini");
// or
vcGantt1.ImportConfiguration
(@"http://members.tripod.de/netronic_te/xgantt_sample.ini");
```

InitializeForWebService

Methode von VcGantt

Nur für internen Gebrauch.

	Datentyp	Beschreibung
Rückgabewert	Void	

InsertLinkRecord

Methode von VcGantt

Mit dieser Methode werden die Daten einer Verbindung zwischen zwei Knoten geladen. Die Daten werden als CSV-String oder als Datenfeld gemäß der im Dialog **Datentabellen verwalten** in der **Relations**-Tabelle vereinbarten Struktur übergeben. Die Methode **EndLoading** sollte am Ende des kompletten Ladevorgangs (Verbindungen und Knoten) einmal aufgerufen werden.

	Datentyp	Beschreibung
Parameter:		
⇨ linkRecordContent	System.Object[]	Inhalt des Verbindungsdatensatzes
Rückgabewert	VcLink	Verbindung

Code-Beispiel VB.NET

```
VcGantt1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning")
VcGantt1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing")
VcGantt1.InsertLinkRecord("1;A100;A105;FS;0")
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
vcGantt1.InsertNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning");
vcGantt1.InsertNodeRecord("A105;Activity 5;13.09.14;18.09.14;7;Testing");
vcGantt1.InsertLinkRecord("1;A100;A105;FS;2");
vcGantt1.EndLoading();
```

InsertNodeRecord

Methode von VcGantt

Mit dieser Methode werden die Daten eines Knotens geladen. Die Daten werden als CSV-String oder Datenfeld gemäß der im Dialog **Datentabellen verwalten** in der **Maindata**-Tabelle vereinbarten Struktur übergeben. Die Methode **EndLoading** sollte am Ende des kompletten Ladevorgangs (Verbindungen und Knoten) einmal aufgerufen werden.

	Datentyp	Beschreibung
Parameter:		
⇨ nodeRecordContent	Data field	Inhalt des Knotendatensatzes
Rückgabewert	VcNode	Knoten

Code-Beispiel VB.NET

```
Dim nodeRecord As String
' data format: "Number;Name;Start date;Finish date;Group code;Group name"
nodeRecord = "A100;Activity 1;12.09.14;17.09.14;5;Planning"
VcGantt1.InsertNodeRecord(nodeRecord)
VcGantt1.EndLoading()

'or
Dim nodeRecord() As Object = New Object(5) {"A100", "Activity 1", "12.09.14",
"17.09.14", "5", "Planning"}
VcGantt1.InsertNodeRecord(nodeRecord)
VcGantt1.EndLoading()
```

Code-Beispiel C#

```
string nodeRecord = "A100;Activity 1;12.09.14;17.09.14;5;Planning";
vcGantt1.InsertNodeRecord(nodeRecord);
vcGantt1.EndLoading();
```

Load

Methode von VcGantt

Mit dieser Methode werden aus der angegebenen Datei die Datensätze der Datentabellen eingelesen, die zu einem früheren Zeitpunkt mit der Methode SaveAsEx (...) im CSV-Format gespeichert wurden. CSV-Dateien können sowohl im ANSI- als auch in Unicode-Zeichensatz gelesen und geschrieben werden. Beim Einlesen erfolgt eine automatische Erkennung der Codierung.

Die Zuordnung der Datensätze zu den einzelnen Datentabellen wird jeweils durch eine entsprechende Identifizierungszeile sichergestellt.

```
**** Name der Tabelle ****
```

Beispiel:

```
**** Maindata ****
1;Node 1;07.05.2007;;5
2;Node 2;14.05.2007;;5
3;Node 3;21.05.2007;;5
**** Relations ****
1;1;2
2;2;3
```

Alle Datensätze von Tabellen, die nicht existieren, werden beim Einlesen ignoriert. Der Inhalt der Datentabellen wird vollständig ersetzt.

	Datentyp	**Beschreibung**
Parameter:		
⇨ fileName	System.String	Dateiname
Rückgabewert	System.Boolean	Datei erfolgreich/nicht erfolgreich geöffnet

Code-Beispiel VB.NET

```
vcgantt1.Load("c:\Data\project1.bar")
```

Code-Beispiel C#

```
vcGantt1.Load(@"c:\Data\project1.bar");
```

MakeARGB

Methode von VcGantt

Mit dieser Methode können Sie aus den vier Einzelwerten einer Farbe einen ARGB-Wert bilden.

	Datentyp	Beschreibung
Parameter:		
⇨ alpha	SystemInt.32	Alpha- Wert
⇨ red	SystemInt.32	Rot- Wert
⇨ green	SystemInt.32	Grün- Wert
⇨ blue	SystemInt.32	Blau- Wert
Rückgabewert	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 = VcGantt1.MakeARGB(alpha,red,green,blue)
```

Code-Beispiel C#

```
long argb;
int alpha = FF;
int red = A0;
int green = 34;
int blue = AB;
argb = VcGantt1.MakeARGB(alpha,red,green,blue);
```

OptimizeTimeScaleStartEnd

Methode von VcGantt

Mit dieser Methode werden Anfang und Ende der Zeitskala in Abhängigkeit von den dargestellten Vorgängen so gesetzt, dass alle Vorgänge zwischen

Anfang und Ende der Zeitskala liegen. Mit dem Parameter **NoOfUnits** können Sie festlegen, um wie viele Zeiteinheiten die Skala früher beginnen soll als der früheste Anfang aller Vorgänge und später enden soll als das späteste Ende aller Vorgänge. Die Zeiteinheit selbst wird auf der Eigenschaftenseite **Allgemeines** eingestellt.

	Datentyp	**Beschreibung**
Parameter:		
⇨ noOfUnits	System.Int16	Anzahl der Zeiteinheiten
Rückgabewert	System.Boolean	Zeitskala erfolgreich/nicht erfolgreich optimiert. Der Rückgabewert ist **false**, wenn sowohl **TimeScaleStart** als auch **TimeScaleEnd** nicht verändert wurden. Wenn keine Vorgänge vorliegen, ist der Rückgabewert stets **false**, weil keine Datumsänderungen stattfinden. Die angegebene Anzahl der Zeiteinheiten ist dabei ohne Bedeutung.

Code-Beispiel VB.NET

```
VcGantt1.OptimizeTimeScaleStartEnd(5)
```

Code-Beispiel C#

```
vcGantt1.OptimizeTimeScaleStartEnd(5);
```

PrintEx

Methode von VcGantt

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
Rückgabewert	VcPrintResultStatus	**Mögliche Werte:**

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 gekennzeichneter 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 = VcGantt1.PrintDirectEx();
if (status != VcPrintResultStatus.vcPrintingSucceeded)
   System.Diagnosis.Trace.WriteLine("Printing failed: "+status.ToString);
```

PrintToFile

Methode von VcGantt

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
Parameter:		
⇨ fileName	System.String	Name der Datei
Rückgabewert	Void	

RecalculateAllStructureCodes

Methode von VcGantt

Mit dieser Methode können Sie den Struktur-Code der Knotenhierarchie für alle Knoten neu berechnen lassen. Eine Neuberechnung erfolgt automatisch nach jeder Änderung. Um die automatische Neuberechnung für bestimmte Aktionen zu unterdrücken, können Sie Anweisungen zwischen VcGantt.SuspendUpdate(true) und VcGantt.SuspendUpdate(false) "klammern".

	Datentyp	Beschreibung
Rückgabewert	Void	

Reset

Methode von VcGantt

Mit dieser Methode können Sie - je nach eingestelltem Wert von resetAction - Objekte (Knoten, Verbindungen, Kalender etc.) aus dem Diagramm löschen bzw. den zur Entwurfszeit auf den Eigenschaftenseiten eingestellten Zustand wiederherstellen.

	Datentyp	Beschreibung
Parameter:		
⇨ resetAction	VcResetAction	Objekte, die gelöscht oder neu initialisiert werden
	Mögliche Werte:	
	.vcEmptyAllDataTables 4	Der komplette Inhalt sämtlicher Datentabellen wird gelöscht, die Datentabellen selbst bleiben bestehen.
	.vcReloadConfiguration 2	Komplette Neuinitialisierung mit der INI-Datei. Alle Einstellungen und erzeugten Objekte verfallen.
	.vcRemoveGroups 0	Alle Gruppen und abhängigen Objekte und damit auch alle Knoten und Links werden entfernt.
	.vcRemoveNodes 1	Alle Knoten und abhängigen Objekte und damit auch alle Links, sofern vorhanden, werden entfernt.
Rückgabewert	System.Boolean	Objekte im Diagramm erfolgreich gelöscht {True}

Code-Beispiel VB.NET

```
VcGantt1.Reset(VcResetAction.vcRemoveNodes)
```

Code-Beispiel C#

```
vcGantt1.Reset(VcResetAction.vcRemoveNodes);
```

SaveAsEx

Methode von VcGantt

Mit dieser Methode werden die Datensätze aller Datentabellen 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**
Parameter:		
⇨ fileName	System.String	Name der zu speichernden Datei
⇨ encoding	VcEncoding	Art der Kodierung
	Mögliche Werte:	
	.vcANSIEncoding 1	Wird eine Datei in der ANSI-Kodierung gespeichert, so geschieht dies in Abhängigkeit von den lokalen Einstellungen des Windows-Betriebssystems, d.h. die Datei enthält Zeichen, die nur in der aktuell eingestellten Sprachversion auch wieder korrekt eingelesen werden können.
	.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 VARCHART-Steuerelement eingelesen werden soll.
Rückgabewert	System.Boolean	Speicherung war erfolgreich/nicht erfolgreich

Code-Beispiel VB.NET

```
VcGantt1.SaveAsEx ("C:\ProjectData.txt", VcEncoding.vcANSIEncoding)
```

Code-Beispiel C#

```
vcGantt1.SaveAsEx (@"C:\ProjectData.txt", VcEncoding.vcANSIEncoding);
```

ScheduleProject

Methode von VcGantt

Mit dieser Methode können Sie eine Vorwärts- und Rückwärtsberechnung des aktuellen Projekts durchführen. Bei alleiniger Übergabe des Starttermins wird zunächst eine Vorwärtsberechnung, dann eine Rückwärtsberechnung durchgeführt. Bei alleiniger Übergabe des Endtermins wird zunächst eine Rückwärtsberechnung, dann eine Vorwärtsberechnung durchgeführt.

	Datentyp	Beschreibung
Parameter:		
⇨ startDate	System.DateTime	Startdatum
⇨ endDate	System.DateTime	Enddatum
Rückgabewert	System.Boolean	Berechnung erfolgreich/nicht erfolgreich

Code-Beispiel VB.NET

```
' Vorwärtsberechung (ASAP)
VcScheduler.ScheduleProject(2.5.2017, newDate(0))

' Rückwärtsberechnung (JIT)
VcScheduler.ScheduleProject(newDate(0), 2.5.2017)
```

Code-Beispiel C#

```
// Vorwärtsberechung (ASAP)
vcScheduler.ScheduleProject(2.5.2017, newDate(0));

// Rückwärtsberechnung (JIT)
vcScheduler.ScheduleProject(newDate(0), 2.5.2017);
```

ScrollComponentStartTo

Methode von VcGantt

Mit dieser Methode können Sie ein grafisches Grundelement des VARCHART-Windows-Forms-Steuerelements (Zeitskala, Diagramm, Histogramm, Tabelle, Tabellenüberschrift usw.) in beliebiger Richtung auf den angegebenen Scrollwert (die Startkoordinate) in 1/100 mm scrollen.

	Datentyp	Beschreibung
Parameter:		
⇦ component	VcComponentType	Typ des grafischen Elements
	Mögliche Werte:	
	.vcAdditionalListComponent 1	zusätzliche Tabelle
	.vcBottomListTitleComponent 14	untere Titelleiste
	.vcBottomRightListTitleComponent 17	Tabellenbereich unten rechts
	.vcBottomTimeScaleComponent 15	untere Zeitskala
	.vcDiagramComponent 4	Diagramm
	.vcHistogramComponent 8	Histogramm
	.vcHistogramVerScaleComponent 7	numerische Skala (vertikale Histogrammskala)
	.vcLegendComponent 10	Legende (zur Zeit ohne Funktion; Rückgabewerte 00)
	.vcListComponent 0	Tabelle
	.vcListTitleComponent 2	Tabellenüberschrift
	.vcRightListComponent 5	rechte Tabelle
	.vcRightListTitleComponent 16	Tabellenüberschrift der rechten Tabelle
	.vcTimeScaleComponent 3	obere Zeitskala
	.vcTopTitleComponent 11	obere Titelleiste
⇦ scrollOrientation	VcScrollOrientation	Richtung, in die gescrollt wird

	Mögliche Werte: .vcHorizontal 1 .vcVertical 2	 horizontales Scrollen vertikales Scrollen
Rückgabewert	System.Boolean	Gewünschter Scrollwert wird/wird nicht zurückgegeben

ScrollToDate

Methode von VcGantt

Mit dieser Methode können Sie in der Zeitskala zu einem bestimmten Datum scrollen. Der Parameter **gapAsNoOfTimeUnits** gibt an, wie viele Zeiteinheiten der Abstand zwischen dem angegebenen Datum und dem linken (**vcLeftAligned**) bzw. rechten (**vcRightAligned**) Rand der Zeitskala betragen soll, wenn Sie durch den Parameter **horAlignment** bestimmt haben, dass das angegebene Datum am linken bzw. rechten Rand des sichtbaren Bereiches der Zeitskala stehen soll. Die Zeiteinheit kann auf der Eigenschaftenseite **Allgemeines** eingestellt werden.

Sind Nicht-Arbeitszeiten kollabiert, werden die kollabierten (unsichtbaren) Zeiten zwar korrekt eingerechnet, aber optisch nicht dargestellt, was zu scheinbaren Abweichungen vom eingestellten Ergebnis führen kann.

	Datentyp	**Beschreibung**
Parameter:		
⇨ date	System.DateTime	Datum
⇨ horAlignment	VcHorizontalAlignment	Horizontale Ausrichtung
	Mögliche Werte: .vcHorCenterAligned -1 .vcLeftAligned -3 .vcRightAligned -2	 horizontal mittig linksbündig rechtsbündig
⇨ gapAsNoOfTimeUnits	System.Int32	Anzahl der Zeiteinheiten
Rückgabewert	System.Boolean	Scrollen erfolgreich/nicht erfolgreich durchgeführt

Code-Beispiel VB.NET

```
VcGantt1.ScrollToDate("20.10.14", VcHorizontalAlignment.vcLeftAligned, 2)
```

Code-Beispiel C#

```
vcGantt1.ScrollToDate(Convert.ToDateTime("20.10.14"),
VcHorizontalAlignment.vcRightAligned, 2);
```

ScrollToGroupLine

Methode von VcGantt

Mit dieser Methode können Sie zu der Zeile einer bestimmten Gruppe scrollen und festlegen, ob diese Gruppe oben, unten oder mittig auf dem Bildschirm erscheinen soll.

	Datentyp	Beschreibung
Parameter:		
⇨ group	VcGroup	Gruppe, zu der gescrollt werden soll
⇨ verAlignment	VcVerticalAlignment	Vertikale Ausrichtung
	Mögliche Werte: .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned -1	 bündig unten bündig oben vertikal mittig
Rückgabewert	System.Boolean	Scrollen erfolgreich/nicht erfolgreich durchgeführt

ScrollToNode

Methode von VcGantt

Mit dieser Methode können Sie zu einem bestimmten Knoten scrollen und festlegen, ob dieser Knoten oben, unten oder mittig auf dem Bildschirm erscheinen soll.

	Datentyp	Beschreibung
Parameter:		
⇨ node	VcNode	Knoten
⇨ verAlignment	VcVerticalAlignment	Vertikale Ausrichtung
	Mögliche Werte: .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned -1	 bündig unten bündig oben vertikal mittig
Rückgabewert	System.Boolean	Scrollen erfolgreich/nicht erfolgreich durchgeführt

Code-Beispiel VB.NET

```
Dim node As VcNode
node = VcGantt1.GetNodeByID(" 2")
VcGantt1.ScrollToNodeLine(node, VcVerticalAlignment.vcVerCenterAligned)
```

Code-Beispiel C#

```
      object[] objDataRecord = new object[5];

         vcGantt1.ExtendedDataTablesEnabled = true;
         vcGantt1.MinimumRowHeight = 1000;

         vcGantt1.TimeScaleEnd = new DateTime(2010, 8, 1);
         vcGantt1.TimeScaleStart = new DateTime(2010, 6, 1);

         objDataRecord[2] = new DateTime(2010, 6, 3);
         objDataRecord[3] = new DateTime(2010, 6, 10);
         objDataRecord[4] = 5;

         VcDataRecordCollection dataRecordCol =
vcGantt1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection;
         for (int i = 1; i < 100; i++)
            {
            objDataRecord[0] = i;
            objDataRecord[1] = "Node " + i.ToString();

            dataRecordCol.Add(objDataRecord);
             }
         vcGantt1.EndLoading();
         vcGantt1.ScrollToNode(vcGantt1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned);
```

ScrollToNodeLine

Methode von VcGantt

Mit dieser Methode können Sie zu der Zeile eines bestimmten Knotens scrollen und festlegen, ob dieser Knoten oben, unten oder mittig auf dem Bildschirm erscheinen soll.

Hinweis: Bei einer optimierten Darstellung werden alle Vorgänge einer Gruppe in einer Zeile ausgegeben. Überschneiden sich Vorgänge einer Gruppe, so werden sie im expandierten Status der Gruppe automatisch untereinander angeordnet, um Überlagerungen zu vermeiden. In diesem Fall bewirkt die Methode **ScrollToNodeLine**, dass zu der entsprechenden Gruppenzeile, in der sich der angegebene Knoten befindet gescrollt, wird. Unter Umständen steht dann der angegebene Knoten nicht mittig auf dem Bildschirm und ist ggf. nicht sofort sichtbar.

	Datentyp	**Beschreibung**
Parameter:		
⇨ node	VcNode	Knoten, zu dem gescrollt werden soll
⇨ verAlignment	VcVerticalAlignment	Vertikale Ausrichtung
	Mögliche Werte: .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned -1	 bündig unten bündig oben vertikal mittig

Rückgabewert	System.Boolean	Scrollen erfolgreich/nicht erfolgreich durchgeführt

Code-Beispiel VB.NET

```
Imports NETRONIC.XGantt
 ...
Dim i As Integer
Dim objDataRecord(2) As Object
Dim dataRecordCol As VcDataRecordCollection

VcGantt1.ExtendedDataTablesEnabled = True
dataRecordCol =
VcGantt1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection
    For i = 1 To 100
            objDataRecord(0) = i
            objDataRecord(1) = "Node " + i.ToString()
            dataRecordCol.Add(objDataRecord)
     Next
     VcGantt1.EndLoading()
     VcGantt1.ScrollToNodeLine(VcGantt1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned)
```

Code-Beispiel C#

```
using NETRONIC.XGantt;
     ...
object[] objDataRecord = new object[2];
vcGantt1.ExtendedDataTablesEnabled = true;
VcDataRecordCollection dataRecordCol =
vcGantt1.DataTableCollection.DataTableByName("Maindata").DataRecordCollection;
     for (int i = 1; i < 100; i++)
         {
         objDataRecord[0] = i;
         objDataRecord[1] = "Node " + i.ToString();
         dataRecordCol.Add(objDataRecord);
          }
vcGantt1.EndLoading();
vcGantt1.ScrollToNodeLine(vcGantt1.GetNodeByID("50"),
VcVerticalAlignment.vcTopAligned);
```

SetImageResource

Methode von VcGantt

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 XGantt-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** folgenden Code ein:

Für Bitmap-Ressourcen:

*vcGantt1.SetImageResource("*PlusImage",*
<namespace>.Properties.Resources.plusImage);

für Metafile-Ressourcen:

*vcGantt1.SetImageResource("*MinusImage", new Metafile(new MemoryStream(<namespace>.Properties.Resources.minusImage)));*

	Datentyp	Beschreibung
Parameter:		
⇨ imageName	System.String	Name, der dem Image-Objekt zugeordnet ist
Rückgabewert	System.Drawing.Image	Image-Objekt

Code-Beispiel C#

```
vcGantt1.SetImageResource("*PlusImage",
<namespace>.Properties.Resources.plusImage);
```

ShowAboutDialog

Methode von VcGantt

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

```
VcGantt1.ShowAboutDialog()
```

Code-Beispiel C#

```
vcGantt1.ShowAboutDialog();
```

ShowEditGroupDialog

Methode von VcGantt

Mit dieser Methode wird der Dialog **Gruppen bearbeiten** für die angegebene Gruppe aufgerufen.

	Datentyp	Beschreibung
Parameter:		
⇨ group	VcGroup	Gruppe, deren Daten editiert werden sollen
Rückgabewert	System.Boolean	Gruppendaten wurden editiert/Editierung abgebrochen

ShowExportGraphicsDialog

Methode von VcGantt

Mit dieser Methode können Sie ein **Speichern unter**-Dialogfeld aufrufen, um die Darstellung abzuspeichern. 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.

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 <= -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 <= -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**
Rückgabewert	System.Boolean	Grafik erfolgreich (true) /nicht erfolgreich (false) exportiert

Code-Beispiel VB.NET

```
VcGantt1.ShowExportGraphicsDialog()
```

Code-Beispiel C#

```
vcGantt1.ShowExportGraphicsDialog();
```

ShowLinkEditDialog

Methode von VcGantt

Mit dieser Methode wird der Dialog **Verbindung bearbeiten** für die angegebene Verbindung aufgerufen.

	Datentyp	Beschreibung
Parameter:		
⇨ link	VcLink	Verbindung, deren Daten editiert werden sollen
Rückgabewert	System.Boolean	Verbindungsdaten wurden editiert/Editierung abgebrochen

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinksClickingEventArgs) Handles VcGantt1.VcLinksLeftClicking
   Dim linkCltn As VcLinkCollection
   Dim link As VcLink
   linkCltn = VcGantt1.LinkCollection
   If linkCltn.Count > 0 Then
      For Each link In linkCltn
         VcGantt1.ShowLinkEditDialog(link)
      Next
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLinksLeftClicking(object sender,
NETRONIC.XGantt.VcLinksClickingEventArgs e)
   {
   if (e.LinkCollection.Count > 0)
      {
      foreach (VcLink link in e.LinkCollection)
         vcGantt1.ShowLinkEditDialog(link);
       }
   }
```

ShowNodeEditDialog

Methode von VcGantt

Mit dieser Methode wird der Dialog **Vorgänge bearbeiten** für den angegebenen Knoten aufgerufen.

	Datentyp	Beschreibung
Parameter:		
⇨ node	VcNode	Knoten, dessen Daten editiert werden sollen
Rückgabewert	System.Boolean	Knotendaten wurden editiert/Editierung abgebrochen

Code-Beispiel VB.NET

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

Code-Beispiel C#

```
private void vcGantt1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   vcGantt1.ShowNodeEditDialog(e.Node);
   }
```

ShowPageSetupDialog

Methode von VcGantt

Mit dieser Methode wird das Dialogfeld **Seite einrichten** aufgerufen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Dialogfeld erfolgreich/nicht erfolgreich aufgerufen

Code-Beispiel VB.NET

```
VcGantt1.ShowPageSetupDialog()
```

Code-Beispiel C#

```
vcGantt1.ShowPageSetupDialog();
```

ShowPrintDialog

Methode von VcGantt

Mit dieser Methode wird der Ausdruck der Grafik ausgelöst. Es wird das Windows-Dialogfeld **Drucken** geöffnet, dabei werden die unter **ShowPageSetupDialog** aktuell eingestellten Parameter verwendet.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Ausdruck erfolgreich/nicht erfolgreich ausgelöst

Code-Beispiel VB.NET

```
VcGantt1.ShowPrintDialog()
```

Code-Beispiel C#

```
vcGantt1.ShowPrintDialog();
```

ShowPrinterSetupDialog

Methode von VcGantt

Mit dieser Methode wird das Windows-Dialogfeld **Drucker einrichten** aufgerufen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Dialogfeld erfolgreich/nicht erfolgreich aufgerufen

Code-Beispiel VB.NET

```
VcGantt1.ShowPrinterSetupDialog()
```

Code-Beispiel C#

```
vcGantt1.ShowPrinterSetupDialog();
```

ShowPrintPreviewDialog

Methode von VcGantt

Mit dieser Methode wird die Druckvorschau aufgerufen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Dialogfeld erfolgreich/nicht erfolgreich aufgerufen

Code-Beispiel VB.NET

```
VcGantt1.ShowPrintPreviewDialog()
```

Code-Beispiel C#

```
vcGantt1.ShowPrintPreviewDialog();
```

SortGroups

Methode von VcGantt

Mit dieser Methode starten Sie bei einer gruppierten Darstellung eine Sortierung der Gruppen bezüglich des eingestellten Sortierparameters für Gruppen **GroupSortingDataFieldIndex** (**GroupingLevel**).

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Gruppen erfolgreich sortiert/nicht erfolgreich sortiert

Code-Beispiel VB.NET

```
VcGantt1.GroupSortingDataFieldIndex(0) = 12
VcGantt1.SortGroups()
```

Code-Beispiel C#

```
vcGantt1.set_GroupSortingDataFieldIndex(0,12);
vcGantt1.SortGroups();
```

SortNodes

Methode von VcGantt

Mit dieser Methode starten Sie eine Sortierung der Vorgänge bezüglich der eingestellten Sortierparameter (**NodeSortingDataFieldIndex** (**sortLevel**) und **NodeSortingOrder** (**sortLevel**)). Ist eine Gruppierung aktiv, so wirkt die Sortierung gruppenweise auf die Vorgänge.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Sortierung erfolgreich/nicht erfolgreich

Code-Beispiel VB.NET

```
VcGantt1.NodeSortingDataFieldIndex(0) = 3
VcGantt1.NodeSortingOrder(0) = VcNodeSortingOrder.vcAscending
VcGantt1.SortNodes()
```

Code-Beispiel C#

```
vcGantt1.set_NodeSortingDataFieldIndex(0,3);
vcGantt1.set_NodeSortingOrder(0, VcNodeSortingOrder.vcAscending);
vcGantt1.SortNodes();
```

SuspendUpdate

Methode von VcGantt

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
Parameter:		
⇨ suspendFlag	System.Boolean	SuspendUpdate(True): Beginn der SuspendUpdate Methode/ SuspendUpdate(False): Ende der SuspendUpdate Methode
Rückgabewert	Void	

Code-Beispiel VB.NET

```
VcGantt1.SuspendUpdate(True)

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

VcGantt1.SuspendUpdate(False)
```

Code-Beispiel C#

```
bool updateFlag = true;
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
int counter = 0;

vcGantt1.SuspendUpdate(true);
if (updateFlag == true)
   {
   foreach (VcNode node in nodeCltn)
      {
      if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.07")) < 0)
         {
         node.set_DataField(13,"X");
         node.Update();
         counter = counter + 1;
         }
      }
   }
   else
   {
   foreach(VcNode node in nodeCltn)
      {
      if (DateTime.Compare(Convert.ToDateTime(node.get_DataField(2)),
Convert.ToDateTime("12.09.07")) < 0)
         {
         node.set_DataField(13,"");
         node.Update();
         counter = counter + 1;
         }
      }
   }
vcGantt1.SuspendUpdate(false);
```

UpdateLinkRecord

Methode von VcGantt

Mit dieser Methode können die Daten eines bestehenden Datensatzes einer Verbindung verändert werden. Die Verbindung wird durch den im Dialog **Datentabellen verwalten** festgelegten Primärschlüssel identifiziert. Diese Methode kommt zur Anwendung, wenn externe Änderungen der Daten in der Grafik nachvollzogen werden sollen. Wenn die Verbindung, die aktualisiert werden soll, noch nicht existiert, so wird sie neu angelegt.

	Datentyp	Beschreibung
Parameter:		
⇨ linkRecordContent	System.Object	Inhalt des Verbindungsdatensatzes
Rückgabewert	VcLink	Aktualisierte Verbindung

Code-Beispiel VB.NET

```
VcGantt1.UpdateLinkRecord("A100;A105;FS;0")
```

Code-Beispiel C#

```
vcGantt1.UpdateLinkRecord("1;A100;A105;FS;0");
```

UpdateNodeRecord

Methode von VcGantt

Mit dieser Methode können die Daten eines bestehenden Datensatzes eines Knotens verändert werden. Der Datensatz wird durch den im Dialog **Datentabellen verwalten** festgelegten Primärschlüssel identifiziert. Diese Methode kommt zur Anwendung, wenn externe Änderungen der Daten in der Grafik nachvollzogen werden sollen.

	Datentyp	Beschreibung
Parameter:		
⇨ nodeRecordContent	System.Object	Inhalt des Knotendatensatzes
Rückgabewert	VcNode	Datensatz des Knotens erfolgreich/nicht erfolgreich aktualisiert

Code-Beispiel VB.NET

```
VcGantt1.UpdateNodeRecord("A100;Activity 1;12.09.14;17.09.14;5;Planning")
```

Code-Beispiel C#

```
vcGantt1.UpdateNodeRecord("A100;Activity 1;12.09.07;17.09.07;5;Planning");
```

UpdateRowNumberFields

Methode von VcGantt

Diese Methode aktualisiert das Feld, in dem sich die Zeilennummern der Knoten befinden. Dieses Feld kann auf der Eigenschaftenseite **Knoten** unter **Zeilennummer-Feld** eingestellt werden. Die Anwendung dieser Methode ist nur dann sinnvoll, wenn weder eine hierarchische Darstellung noch eine Gruppierung gewählt wurde.

	Datentyp	Beschreibung
Rückgabewert	Void	

Code-Beispiel VB.NET

```
VcGantt1.UpdateRowNumberFields()
VcGantt1.SaveAs("c:\tmp\data.bar")
```

Code-Beispiel C#

```
vcGantt1.UpdateRowNumberFields();
vcGantt1.SaveAs(@"c:\tmp\data.bar");
```

Zoom

Methode von VcGantt

Mit dieser Methode wird die augenblickliche Bildschirmdarstellung um den angegebenen Wert prozentual vergrößert (Zoomfaktor > 100) oder verkleinert (Zoomfaktor < 100).

Siehe auch die VcGantt-Methode **FitChartIntoView()** und die Eigenschaft **ZoomFactor**.

	Datentyp	Beschreibung
Parameter:		
⇨ zoomFactor	System.Int16	relativer Zoomfaktor {11...999}, andere Werte bleiben unberücksichtigt
Rückgabewert	System.Boolean	Zoomen erfolgreich/nicht erfolgreich durchgeführt

Code-Beispiel VB.NET

```
VcGantt1.Zoom(120)
```

Code-Beispiel C#

```
vcGantt1.Zoom(120);
```

Ereignisse

KeyDown

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender eine Taste drückt, während VARCHART XGantt den Fokus hat. Mit Hilfe der Key-Ereignisse können Sie mit Hilfe der Tastatur Funktionen des VARCHART Windows Forms auslösen. (Zum Interpretieren von ANSI-Zeichen verwenden Sie das KeyPress-Ereignis.)

	Datentyp	Beschreibung
Eigenschaften:		
⇨ keyCode	System.Int16	Tasten-Code wie vbKeyF1 (F1-Taste) oder vbKeyHome (POS1-Taste)
⇨ 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 niederstwertigen 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.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_KeyDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles VcGantt1.KeyDown
   MsgBox("key pressed")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_KeyDown(object sender, System.Windows.Forms.KeyEventArgs
e)
   {
   MessageBox.Show("key pressed");
   }
```

KeyPress

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender eine ANSI-Taste drückt und wieder loslässt, während VARCHART XGantt den Fokus hat. Mit Hilfe der

Key-Ereignisse können Sie mit Hilfe der Tastatur Funktionen des VARCHART Windows Forms auslösen.

	Datentyp	Beschreibung
Eigenschaften:		
⇨ keyAscii	System.Int16	Eine Ganzzahl, die den numerischen Tasten-Code einer Standard-ANSI-Taste zurückgibt. KeyAscii wird als Referenz übergeben. Wird das Argument geändert, wird ein anderes Zeichen an das Objekt gesendet. Das Ändern von KeyAscii auf 0 hebt den Tastenanschlag auf, d.h. das Objekt erhält kein Zeichen.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_KeyPress(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyPressEventArgs) Handles VcGantt1.KeyPress
   MsgBox("key pressed and released")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_KeyPress(object sender,
System.Windows.Forms.KeyPressEventArgs e)
   {
   MessageBox.Show("key pressed and released");
   }
```

KeyUp

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender eine Taste loslässt, während VARCHART XGantt den Fokus hat. Mit Hilfe der Key-Ereignisse können Sie mit Hilfe der Tastatur Funktionen des VARCHART Windows Forms auslösen. (Zum Interpretieren von ANSI-Zeichen verwenden Sie das KeyPress-Ereignis.)

	Datentyp	Beschreibung
Eigenschaften:		
⇨ keyCode	System.Int16	Tasten-Code wie vbKeyF1 (F1-Taste) oder vbKeyHome (POS1-Taste)
⇨ 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 niederstwertigen 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.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_KeyUp(ByVal sender As Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles VcGantt1.KeyUp
   MsgBox("key released")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_KeyUp(object sender, System.Windows.Forms.KeyEventArgs e)
   {
   MessageBox.Show("key released");
   }
```

VcBoxCreated

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn das interaktive Anlegen einer Box abgeschlossen ist. Das Boxobjekt wird als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxCreatedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxCreatedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ box	VcBox	Angelegte Box

VcBoxCreating

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender interaktiv eine Box erzeugt.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcBoxCreated**.

Durch Setzen des Rückgabestatus kann die Anlage verhindert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxCreatingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxCreatingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ xOffset	System.Int32	x-Position der Box
⇨ yOffset	System.Int32	y-Position der Box
⇨ width	System.Int32	Breite der Box
⇨ height	System.Int32	Höhe der Box
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Box wird nicht angelegt. Die Box wird angelegt.

VcBoxLeftClicking

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ box	VcBox	Getroffene Box
⇨ x	System.Int32	X-Koordinate des Mauszeigers

⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcBoxLeftClicking(ByVal sender As Object, ByVal e As
NETRONICXGantt.VcBoxClickingEventArgs) Handles VcGantt1.VcBoxLeftClicking
   TextBox1.Text = e.Box.FieldText(1)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcBoxLeftClicking(object sender,
VcGanttLibVcBoxClickingEventArgs e)
   {
   textBox1.Text = e.Box.get_FieldText(1);
   }
```

VcBoxLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Box doppelklickt. Das getroffene VcBox-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter mitgegeben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ box	VcBox	Getroffene Box
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcBoxLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcBoxClickingEventArgs) Handles VcGantt1.VcBoxLeftDoubleClicking
   e.Box.FieldText(0) = TextBox1.Text
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcBoxLeftDoubleClicking(object sender,
VcGanttLib.VcBoxClickingEventArgs e)
   {
   e.Box.set_FieldText(1, textBox1.Text);
   }
```

VcBoxModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Modifizierung der Box abgeschlossen ist. Das veränderte VcBox-Objekt und der Modifikationstyp werden als Parameter mitgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ box	VcBox	Veränderte Box
⇨ modificationType	VcBoxModificationTypes	Art der Veränderung
	Mögliche Werte:	
	.vcBMTAnchoringModified 16	Verankerung der Box geändert
	.vcBMTAnything 1	beliebige Veränderung
	.vcBMTNothing 0	keine Veränderung
	.vcBMTSizeModified 8	Größe der Box verändert
	.vcBMTTextModified 4	Text der Box verändert
	.vcBMTXYOffsetModified 2	Offset verändert

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcBoxModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcBoxModifiedEventArgs) Handles VcGantt1.VcBoxModified
   MsgBox("The box has been modified")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcBoxModified(object sender,
NETRONIC.XGantt.VcBoxModifiedEventArgs e)
   {
   MessageBox.Show("The box has been modified");
   }
```

VcBoxModifying

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn eine Box interaktiv verändert wurde. Das veränderte VcBox-Objekt und der Modifikationstyp werden als Parameter mitgegeben.

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 kann die Änderung verhindert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxModifyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ box	VcBox	Veränderte Box
⇨ modificationType	VcBoxModificationTypes	Art der Veränderung
	Mögliche Werte:	
	.vcBMTAnchoringModified 16	Verankerung der Box geändert
	.vcBMTAnything 1	beliebige Veränderung
	.vcBMTNothing 0	keine Veränderung
	.vcBMTSizeModified 8	Größe der Box verändert
	.vcBMTTextModified 4	Text der Box verändert
	.vcBMTXYOffsetModified 2	Offset verändert
⇔ returnStatus	VcReturnStatus	Rückgabestatus

Mögliche Werte:	
.vcRetStatFalse 0	Die Veränderung wird rückgängig gemacht.
.vcRetStatOK 1	Die Veränderung wird durchgeführt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcBoxModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcBoxModifyingEventArgs) Handles VcGantt1.VcBoxModifying
   Select Case e.ModificationType
            Case VcBoxModificationTypes.vcBMTAnything :
               MsgBox("Box modification")
            Case VcBoxModificationTypes.vcBMTXYOffsetModified :
               MsgBox("Offset changed")
            Case VcBoxModificationTypes.vcBMTTextModified :
               MsgBox("Box field text changed")
        End Select
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcBoxModifying(obje!ct sender,
NETRONIC.XGantt.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 VcGantt

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 Cursors als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcBoxClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcBoxClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ box	VcBox	Getroffene Box
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcBoxRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcBoxClickingEventArgs) Handles VcGantt1.VcBoxRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcBoxRightClicking(object sender,
NETRONIC.XGantt.VcBoxClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   }
```

VcCalendarGridRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf eine Kalendergitter klickt. Das getroffene Objekt wird zusammen mit der Position (x,y-Koordinaten) des Cursors als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

Dieses Ereignis wird nur ausgelöst, wenn das Kalendergitter identifizierbar ist, d.h. wenn die Kalendergitter-Eigenschaft **Identifiable** auf **True** gesetzt wurde.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCalendarGridClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCalendarGridClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ calendarGrid	VcCalendarGrid	Getroffenes Kalendergitter
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCalendarGridRightClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcCalendarGridClickingEventArgs) Handles
VcGantt1.VcCalendarGridRightClicking
   MsgBox(e.CalendarGrid.Name)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCalendarGridRightClicking(object sender,
NETRONIC.XGantt.VcCalendarGridClickingEventArgs e)
   {
   MessageBox.Show(e.CalendarGrid.Name);
   }
```

VcComponentScrolled

Ereignis von VcGantt

Mit diesem Ereignis können Sie bei jeder interaktiven Scrollaktion Folgendes ermitteln:

1. die Komponente, die gescrollt wird (nur vcDiagramComponent, vcHistogramComponent, vcListComponent und vcRightListComponent werden als „Masterscroller“ berücksichtigt, da alle anderen Komponenten abhängig mitgescrollt werden)

2. die Bewegungsorientierung (horizontal oder vertikal)

3. die Art der Benutzeraktion.

Hinweis: Die tatsächliche Scrollaktion ergibt sich aus der Kombination der Parameter **orientation** und **scrollAction**, da in Windows-Programmen die up/left- und down/right-Aktionen jeweils dieselbe Nummer haben. Z. B.:

vcScrollActionSBPageLeft = vcScrollActionSBPageUp = 2

vcScrollActionThumbTrackLeft = vcScrollActionThumbTrackUp = 107

Das folgende Beispiel zeigt die Auswertung über **orientation** für **VcScrollActionSBPageLeft** und **vcScrollActionSBPageUp**, die beide den Wert 2 haben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcComponentScrolledEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcComponentScrolledEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ component	VcComponentType	Komponententyp
	Mögliche Werte:	
	.vcAdditionalListComponent 1	zusätzliche Tabelle
	.vcBottomListTitleComponent 14	untere Titelleiste
	.vcBottomRightListTitleComponent 17	Tabellenbereich unten rechts
	.vcBottomTimeScaleComponent 15	untere Zeitskala
	.vcDiagramComponent 4	Diagramm
	.vcHistogramComponent 8	Histogramm
	.vcHistogramVerScaleComponent 7	numerische Skala (vertikale Histogramm-skala)
	.vcLegendComponent 10	Legende (zur Zeit ohne Funktion; Rückgabewerte 00)
	.vcListComponent 0	Tabelle
	.vcListTitleComponent 2	Tabellenüberschrift
	.vcRightListComponent 5	rechte Tabelle
	.vcRightListTitleComponent 16	Tabellenüberschrift der rechten Tabelle
	.vcTimeScaleComponent 3	obere Zeitskala
	.vcTopTitleComponent 11	obere Titelleiste
⇨ orientation	VcScrollOrientation	Scrollrichtung
	Mögliche Werte:	
	.vcHorizontal 1	horizontales Scrollen
	.vcVertical 2	vertikales Scrollen
⇨ scrollAction	VcScrollAction	Art des Scrollvorgangs
	Mögliche Werte:	
	.vcScrollActionAutoscrollDown 102	Die Ansicht verschob sich automatisch nach unten.
	.vcScrollActionAutoscrollLeft 101	Die Ansicht verschob sich automatisch nach links.
	.vcScrollActionAutoscrollRight 102	Die Ansicht verschob sich automatisch nach rechts.
	.vcScrollActionAutoscrollUp 101	Die Ansicht verschob sich automatisch nach oben.

.vcScrollActionMouseWheelDown 106	Bei gedrücktem Mausrad wurde die Maus nach unten geschoben.
.vcScrollActionMouseWheelLeft 105	Bei gedrücktem Mausrad wurde die Maus nach links geschoben.
.vcScrollActionMouseWheelRight 106	Bei gedrücktem Mausrad wurde die Maus nach rechts geschoben.
.vcScrollActionMouseWheelUp 105	Bei gedrücktem Mausrad wurde die Maus nach oben geschoben.
.vcScrollActionSBLineDown 1	Die Ansicht verschob sich zur unteren Begrenzungslinie.
.vcScrollActionSBLineLeft 0	Die Ansicht verschob sich zur linken Begrenzungslinie.
.vcScrollActionSBLineRight 1	Die Ansicht verschob sich zur rechten Begrenzungslinie.
.vcScrollActionSBLineUp 0	Die Ansicht verschob sich zur oberen Begrenzungslinie.
.vcScrollActionSBNothing -1	Die Ansicht wurde nicht verschoben.
.vcScrollActionSBPageDown 3	Die Ansicht wurde um eine Seite nach unten versetzt.
.vcScrollActionSBPageLeft 2	Die Ansicht wurde um eine Seite nach links versetzt.
.vcScrollActionSBPageRight 3	Die Ansicht wurde um eine Seite nach rechts versetzt.
.vcScrollActionSBPageUp 2	Die Ansicht wurde um eine Seite nach oben versetzt
.vcScrollActionSBThumbPosition 4	Die Versetzung um einen Schritt ist abgeschlossen.
.vcScrollActionSBThumbTrack 5	Die Ansicht wurde um einen Schritt versetzt.
.vcScrollActionScrollEnd 104	Scrollen über die Taste **Ende** oder über das Kontextmenü an das Diagrammende (rechts unten)
.vcScrollActionScrollHome 103	Scrollen über die Taste **Pos 1** oder über das Kontextmenü an den Diagrammanfang (links oben)
.vcScrollActionThumbTrackDown 108	Thumb (Balken im Scrollbar) nach unten geschoben
.vcScrollActionThumbTrackLeft 107	Thumb (Balken im Scrollbar) nach links geschoben
.vcScrollActionThumbTrackRight 108	Thumb (Balken im Scrollbar) nach rechts geschoben
.vcScrollActionThumbTrackUp 107	Thumb (Balken im Scrollbar) nach oben geschoben

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcComponentScrolled(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcComponentScrolledEventArgs) Handles
VcGantt1.VcComponentScrolled
   If e.ScrollOrientation = VcScrollOrientation.vcHorizontal And e.ScrollAction
= VcScrollAction.vcScrollActionSBPageLeft Then
      MsgBox("Scrolled left")
   ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageRight Then
       MsgBox("Scrolled right")
   ElseIf e.ScrollOrientation = VcScrollOrientation.vcVertical And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageUp Then
        MsgBox("Scrolled up")
    ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageDown Then
         MsgBox("Scrolled down")
    End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcComponentScrolled(object sender,
NETRONIC.XGantt.VcComponentScrolledEventArgs e)
   {
   if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal && e.ScrollAction
== VcScrollAction.vcScrollActionSBPageLeft)
      MessageBox.Show("Scrolled left");
   else if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageRight)
      MessageBox.Show("Scrolled right");
   else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageUp)
      MessageBox.Show("Scrolled up");
   else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageDown)
      MessageBox.Show("Scrolled down");
   }
```

VcComponentScrolling

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn Sie einen Scrollvorgang angefordert haben, aber bevor der integrierte Scroll-Automatismus ausgeführt wird.

Mit diesem Ereignis können Sie bei jeder interaktiven Scrollaktion Folgendes ermitteln:

1. die Komponente, die gescrollt wird (nur vcDiagramComponent, vcHistogramComponent, vcListComponent und vcRightListComponent werden als „Masterscroller“ berücksichtigt, da alle anderen Komponenten abhängig mitgescrollt werden)

2. die Bewegungsorientierung (horizontal oder vertikal)

3. die Art der Benutzeraktion.

Setzt man den Rückgabestatus auf **vcRetStatFalse**, so wird der integrierte Scroll-Automatismus unterdrückt, und Sie können in Ihrer Anwendung selbst auf das Ereignis reagieren.

Hinweis: Die tatsächliche Scrollaktion ergibt sich aus der Kombination der Parameter **orientation** und **scrollAction**, da in Windows-Programmen die up/left- und down/right-Aktionen jeweils dieselbe Nummer haben. Z. B.:

vcScrollActionSBPageLeft = vcScrollActionSBPageUp = 2

vcScrollActionThumbTrackLeft = vcScrollActionThumbTrackUp = 107

Das folgende Beispiel zeigt die Auswertung über **orientation** für **VcScrollActionSBPageLeft** und **vcScrollActionSBPageUp**, die beide den Wert 2 haben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcComponentScrollingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcComponentScrollingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ component	VcComponentType	Komponententyp
	Mögliche Werte:	
	.vcAdditionalListComponent 1	zusätzliche Tabelle
	.vcBottomListTitleComponent 14	untere Titelleiste
	.vcBottomRightListTitleComponent 17	Tabellenbereich unten rechts
	.vcBottomTimeScaleComponent 15	untere Zeitskala
	.vcDiagramComponent 4	Diagramm
	.vcHistogramComponent 8	Histogramm
	.vcHistogramVerScaleComponent 7	numerische Skala (vertikale Histogrammskala)
	.vcLegendComponent 10	Legende (zur Zeit ohne Funktion; Rückgabewerte 00)
	.vcListComponent 0	Tabelle
	.vcListTitleComponent 2	Tabellenüberschrift
	.vcRightListComponent 5	rechte Tabelle
	.vcRightListTitleComponent 16	Tabellenüberschrift der rechten Tabelle
	.vcTimeScaleComponent 3	obere Zeitskala
	.vcTopTitleComponent 11	obere Titelleiste
⇨ histogramsHeightRatio		Höhenverhältnis des Histogramms zum Gesamtdiagramm
⇨ orientation	VcScrollOrientation	Scrollrichtung
	Mögliche Werte:	
	.vcHorizontal 1	horizontales Scrollen
	.vcVertical 2	vertikales Scrollen
⇨ scrollAction	VcScrollAction	Art des Scrollvorgangs
	Mögliche Werte:	
	.vcScrollActionAutoscrollDown 102	Die Ansicht verschob sich automatisch nach unten.
	.vcScrollActionAutoscrollLeft 101	Die Ansicht verschob sich automatisch nach links.
	.vcScrollActionAutoscrollRight 102	Die Ansicht verschob sich automatisch nach rechts.
	.vcScrollActionAutoscrollUp 101	Die Ansicht verschob sich automatisch nach oben.
	.vcScrollActionMouseWheelDown 106	Bei gedrücktem Mausrad wurde die Maus nach unten geschoben.
	.vcScrollActionMouseWheelLeft 105	Bei gedrücktem Mausrad wurde die Maus nach links geschoben.

	.vcScrollActionMouseWheelRight 106	Bei gedrücktem Mausrad wurde die Maus nach rechts geschoben.
	.vcScrollActionMouseWheelUp 105	Bei gedrücktem Mausrad wurde die Maus nach oben geschoben.
	.vcScrollActionSBLineDown 1	Die Ansicht verschob sich zur unteren Begrenzungslinie.
	.vcScrollActionSBLineLeft 0	Die Ansicht verschob sich zur linken Begrenzungslinie.
	.vcScrollActionSBLineRight 1	Die Ansicht verschob sich zur rechten Begrenzungslinie.
	.vcScrollActionSBLineUp 0	Die Ansicht verschob sich zur oberen Begrenzungslinie.
	.vcScrollActionSBNothing -1	Die Ansicht wurde nicht verschoben.
	.vcScrollActionSBPageDown 3	Die Ansicht wurde um eine Seite nach unten versetzt.
	.vcScrollActionSBPageLeft 2	Die Ansicht wurde um eine Seite nach links versetzt.
	.vcScrollActionSBPageRight 3	Die Ansicht wurde um eine Seite nach rechts versetzt.
	.vcScrollActionSBPageUp 2	Die Ansicht wurde um eine Seite nach oben versetzt
	.vcScrollActionSBThumbPosition 4	Die Versetzung um einen Schritt ist abgeschlossen.
	.vcScrollActionSBThumbTrack 5	Die Ansicht wurde um einen Schritt versetzt.
	.vcScrollActionScrollEnd 104	Scrollen über die Taste **Ende** oder über das Kontextmenü an das Diagrammende (rechts unten)
	.vcScrollActionScrollHome 103	Scrollen über die Taste **Pos 1** oder über das Kontextmenü an den Diagrammanfang (links oben)
	.vcScrollActionThumbTrackDown 108	Thumb (Balken im Scrollbar) nach unten geschoben
	.vcScrollActionThumbTrackLeft 107	Thumb (Balken im Scrollbar) nach links geschoben
	.vcScrollActionThumbTrackRight 108	Thumb (Balken im Scrollbar) nach rechts geschoben
	.vcScrollActionThumbTrackUp 107	Thumb (Balken im Scrollbar) nach oben geschoben
⇨ delta	System.Int32	Länge des Scrollvorgangs (in Pixeln)
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcComponentScrolling(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcComponentScrollingEventArgs) Handles
VcGantt1.VcComponentScrolling
   If e.ScrollOrientation = VcScrollOrientation.vcHorizontal And e.ScrollAction
= VcScrollAction.vcScrollActionSBPageLeft Then
      MsgBox("Scrolled left")
   ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageRight Then
      MsgBox("Scrolled right")
   ElseIf e.ScrollOrientation = VcScrollOrientation.vcVertical And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageUp Then
      MsgBox("Scrolled up")
   ElseIf e.ScrollOrientation = VcScrollOrientation.vcHorizontal And
e.ScrollAction = VcScrollAction.vcScrollActionSBPageDown Then
      MsgBox("Scrolled down")
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcComponentScrolling(object sender,
NETRONIC.XGantt.VcComponentScrollingEventArgs e)
   {
   if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal && e.ScrollAction
== VcScrollAction.vcScrollActionSBPageLeft)
      MessageBox.Show("Scrolled left");
   else if (e.ScrollOrientation == VcScrollOrientation.vcHorizontal &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageRight)
      MessageBox.Show("Scrolled right");
   else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageUp)
      MessageBox.Show("Scrolled up");
   else if (e.ScrollOrientation == VcScrollOrientation.vcVertical &&
e.ScrollAction == VcScrollAction.vcScrollActionSBPageDown)
      MessageBox.Show("Scrolled down");
   }
```

VcCurveLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Histogrammkurve klickt, und bevor die Kurve markiert wird. Durch Setzen des VcReturnStatus auf **vcRetStatFalse** kann das Markieren der Kurve verhindert werden. Trotzdem können die Kurvenwerte verändert werden. Eine Möglichkeit zur Unterdrückung gibt es zur Zeit nicht. Das getroffene Kurvenobjekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurveClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurveClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Histogrammkurve
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Kurve wird nicht markiert. Die Kurve wird markiert.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurveLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurveClickingEventArgs) Handles VcGantt1.VcCurveLeftClicking
   e.Curve.LineColor = Color.Blue
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurveLeftClicking(object sender,
NETRONIC.XGantt.VcCurveClickingEventArgs e)
   {
   e.Curve.LineColor = Color.LightSteelBlue;
   }
```

VcCurveLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender auf eine Histogrammkurve mit der linken Maustaste doppelklickt. Das getroffene VcCurve-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurveClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurveClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Histogrammkurve
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ 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 VcGantt1_VcCurveLeftDoubleClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcCurveClickingEventArgs) Handles
VcGantt1.VcCurveLeftDoubleClicking
   Call MsgBox("x: " + e.X.ToString() + "y: " + e.Y.ToString())
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurveLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcCurveClickingEventArgs e)
   {
   MessageBox.Show("x: " + e.X.ToString() + " y: " + e.Y.ToString());
   }
```

VcCurveModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Modifizierung der angegebenen Kurve abgeschlossen ist.

Das Kurven-Objekt wird als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcCurve	Veränderte Kurve

VcCurveModifying

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn eine Histogrammkurve interaktiv verändert wurde. Das gilt sowohl für das interaktive Verändern von über die API gesetzten Histogrammkurven, als auch für das Verändern von layergenerierten Histogrammkurven durch das Verschieben von Knoten. Die veränderte Kurve, der Anfang und das Ende des Bereiches, in dem sie verändert wurde, sowie der Wert, um den der Kurvenpunkt in y-Richtung verändert wurde, werden als Parameter zurückgegeben. Die Art der veränderten Kurve kann über die VcCurve-Eigenschaft **CurveSource** erfragt werden.

Hinweis: Für jeden veränderten Layer, der zu einer Veränderung einer layergenerierten Kurve beiträgt, kommt das Ereignis **VcCurveModifying** je zweimal (einmal für den Anfangsbereich und einmal für den Endbereich der Veränderung).

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcCurveModified**.

Durch Setzen des Rückgabestatus kann die Änderung verhindert werden.

Hinweis: Mit **VcCurveModifyingEx** kann der y-Wert auch ein Fließkomma-Wert sein.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurveModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurveModifyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ curve	VcCurve	Veränderte Kurve
⇨ date1	System.DateTime	Anfang des Bereiches, in dem die Kurve verändert wurde
⇨ date2	System.DateTime	Ende des Bereiches, in dem die Kurve verändert wurde

⇨ increment	System.Int32	Wert, um den die Kurve in y-Richtung verändert wurde
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Veränderung wird rückgängig gemacht. Die Veränderung wird durchgeführt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurveModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurveModifyingEventArgs) Handles VcGantt1.VcCurveModifying
   Select Case e.Curve.CurveSource
      Case VcCurveSource.vcCalculateFromLayer
         MsgBox("The curve is calculated from layers. Increment:" +
e.Increment.ToString() + " Changed start date: " + e.LeftDate + "Changed end
date: " + e.RightDate)
      Case VcCurveSource.vcSetCurve
         MsgBox("Curve set via API. Increment:" + e.Increment.ToString() + "
Changed start date: " + e.LeftDate + "Changed end date: " + e.RightDate)
    End Select
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurveModifying(object sender,
NETRONIC.XGantt.VcCurveModifyingEventArgs e)
   {
   switch (e.Curve.CurveSource)
      {
      case VcCurveSource.vcCalculateFromLayer:
         MessageBox.Show("The curve is calculated from layers. Increment: " +
e.Increment.ToString() + " Changed start date: " + e.LeftDate + " Changed end
date: " + e.RightDate);
         break;
      case VcCurveSource.vcSetCurve:
         MessageBox.Show("Curve set via API. Increment: " +
e.Increment.ToString() + " Changed start date: " + e.LeftDate + " Changed end
date: " + e.RightDate);
         break;
       }
   }
```

VcCurveModifyingEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn eine Histogrammkurve interaktiv verändert wurde. Das gilt sowohl für das interaktive Verändern von über die API gesetzten Histogrammkurven, als auch für das Verändern von layergenerierten Histogrammkurven durch das Verschieben von Knoten. Die veränderte Kurve, der Anfang und das Ende des Bereiches, in dem sie verändert wurde, sowie der Wert, um den der Kurvenpunkt in y-Richtung verändert wurde, werden als Parameter zurückgegeben. Die Art der veränderten Kurve kann über die VcCurve-Eigenschaft **CurveSource** erfragt werden.

Hinweis: Für jeden veränderten Layer, der zu einer Veränderung einer layergenerierten Kurve beiträgt, kommt das Ereignis **VcCurveModifying** je zweimal (einmal für den Anfangsbereich und einmal für den Endbereich der Veränderung).

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcCurveModified**.

Durch Setzen des Rückgabestatus kann die Änderung verhindert werden.

Hinweis: Im Vergleich zum Ereignis **VcCurveModifying** kann hier der Parameter **increment** als Fließkommazahl übergeben werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurveModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurveModifyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ curve	VcCurve	Veränderte Kurve
⇨ leftDate	System.DateTime	Anfang des Bereiches, in dem die Kurve verändert wurde
⇨ rightDate	System.DateTime	Ende des Bereiches, in dem die Kurve verändert wurde
⇨ increment	System.Int32	Wert, um den die Kurve in y-Richtung verändert wurde
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Veränderung wird rückgängig gemacht. Die Veränderung wird durchgeführt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurveModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurveModifyingEventArgs) Handles VcGantt1.VcCurveModifying
   Select Case e.Curve.CurveSource
      Case VcCurveSource.vcCalculateFromLayer
         MsgBox("The curve is calculated from layers. Increment:" +
e.Increment.ToString() + " Changed start date: " + e.LeftDate + "Changed end
date: " + e.RightDate)
      Case VcCurveSource.vcSetCurve
         MsgBox("Curve set via API. Increment:" + e.Increment.ToString() + "
Changed start date: " + e.LeftDate + "Changed end date: " + e.RightDate)
    End Select
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurveModifying(object sender,
NETRONIC.XGantt.VcCurveModifyingEventArgs e)
   {
   switch (e.Curve.CurveSource)
      {
      case VcCurveSource.vcCalculateFromLayer:
         MessageBox.Show("The curve is calculated from layers. Increment: " +
e.Increment.ToString() + " Changed start date: " + e.LeftDate + " Changed end
date: " + e.RightDate);
         break;
      case VcCurveSource.vcSetCurve:
         MessageBox.Show("Curve set via API. Increment: " +
e.Increment.ToString() + " Changed start date: " + e.LeftDate + " Changed end
date: " + e.RightDate);
         break;
       }
   }
```

VcCurvePointDeleting

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs einen Punkt einer über die API gesetzten Histogrammkurve löscht. Die getroffene Histogrammkurve, das Datum und der y-Wert des gelöschten Kurvenpunkts werden als Parameter zurückgegeben, so dass noch eine Überprüfung vorgenommen werden kann. Durch Setzen des Rückgabestatus kann der Löschvorgang verhindert werden.

Hinweis: Mit **VcCurvePointDeletingEx** kann der y-Wert auch als Fließkomma-Wert übergeben werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurvePointDeletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurvePointDeletingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Histogrammkurve
⇨ pointDate	System.DateTime	Datum des gelöschten Kurvenpunkts
⇨ value	System.Int32	y-Wert des gelöschten Kurvenpunkts
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der Kurvenpunkt wird nicht gelöscht. Der Kurvenpunkt wird gelöscht.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurvePointDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurvePointDeletingEventArgs) Handles
VcGantt1.VcCurvePointDeleting
   If MsgBox("Do you want to delete this curve point (date:" + e.PointDate + ",
y value:" + e.Value + ")?", MsgBoxStyle.YesNo, "Deleting curve point") =
MsgBoxResult.No Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurvePointDeleting(object sender,
NETRONIC.XGantt.VcCurvePointDeletingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to delete this curve
point (date:" + e.PointDate + ", y value:" + e.Value + ")?", "Deleting curve
point", MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcCurvePointDeletingEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs einen Punkt einer über die API gesetzten Histogrammkurve löscht. Die getroffene Histogrammkurve, das Datum und der y-Wert des gelöschten Kurvenpunkts werden als Parameter zurückgegeben, so dass noch eine Überprüfung vorgenommen werden kann. Durch Setzen des Rückgabestatus kann der Löschvorgang verhindert werden.

Hinweis: Im Vergleich zum Ereignis **VcCurvePointDeleting** kann hier der Parameter **value** als Fließkomma-Zahl übergeben werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurvePointDeletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurvePointDeletingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Histogrammkurve
⇨ pointDate	System.DateTime	Datum des gelöschten Kurvenpunkts
⇨ value	System.Int32	y-Wert des gelöschten Kurvenpunkts
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der Kurvenpunkt wird nicht gelöscht. Der Kurvenpunkt wird gelöscht.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurvePointDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurvePointDeletingEventArgs) Handles
VcGantt1.VcCurvePointDeleting
   If MsgBox("Do you want to delete this curve point (date:" + e.PointDate + ",
y value:" + e.Value + ")?", MsgBoxStyle.YesNo, "Deleting curve point") =
MsgBoxResult.No Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurvePointDeleting(object sender,
NETRONIC.XGantt.VcCurvePointDeletingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to delete this curve
point (date:" + e.PointDate + ", y value:" + e.Value + ")?", "Deleting curve
point", MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcCurvePointInserting

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs im Histogramm den Einfügemodus eingeschaltet und dann mit einem Klick der linken Maustaste auf eine Histogrammkurve einen Kurvenpunkt in eine über

die API gesetzte Histogrammkurve eingefügt hat. Die getroffene Histogrammkurve, das Datum und der y-Wert des eingefügten Kurvenpunkts werden als Parameter zurückgegeben, so dass beispielsweise noch eine Überprüfung vorgenommen werden kann. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird das Einfügen des Kurvenpunkts wieder rückgängig gemacht.

Hinweis: Mit **VcCurvePointInsertingEx** kann der y-Wert auch ein Fließkomma-Wert sein.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurvePointInsertingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurvePointInsertingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Histogrammkurve
⇨ pointDate	System.DateTime	Datum des eingefügten Kurvenpunkts
⇨ value	System.Int32	y-Wert des eingefügten Kurvenpunkts
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcCurvePointInserting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurvePointInsertingEventArgs) Handles
VcGantt1.VcCurvePointInserting
   If MsgBox("Do you want to insert this curve point (date: " + e.PointDate + "
y value: " + e.Value + ")?", MsgBoxStyle.YesNo, "Deleting curve point") =
MsgBoxResult.No Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurvePointInserting(object sender,
NETRONIC.XGantt.VcCurvePointInsertingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to insert this curve point
(date:" + e.PointDate + ", y value:" + e.Value + ")?", "Inserting curve point",
MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcCurvePointInsertingEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs im Histogramm den Einfügemodus eingeschaltet und dann mit einem Klick der linken Maustaste auf eine Histogrammkurve einen Kurvenpunkt in eine über die API gesetzte Histogrammkurve eingefügt hat. Die getroffene Histogrammkurve, das Datum und der y-Wert des eingefügten Kurvenpunkts werden als Parameter zurückgegeben, so dass beispielsweise noch eine Überprüfung vorgenommen werden kann. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird das Einfügen des Kurvenpunkts wieder rückgängig gemacht.

Hinweis: Im Vergleich zu **VcCurvePointInserting** kann der y-Wert auch ein Fließkomma-Wert sein.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurvePointInsertingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurvePointInsertingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Histogrammkurve
⇨ pointDate	System.DateTime	Datum des eingefügten Kurvenpunkts
⇨ value	System.Int32	y-Wert des eingefügten Kurvenpunkts
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcCurvePointInserting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurvePointInsertingEventArgs) Handles
VcGantt1.VcCurvePointInserting
   If MsgBox("Do you want to insert this curve point (date: " + e.PointDate + "
y value: " + e.Value + ")?", MsgBoxStyle.YesNo, "Deleting curve point") =
MsgBoxResult.No Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

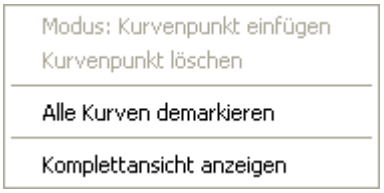
Code-Beispiel C#

```
private void vcGantt1_VcCurvePointInserting(object sender,
NETRONIC.XGantt.VcCurvePointInsertingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to insert this curve point
(date:" + e.PointDate + ", y value:" + e.Value + ")?", "Inserting curve point",
MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcCurveRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf eine Histogrammkurve klickt. Das getroffene Kurvenobjekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.



Oben: integriertes Kontextmenü

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcCurveClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcCurveClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ curve	VcCurve	Getroffene Kurve
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcCurveRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcCurveClickingEventArgs) Handles VcGantt1.VcCurveRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcCurveRightClicking(object sender,
NETRONIC.XGantt.VcCurveClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcDataModified

Ereignis von VcGantt

Dieses Ereignis tritt immer dann auf, wenn Daten interaktiv im Chart verändert werden, also explizit nach den folgenden Ereignissen:

- VcBoxModified
- VcCurveModifying
- VcCurvePointDeleting
- VcGroupModified
- 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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇦ (no parameter)		Kein Parameter

VcDataRecordCreated

Ereignis von VcGantt

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. **VcNode-Created** und **VcLinkCreated**).

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataRecordCreatedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordCreatedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ dataRecord	VcDataRecord	Angelegtes Datensatz-Objekt
⇨ creationType	VcCreationType	Typ des Anlegens von Datensätzen
	Mögliche Werte:	
	.vcDataRecordCreated 6	Datensatz wurde durch Interaktion angelegt
	.vcDataRecordCreatedByResourceScheduling 5	Datensatz wurde automatisch durch Ressourcenplanung angelegt
	.vcLinkCreated 2	Verbindung wurde durch Interaktion angelegt
	.vcNodeCreated 1	Knoten durch "Stempeln" angelegt
⇨ isLast	System.Boolean	**True**: Angelegter Datensatz ist der einzige oder der letzte Datensatz einer Menge
		False: Angelegter Datensatz ist nicht der einzige oder der letzte Datensatz einer Menge

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDataRecordCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDataRecordCreatedEventArgs) Handles
VcGantt1.VcDataRecordCreated
        MsgBox(e.DataRecord.AllData)
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDataRecordCreated(object sender,
NETRONIC.XGantt.VcDataRecordCreatedEventArgs e)
   {
   MessageBox.Show(e.DataRecord.AllData.ToString());
   }
```

VcDataRecordCreating

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataRecordCreatingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordCreatingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ 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 VcVcGantt1_VcDataRecordCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDataRecordCreatedEventArgs) Handles
VcVcGantt1.VcDataRecordCreated
        MsgBox(e.DataRecord.AllData)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDataRecordCreated(object sender,
NETRONIC.XGantt.VcDataRecordCreatedEventArgs e)
{
MessageBox.Show(e.DataRecord.AllData.ToString());
}
```

VcDataRecordDeleted

Ereignis von VcGantt

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**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataRecordDeletedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordDeletedEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ dataRecord	VcDataRecord	Gelöschter Datensatz
⇨ isLast	System.Boolean	**True**: Gelöschter Datensatz ist der einzige oder der letzte Datensatz einer Menge **False**: Gelöschter Datensatz ist nicht der einzige oder der letzte Datensatz einer Menge

VcDataRecordDeleting

Ereignis von VcGantt

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**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataRecordDeletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordDeletingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ dataRecord	VcDataRecord	Gelöschtes Datensatz-Objekt
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der Datensatz wird nicht gelöscht. Der Datensatz wird gelöscht.

Code-Beispiel VB.NET

```
Private Sub VcVcGantt1_VcDataRecordDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDataRecordDeletingEventArgs) Handles
VcVcGantt1.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 vcGantt1_VcDataRecordDeleting(object sender,
NETRONIC.XGantt.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 VcGantt

Dieses Ereignis tritt ein, wenn die Modifizierung des Datensatzes abgeschlossen ist.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataRecordModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ dataRecord	VcDataRecord	Veränderter Datensatz

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDataRecordModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDataRecordModifiedEventArgs) Handles
VcGantt1.VcDataRecordModified
   MsgBox("The data record has been modified")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDataRecordModified(object sender,
NETRONIC.XGantt.VcDataRecordModifiedEventArgs e)
   {
   MessageBox.Show("The data record has been modified");
   }
```

VcDataRecordModifying

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDataRecordModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDataRecordModifyingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ dataRecord	VcDataRecord	Veränderter Datensatz
⇨ modificationType	VcModificationTypes	Art der Veränderung
	Mögliche Werte:	
	.vcAnything 1	Änderungstyp nicht näher bestimmt
	.vcChangedGroup 16	Zuordnung des Knotens zu einer Gruppe wurde verändert (nur für Knoten).
	.vcEndModified 4	Ende des Knotens wurde verändert (nur für Knoten).
	.vcHierarchyModified 64	Hierarchie der Knoten wurde verändert
	.vcModifiedByResourceScheduling 128	Änderung durch Ressourcenplanung (nur für Datensatz-Objekt)
	.vcModifiedBySchedule 32	Änderung durch neue Zeitberechnung
	.vcMoved 8	Objekt wurde verschoben.
	.vcNothing 0	Keine Änderung
	.vcStartModified 2	Anfang des Knotens wurde verändert (nur für Knoten).
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.vcRetStatFalse 0	Die Veränderung wird rückgängig gemacht.
	.vcRetStatOK 1	Die Veränderung wird durchgeführt.

VcDataRecordNotFound

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	

⇨ e	VcDataRecordNotFoundEventArgs

	Datentyp	Beschreibung
Eigenschaften:		
⇦ index	System.Int32	Index des Feldes, das den Schlüssel des abhängigen Datensatzes enthält

VcDateLineModifying

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn eine Stichtaglinie durch den Anwender interaktiv verschoben wurde. Die veränderte Stichtaglinie wird als Parameter zurückgegeben.

Durch Setzen des Rückgabestatus kann die Änderung verhindert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDateLineModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDateLineModifyingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ dateLine	VcDateLine	Stichtaglinie
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Veränderung wird rückgängig gemacht. Die Veränderung wird durchgeführt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDateLineModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDateLineModifyingEventArgs) Handles
VcGantt1.VcDateLineModifying
   MsgBox(e.DateLine.Date)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDateLineModifying(object sender,
NETRONIC.XGantt.VcDateLineModifyingEventArgs e)
   {
   MessageBox.Show(e.DateLine.Date.ToString());
   }
```

VcDateLineRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf eine Stichtaglinie klickt. Das getroffene Objekt wird zusammen mit der Position (x,y-Koordinaten) des Cursors als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDateLineClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDateLineClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ dateLine	VcDateLine	Getroffene Terminlinie
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDateLineRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDateLineClickingEventArgs) Handles
VcGantt1.VcDateLineRightClicking
   MsgBox(e.DateLine.Name)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDateLineRightClicking(object sender,
NETRONIC.XGantt.VcDateLineClickingEventArgs e)
   {
   MessageBox.Show(e.DateLine.Name);
   }
```

VcDateShowing

Ereignis von VcGantt

Dieses Ereignis tritt bei jeder Mausbewegung im Diagramm- oder Zeitskalenbereich ein. Es wird das Datum der Mausposition als Parameter zurückgeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDateShowingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDateShowingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ dateVal	System.DateTime	Datum

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDateShowing(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDateShowingEventArgs) Handles VcGantt1.VcDateShowing
   TextBox1.Text = e.CurDate
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDateShowing(object sender,
NETRONIC.XGantt.VcDateShowingEventArgs e)
   {
   textBox1.Text = e.CurDate.ToString();
   }
```

VcDiagramHorizontalScrolled

Ereignis von VcGantt

Dieses Ereignis tritt ein, nachdem der Scrollvorgang ausgeführt wurde. Das neue Anfangs- und Enddatum des sichtbaren Diagrammbereichs werden

übergeben. Anhand des Parameters **scrollAction** erhalten Sie Informationen darüber, welcher Art der ausgeführte Scrollvorgang war.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDiagramHorizontalScrolledEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDiagramHorizontalScrolledEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ newStartDate	System.DateTime	Neues Anfangsdatum des sichtbaren Diagrammbereiches
⇨ newEndDate	System.DateTime	Neues Enddatum des sichtbaren Diagrammbereiches
⇨ scrollAction	VcScrollAction	Art des Scrollvorgangs
	Mögliche Werte:	
	.vcScrollActionAutoscrollDown 102	Die Ansicht verschob sich automatisch nach unten.
	.vcScrollActionAutoscrollLeft 101	Die Ansicht verschob sich automatisch nach links.
	.vcScrollActionAutoscrollRight 102	Die Ansicht verschob sich automatisch nach rechts.
	.vcScrollActionAutoscrollUp 101	Die Ansicht verschob sich automatisch nach oben.
	.vcScrollActionMouseWheelDown 106	Bei gedrücktem Mausrad wurde die Maus nach unten geschoben.
	.vcScrollActionMouseWheelLeft 105	Bei gedrücktem Mausrad wurde die Maus nach links geschoben.
	.vcScrollActionMouseWheelRight 106	Bei gedrücktem Mausrad wurde die Maus nach rechts geschoben.
	.vcScrollActionMouseWheelUp 105	Bei gedrücktem Mausrad wurde die Maus nach oben geschoben.
	.vcScrollActionSBLineDown 1	Die Ansicht verschob sich zur unteren Begrenzungslinie.
	.vcScrollActionSBLineLeft 0	Die Ansicht verschob sich zur linken Begrenzungslinie.
	.vcScrollActionSBLineRight 1	Die Ansicht verschob sich zur rechten Begrenzungslinie.
	.vcScrollActionSBLineUp 0	Die Ansicht verschob sich zur oberen Begrenzungslinie.
	.vcScrollActionSBNothing -1	Die Ansicht wurde nicht verschoben.
	.vcScrollActionSBPageDown 3	Die Ansicht wurde um eine Seite nach unten versetzt.
	.vcScrollActionSBPageLeft 2	Die Ansicht wurde um eine Seite nach links versetzt.
	.vcScrollActionSBPageRight 3	Die Ansicht wurde um eine Seite nach rechts versetzt.
	.vcScrollActionSBPageUp 2	Die Ansicht wurde um eine Seite nach oben versetzt
	.vcScrollActionSBThumbPosition 4	Die Versetzung um einen Schritt ist abgeschlossen.

.vcScrollActionSBThumbTrack 5	Die Ansicht wurde um einen Schritt versetzt.
.vcScrollActionScrollEnd 104	Scrollen über die Taste **Ende** oder über das Kontextmenü an das Diagrammende (rechts unten)
.vcScrollActionScrollHome 103	Scrollen über die Taste **Pos 1** oder über das Kontextmenü an den Diagrammanfang (links oben)
.vcScrollActionThumbTrackDown 108	Thumb (Balken im Scrollbar) nach unten geschoben
.vcScrollActionThumbTrackLeft 107	Thumb (Balken im Scrollbar) nach links geschoben
.vcScrollActionThumbTrackRight 108	Thumb (Balken im Scrollbar) nach rechts geschoben
.vcScrollActionThumbTrackUp 107	Thumb (Balken im Scrollbar) nach oben geschoben

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDiagramHorizontalScrolled(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcDiagramHorizontalScrolledEventArgs) Handles
VcGantt1.VcDiagramHorizontalScrolled
   MsgBox(e.CurStartDate + e.CurEndDate)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDiagramHorizontalScrolled(object sender,
NETRONIC.XGantt.VcDiagramHorizontalScrolledEventArgs e)
   {
   MessageBox.Show(e.CurStartDate.ToString() + "\r\n" +
e.CurEndDate.ToString());
   }
```

VcDiagramHorizontalScrolling

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn Sie einen Scrollvorgang angefordert haben, aber bevor der integrierte Scroll-Automatismus ausgeführt wird. Das ursprüngliche Anfangs- und Enddatum des sichtbaren Diagrammbereichs werden zurückgegeben. Anhand des Parameters **scrollAction** erhalten Sie Informationen darüber, welcher Art der ausgelöste Scrollvorgang ist. Setzt man den Rückgabestatus auf **vcRetStatFalse**, so wird der integrierte Scroll-Automatismus unterdrückt, und Sie können in Ihrer Anwendung selbst auf das Ereignis reagieren.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDiagramHorizontalScrollingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDiagramHorizontalScrollingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ curStartDate	System.DateTime	Aktuelles Anfangsdatum des sichtbaren Diagrammbereiches
⇨ curEndDate	System.DateTime	Aktuelles Enddatum des sichtbaren Diagrammbereiches
⇨ scrollAction	VcScrollAction	Art des Scrollvorgangs
	Mögliche Werte:	
	.vcScrollActionAutoscrollDown 102	Die Ansicht verschob sich automatisch nach unten.
	.vcScrollActionAutoscrollLeft 101	Die Ansicht verschob sich automatisch nach links.
	.vcScrollActionAutoscrollRight 102	Die Ansicht verschob sich automatisch nach rechts.
	.vcScrollActionAutoscrollUp 101	Die Ansicht verschob sich automatisch nach oben.
	.vcScrollActionMouseWheelDown 106	Bei gedrücktem Mausrad wurde die Maus nach unten geschoben.
	.vcScrollActionMouseWheelLeft 105	Bei gedrücktem Mausrad wurde die Maus nach links geschoben.
	.vcScrollActionMouseWheelRight 106	Bei gedrücktem Mausrad wurde die Maus nach rechts geschoben.
	.vcScrollActionMouseWheelUp 105	Bei gedrücktem Mausrad wurde die Maus nach oben geschoben.
	.vcScrollActionSBLineDown 1	Die Ansicht verschob sich zur unteren Begrenzungslinie.
	.vcScrollActionSBLineLeft 0	Die Ansicht verschob sich zur linken Begrenzungslinie.
	.vcScrollActionSBLineRight 1	Die Ansicht verschob sich zur rechten Begrenzungslinie.
	.vcScrollActionSBLineUp 0	Die Ansicht verschob sich zur oberen Begrenzungslinie.
	.vcScrollActionSBNothing -1	Die Ansicht wurde nicht verschoben.
	.vcScrollActionSBPageDown 3	Die Ansicht wurde um eine Seite nach unten versetzt.
	.vcScrollActionSBPageLeft 2	Die Ansicht wurde um eine Seite nach links versetzt.
	.vcScrollActionSBPageRight 3	Die Ansicht wurde um eine Seite nach rechts versetzt.
	.vcScrollActionSBPageUp 2	Die Ansicht wurde um eine Seite nach oben versetzt
	.vcScrollActionSBThumbPosition 4	Die Versetzung um einen Schritt ist abgeschlossen.
	.vcScrollActionSBThumbTrack 5	Die Ansicht wurde um einen Schritt versetzt.
	.vcScrollActionScrollEnd 104	Scrollen über die Taste **Ende** oder über das Kontextmenü an das Diagrammende (rechts unten)
	.vcScrollActionScrollHome 103	Scrollen über die Taste **Pos 1** oder über das Kontextmenü an den Diagrammanfang (links oben)
	.vcScrollActionThumbTrackDown 108	Thumb (Balken im Scrollbar) nach unten geschoben
	.vcScrollActionThumbTrackLeft 107	Thumb (Balken im Scrollbar) nach links geschoben
	.vcScrollActionThumbTrackRight 108	Thumb (Balken im Scrollbar) nach rechts geschoben

	.vcScrollActionThumbTrackUp 107	Thumb (Balken im Scrollbar) nach oben geschoben
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcDiagramHorizontalScrolling(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcDiagramHorizontalScrollingEventArgs) Handles
VcGantt1.VcDiagramHorizontalScrolling
   If e.CurStartDate > "01.01.2014" Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDiagramHorizontalScrolling(object sender,
NETRONIC.XGantt.VcDiagramHorizontalScrollingEventArgs e)
   {
   if (DateTime.Compare(e.CurStartDate, Convert.ToDateTime("01.05.14
00:00:00")).Equals(true))
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcDiagramLeftClicking

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDiagramClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDiagramClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers

⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcDiagramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramLeftClicking
   MsgBox("x: " + e.X.ToString() + " y: " + e.Y.ToString())
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDiagramLeftClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
   {
   MessageBox.Show("x: " + e.X.ToString() + " y: " + e.Y.ToString());
   }
```

VcDiagramLeftDoubleClicking

Ereignis von VcGantt

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 Cursors) wird als Parameter zurückgegeben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDiagramClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDiagramClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcDiagramLeftDoubleClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramLeftDoubleClicking
   VcGantt1.Zoom(90)
End Sub
```

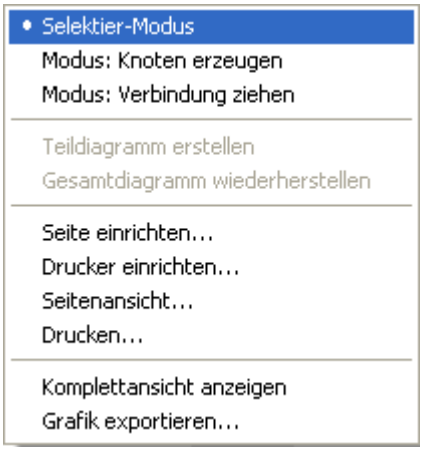
Code-Beispiel C#

```
private void vcGantt1_VcDiagramLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
   {
   vcGantt1.Zoom(90);
   }
```

VcDiagramRightClicking

Ereignis von VcGantt

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 zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.



Oben: integriertes Kontextmenü

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDiagramClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDiagramClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcDiagramRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcDiagramClickingEventArgs) Handles
VcGantt1.VcDiagramRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcDiagramRightClicking(object sender,
NETRONIC.XGantt.VcDiagramClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcDragCompleting

Ereignis von VcGantt

Dieses Ereignis wird zum Abschluss eines Drag&Drop-Vorgangs bei der Quellkomponente ausgelöst und gibt den Drop-Effekt bekannt.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDragCompletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDragCompletingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ DropEffect	System.Windows.Forms.DragDropEffects	Auswirkungen einer Drag & Drop-Operation

VcDragOver

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender Daten über ein Ziel zieht.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDragEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDragEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ AllowedEffects	System.Windows.Forms.DragDropEffects	Ruft ab, welche Drag & Drop-Operationen von der Quelle des Ziehereignisses zugelassen werden.
⇨ Data	System.Windows.Forms.IDataObject	Erfragt das IDataObject, das die Daten enthält, die diesem Ereignis zugeordnet sind.
⇨ Effect	System.Windows.Forms.DragDropEffects	Ruft den Ablageeffekt des Ziels in einer Drag & Drop-Operation ab oder legt dieses fest.
⇨ KeyState	System.Int32	Erfragt den aktuellen Zustand der UMSCHALTTASTE, STRG, ALT sowie den Zustand der Maustasten ab.
⇨ X	System.Int32	Ruft die x-Koordinate des Mauszeigers in Bildschirmkoordinaten ab.
⇨ Y	System.Int32	Ruft die y-Koordinate des Mauszeigers in Bildschirmkoordinaten ab.
⇨ Start	System.DateTime	Ruft den Starttermin ab, der im Tooltip beim Ziehen eines Knotenobjekts erscheint.
⇨ End	System.DateTime	Ruft den Endtermin ab, der im Tooltip beim Ziehen eines Knotenobjekts erscheint.

VcDragStarting

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcDragStartingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcDragStartingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ allowedEffects	System.Windows.Forms.AllowedEffects	Erlaubte DropEffects

VcErrorOccurring

Ereignis von VcGantt

Dieses Ereignis tritt nur dann auf, wenn ein unvorhergesehener Fehler im Code des VARCHART XGantt 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 Windows-Forms-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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt

⇨ e	VcErrorOcurringEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcErrorOcurringEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇦ ReturnStatus	System.Boolean	Wird der ReturnStatus auf **vcRetStatFalse** gesetzt, erscheint keine MessageBox.
⇨ Text	System.String	Fehlerbeschreibungstext

VcFieldSelecting

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Zelle einer Tabelle oder das Feld einer Box selektiert wird. Die Selektion kann durch Setzen des Rückgabestatus verhindert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcFieldSelectingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcFieldSelectingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ editObject	VcObject	Editiertes Objekt
⇨ editObjectType	VcObjectType	Objekttyp
	Mögliche Werte:	
	.vcObjTypeBox 15	Objekttyp **Box**
	.vcObjTypeCalendarGrid 18	Objekttyp **Kalendergitter**
	.vcObjTypeCurve 12	Objekttyp **Kurve**
	.vcObjTypeDateLine 9	Objekttyp **Stichtaglinie**
	.vcObjTypeGroup 7	Objekttyp **Gruppe**
	.vcObjTypeGroupInDiagram 11	Objekttyp **Gruppe im Knotenbereich**
	.vcObjTypeGroupInTable 7	Objekttyp **Gruppe im Tabellenbereich**
	.vcObjTypeHistogram 13	Objekttyp **Histogramm**
	.vcObjTypeLayer 8	Objekttyp **Layer**
	.vcObjTypeLinkCollection 3	Objekttyp **LinkCollection**
	.vcObjTypeNodeInDiagram 2	Objekttyp **Knoten im Knotenbereich**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNodeInTable 1	Objekttyp **Knoten im Tabellenbereich**

	.vcObjTypeNone 0 .vcObjTypeNumericScale 10 .vcObjTypeSummaryNode 14 .vcObjTypeTable 4 .vcObjTypeTableCaption 5 .vcObjTypeTimeScale 6	kein Objekt Objekttyp **Werteskala** Objekttyp **Summenbalken** Objekttyp **Tabelle** Objekttyp **Tabellenüberschrift** Objekttyp **Zeitskala**
⇨ 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	
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Das Feld wird nicht ausgewählt. Das Feld wird ausgewählt.

VcGroupDeleting

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender interaktiv eine Gruppe löscht. Dabei wird die Gruppe als Parameter zurückgegeben. Durch Setzen des Rückgabestatus kann der Löschvorgang verhindert werden. Es können nur Gruppen gelöscht werden, die keine Elemente enthalten.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupDeletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupDeletingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ group	VcGroup	Gelöschte Gruppe
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Gruppe wird nicht gelöscht. Die Gruppe wird gelöscht.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcGroupDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupDeletingEventArgs) Handles VcGantt1.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 vcGantt1_VcGroupDeleting(object sender,
NETRONIC.XGantt.VcGroupDeletingEventArgs e)
   {
   if (e.Group.Name == "A")
      {
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      MessageBox.Show("Group A cannot be deleted");
      }
   }
```

VcGroupLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender auf eine Gruppenüberschrift in der Tabelle mit der linken Maustaste klickt. Das getroffene Group-Objekt wird zusammen mit der Position (x,y-Koordinaten des Mauszeigers) als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ group	VcGroup	Getroffene Gruppe
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcGroupLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupClickingEventArgs) Handles VcGantt1.VcGroupLeftClicking
   MsgBox(e.Group.SubGroups.Count)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupLeftClicking(object sender,
NETRONIC.XGantt.VcGroupClickingEventArgs e)
   {
   MessageBox.Show(e.Group.SubGroups.Count.ToString());
   }
```

VcGroupLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender auf eine Gruppenüberschrift in der Tabelle mit der linken Maustaste doppelt klickt. Das getroffene Group-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Cursors als Parameter zurückgegeben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ group	VcGroup	Getroffene Gruppe
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ 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 VcGantt1_VcGroupLeftDoubleClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcGroupClickingEventArgs) Handles
VcGantt1.VcGroupLeftDoubleClicking
   MsgBox(e.Group.Name)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcGroupClickingEventArgs e)
   {
   MessageBox.Show(e.Group.Name);
   }
```

VcGroupModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Modifizierung der Gruppe abgeschlossen ist.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ group	VcGroup	Veränderte Gruppe

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcGroupModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupModifiedEventArgs) Handles VcGantt1.VcGroupModified
  MsgBox("The group has been modified.")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupModified(object sender,
NETRONIC.XGantt.VcGroupModifiedEventArgs e)
   {
   MessageBox.Show("The group has been modified");
   }
```

VcGroupModifying

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender eine Gruppe interaktiv verändert. Das getroffene Group-Objekt, die Art der Veränderung und der Rückgabestatus werden als Parameter zurückgegeben. Über den Parameter **modificationType** erhalten Sie nähere Informationen über die Art der Veränderung.

Die mit diesem Ereignis übermittelten Daten dürfen nur gelesen, aber nicht verändert werden. Um sie zu verändern, verwenden Sie bitte das Ereignis **VcGroupModified**.

Durch Setzen des Rückgabestatus kann die Änderung verhindert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupModifyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇦ Rückgabewert	Void	
⇨ oldGroup	VcGoup	Gruppe vor der Veränderung
⇨ group	VcGroup	Veränderte Gruppe
⇨ group	VcGroup	Zu verändernde Gruppe
⇨ modificationType	VcGroupModificationTypes	Art der Veränderung
	Mögliche Werte:	
	.vcGMTAnything 1	Änderungstyp **Nicht näher bestimmt**
	.vcGMTMinusPressed 2	Änderungstyp **Minus-Symbol angeklickt**
	.vcGMTNothing 0	Änderungstyp **Nichts**
	.vcGMTPlusPressed 4	Änderungstyp **Plus-Symbol angeklickt**
⇨ modificationType	VcModificationTypes	Art der Veränderung
	Mögliche Werte:	
	.vcAnything 1	Änderungstyp nicht näher bestimmt
	.vcChangedGroup 16	Zuordnung des Knotens zu einer Gruppe wurde verändert (nur für Knoten).
	.vcEndModified 4	Ende des Knotens wurde verändert (nur für Knoten).

	.vcHierarchyModified 64	Hierarchie der Knoten wurde verändert
	.vcModifiedByResourceScheduling 128	Änderung durch Ressourcenplanung (nur für Datensatz-Objekt)
	.vcModifiedBySchedule 32	Änderung durch neue Zeitberechnung
	.vcMoved 8	Objekt wurde verschoben.
	.vcNothing 0	Keine Änderung
	.vcStartModified 2	Anfang des Knotens wurde verändert (nur für Knoten).
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.vcRetStatFalse 0	Die Veränderung wird rückgängig gemacht.
	.vcRetStatOK 1	Die Veränderung wird durchgeführt.
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcGroupModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupModifyingEventArgs) Handles VcGantt1.VcGroupModifying
   Select Case e.ModificationType
      Case VcGroupModificationTypes.vcGMTNothing
         MsgBox("No modification")
      Case VcGroupModificationTypes.vcGMTAnything
         MsgBox("Any modification")
      Case VcGroupModificationTypes.vcGMTMinusPressed
         MsgBox("Collapsing group:" + e.Group.Name)
      Case VcGroupModificationTypes.vcGMTPlusPressed
         MsgBox("Expanding group" + e.Group.Name)
    End Select
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupModifying(object sender,
NETRONIC.XGantt.VcGroupModifyingEventArgs e)
   {
   switch (e.ModificationType)
      {
      case VcGroupModificationTypes.vcGMTNothing:
         MessageBox.Show("No modification");
         break;
      case VcGroupModificationTypes.vcGMTAnything:
         MessageBox.Show("Any modification");
         break;
      case VcGroupModificationTypes.vcGMTMinusPressed:
         MessageBox.Show("Collapsing group: " + e.Group.Name);
         break;
     case VcGroupModificationTypes.vcGMTPlusPressed:
         MessageBox.Show("Expanding group: " + e.Group.Name);
         break;
       }
   }
```

VcGroupRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender in der Tabelle auf eine Gruppenüberschrift mit der rechten Maustaste klickt. Das getroffene Group-Objekt und die Mausposition (x,y-Koordinaten) werden als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ group	VcGroup	Getroffene Gruppe
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcGroupRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupClickingEventArgs) Handles VcGantt1.VcGroupRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupRightClicking(object sender,
NETRONIC.XGantt.VcGroupClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcGroupsMarked

Ereignis von VcGantt

Mit diesem Ereignis wird das Ende einer Markieroperation von Gruppen angezeigt.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcGroupsMarkedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcGroupsMarkedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇦ (no parameter)		Kein Parameter

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcGroupsMarked(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupsMarkedEventArgs) Handles VcGantt1.VcGroupsMarked
   MsgBox("Groups have been marked successfully.")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupsMarked(object sender,
NETRONIC.XGantt.VcGroupsMarkedEventArgs e)
   {
   MessageBox.Show("Groups have been marked successfully.");
   }
```

VcGroupsMarking

Ereignis von VcGantt

Mit diesem Ereignis wird bekanntgegeben, dass der Benutzer Gruppen zum Markieren ausgewählt oder markierte Gruppen durch einen Klick in den leeren Diagrammbereich demarkiert hat. In der Gruppenauflistung (GroupCollection-Objekt) sind die beim letzten Markiervorgang ausgewählten Gruppen verzeichnet. Falls durch einen Klick ins leere Diagramm demarkiert wurde, ist die GroupCollection 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 **VcGroupsMarked**

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodesMarkingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodesMarkingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ groupCollection	VcGroupCollection	GroupCollection, die die vom Anwender selektierten Knoten enthält. Wenn in das Diagramm geklickt wurde, ist die GroupCollection leer.
⇦ returnStatus	VcReturnStatus	Rückgabestatus

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcGroupsMarking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcGroupsMarkingEventArgs) Handles VcGantt1.VcGroupsMarking
   If MsgBox("Mark this group?", MsgBoxStyle.YesNo, "Marking groups") =
MsgBoxResult.No Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcGroupsMarking(object sender,
NETRONIC.XGantt.VcGroupsMarkingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Mark this group?", "Marking groups",
MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcHelpRequested

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHelpRequestedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHelpRequestedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ DialogType	VcDialogType	Dialog, für den Hilfe angefordert wurde
	Mögliche Werte:	
	.vcEditDataRecordDialog 5400	Hilfe wurde für den **Datensatz bearbeiten** Dialog angefordert
	.vcEditTimeScaleDialog 5409	Hilfe wurde für den **Zeitskala bearbeiten** Dialog angefordert
	.vcPageSetupDialog 4097	Hilfe wurde für den **Seite einrichten** Dialog angefordert
	.vcPrintPreviewDialog 4096	Hilfe wurde für den **Druckvorschau** Dialog angefordert

VcHistogramCurveNameShowingInMenu

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Namen von per API definierten Histogrammkurven in einem Kontextmenü angezeigt werden. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** werden die Namen der Histogrammkurven nicht im Kontextmenü angezeigt.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHistogramCurveNameShowingInMenuEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHistogramCurveNameShowingInMenuEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ histogram	VcHistogram	Getroffenes Histogramm

⇨ curveName	System.String	Name der Histogrammkurve
⇔ returnStatus	VcReturnStatus	Rückgabewert
	Mögliche Werte:	
	.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 VcGantt1_VcHistogramCurveNameShowingInMenu(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcHistogramCurveNameShowingInMenuEventArgs) Handles
VcGantt1.VcHistogramCurveNameShowingInMenu
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcHistogramCurveNameShowingInMenu(object sender,
NETRONIC.XGantt.VcHistogramCurveNameShowingInMenuEventArgs e)
   {
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcHistogramLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf ein Histogramm klickt. Das getroffene Histogramm-Objekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter zurückgegeben.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHistogramClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHistogramClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ histogram	VcHistogram	Getroffenes Histogramm
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus

Mögliche Werte:	
.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 VcGantt1_VcHistogramLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcHistogramClickingEventArgs) Handles
VcGantt1.VcHistogramLeftClicking
   Call MsgBox("Histogram:" + e.Histogram.Name + " x:" + e.X.ToString() + " y: "
+ e.Y.ToString())
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcHistogramLeftClicking(object sender,
NETRONIC.XGantt.VcHistogramClickingEventArgs e)
   {
   MessageBox.Show("Histogram: " + e.Histogram.Name + " x: " + e.X.ToString() +
" y: " + e.Y.ToString());
   }
```

VcHistogramLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf ein Histogramm doppelklickt. Das getroffene Histogramm-Objekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter übergeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHistogramClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHistogramClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ histogram	VcHistogram	Getroffenes Histogramm
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	

.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 VcGantt1_VcHistogramLeftDoubleClicking(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcHistogramClickingEventArgs) Handles
VcGantt1.VcHistogramLeftDoubleClicking
   MsgBox(e.Histogram.Name)
End Sub
```

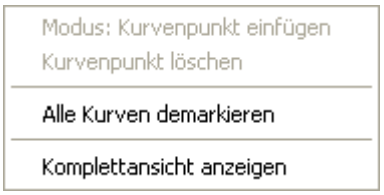
Code-Beispiel C#

```
private void vcGantt1_VcHistogramLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcHistogramClickingEventArgs e)
   {
   MessageBox.Show(e.Histogram.Name);
   }
```

VcHistogramRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf ein Histogramm klickt. Das getroffene Histogramm-Objekt wird zusammen mit der Position des Mauszeigers (x,y-Koordinaten) als Parameter übergeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.



Oben: integriertes Kontextmenü

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHistogramClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHistogramClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ histogram	VcHistogram	Getroffenes Histogramm
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcHistogramRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcHistogramClickingEventArgs) Handles
VcGantt1.VcHistogramRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcHistogramRightClicking(object sender,
NETRONIC.XGantt.VcHistogramClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcHistogramsHeightChanged

Ereignis von VcGantt

Dieses Ereignis tritt ein, nachdem das interaktiv veränderte Höhenverhältnis zwischen Diagrammbereich und dem oder den Histogrammen verändert wurde. Das Histogramm-Collection-Objekt und das Diagramm / Histogramm-Höhenverhältnis werden als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHistogramsHeightChangedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHistogramsHeightChangedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ histogramCollection	VcHistogramCollection	Histogramm-Kollektion
⇨ histogramsHeightRatio	System.Int32	Höhenverhältnis der Histogramme zum Gesamtdiagramm

VcHistogramsHeightChanging

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender interaktiv das Höhenverhältnis zwischen Diagrammbereich und dem oder den Histogrammen verändert hat. Die Auflistung der Histogramme und das Diagramm / Histogramm-Höhenverhältnis werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus auf vcRetStatFalse wird die Änderung rückgängig gemacht.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcHistogramsHeightChangingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcHistogramsHeightChangingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ histogramCollection	VcHistogramCollection	Histogramm-Kollektion
⇨ histogramsHeightRatio	System.Int32	Höhenverhältnis der Histogramme zum Gesamtdiagramm
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Höhe wird nicht geändert. Die Höhe wird geändert.

VcHistogramsHeightChangingEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender die Histogrammhöhe(n) interaktiv verändert hat. Die Histogramme und das neue Diagramm-

/Histogramm-Höhenverhältnis werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

Dieses Ereignis unterscheidet sich von dem Ereignis **VcHistogramHeight-Changing** durch die Rückgabe des Parameters Diagramm-/Histogramm-Höhenverhältnis als Datentyp "Double", mit dem eine höhere Genauigkeit erzielt wird. Die Verwendung dieses Ereignisses muss zuvor über die Eigenschaft **UseHigherDiagramHistogramHeightRatioPrecision** oder über die Option **Höhere Genauigkeit für die Höhenverhältnisse zwischen Diagramm und Histogramm** auf der Eigenschaftenseite **Allgemeines** aktiviert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	
⇨ e	VcHistogramsHeightChangingExEventArgs	

	Datentyp	Beschreibung

VcInPlaceEditorShowing

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der im Programm implementierte Editor gestartet wird.

Das Ereignis wird erst aktiviert, wenn die entsprechenden Eigenschaften **InPlaceEditingOnGroupsInDiagramEnabled, InPlaceEditingOnGroupsInTableEnabled, InPlaceEditingOnNodesInDiagramEnabled, InPlaceEditingOnNodesInTableEnabled** auf True gesetzt sind.

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcInPlaceEditorShowingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcInPlaceEditorShowingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ editObject	VcObject	Editiertes Objekt
⇨ editObjectType	VcObjectType	Objekttyp
	Mögliche Werte:	
	.vcObjTypeBox 15	Objekttyp **Box**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNodeInTable 1	Objekttyp **Knoten im Tabellenbereich**
	.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	
	Mögliche Werte:	
	.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 VcGantt1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs) Handles
VcGantt1.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 + VcGantt1.Left
                    TextBox1.Top = e.FldRectVisible.Top + VcGantt1.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 + VcGantt1.Left
                    DateTimePicker1.Top = e.FldRectVisible.Top + VcGantt1.Top
                    DateTimePicker1.Value = node.DataField(0)
                    DateTimePicker1.Visible = True
                    DateTimePicker1.Focus()
                Case 13     'Employee
                    ComboBox1.Left = e.FldRectVisible.Left + VcGantt1.Left
                    ComboBox1.Top = e.FldRectVisible.Top + VcGantt1.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 vcGantt1_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 + vcGantt1.Left;
            textBox1.Top = e.FldRectVisible.Top + vcGantt1.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 + vcGantt1.Left;
            dateTimePicker1.Top = e.FldRectVisible.Top + vcGantt1.Top;
            dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
            dateTimePicker1.Visible = true;
            dateTimePicker1.Focus();
            break;
         case 13: //Employee
            comboBox1.Left = e.FldRectVisible.Left + vcGantt1.Left;
            comboBox1.Top = e.FldRectVisible.Top + vcGantt1.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;
        }
   }
```

VcInteractionEnded

Ereignis von VcGantt

Dieses Ereignis tritt im Rahmen des LiveUpdate ein, sobald eine Interaktion abgeschlossen wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	
⇨ e	VcInteractionEndedEventArgs	

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ InInteractionMode	VcInInteractionMode	Art der Interaktion

Mögliche Werte:	
.vcIIMCopyMoveNode 1014	Kopierter Knoten wird bewegt
.vcIIMCopyNode 1007	Knoten wird kopiert
.vcIIMCreateLinkChangeSuccessor 1101	Modus Verbindung erzeugen: Nachfolger wird gewechselt
.vcIIMCreateNodeResizeRightX 1012	Modus Knoten erzeugen: Starttermin eines Layers
.vcIIMCreateResizeObjectContainerWidthHight 1072	Größe einer Textbox verändern
.vcIIMDragDropNode 1018	Knoten durch Drag and Drop verschieben
.vcIIMDragDropNodeInTable 1019	Knoten in Tabelle durch Drag and Drop verschieben
.vcIIMModifySectionStartDate 1061	Startdatum des Zeitskalenabschnitts verändern
.vcIIMMoveCurvePointX 1052	Verschieben von Kurvenpunkt X
.vcIIMMoveCurvePointXandY 1051	Verschieben von Kurvenpunkten X und Y
.vcIIMMoveCurvePointY 1053	Verschieben von Kurvenpunkt Y
.vcIIMMoveGroupInDiagram 1100	Gruppe im Diagramm wird verschoben
.vcIIMMoveGroupInTable 1009	Gruppe in Tabelle wird verschoben
.vcIIMMoveHorValueLine 1031	Horizontales Verschieben einer Dateline
.vcIIMMoveLayer 1004	Verschieben eines Layers
.vcIIMMoveNode 1001	Verschieben eines Knotens
.vcIIMMoveNode 1001	Knoten durch Drag and Drop verschieben
.vcIIMMoveNodeInRow 1002	Knoten in Zeile verschieben
.vcIIMMoveNodeInTable 1008	Knoten in Tabelle verschieben
.vcIIMMoveNodeVertical 1003	Knoten vertikal verschieben
.vcIIMMoveObjectContainer 1073	Bewegen einer Textbox
.vcIIMMoveSash 1026	Sash verschieben
.vcIIMResizeBasicUnitWidth 1062	Ändern der Grundeinheitsbreite
.vcIIMResizeLeftX 1005	Starttermin eines Layers
.vcIIMResizeNumericBasicUnitWidth 1063	Ändern der numerischen Grundeinheitsbreite
.vcIIMResizeObjectContainerHeight 1075	Höhe eine Textbox ändern
.vcIIMResizeObjectContainerWidth 1074	Breite einer Textbox ändern
.vcIIMResizeObjectContainerWidthHeight 1076	Höhe und Breite einer Textbox ändern
.vcIIMResizeRightX 1006	Endtermin eines Layers
.vcIIMUnKnown -1	Wird i.d.R. ncht über die Eventargs zurückgegeben, kann aber z.B. genutz werden, um eine Variable als nicht gesetzt zu kennzeichnen
.vcIIMvcIIMResizeLeftTableColumnWidth 1041	Spaltenbreite der linken Tabelle ändern

	.vcIIMvcIIMResizeRightTableColumnWidth 1042	Spaltenbreite der rechten Tabelle ändern
⇨ InteractionObject	InteractionObject	von der Interaktion betroffenes Objekt
⇨ ObjectType	InteractionType	Typ des von der Interaktion betroffenen Objekts

VcInteractionModeChanged

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Interaktionsmodus im Kontextmenü geändert wurde und die Änderung nicht durch Setzen des Rückgabewerts auf **vcRetStatFalse** im Ereignis **VcInteractionModeChanging** verhindert wurde.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	
⇨ e	VcInteractionModeChangedEventArgs	

	Datentyp	Beschreibung

VcInteractionModeChanging

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Interaktionsmodus im Kontextmenü geändert wurde.

Durch Setzen des Rückgabewerts auf **vcRetStatFalse** wird die Änderung nicht übernommen und das Event **VcInteractionModeChanged** nicht ausgelöst.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	

⇨ e	VcInteractionModeChangingEventArgs

	Datentyp	Beschreibung
Eigenschaften:		
⇨ NewInteractionMode	VcInteractionMode	Ausgewählter Interaktionsmodus
	Mögliche Werte:	
	.vcCreateBox 36	Erzeugemodus für Boxen
	.vcCreateLink 4	Erzeugemodus für Verbindungen
	.vcCreateNode 2	Erzeugemodus für Knoten
	.vcDeleteLink 5	Löschmodus für Verbindungen
	.vcDeleteNode 3	Löschmodus für Knoten
	.vcPanning 6	Verschiebemodus
	.vcPointer 0	Selektiermodus
⇨ returnstatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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.

VcInteractionObjectChanged

Ereignis von VcGantt

Dieses Ereignis tritt im Rahmen des LiveUpdate ein, wenn bei einer Interaktion initial noch kein Objekt zur Verfügung steht (z.B. Knoten- oder Boxerzeugung) und sobald intern das Objekt erzeugt wurde.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	
⇨ e	VcInteractionObjectChangedEventArgs	

	Datentyp	Beschreibung
Eigenschaften:		
⇨ InInteractionMode	VcInInteractionMode	Art der Interaktion
	Mögliche Werte:	

.vcIIMCopyMoveNode 1014	Kopierter Knoten wird bewegt
.vcIIMCopyNode 1007	Knoten wird kopiert
.vcIIMCreateLinkChangeSuccessor 1101	Modus Verbindung erzeugen: Nachfolger wird gewechselt
.vcIIMCreateNodeResizeRightX 1012	Modus Knoten erzeugen: Starttermin eines Layers
.vcIIMCreateResizeObjectContainerWidthHight 1072	Größe einer Textbox verändern
.vcIIMDragDropNode 1018	Knoten durch Drag and Drop verschieben
.vcIIMDragDropNodeInTable 1019	Knoten in Tabelle durch Drag and Drop verschieben
.vcIIMModifySectionStartDate 1061	Startdatum des Zeitskalenabschnitts verändern
.vcIIMMoveCurvePointX 1052	Verschieben von Kurvenpunkt X
.vcIIMMoveCurvePointXandY 1051	Verschieben von Kurvenpunkten X und Y
.vcIIMMoveCurvePointY 1053	Verschieben von Kurvenpunkt Y
.vcIIMMoveGroupInDiagram 1100	Gruppe im Diagramm wird verschoben
.vcIIMMoveGroupInTable 1009	Gruppe in Tabelle wird verschoben
.vcIIMMoveHorValueLine 1031	Horizontales Verschieben einer Dateline
.vcIIMMoveLayer 1004	Verschieben eines Layers
.vcIIMMoveNode 1001	Verschieben eines Knotens
.vcIIMMoveNode 1001	Knoten durch Drag and Drop verschieben
.vcIIMMoveNodeInRow 1002	Knoten in Zeile verschieben
.vcIIMMoveNodeInTable 1008	Knoten in Tabelle verschieben
.vcIIMMoveNodeVertical 1003	Knoten vertikal verschieben
.vcIIMMoveObjectContainer 1073	Bewegen einer Textbox
.vcIIMMoveSash 1026	Sash verschieben
.vcIIMResizeBasicUnitWidth 1062	Ändern der Grundeinheitsbreite
.vcIIMResizeLeftX 1005	Starttermin eines Layers
.vcIIMResizeNumericBasicUnitWidth 1063	Ändern der numerischen Grundeinheitsbreite
.vcIIMResizeObjectContainerHeight 1075	Höhe eine Textbox ändern
.vcIIMResizeObjectContainerWidth 1074	Breite einer Textbox ändern
.vcIIMResizeObjectContainerWidthHeight 1076	Höhe und Breite einer Textbox ändern
.vcIIMResizeRightX 1006	Endtermin eines Layers
.vcIIMUnKnown -1	Wird i.d.R. ncht über die Eventargs zurückgegeben, kann aber z.B. genutz werden, um eine Variable als nicht gesetzt zu kennzeichnen
.vcIIMvcIIMResizeLeftTableColumnWidth 1041	Spaltenbreite der linken Tabelle ändern
.vcIIMvcIIMResizeRightTableColumnWidth 1042	Spaltenbreite der rechten Tabelle ändern

⇨ InteractionObject	InteractionObject	von der Interaktion betroffenes Objekt
⇨ ObjectType	InteractionType	Typ des von der Interaktion betroffenen Objekts

VcInteractionStarted

Ereignis von VcGantt

Dieses Ereignis tritt im Rahmen des LiveUpdate ein, sobald bei einer Interaktion die linke Maustaste gedrückt wird und liefert Informationen darüber, auf welchem Objekt (Objekt und Objekttyp) der Mauszeiger steht.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcInteractionStartedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcInteractionStartedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ InInteractionMode	VcInInteractionMode	Art der Interaktion
	Mögliche Werte:	
	.vcIIMCopyMoveNode 1014	Kopierter Knoten wird bewegt
	.vcIIMCopyNode 1007	Knoten wird kopiert
	.vcIIMCreateLinkChangeSuccessor 1101	Modus Verbindung erzeugen: Nachfolger wird gewechselt
	.vcIIMCreateNodeResizeRightX 1012	Modus Knoten erzeugen: Starttermin eines Layers
	.vcIIMCreateResizeObjectContainerWidthHight 1072	Größe einer Textbox verändern
	.vcIIMDragDropNode 1018	Knoten durch Drag and Drop verschieben
	.vcIIMDragDropNodeInTable 1019	Knoten in Tabelle durch Drag and Drop verschieben
	.vcIIMModifySectionStartDate 1061	Startdatum des Zeitskalenabschnitts verändern
	.vcIIMMoveCurvePointX 1052	Verschieben von Kurvenpunkt X
	.vcIIMMoveCurvePointXandY 1051	Verschieben von Kurvenpunkten X und Y
	.vcIIMMoveCurvePointY 1053	Verschieben von Kurvenpunkt Y
	.vcIIMMoveGroupInDiagram 1100	Gruppe im Diagramm wird verschoben

	.vcIIMMoveGroupInTable 1009	Gruppe in Tabelle wird verschoben
	.vcIIMMoveHorValueLine 1031	Horizontales Verschieben einer Dateline
	.vcIIMMoveLayer 1004	Verschieben eines Layers
	.vcIIMMoveNode 1001	Verschieben eines Knotens
	.vcIIMMoveNode 1001	Knoten durch Drag and Drop verschieben
	.vcIIMMoveNodeInRow 1002	Knoten in Zeile verschieben
	.vcIIMMoveNodeInTable 1008	Knoten in Tabelle verschieben
	.vcIIMMoveNodeVertical 1003	Knoten vertikal verschieben
	.vcIIMMoveObjectContainer 1073	Bewegen einer Textbox
	.vcIIMMoveSash 1026	Sash verschieben
	.vcIIMResizeBasicUnitWidth 1062	Ändern der Grundeinheitsbreite
	.vcIIMResizeLeftX 1005	Starttermin eines Layers
	.vcIIMResizeNumericBasicUnitWidth 1063	Ändern der numerischen Grundeinheitsbreite
	.vcIIMResizeObjectContainerHeight 1075	Höhe eine Textbox ändern
	.vcIIMResizeObjectContainerWidth 1074	Breite einer Textbox ändern
	.vcIIMResizeObjectContainerWidthHeight 1076	Höhe und Breite einer Textbox ändern
	.vcIIMResizeRightX 1006	Endtermin eines Layers
	.vcIIMUnKnown -1	Wird i.d.R. ncht über die Eventargs zurückgegeben, kann aber z.B. genutz werden, um eine Variable als nicht gesetzt zu kennzeichnen
	.vcIIMvcIIMResizeLeftTableColumnWidth 1041	Spaltenbreite der linken Tabelle ändern
	.vcIIMvcIIMResizeRightTableColumnWidth 1042	Spaltenbreite der rechten Tabelle ändern
⇨ InteractionObject	InteractionObject	von der Interaktion betroffenes Objekt
⇨ ObjectType	InteractionType	Typ des von der Interaktion betroffenen Objekts

VcLegendViewClosed

Ereignis von VcGantt

Dieses Ereignis wird aufgerufen, wenn das Popup-Fenster der Legenden-ansicht geschlossen wird.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLEgendViewClosedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLEgendViewClosedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇦ (no parameter)		

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcLegendViewClosed(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLegendViewClosedEventArgs) Handles VcGantt1.VcLegendViewClosed
   MsgBox("Do you want to close the legend view window?", MsgBoxStyle.OKCancel)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLegendViewClosed(object sender,
NETRONIC.XGantt.VcLegendViewClosedEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to close the legend view
window?", "Closing legend view window", MessageBoxButtons.OKCancel);
   }
```

VcLinkCreated

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn das interaktive Anlegen einer Verbindung zwischen zwei Knoten beendet ist. Das Verbindungs-Objekt, der Typ des Anlegens der Verbindung (hier immer **VcLinkCreated**) und die Information, ob die angelegte Verbindung die einzige Verbindung bzw. die letzte Verbindung einer Menge ist (derzeit immer **True**), werden als Parameter zurückgegeben, so dass eine Datenvalidierung vorgenommen werden kann.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinkCreatedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinkCreatedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ link	VcLink	Angelegte Verbindung
⇨ creationType	VcCreationType	Typ des Anlegens der Verbindung
	Mögliche Werte:	
	.vcLinkCreated 2	Verbindung wurde durch Interaktion angelegt

⇨ isLast	System.Boolean	Angelegte Verbindung ist/ist nicht die einzige bzw. die letzte Verbindung einer Menge

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcLinkCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinkCreatedEventArgs) Handles VcGantt1.VcLinkCreated
        MsgBox(e.Link.AllData)
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLinkCreated(object sender,
NETRONIC.XGantt.VcLinkCreatedEventArgs e)
   {
   MessageBox.Show(e.Link.AllData.ToString());
   }
```

VcLinkCreating

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender interaktiv eine Verbindung zwischen zwei Knoten erzeugt hat. Das neu erzeugte Link-Objekt wird als Parameter zurückgegeben, so dass eine Validierung und ggf. ein Datenbank-Eintrag 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 **VcLinkCreated**.

Durch Setzen des Rückgabestatus kann die Anlage verhindert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinkCreatingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinkCreatingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ link	VcLink	Angelegte Verbindung
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0	Die Verbindung wird nicht angelegt.

.vcRetStatOK 1	Die Verbindung wird angelegt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcLinkCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinkCreatedEventArgs) Handles VcGantt1.VcLinkCreated
MsgBox(e.Link.AllData)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLinkCreated(object sender,
NETRONIC.XGantt.VcLinkCreatedEventArgs e)
{
   MessageBox.Show(e.Link.AllData.ToString());
}
```

VcLinkDeleted

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinkDeletedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinkDeletedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ link	VcLink	Gelöschte Verbindung
⇨ isLast	System.Boolean	Gelöschte Verbindung ist/ist nicht die einzige bzw. die letzte Verbindung einer Menge.

VcLinkDeleting

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs eine Verbindung löscht. Die betroffene Verbindung 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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinkDeletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinkDeletingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ link	VcLink	Gelöschte Verbindung
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Verbindung wird nicht gelöscht. Die Verbindung wird gelöscht.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcLinkDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinkDeletingEventArgs) Handles VcGantt1.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 vcGantt1_VcLinkDeleting(object sender,
NETRONIC.XGantt.VcLinkDeletingEventArgs e)
   {
   // deny deletion of link with a certain predecessor
   if (e.Link.PredecessorNode.get_DataField(0).Equals("1"))
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcLinksLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Verbindung bzw. eine Gruppe sich überlagernder Verbindungen klickt. Das getroffene LinkCollection-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinksClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinksClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ linkCltn	VcLinkCollection	Getroffenes LinkCollection-Objekt
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcLinksLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinksClickingEventArgs) Handles VcGantt1.VcLinksLeftClicking
   Dim linkCltn As VcLinkCollection
   Dim link As VcLink
   linkCltn = VcGantt1.LinkCollection
   'set certain data field of all links
   For Each link In linkCltn
      link.DataField(2) = "A"
  Next
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLinksLeftClicking(object sender,
NETRONIC.XGantt.VcLinksClickingEventArgs e)
   {
   VcLinkCollection linkCltn = vcGantt1.LinkCollection;
   // set certain data field of all links
   foreach (VcLink link in linkCltn)
      link.set_DataField(2, "A");
   }
```

VcLinksLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine Verbindung bzw. eine Gruppe sich überlagernder Verbindungen doppelklickt. Das getroffene LinkCollection-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinksClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinksClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ linkCltn	VcLinkCollection	Getroffenes LinkCollection-Objekt
⇨ x	System.Int32	X-Koordinate ddes Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ 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 VcGantt1_VcLinksLeftDoubleClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcLinksClickingEventArgs) Handles
VcGantt1.VcLinksLeftDoubleClicking
        MsgBox("Show your own dialog")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLinksLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcLinksClickingEventArgs e)
   {
   MessageBox.Show("Show your own dialog");
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcLinksRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf eine Verbindung bzw. eine Gruppe sich überlagernder Verbindungen klickt. Das getroffene Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcLinksClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcLinksClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ linkCltn	VcLinkCollection	Getroffenes LinkCollection-Objekt
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcLinksRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinksClickingEventArgs) Handles VcGantt1.VcLinksRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcLinksRightClicking(object sender,
NETRONIC.XGantt.VcLinksClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcNodeCreated

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn das interaktive Anlegen eines Knotens abgeschlossen ist. Das Knotenobjekt, der Typ des Knotenanlegens (hier **vcNodeCreated**) 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**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeCreatedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeCreatedEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ node	VcNode	Angelegter Knoten
⇨ creationType	VcCreationType	Typ des Anlegens von Knoten bzw. Verbindungen
	Mögliche Werte:	
	.vcDataRecordCreated 6	Datensatz wurde durch Interaktion angelegt
	.vcDataRecordCreatedByResourceScheduling 5	Datensatz wurde automatisch durch Ressourcenplanung angelegt
	.vcNodeCreated 1	Knoten durch "Stempeln" angelegt
⇨ isLast	System.Boolean	Angelegter Knoten ist/ist nicht der einzige Knoten bzw. der letzte Knoten einer Menge

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeCreated(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeCreatedEventArgs) Handles VcGantt1.VcNodeCreated
        MsgBox(e.Node.AllData)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeCreated(object sender,
NETRONIC.XGantt.VcNodeCreatedEventArgs e)
{
MessageBox.Show(e.Node.AllData.ToString());
}
```

VcNodeCreating

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender interaktiv einen Knoten erzeugt. Das Knotenobjekt wird als Parameter zurückgegeben, so dass eine Datenvalidierung 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 **VcNodeCreated**.

Durch Setzen des Rückgabestatus kann die Anlage verhindert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeCreatingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeCreatingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Anzulegender Knoten
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der Knoten wird nicht angelegt. Der Knoten wird angelegt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeCreating(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeCreatingEventArgs) Handles VcGantt1.VcNodeCreating
        MsgBox("Show your own dialog")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeCreating(object sender,
NETRONIC.XGantt.VcNodeCreatingEventArgs e)
   {
   MessageBox.Show("Show your own dialog");
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcNodeDeleted

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeDeletedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeDeletedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Gelöschter Knoten
⇨ isLast	System.Boolean	Gelöschter Knoten ist/ist nicht der letzte Knoten einer Menge

VcNodeDeleting

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit Hilfe des Kontextmenüs einen Knoten löscht. Der betroffene Knoten 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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeDeletingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeDeletingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Gelöschter Knoten
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der Knoten wird nicht gelöscht. Der Knoten wird gelöscht.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeDeleting(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeDeletingEventArgs) Handles VcGantt1.VcNodeDeleting
        'deny the deletion of the last node in the chart
        If VcGantt1.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 vcGantt1_VcNodeDeleting(object sender,
NETRONIC.XGantt.VcNodeDeletingEventArgs e)
   {
   //deny the deletion of the last node in the chart
   if (vcGantt1.NodeCollection.Count == 1)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      MessageBox.Show("The last node in the chart cannot be deleted.");
   }
```

VcNodeLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender auf einen Knoten (location = vcInDiagram) oder auf einen vorgangsbezogenen Tabelleneintrag (location = vcInTable) mit der linken Maustaste klickt. Das getroffene Knotenobjekt

wird zusammen mit der Cursorposition (x,y-Koordinaten) als Parameter übergeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Getroffener Knoten
⇨ location	VcLocation	Lokalisierung im Diagramm
	Mögliche Werte: .vcInDiagram 1 .vcInTable 0	 im Knotenbereich im Tabellenbereich
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ 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 VcGantt1_VcNodeLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.VcNodeLeftClicking
        'change data field of the node
        e.Node.DataField(4) = 1 - Convert.ToInt64(e.Node.DataField(4))
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeLeftClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   //change data field of the node
    e.Node.set_DataField(4,Convert.ToInt64(e.Node.get_DataField(4)));
   }
```

VcNodeLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender auf einen Knoten (location = vcInDiagram) oder auf einen vorgangsbezogenen Tabelleneintrag (location = vcInTable) mit der linken Maustaste doppelt klickt. Das getroffene Knotenobjekt wird zusammen mit der Cursorposition (x,y-Koordinaten) als Parameter übergeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird das standardmäßige Dialogfeld **Vorgänge bearbeiten** unterdrückt.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ node	VcNode	Getroffener Knoten
⇨ location	VcLocation	Lokalisierung im Diagramm
	Mögliche Werte: .vcInDiagram 1 .vcInTable 0	 im Knotenbereich im Tabellenbereich
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der **Daten bearbeiten** Dialog erscheint nicht. Der **Daten bearbeiten** Dialog erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeLeftDoubleClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles
VcGantt1.VcNodeLeftDoubleClicking
        MsgBox("Show your own dialog")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   MessageBox.Show("Show your own dialog");
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcNodeModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Modifizierung des angegebenen Knotens abgeschlossen ist.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Veränderter Knoten
⇨ isLast	System.Boolean	Veränderter Knoten ist/ist nicht der einzige Knoten bzw. der letzte Knoten einer Menge

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcGantt1.VcNodeModifying
   'revoke the modification if the node would change the group
   If e.ModificationType And VcModificationTypes.vcChangedGroup Then
      MsgBox("The node cannot be moved to a different group.")
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeModifying(object sender,
NETRONIC.XGantt.VcNodeModifyingEventArgs e)
   {
   //revoke the modification if the node would change the group
   if (VcModificationTypes.vcChangedGroup.Equals(true))
      {
      MessageBox.Show("The node cannot be moved into another group.");
       }
   }
```

VcNodeModifiedEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Modifizierung des angegebenen Knotens abgeschlossen ist.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeModifiedExEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeModifiedExEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Veränderter Knoten
⇨ isLast	System.Boolean	Veränderter Knoten ist/ist nicht der einzige Knoten bzw. der letzte Knoten einer Menge
⇨ modificationType	VcModificationTypes	Änderungstyp
	Mögliche Werte:	
	.vcAnything 1	Änderungstyp nicht näher bestimmt
	.vcChangedGroup 16	Zuordnung des Knotens zu einer Gruppe wurde verändert (nur für Knoten).
	.vcEndModified 4	Ende des Knotens wurde verändert (nur für Knoten).
	.vcHierarchyModified 64	Hierarchie der Knoten wurde verändert
	.vcModifiedByResourceScheduling 128	Änderung durch Ressourcenplanung (nur für Datensatz-Objekt)
	.vcModifiedBySchedule 32	Änderung durch neue Zeitberechnung
	.vcMoved 8	Objekt wurde verschoben.
	.vcNothing 0	Keine Änderung
	.vcStartModified 2	Anfang des Knotens wurde verändert (nur für Knoten).

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeModifiedEx(ByVal sender As System.Object, _
                                      ByVal e As
NETRONIC.XGantt.VcNodeModifiedExEventArgs)
                                    Handles VcGantt1.VcNodeModifiedEx
     'modify a record in the underlying database of the application
     modifyDataRecord(e.Node.AllData)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeModifiedEx(object sender,

VcNodeModifiedExEventArgs e)
{
  //modify a record in the underlying database of the application
  modifyDataRecord(e.Node.AllData);
}
```

VcNodeModifying

Ereignis von VcGantt

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**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeModifyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ oldNode	VcNode	Knoten vor der Veränderung
⇨ node	VcNode	zu verändernder Knoten
⇨ modificationType	VcModificationTypes	Art der Veränderung
		(Auch eine Kombination der Werte ist möglich.)
	Mögliche Werte:	
	.vcAnything 1	Änderungstyp nicht näher bestimmt

	.vcChangedGroup 16	Zuordnung des Knotens zu einer Gruppe wurde verändert (nur für Knoten).
	.vcEndModified 4	Ende des Knotens wurde verändert (nur für Knoten).
	.vcHierarchyModified 64	Hierarchie der Knoten wurde verändert
	.vcModifiedByResourceScheduling 128	Änderung durch Ressourcenplanung (nur für Datensatz-Objekt)
	.vcModifiedBySchedule 32	Änderung durch neue Zeitberechnung
	.vcMoved 8	Objekt wurde verschoben.
	.vcNothing 0	Keine Änderung
	.vcStartModified 2	Anfang des Knotens wurde verändert (nur für Knoten).
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.vcRetStatFalse 0	Die Veränderung wird rückgängig gemacht.
	.vcRetStatOK 1	Die Veränderung wird durchgeführt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcGantt1.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
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeModifying(object sender,
NETRONIC.XGantt.VcNodeModifyingEventArgs e)
   {
   //revoke the modification if the node would change the group
   if (e.ModificationType == VcModificationTypes.vcChangedGroup)
      {
      MessageBox.Show("The node cannod be moved into another group.");
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      }
   }
```

VcNodeResizeStarting

Ereignis von VcGantt

Dieses Ereignis wird aufgerufen, wenn der Benutzer beginnt, einen Layer an einem Knoten interaktiv zu verlängern bzw. zu verkürzen. Dies kann dazu dienen, kleinere Änderungen am XGantt vorzunehmen wie z.B. die Schrittweite knotenabhängig (TimeUnitsPerStep) anzupassen.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	

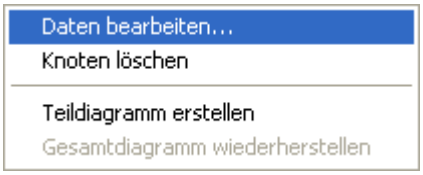
⇨ e	VcGroupDeletingEventArgs

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Größenveränderter Knoten

VcNodeRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender auf einen Knoten (location = vcInDiagram) oder auf einen vorgangsbezogenen Tabelleneintrag (location = vcInTable) mit der rechten Maustaste klickt. Das getroffene Knotenobjekt wird zusammen mit der Cursorposition (x,y-Koordinaten) als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.



Oben: integriertes Kontextmenü

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodeClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodeClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ node	VcNode	Getroffener Knoten
⇨ location	VcLocation	Lokalisierung im Chart

	Mögliche Werte: .vcInDiagram 1 .vcInTable 0	 im Knotenbereich im Tabellenbereich
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.VcNodeRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcNodesMarked

Ereignis von VcGantt

Mit diesem Ereignis wird das Ende einer Markieroperation von Knoten angezeigt.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodesMarkedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodesMarkedEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇦ (no parameter)		Kein Parameter

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodesMarked(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodesMarkedEventArgs) Handles VcGantt1.VcNodesMarked
   MsgBox("Nodes have been marked successfully.")
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodesMarked(object sender,
NETRONIC.XGantt.VcNodesMarkedEventArgs e)
   {
   MessageBox.Show("Nodes have been marked successfully.");
   }
```

VcNodesMarking

Ereignis von VcGantt

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 Knotenauflistung (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**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNodesMarkingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNodesMarkingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ nodeCollection	VcNodeCollection	NodeCollection, die die vom Anwender selektierten Knoten enthält. Wenn in das Diagramm geklickt wurde, ist die NodeCollection leer.
⇦ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	

.vcRetStatFalse 0	Die Markierung muss manuell erfolgen.
.vcRetStatOK 1	Die Markierung erfolgt automatisch.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodesMarking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodesMarkingEventArgs) Handles VcGantt1.VcNodesMarking
   If MsgBox("Mark this node?", MsgBoxStyle.YesNo, "Marking nodes") =
MsgBoxResult.No Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodesMarking(object sender,
NETRONIC.XGantt.VcNodesMarkingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Mark this node?", "Marking nodes",
MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.No)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcNumericScaleLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf die numerische Skala klickt. Das getroffene NumericScale-Objekt wird zusammen mit der Cursorposition (x,y-Koordinaten) des Cursors als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNumericScaleClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNumericScaleClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ numericScale	VcNumericScale	Getroffene Werteskala
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcNumericScaleLeftClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcNumericScaleClickingEventArgs) Handles
VcGantt1.VcNumericScaleLeftClicking
   e.NumericScale.BackgroundColor = Color.Blue
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNumericScaleLeftClicking(object sender,
NETRONIC.XGantt.VcNumericScaleClickingEventArgs e)
   {
   e.NumericScale.BackgroundColor = Color.LightSteelBlue;
   }
```

VcNumericScaleLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf die numerische Skala doppelt klickt. Das getroffene NumericScale-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNumericScaleClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNumericScaleClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ numericScale	VcNumericScale	Getroffene Werteskala
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcNumericScaleLeftDoubleClicking(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcNumericScaleClickingEventArgs) Handles
VcGantt1.VcNumericScaleLeftDoubleClicking
   e.NumericScale.MajorTicks = TextBox1.Text
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNumericScaleLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcNumericScaleClickingEventArgs e)
   {
   e.NumericScale.MajorTicks = textBox1.Text;
   }
```

VcNumericScaleRescaling

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender die numerische Skala interaktiv skaliert. Die getroffene numerische Skala und die neue Breite der Grundeinheit werden als Parameter zurückgegeben, so dass Sie die Zulässigkeit der Skalierung prüfen können. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNumericScaleRescalingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNumericScaleRescalingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ numericScale	VcNumericScale	Getroffene Werteskala
⇨ newBasicUnitWidth	System.Int32	Neue Breite der Grundeinheit
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die numerische Skala wird nicht verändert. Die numerische Skala wird verändert.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNumericScaleRescaling(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNumericScaleRescalingEventArgs) Handles
VcGantt1.VcNumericScaleRescaling
   Select Case e.NewBasicUnitWidth
      Case Is <= 1000
         MsgBox("New basic unit width:" + e.NewBasicUnitWidth)
      Case Is > 1000
         MsgBox("The maximum basic unit width is 1000.")
         e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End Select
End Sub
```

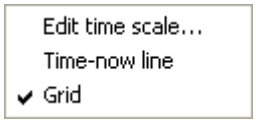
Code-Beispiel C#

```
private void vcGantt1_VcNumericScaleRescaling(object sender,
NETRONIC.XGantt.VcNumericScaleRescalingEventArgs e)
   {
   switch (e.NewBasicUnitWidth)
      {
      case e.NewBasicUnitWidth.CompareTo(1000):
         MessageBox.Show("New basic unit width: " + e.NewBasicUnitWidth);
         break;
      case e.TimeScale.UnitWidth > 1000:
          MessageBox.Show("The maximum basic unit width is 1000");
          e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
          break;
        }
   }
```

VcNumericScaleRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf die numerische Skala klickt. Das getroffene NumericScale-Objekt wird zusammen mit der Position (x,y-Koordinaten) des Mauszeigers als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position ein eigenes Kontextmenü anzeigen.



	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcNumericScaleClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcNumericScaleClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ numericScale	VcNumericScale	Getroffene Werteskala
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNumericScaleRightClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcNumericScaleClickingEventArgs) Handles
VcGantt1.VcNumericScaleRightClicking
   If MsgBox("Change unit label of numeric scale?", MsgBoxStyle.YesNo) =
MsgBoxResult.Yes Then
      e.NumericScale.UnitLabel = TextBox1.Text
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNumericScaleRightClicking(object sender,
NETRONIC.XGantt.VcNumericScaleClickingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Change unit label of numeric scale?",
"Changing numeric scale", MessageBoxButtons.YesNo);
   if (retVal ==  DialogResult.Yes)
      e.NumericScale.UnitLabel = textBox1.Text;
   }
```

VcObjectDrawing

Ereignis von VcGantt

Dieses Ereignis tritt auf, bevor ein Objekt gezeichnet wird. Damit können Sie das Objekt nach Ihren Wünschen durch eigenen Programmcode selber gestalten. Das nachfolgende, von der Komponente vorgesehene Zeichnen des Objekts kann durch Setzen des ReturnStatus auf **vcRetStatFalse** unterbunden werden.

Die ObjectDraw-Ereignisse werden nur ausgelöst, wenn dieses Verhalten für einen Objekttyp eingeschaltet wurde. Sie können sowohl Layer als auch benutzerdefinierte Skalenbeschriftungen zeichnen.

Um einen Layer zu zeichnen, setzen Sie zur Laufzeit die Eigenschaft **ObjectDrawEventsEnabled** des Objekts vom Typ **VcLayer** auf **True** oder

aktivieren zur Design-Zeit im Dialog **Balkenaussehen festlegen** die Option **ObjectDraw-Ereignisse** für den gewünschten Layer.

Zum Zeichnen einer benutzerdefinierten Skalenbeschriftung aktivieren Sie zur Design-Zeit im Dialog **Histogramme bearbeiten** unter **Skalenstreifen** die Option **ObjectDraw-Ereignisse** für den gewünschten Skalenstreifen.

Wenn Sie einem durch VARCHART XGantt gezeichneten Layer oder einer Skalenbezeichnung noch etwas hinzufügen möchten, können Sie dies im Ereignis **VcObjectDrawn**.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcObjectDrawingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcObjectDrawingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
Graphics	System.Drawing.Graphics	Gerätekontext
ObjectToDraw	Object	Objekt, das gezeichnet wird
ObjectType	VcObjectType	Typ des zu zeichnenden Objekts
	Mögliche Werte: .vcObjTypeNodeInDiagram 2 .vcObjTypeNodeInLegend 17 .vcObjTypeNumericScale 10 .vcObjTypeSummaryNode 14	 Objekttyp **Knoten im Knotenbereich** Objekttyp **Knoten im Legendenbereich** Objekttyp **Werteskala** Objekttyp **Summenbalken**
SubObject	Object	Unterobjekt, das kontextabhängig mitgegeben wird
SubObjectType	VcObjectType	Typ des Unterobjekts
	Mögliche Werte: .vcObjTypeLayer 8	 Objekttyp **Layer**
CompleteRect	VcRect	Rechteck in Gerätekoordinaten, in die das komplette Objekt gezeichnet werden soll
UpdateRect	VcRect	Rechteck in Gerätekoordinaten, das den Updatebereich bezeichnet. Dieser kann gleich groß wie oder kleiner als das Rechteck in completeRect sein
ReturnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Das Objekt wird nicht gezeichnet. Das Objekt wird gezeichnet.

LineWidth	System.Int32	Stärke einer dünnen Linie. Kann verwendet werden, um bei Zeichenbefehlen die Linienstärke der Auflösung des Gerätekontextes (Bildschirm bzw. Drucker) anzupassen.
xZoomFactor	System.Int16	Dieser Parameter gibt den aktuellen Zoomfaktor in X-Richtung an, der eine Umrechnung von Distanzen in 1/100 Millimetern nach Pixeln und umgekehrt zulässt. Der Zoomfaktor ist jeweils auf das Ausgabegerät bezogen (also Bildschirm, Druckvorschau oder Drucker).
yZoomFactor	System.Int16	Dieser Parameter gibt den aktuellen Zoomfaktor in Y-Richtung an, der eine Umrechnung von Distanzen in 1/100 Millimetern nach Pixeln und umgekehrt zulässt. Der Zoomfaktor ist jeweils auf das Ausgabegerät bezogen (also Bildschirm, Druckvorschau oder Drucker).

VcObjectDrawn

Ereignis von VcGantt

Dieses Ereignis tritt auf, nachdem ein Objekt gezeichnet wurde. Damit können Sie durch eigenen Programmcode die durch VARCHART XGantt gezeichneten Objekte gestalterisch ergänzen.

Die ObjectDraw-Ereignisse werden nur ausgelöst, wenn dieses Verhalten für einen Objekttyp eingeschaltet wurde. Sie können sowohl Layer als auch benutzerdefinierte Skalenbeschriftungen zeichnen.

Um einen Layer zu zeichnen, setzen Sie zur Laufzeit die Eigenschaft **Object-DrawEventsEnabled** des Objekts vom Typ **VcLayer** auf **True** oder aktivieren zur Design-Zeit auf der Eigenschaftenseite **Objekte** unter **Layer** die Option **ObjectDraw-Ereignisse** für den gewünschten Layer.

Zum Zeichnen einer benutzerdefinierte Skalenbeschriftung aktivieren Sie zur Design-Zeit im Dialog **Histogramme bearbeiten** unter **Skalenstreifen** die Option **ObjectDraw-Ereignisse** für den gewünschten Skalenstreifen.

Wenn Sie das standardmäßige Zeichnen der Layer oder Skalenbeschriftungen unterbinden und durch eigenen Programmcode ersetzen wollen, dann verwenden Sie bitte das Ereignis **VcObjectDrawing**.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcObjectDrawnEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcObjectDrawnEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
Graphics	System.Drawing.Graphics	Gerätekontext
ObjectType	VcObjectType	Typ des gezeichneten Objekts
	Mögliche Werte:	
	.vcObjTypeNodeInDiagram 2	Objekttyp **Knoten im Knotenbereich**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNumericScale 10	Objekttyp **Werteskala**
	.vcObjTypeSummaryNode 14	Objekttyp **Summenbalken**
SubObject	Object	Unterobjekt, das kontextabhängig mitgegeben wurde
SubObjectType	VcObjectType	Typ des Unterobjekts
	Mögliche Werte:	
	.vcObjTypeLayer 8	Objekttyp **Layer**
CompleteRect	VcRect	Rechteck in Gerätekoordinaten, in das das komplette Objekt gezeichnet wurde
UpdateRect	VcRect	Rechteck in Gerätekoordinaten, das den Updatebereich bezeichnet. Dieser kann gleich groß wie oder kleiner als das Rechteck in completeRect sein
LineWidth	SystemInt.32	Stärke einer dünnen Linie. Kann verwendet werden, um bei Zeichenbefehlen die Linienstärke der Auflösung des Gerätekontextes (Bildschirm bzw. Drucker) anzupassen.
xZoomFactor	System.Int16	Dieser Parameter gibt den aktuellen Zoomfaktor in X-Richtung an, der eine Umrechnung von Distanzen in 1/100 Millimetern nach Pixeln und umgekehrt zulässt. Der Zoomfaktor ist jeweils auf das Ausgabegerät bezogen (also Bildschirm, Druckvorschau oder Drucker).
yZoomFactor	System.Int16	Dieser Parameter gibt den aktuellen Zoomfaktor in Y-Richtung an, der eine Umrechnung von Distanzen in 1/100 Millimetern nach Pixeln und umgekehrt zulässt. Der Zoomfaktor ist jeweils auf das Ausgabegerät bezogen (also Bildschirm, Druckvorschau oder Drucker).

VcResourceSchedulingProgressing

Ereignis von VcGantt

Bei der Ressourcenplanung informiert dieses Ereignis über den Fortschritt des Planungsprozesses. Es wird die Anzahl der bereits geplanten Arbeitsaufträge und die Gesamtzahl der Aufträge bekannt gegeben. Durch Setzen des Rückgabestatus auf **vcRetStatFalse** wird die Planung abgebrochen.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcResourceSchedulingProgressingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcResourceSchedulingProgressingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ ScheduledJobCount	System.Int32	Anzahl der eingeplanten Arbeitsaufträge
⇨ TotalJobCount	System.Int32	Gesamtzahl der Arbeitsaufträge
⇦ ReturnStatus	System.Object	Rückgabestatus
		vcRetStatFalse: Der Planungsprozess wird abgebrochen
		vcRetStatDefault: Der Planungsprozess wird weitergeführt

VcResourceSchedulingWarning

Ereignis von VcGantt

Dieses Ereignis wird ausgelöst, wenn die Ressourcenplanung (s. Methode **process** im Objekt VcResourceScheduler2) Inkonsistenzen in den Datensätzen findet. Dieses Ereignis ermöglicht, bestimmte Fehler in der Datendefinition herauszufinden. Sie haben die Möglichkeit, die Ressourcenplanung abzubrechen, indem Sie den Rückgabestatus setzen.

	Datentyp	Beschreibung
Eigenschaften:		
⇦ WarningType	VcResSchedWarningType	Art der Warnung

Mögliche Werte:	
.vcResSched-AssignmentLoadPer-ItemIsZero 23	Beim angegebenen Zuweisungsdatensatz wird der Inhalt des Datenfeldes **LoadOrConsumption-PerItem** zur Zahl 0 ausgewertet, was dazu führt, dass die Zuweisung bei der Planung ignoriert wird.
.vcResSched-AssignmentNo-OperationID 3	Im mitübergebenen Zuweisungsdatensatz ist das Datenfeld für die Operationsdatensatz-ID leer, dadurch wird die Zuweisung im weiteren Ablauf ignoriert.
.vcResSched-AssignmentNo-ResourceID 1	Im mitübergebenen Zuweisungsdatensatz sind alle Datenfelder für Ressourcendatensatz-IDs leer, dadurch wird die Zuweisung im weiteren Ablauf ignoriert.
.vcResSched-AssignmentNoData-Records 0	Es sind keine Zuweisungsdatensätze vorhanden; der Parameter **DataRecord** ist **null**.
.vcResSched-Assignment-NoResourceID 2	Im mitübergebenen Zuweisungsdatensatz wurde der Ressourcendatensatz zur dort angegebenen Ressourcendatensatz-ID nicht gefunden, dadurch wird die Zuweisung im weiteren Ablauf ignoriert.
.vcResSched-AssignmentOperation-NotFound 4	Im mitübergebenen Zuweisungsdatensatz wurde der Operationsdatensatz zur dort angegebenen Operationsdatensatz-ID nicht gefunden, dadurch wird die Zuweisung im weiteren Ablauf ignoriert.
.vcResSched-AssignmentTiming-ResourceMultiple 5	Der mitübergebene Zuweisungsdatensatz stellt eine unerlaubte zweite oder weitere Zuweisung einer Operation zu einer Ressource vom Typ **vcRes-SchedTiming** dar. Dadurch wird die Zuweisung im weiteren Ablauf ignoriert.
.vcResSchedOperation-LoadPerItemIsZero 24	Beim angegebenen Operationsdatensatz wird der Inhalt des Datenfeldes für **LoadPerItem** zur Zahl 0 ausgewertet, was dazu führt, dass die Operation bei der Planung ignoriert wird.
.vcResSchedOperation-NoTaskID 6	Im mitübergebenen Operationsdatensatz ist das Datenfeld für die Auftragsdatensatz-ID leer, dadurch wird die Operation im weiteren Ablauf ignoriert.
.vcResSchedOperation-OverlapQuantityOutOf-Range 19	Die Warnung tritt auf, wenn die Überlappungs-menge (OverlapQuantity) einer Operation größer ist als die Menge des zugehörigen Auftrags. Dies führt in der Folge dazu, dass der Auftrag nicht eingeplant werden kann.
.vcResSchedOperation-StartLockDateOutOf-Range 15	Die Warnung tritt auf, wenn das feste Startdatum (StartLockDate) einer Operation nicht zwischen Freigabe- und Fälligkeitsdatum des Auftrags (ReleaseDate und DueDate) liegt. Dies führt in der Folge dazu, dass der Auftrag nicht eingeplant werden kann.
.vcResSchedOperation-TaskNotFound 7	Im mitübergebenen Operationsdatensatz wurde der Auftragsdatensatz zur dort angegebenen Auftrags-datensatz-ID nicht gefunden, dadurch wird die Operation im weiteren Ablauf ignoriert.
.vcResSchedOperation-WorkInProcessOutOf-Range 20	Die Warnung tritt auf, wenn die bereits fertig gestellte Menge in dem mit übergebenen Operationsdatensatz größer ist als die Arbeits-menge des zugehörigen Auftrags. Dies führt in der Folge dazu, dass der Auftrag nicht eingeplant werden kann.
.vcResSchedResource-CalendarNotFound 22	Die Warnung tritt auf, wenn zum Namen eines Kalenders, der im Ressourcen-Datenfeld mit dem Index aus der Eigenschaft ResourceCalendarName-FieldIndex gesetzt ist, kein Kalenderobjekt vorhanden ist.

.vcResSchedResource-GroupResourceNot-Found 10	Im mitübergebenen Ressourcendatensatz wurde der Ressourcendatensatz zur dort angegebenen Gruppenressourcendatensatz-ID nicht gefunden, dadurch kann die Ressource keiner Gruppe zugeordnet werden.
.vcResSchedResource-HistogramNotFound 21	Die Warnung tritt auf, wenn das namensgleiche Histogramm der Ressource nicht vorhanden ist.
.vcResSchedResource-InputCurveNot-Found 11	Die Eingabekurve zum mitübergebenen Ressourcendatensatz konnte nicht gefunden werden. Eingabekurven sind bei Ressourcen vom Typ **vcResSchedTiming** und **vcResSchedWork** Kapazitätskurven und bei Ressourcen vom Typ **vcResSchedMaterial** sind sie Lieferkurven.
.vcResSchedResource-InputCurveIs-CompletelyZero 12	Alle Werte der Eingabekurve zum mitübergebenen Ressourcendatensatz sind Null. Eingabekurven sind bei Ressourcen vom Typ **vcResSchedTiming** und **vcResSchedWork** Kapazitätskurven und bei Ressourcen vom Typ **vcResSchedMaterial** sind sie Lieferkurven.
.vcResSchedResource-OutputCurveNot-Found 13	Die Ausgabekurve zum mitübergebenen Ressourcendatensatz konnte nicht gefunden werden. Ausgabekurven sind bei Ressourcen vom Typ **vcResSchedTiming** und **vcResSchedWork** Belegungskurven und bei Ressourcen vom Typ **vcResSchedMaterial** sind sie Lagerbestandskurven.
.vcResSchedResource-OutputCurveOfFalse-Type 14	Die Ausgabekurve zum mitübergebenen Ressourcendatensatz kann nicht benutzt werden, weil sie nicht vom Typ **vcSetCurve** ist (s. Methode **CurveType** beim Objekt **VcCurve**). Ausgabekurven sind bei Ressourcen vom Typ **vcResSchedTiming** und **vcResSchedWork** Belegungskurven und bei Ressourcen vom Typ **vcResSchedMaterial** sind sie Lagerbestandskurven.
.vcResSchedTask-CapacityBeyond-Limit 25	Diese Warnung tritt auf, wenn in einem Auftrag mindestens eine Operation vorhanden ist, deren Kapazitätsanforderung über einem internen Limit liegt. Die Kapazitätsanforderung ergibt sich aus der Auftragsmenge beim Auftragsdatensatz, dem LoadPerItem bei der Operation sowie gegebenenfalls einem Effizienzfaktor bei der zu belegenden Ressource. Das Limit liegt zur Zeit bei 100000.
.vcResSchedTaskDue-DateEarlierThan-ReleaseDate 9	Im mitübergebenen Auftragsdatensatz liegt das Fälligkeitsdatum früher als das Freigabedatum, dadurch wird der Auftrag im weiteren Ablauf ignoriert.
.vcResSchedTaskDue-DateEqualToRelease-Date 18	Die Warnung tritt auf, wenn das Freigabedatum des Auftrags (ReleaseDate) mit dem Fälligkeitsdatum (DueDate) identisch ist. Dies führt in der Folge dazu, dass der Auftrag nicht eingeplant werden kann.
.vcResSchedTaskDue-DateOutOfRange 17	Die Warnung tritt auf, wenn das Fälligkeitsdatum (DueDate) eines Auftrags nicht zwischen Planning-StartDate und PlanningEndDate oder im per Default gesetzten sichtbaren Bereich liegt. Falls auch das Freigabedatum außerhalb des erlaubten Zeitraumes liegt, kann der Auftrag nicht eingeplant werden.
.vcResSchedTask-QuantityIsZero 8	Im mitübergebenen Auftragsdatensatz ist die Auftragsmenge Null, dadurch wird der Auftrag im weiteren Ablauf ignoriert.

	.vcResSchedTask-ReleaseDateOutOf-Range 16	Die Warnung tritt auf, wenn das Freigabedatum (ReleaseDate) eines Auftrags nicht zwischen PlanningStartDate und PlanningEndDate oder den per Default gesetzten Daten liegt. Falls auch das Fälligkeitsdatum außerhalb des erlaubten Zeit-raumes liegt, kann der Auftrag nicht eingeplant werden.
⇦ DataRecord	VcDataRecord	Datensatz, auf den sich die Warnung bezieht
⇦ ReturnStatus	VcReturnStatus	Rückgabestatus **vcRetStatFalse**: Der Planungsprozess wird abgebrochen **vcRetStatDefault**: Der Planungsprozess wird weitergeführt
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Ressourcenplanung wird abgebrochen. Die Ressourcenplanung wird fortgeführt.

VcSashButtonClicked

Ereignis von VcGantt

Um dieses Ereignis nutzen zu können, muss eine spezielle Einstellung in der Konfigurationsdatei vorgenommen werden. Bitte kontaktieren Sie bei Bedarf NETRONIC.

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf eine zuvor auf dem Sash positionierte Schaltfläche klickt. Das getroffene Sash-Objekt wird zusammen mit dem Index der angeklickten Schaltfläche (0 oder 1) als Parameter zurückgegeben.

Hinweis: Die Bitmaps/WMFs Pfeile links/rechts/oben/unten können bequem über die Methode **VcGantt.SetImageResource** auf dem Sash positioniert werden, mit folgenden Angaben für den Parameter **imageName** :

****SashHorizontalFirstButton**: Pfeil links

****SashHorizontalSecondButton**: Pfeil rechts

****SashVerticalFirstButton**: Pfeil oben

****SashVerticalSecondButton**: Pfeil unten

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt

⇨ e	VcSashButtonClickedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcSashButtonClickedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ SashType	VcSashType	Typ des getroffenen Trennbalkens
	Mögliche Werte:	
	.vcDiagramHistogramSash 3	Horizontaler Trennbalken zwischen Diagramm und Histogramm
	.vcLeftTableHistogramSash 1	Vertikaler Trennbalken zwischen linker Tabelle und Diagramm
	.vcRightTableHistogramSash 2	Vertikaler Trennbalken zwischen rechter Tabelle und Diagramm, nur verfügbar falls in der ini-Datei eine zweite Tabelle eingeschaltet wurde

VcStatusLineTextShowing

Ereignis von VcGantt

Dieses Ereignis tritt immer dann ein, wenn eine Information von allgemeiner Bedeutung bereitgestellt wird. Das kann ein funktionaler Hinweis sein (z. B. beim Laden), aber auch Informationen über einen Knoten, auf den die Maus gerade zeigt.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcStatusLineTextShowingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcStatusLineTextShowingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ text	System.String	Informationstext

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcStatusLineTextShowing(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcStatusLineTextShowingEventArgs) Handles
VcGantt1.VcStatusLineTextShowing
   TextBox1.Text = e.Text
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcStatusLineTextShowing(object sender,
NETRONIC.XGantt.VcStatusLineTextShowingEventArgs e)
   {
   textBox1.Text = e.Text;
   }
```

VcTableCaptionLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender in den Titelbereich der Tabelle mit der linken Maustaste klickt. Das getroffene Tabellen-Objekt wird zusammen mit der Spaltennummer und der Cursorposition (x,y-Koordinaten) als Parameter zurückgegeben. Wenn die Darstellung weder gruppiert noch hierarchisch ist, so werden die Vorgänge nach der getroffenen Tabellenspalte sortiert.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTableCaptionClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableCaptionClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ table	VcTable	Getroffene Tabelle
⇨ columnNumber	System.Int32	Index der getroffenen Tabellenspalte
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ 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 VcGantt1_VcTableCaptionLeftClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcTableClickingEventArgs) Handles
VcGantt1.VcTableCaptionLeftClicking
VcGantt1.LeftTable.TableFormatCollection.FirstFormat.FormatField(0).BackColor =
Color.Blue
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTableCaptionLeftClicking(object sender,
NETRONIC.XGantt.VcTableClickingEventArgs e)
   {

vcGantt1.LeftTable.TableFormatCollection.FirstFormat().get_FormatField(0).BackCo
lor = Color.LightSteelBlue;
   }
```

VcTableCaptionLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender in den Titelbereich der Tabelle mit der linken Maustaste doppelt klickt. Das getroffene Table-Objekt wird zusammen mit der Spaltennummer und der Cursorposition (x,y-Koordinaten) als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTableCaptionClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableCaptionClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ table	VcTable	Getroffene Tabelle
⇨ columnNumber	System.Int32	Index der getroffenen Tabellenspalte
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇦ 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 VcGantt1_VcTableCaptionLeftDoubleClicking(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcTableClickingEventArgs) Handles
VcGantt1.VcTableCaptionLeftDoubleClicking
   VcGantt1.LeftTable.Visible = True
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTableCaptionLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcTableClickingEventArgs e)
   {
   vcGantt1.LeftTable.Visible = true;
   }
```

VcTableCaptionRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender in den Titelbereich der Tabelle mit der rechten Maustaste klickt. Das getroffene Table-Objekt wird zusammen mit der Spaltennummer und der Cursorposition (x,y-Koordinaten) als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTableCaptionClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableCaptionClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ table	VcTable	Getroffene Tabelle
⇨ columnNumber	System.Int32	Index der getroffenen Tabellenspalte
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTableCaptionRightClicking(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcTableClickingEventArgs) Handles
VcGantt1.VcTableCaptionRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTableCaptionRightClicking(object sender,
NETRONIC.XGantt.VcTableClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcTableColumnWidthChanged

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender die Breite einer Tabellenspalte interaktiv verändert hat. Die Tabelle, der Index der geänderten Spalte und die neue Spaltenbreite (in 1/100 mm) werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	
⇨ e	VcTableColumnWidthChangedEventArgs	

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ table	VcTable	Tabelle
⇨ columnNumber	System.Int16	Index der geänderten Spalte
⇨ currentWidth	System.Int32	Neue Spaltenbreite
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Spaltenbreite wird nicht geändert. Die Spaltenbreite wird geändert.

VcTableColumnWidthChanging

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender die Breite einer Tabellenspalte interaktiv verändert. Die Tabelle, der Index der geänderten Spalte und die neue Spaltenbreite (in 1/100 mm) werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTableColumnWidthChangingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableColumnWidthChangingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ table	VcTable	Tabelle
⇨ columnNumber	System.Int16	Index der geänderten Spalte
⇨ currentWidth	System.Int32	Neue Spaltenbreite
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.vcRetStatFalse 0	Die Spaltenbreite wird nicht geändert.
	.vcRetStatOK 1	Die Spaltenbreite wird geändert.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTableColumnWidthChanging(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcTableColumnWidthChangingEventArgs) Handles
VcGantt1.VcTableColumnWidthChanging
   If e.CurrentWidth > 5000 Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
      VcGantt1.LeftTable.ColumnWidth(index) = 5000
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTableColumnWidthChanging(object sender,
NETRONIC.XGantt.VcTableColumnWidthChangingEventArgs e)
   {
   if (e.CurrentWidth > 5000)
      {
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      vcGantt1.LeftTable.set_ColumnWidth(1,5000);
      }
   }
```

VcTableColumnWidthOptimizing

Ereignis von VcGantt

Dieses Ereignis tritt bei einem Doppelklick auf die Trennlinie zwischen zwei Tabellenspalten ein, sofern auf der Eigenschaftenseite **Allgemeines** die Option **Tabellenspalten optimierbar** aktiviert wurde oder die Eigenschaft **TableColumnWidthOptimizationAllowed** gesetzt wurde. Die linke Tabellenspalte wird dann automatisch optimiert. Die Tabelle und der Index der optimierten Spalte werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTableColumnWidthOptimizingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableColumnWidthOptimizingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ table	VcTable	Tabelle
⇨ index	System.Int16	Index der geänderten Spalte
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Spaltenbreite wird nicht optimiert. Die Spaltenbreite wird optimiert.

VcTableWidthChanging

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender die Tabellenbreite interaktiv verändert hat. Die Tabelle und das neue Tabelle/Diagramm-Breitenverhältnis werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt

⇨ e	VcTableWidthChangingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableWidthChangingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ table	VcTable	Tabelle
⇨ tableWidthRatio	System.Int32	Breitenverhältnis der Tabelle zum Gesamtdiagramm (einschließlich Tabelle)
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.vcRetStatFalse 0	Die Tabelle wird nicht geändert.
	.vcRetStatOK 1	Die Tabelle wird geändert.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTableWidthChanging(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcTableWidthChangingEventArgs) Handles
VcGantt1.VcTableWidthChanging
   If e.TableDiagramWidthRatio > 30 Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
      VcGantt1.LeftTableDiagramWidthRatio = 30
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTableWidthChanging(object sender,
NETRONIC.XGantt.VcTableWidthChangingEventArgs e)
   {
   if (e.TableDiagramWidthRatio > 30)
      {
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      vcGantt1.LeftTableDiagramWidthRatio = 30;
      }
   }
```

VcTableWidthChangingEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender die Tabellenbreite interaktiv verändert hat. Die Tabelle und das neue Tabelle/Diagramm-Breitenverhältnis werden als Parameter zurückgegeben. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

Dieses Ereignis unterscheidet sich von dem Ereignis **VcTableWidthChanging** durch die Rückgabe des Parameters Tabelle/Diagramm-Breitenverhältnis als Datentyp "Double", mit dem eine

höhere Genauigkeit erzielt wird. Die Verwendung dieses Ereignisses muss zuvor über die Eigenschaft **UseHigherTableDiagramWidthRatioPrecision** oder über die Option **Höhere Genauigkeit für die Breitenverhältnisse zwischen Tabelle(n) und Diagramm** auf der Eigenschaftenseite **Allgemeines** aktiviert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTableWidthChangingExEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTableWidthChangingExEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ table	VcTable	Tabelle
⇨ tableWidthRatio	System.Double	Breitenverhältnis der Tabelle zum Gesamtdiagramm (einschließlich Tabelle)
⇦ returnStatus	VcReturnStatus	Rückgabestatus

VcTextEntrySupplying

Ereignis von VcGantt

Dieses Ereignis tritt nur auf, wenn Sie die VcGantt-Eigenschaft **TextEntry-SupplyingEventEnabled** auf **True** gesetzt haben. Das Ereignis tritt auf, wenn ein Text ausgegeben werden soll. Sie können hier alle vorgegebenen Texte durch eigene Texte ersetzen, z. B. um sie in unterschiedliche Sprachen zu übersetzen. Das betrifft die Kontextmenüs, Dialogfelder, Infoboxen, Fehlermeldungen und Monats- und Tagesnamen.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTextEntrySupplyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTextEntrySupplyingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ controlIndex	VcTextEntryIndex	Textkonstante, deren Inhalt ersetzt werden soll
	Mögliche Werte:	
	.vcTXECtxmenArrowMode 2116	Text im Kontextmenü: **Selektier-Modus**
	.vcTXECtxmenBarGroupSepLine 2111	Nicht mehr verwendete Konstante, an der API noch sichtbar
	.vcTXECtxmenCancelGrouping 2108	Nicht mehr verwendete Konstante, an der API noch sichtbar
	.vcTXECtxmenCreateBoxMode 2135	Text im Kontextmenü: **Modus: Box erzeugen**
	.vcTXECtxmenCreateLinkMode 2118	Text im Kontextmenü: **Verbindung ziehen**
	.vcTXECtxmenCreateNodeMode 2117	Text im Kontextmenü: **Knoten erzeugen**
	.vcTXECtxmenDateLineGrid 2106	Text im Kontextmenü: **Gitternetzlinien**
	.vcTXECtxmenDeleteCurvePoint 2131	Text im Kontextmenü: **Kurvenpunkt löschen**
	.vcTXECtxmenDeleteLink 2102	Text im Kontextmenü: **Verbindung löschen**
	.vcTXECtxmenDeleteNode 2101	Text im Kontextmenü: **Knoten löschen**
	.vcTXECtxmenEditGroup 2160	Text im Kontextmenü der **Gruppe:** Gruppendaten bearbeiten
	.vcTXECtxmenEditLink 2154	Text im Kontextmenü: **Verbindung bearbeiten**
	.vcTXECtxmenEditNode 2100	Text im Kontextmenü: **Daten bearbeiten**
	.vcTXECtxmenFilePrint 2122	Text im Kontextmenü: **Drucken**
	.vcTXECtxmenFilePrintPreview 2121	Text im Kontextmenü: **Druckvorschau**
	.vcTXECtxmenFilePrintSetup 2120	Text im Kontextmenü: **Drucker einrichten**
	.vcTXECtxmenFullDiagram 2156	Text im Kontextmenü **Gesamtdiagramm wiederherstellen**
	.vcTXECtxmenGraphicExport 2123	Text im Kontextmenü: **Grafik exportieren**
	.vcTXECtxmenGroupCollapse 2114	Text im Kontextmenü: **Gruppe kollabieren**
	.vcTXECtxmenGroupCollapseRowsBelow 2129	Text im Kontextmenü: **Zeilen der Gruppe kollabieren**
	.vcTXECtxmenGroupDelete 2115	Text im Kontextmenü: **Gruppe löschen**
	.vcTXECtxmenGrouped 2107	Nicht mehr verwendete Konstante, an der API noch sichtbar
	.vcTXECtxmenGroupExpand 2113	Text im Kontextmenü: **Gruppe expandieren**
	.vcTXECtxmenGroupExpandRowsBelow 2128	Text im Kontextmenü: **Zeilen der Gruppe expandieren**
	.vcTXECtxmenGroupNodesBelow 2126	Nicht mehr verwendete Konstante, an der API noch sichtbar
	.vcTXECtxmenGroupNodesInOneRow 2127	Text im Kontextmenü für **Gruppen**: **Alle Knoten in einer Zeile**
	.vcTXECtxmenGroupNodesOptimized 2124	Text im Kontextmenü für **Gruppen**: **Knoten optimiert**
	.vcTXECtxmenGroupNodesOverlaid 2125	Text im Kontextmenü für **Gruppen**: **Knoten überlappend**
	.vcTXECtxmenGroupOutlineIndent 2134	Text im Kontextmenü: **Tiefer stufen**
	.vcTXECtxmenGroupOutlineOutdent 2133	Text im Kontextmenü: **Höher stufen**
	.vcTXECtxmenGroupSortingOptions 2110	Text im Kontextmenü: **Sortieroptionen für Gruppen**
	.vcTXECtxmenInsertCurvePointMode 2130	Text im Kontextmenü: **Kurvenpunkt einfügen**

.vcTXECtxmenInvertSelection 2103	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXECtxmenPageLayout 2119	Text im Kontextmenü: **Seite einrichten**
.vcTXECtxmenReOptimizeNodesInGroup 2136	Text im Kontextmenü für**Knoten erneut optimieren**
.vcTXECtxmenShowLegendView 2157	Text im Kontextmenü **Legendenansicht anzeigen**
.vcTXECtxmenShowWorldView 2157	Text im Kontextmenü **Komplettansicht anzeigen**
.vcTXECtxmenSubDiagram 2155	Text im Kontextmenü **Teildiagramm erstellen**
.vcTXECtxmenTimeScaleEditor 2104	Text im Kontextmenü: **Zeitskala bearbeiten**
.vcTXECtxmenToggleDateLine 2105	Text im Kontextmenü: **Stichtaglinie**
.vcTXECtxmenUnmarkAllCurves 2136	Text im Kontextmenü des **Histogramms:** Alle Kurven demarkieren
.vcTXEDlgLegArrangement 2046	Text im Dialog **Legendenattribute**: **Anordnung**
.vcTXEDlgLegBottomMargin 2052	Text im Dialog **Legendenattribute**: **Unterer Rand:**
.vcTXEDlgLegFixedToColumns 2048	Text im Dialog **Legendenattribute**: **nach Spaltenanzahl**
.vcTXEDlgLegFixedToRows 2047	Text im Dialog **Legendenattribute**: **nach Zeilenanzahl**
.vcTXEDlgLegFixedToRowsAndColumns 2049	Text im Dialog **Legendenattribute**: **nach Zeilen- und Spaltenanzahl**
.vcTXEDlgLegIdcancel 2042	Schaltfläche im Dialog **Legendenattribute**: **Abbrechen**
.vcTXEDlgLegIdd 2040	Dialog Legendenattribute Beschriftung der Titelzeile
.vcTXEDlgLegIdok 2041	Schaltflächentext im Dialog **Legendenattribute**: **OK**
.vcTXEDlgLegLegendElements 2045	Text im Dialog **Legendenattribute**: **Legendenelemente**
.vcTXEDlgLegLegendFont 2053	Schaltfläche im Dialog **Legendenattribute**: **Schriftart...** für Legende
.vcTXEDlgLegLegendTitleFont 2044	Schaltfläche im Dialog **Legendenattribute**: **Schriftart...** für Legendentitel
.vcTXEDlgLegLegendTitleVisible 2043	Text im Dialog **Legendenattribute**: **Legendentitel sichtbar**
.vcTXEDlgLegMargins 2050	Text im Dialog **Legendenattribute**: **Ränder**
.vcTXEDlgLegTopMargin 2051	Text im Dialog **Legendenattribute**: **Oberer Rand:**
.vcTXEDlgNedCaptionPrefix 2024	Dialog **Vorgänge bearbeiten**,: Text für Beschriftungszeile: “Knoten”
.vcTXEDlgNedIdapply 2027	Dialog **Vorgänge bearbeiten**, "Übernehmen"-Schaltfläche
.vcTXEDlgNedIdcancel 2016	Text im Dialog **Vorgänge bearbeiten**: **Abbrechen**
.vcTXEDlgNedIdclose 2029	Dialog **Vorgänge bearbeiten**: **Schließen**-Schaltfläche
.vcTXEDlgNedIdd 2014	Überschrift des Dialogs **Vorgänge bearbeiten**
.vcTXEDlgNedIdhelp 2028	Dialog **Vorgänge bearbeiten**: **Hilfe**-Schaltfläche
.vcTXEDlgNedIdok 2015	Text im Dialog **Vorgänge bearbeiten**: **OK**
.vcTXEDlgNedNamesColStr 2018	Text im Dialog **Vorgänge bearbeiten**: **Datenfelder**
.vcTXEDlgNedTTGotoFirst 2032	Dialog **Vorgänge bearbeiten**: Tooltiptext **Ersten ausgewählten Vorgang anzeigen**

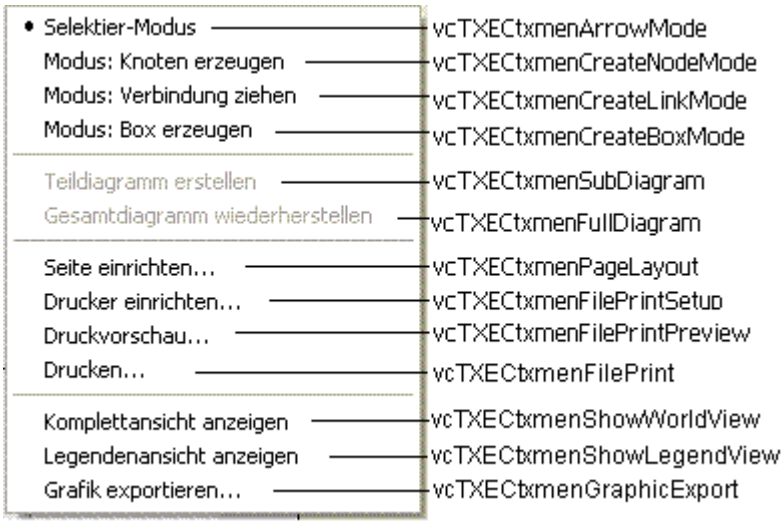
.vcTXEDlgNedTTGotoLast 2035	Dialog **Vorgänge bearbeiten**, Tooltip “Letzten ausgewählten Vorgang anzeigen“
.vcTXEDlgNedTTGotoPrev 2033	Dialog **Vorgänge bearbeiten**: Tooltiptext **Vorherigen ausgewählten Vorgang anzeigen**
.vcTXEDlgNedValuesColStr 2019	Text im Dialog **Vorgänge bearbeiten**: **Werte**
.vcTXEDlgTscEndDate 2012	Text im Dialog **Zeitskala bearbeiten**: **Ende**
.vcTXEDlgTscIdcancel 2010	Dialog **Zeitskala bearbeiten**: Schaltflächentext **Abbrechen**
.vcTXEDlgTscIdd 2008	Dialog **Zeitskala bearbeiten**: Text der Titelleiste
.vcTXEDlgTscIdok 2009	Dialog **Zeitskala bearbeiten**: Schaltflächentext **OK**
.vcTXEDlgTscScale 2013	Text im Dialog **Zeitskala bearbeiten**: **Skala**
.vcTXEDlgTscStartDate 2011	Text im Dialog **Zeitskala bearbeiten**: **Anfang**
.vcTXEErrTxtCannotMoveToEmptyRow 2735	Meldungstext: "Knoten kann nicht in nicht existierende Gruppe eingefügt werden."
.vcTXEErrTxtEndNotEarlierThanNextSect 2734	Meldungstext: "Enddatum ""%s"" ist nicht früher als Enddatum des nächsten Zeitskalenabschnitts.\n\nDas alte Datum wird wieder eingefügt."
.vcTXEErrTxtEndNotLaterThanStart 2732	Meldungstext: "Enddatum ""%s"" ist nicht später als das Startdatum.\n\nDas alte Datum wird wieder eingefügt."
.vcTXEErrTxtEntryTooLong 2730	Meldungstext: "Eintrag ist zu lang, %s Zeichen sind möglich."
.vcTXEErrTxtSpinNoButton 2727	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinNumberFormatFloat 2724	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinNumberFormatInt 2723	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinNumberMissing 2722	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinNumberTooHigh 2725	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinNumberTooLow 2726	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinUnitInsert 2720	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinUnitNotInsert 2721	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinWrongFormatString 2728	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinWrongUnitInserted 2718	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtSpinWrongUnitNotInserted 2719	Nicht mehr verwendete Konstante, an der API noch sichtbar
.vcTXEErrTxtStartNotEarlierThanEnd 2731	Meldungstext: "Startdatum ""%s"" ist nicht früher als das Enddatum.\n\nDas alte Datum wird wieder eingefügt."
.vcTXEErrTxtStartNotLaterThanPrevSect 2733	Meldungstext: "Startdatum ""%s"" ist nicht später als der Start des vorangehenden Zeitskalenabschnitts.\n\nDas alte Datum wird wieder eingefügt."
.vcTXEErrTxtWrongLongInteger 2729	Meldungstext: "Eintrag ist kein Integer oder zu lang."
.vcTXEInfWndChangeEndDate 2615	Tooltip-Text: **Ende verschieben**

.vcTXEInfWndChangeSectionStartDate 2618	Tooltip-Text: **Startdatum des Achsenabschnitts wird verändert.**
.vcTXEInfWndChangeStartDate 2614	Tooltip-Text: <**Anfang verschieben**
.vcTXEInfWndCopyActivity 2619	Tooltip-Text: **Vorgang kopieren**
.vcTXEInfWndCreateActivity 2611	Tooltip-Text: **Knoten neu anlegen**
.vcTXEInfWndDate 2620	Tooltip-Text: **Datum**
.vcTXEInfWndDateValue 12620	Tooltip-Text: **Datum**
.vcTXEInfWndDayPl 2604	Tooltip-Text: **Tage**
.vcTXEInfWndDaySi 2603	Tooltip-Text: **Tag**
.vcTXEInfWndDuration 2602	Tooltip-Text: **Dauer**
.vcTXEInfWndDurationValue 12602	Tooltip-Text: **Dauer**
.vcTXEInfWndEnd 2601	Tooltip-Text: **Ende**
.vcTXEInfWndEndValue 12601	Tooltip-Text: **Enddatum**
.vcTXEInfWndHourPl 2606	Tooltip-Text: **Stunden**
.vcTXEInfWndHourSi 2605	Tooltip-Text: **Stunde**
.vcTXEInfWndMinPl 2608	Tooltip-Text: **Minuten**
.vcTXEInfWndMinSi 2607	Tooltip-Text: **Minuten**
.vcTXEInfWndMoveActivity 2612	Tooltip-Text: **Vorgang verschieben**
.vcTXEInfWndMoveDateLine 2622	Tooltip-Text: **Stichtaglinie verschieben**
.vcTXEInfWndMoveLayer 2613	Tooltip-Text: **Layer verschieben**
.vcTXEInfWndResizeBUW 2616	Tooltip-Text: **Auflösung des Skalenabschnitts verändern**
.vcTXEInfWndResizeNumericBUW 2617	Tooltipp-Text: **Auflösung der Skala verändern**
.vcTXEInfWndSecPl 2610	Tooltipp-Text: **Sekunden**
.vcTXEInfWndSecSi 2609	Tooltip-Text: **Sekunde**
.vcTXEInfWndStart 12600	Tooltip-Text: **Startdatum der Stichtaglinie**
.vcTXEInfWndStart 2600	Tooltip-Text: **Anfang**
.vcTXEPrctBtAll 2306	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Übersicht**
.vcTXEPrctBtApply 2318	Schaltflächen-Text im **Seite einrichten**-Dialogs: **Anwenden**
.vcTXEPrctBtCancel 2302	Schaltflächen-Text im Druck-Info-Fenster: **Abbrechen**
.vcTXEPrctBtClose 2303	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Schließen**
.vcTXEPrctBtFitToPage 2308	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Einpassen**
.vcTXEPrctBtNext 2305	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Weiter**
.vcTXEPrctBtOk 2301	Schaltflächen-Text des Seitenlayout-Dialogs: **OK**
.vcTXEPrctBtPageLayout 2311	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Seite einrichten**
.vcTXEPrctBtPrevious 2304	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Vorher**
.vcTXEPrctBtPrint 2313	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Drucken**
.vcTXEPrctBtPrinterSetup 2312	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Drucker einrichten**
.vcTXEPrctBtSingle 2307	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Einzelseite**
.vcTXEPrctBtZoomPrint 2319	Schaltflächen-Text des **Druckvorschau**-Dialogs: **Ausschnitt drucken…**
.vcTXEPrctDtAddCuttingMarks 2514	Text des **Seite einrichten**-Dialogs: **Zuschnittmarken**
.vcTXEPrctDtAdjustTimescale 2560	Text des **Seite einrichten**-Dialogs: **Zeitskala an Breite der Seite anpassen**

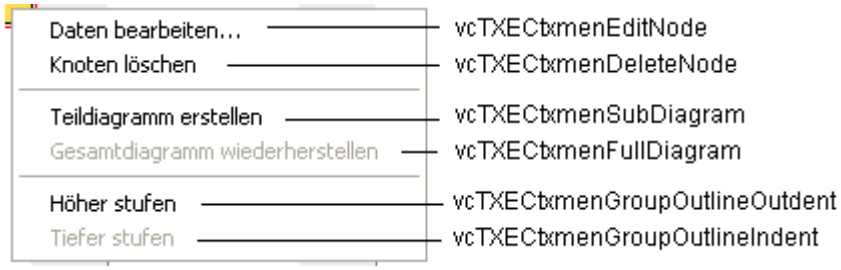
.vcTXEPrctDtAdoptTableWidthOfView 2591	Text des **Seite einrichten**-Dialogs: **Aussehen von Bildschirmansicht übernehmen**
.vcTXEPrctDtAlignment 2526	Text des **Seite einrichten**-Dialogs: **Ausrichtung**
.vcTXEPrctDtAlignmentItems 2583	Text des **Seite einrichten**-Dialogs: **Oben links|Oben|Oben rechts|Links|Mittig|Rechts|Unten links|Unten|Unten rechts**
.vcTXEPrctDtBottom 2521	Text des **Seite einrichten**-Dialogs: **Unten**
.vcTXEPrctDtCm 2530	Text des **Seite einrichten**-Dialogs: **cm**
.vcTXEPrctDtCombinedFitToPage 2574	Text des **Seite einrichten**-Dialogs: **Zoomen mit horizontaler Anpassung**
.vcTXEPrctDtCurrentValues 2581	Text des **Seite einrichten**-Dialogs: **Aktuell**
.vcTXEPrctDtEnableDiagram 2559	Text des **Seite einrichten**-Dialogs: **Diagramm anzeigen**
.vcTXEPrctDtEnableTable 2558	Text des **Seite einrichten**-Dialogs: **Tabelle anzeigen**
.vcTXEPrctDtFitToPage 2508	Text des **Seite einrichten**-Dialogs: **Anpassen an Seitenzahl**
.vcTXEPrctDtFoldingMarksItems 2577	Text des **Seite einrichten**-Dialogs: **Form A|Form B|Form C**
.vcTXEPrctDtFoldingMarksText 2576	Text des **Seite einrichten**-Dialogs:**"&Faltmarkierungen (DIN 824)**
.vcTXEPrctDtFooterGroup 2584	Text des **Seite einrichten**-Dialogs: **Fußzeile**
.vcTXEPrctDtFrameOutside 2515	Text des **Seite einrichten**-Dialogs: **Rahmen außen**
.vcTXEPrctDtInch 2588	Text des **Seite einrichten**-Dialogs: **Zoll**
.vcTXEPrctDtLeft 2520	Text des **Seite einrichten**-Dialogs: **Links**
.vcTXEPrctDtMargins 2529	Text des **Seite einrichten**-Dialogs: **Mindestgrößen für die Seitenränder**
.vcTXEPrctDtMaxPages 2580	Text des **Seite einrichten**-Dialogs: **Seiten**
.vcTXEPrctDtOff 2557	Text **Aus** Dialog
.vcTXEPrctDtOptions 2528	Text des **Seite einrichten**-Dialogs: **Seitenaufteilung**
.vcTXEPrctDtPageDescription 2562	Text des **Seite einrichten**-Dialogs: **Text**
.vcTXEPrctDtPageLayout 2532	Fenstertitel des **Seite einrichten**-Dialogs
.vcTXEPrctDtPageNumberingItems 2582	Text des **Seite einrichten**-Dialogs: **Zeile.Spalte|Spalte.Zeile|Seite/Anzahl**
.vcTXEPrctDtPageNumbers 2518	Text des **Seite einrichten**-Dialogs: **Seiten&nummerierung**
.vcTXEPrctDtPagePadding 2585	Text des **Seite einrichten**-Dialogs: **Seiten mit Leerraum auff&üllen**
.vcTXEPrctDtPagePreview 2533	Fenstertitel des Dialogs **Druckvorschau**
.vcTXEPrctDtPagesMaxHeight 2511	Text des **Seite einrichten**-Dialogs: **Maximale Höhe**
.vcTXEPrctDtPagesMaxWidth 2510	Text des **Seite einrichten**-Dialogs: **Maximale Breite**
.vcTXEPrctDtPercent 2509	Text des **Seite einrichten**-Dialogs: **%**
.vcTXEPrctDtPrintDate 2564	Text des **Seite einrichten**-Dialogs: **&Druckdatum**
.vcTXEPrctDtPrintingPage 2556	Text im Druck-Info-Fenster: **Seite %1 von %2 wird gedruckt auf**
.vcTXEPrctDtReduceExpand 2507	Text des **Seite einrichten**-Dialogs: **Zoomfaktor**

.vcTXEPrctDtRepeatTable 2565	Text des **Seite einrichten**-Dialogs für Balkenpläne: **Titel/ Tabelle/ Zeitskala/Legende wiederholen**
.vcTXEPrctDtRight 2522	Text des **Seite einrichten**-Dialogs: **Rechts**
.vcTXEPrctDtScaling 2527	Text des **Seite einrichten**-Dialogs: **Skalierung**
.vcTXEPrctDtScalingMode 2578	Text des **Seite einrichten**-Dialogs **&Modus:**
.vcTXEPrctDtStatusBarCurrentValues 2586	**Statuszeilentext** des **Druckvorschau**-Dialogs: **%1 Seiten in %2 Zeilen und %3 Spalten**
.vcTXEPrctDtStatusBarSelectedPage 2587	**Statuszeilentext** des **Druckvorschau**-Dialogs: **Seite %1 selektiert (in Zeile %2, Spalte %3)**
.vcTXEPrctDtTableColumnRange 2575	Text des **Seite einrichten**-Dialogs: **Tabellenspalten (1-5;7)**
.vcTXEPrctDtTimeColumnEnd 2590	Text des **Seite einrichten**-Dialogs: **Zeitskalenende:**
.vcTXEPrctDtTimeColumnStart 2589	Text des **Seite einrichten**-Dialogs: **Zeitskalenstart:**
.vcTXEPrctDtTop 2519	Text des **Seite einrichten**-Dialogs: **Oben**
.vcTXEPrctDtZoomFactor 2579	Text des **Seite einrichten**-Dialogs:**&Zoomfaktor:**
.vcTXEPrctMtAdjustBottomAndTopMargin 2437	Meldungstext: **Der untere Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.\r\nAußerdem wird der obere Rand auf %2 cm reduziert.**
.vcTXEPrctMtAdjustLeftAndRightMargin 2434	Meldungstext: **Der linke Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.\r\nAußerdem wird der rechte Rand auf %2 cm reduziert.**
.vcTXEPrctMtAdjustRightAndLeftMargin 2435	Meldungstext: **Der rechte Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.\r\nAußerdem wird der linke Rand auf %2 cm reduziert.**
.vcTXEPrctMtAdjustTopAndBottomMargin 2436	Meldungstext: **Der obere Rand ist außerhalb des Wertebereichs und wird deshalb auf %1 cm reduziert.\r\nAußerdem wird der untere Rand auf %2 cm reduziert.**
.vcTXEPrctMtBottomMargin 2409	Meldungstext: **Unterer Rand ...**
.vcTXEPrctMtIncompatibleVcVersion 2414	Meldungstext: **VcVersion inkompatibel**
.vcTXEPrctMtLeftMargin 2406	**Der linke Rand ist außerhalb des Wertebereichs und wird deshalb auf %s cm reduziert.**
.vcTXEPrctMtPrinterNotInstalled 2411	Meldungstext: **Kein Drucker installiert**
.vcTXEPrctMtPrintingNotPossible 2402	Meldungstext: **Drucken z. Zt nicht möglich**
.vcTXEPrctMtRightMargin 2408	Meldungstext: **Der rechte Rand ist außerhalb des Wertebereichs und wird deshalb auf %s cm reduziert.**
.vcTXEPrctMtSelectPaperSize 2413	Meldungstext: **Gewählte Blattgröße zu klein**
.vcTXEPrctMtTopMargin 2407	Meldungstext **Der obere Rand ist außerhalb des Wertebereichs und wird deshalb auf %s cm reduziert.**
.vcTXEPrctMtValueOutOfRange 2404	Meldungstext: **Außerhalb des Wertebereichs %1 bis %2**
.vcTXEPrctMtWillBeAdjustedTo 2410	Meldungstext: **Wird korrigiert auf**

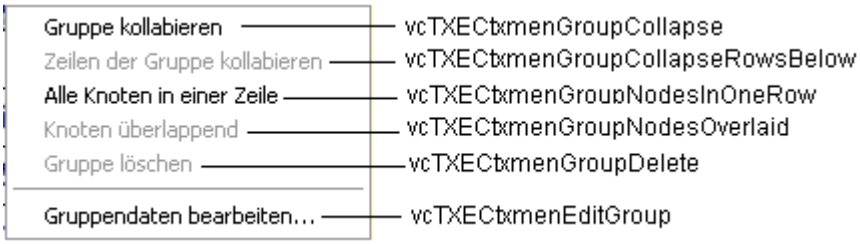
	.vcTXERelTypeLongFF 3001	Text im Dialog **Verbindungen bearbeiten**: : **Ende-Ende (FF)**
	.vcTXERelTypeLongFS 3000	Text im Dialog **Verbindungen bearbeiten**: **Ende-Anfang (FS)**
	.vcTXERelTypeLongSF 3003	Text im Dialog **Verbindungen bearbeiten**: **Anfang-Ende (SF)**
	.vcTXERelTypeLongSS 3002	Text im Dialog **Verbindungen bearbeiten**: **Anfang-Anfang (SS)**
	.vcTXERibAM 2225	Text im Skalenstreifen für **vormittags**
	.vcTXERibCW 2223	Text im Skalenstreifen für **Kalenderwoche**
	.vcTXERibDay0 2212	Text im Skalenstreifen für **Montag**
	.vcTXERibDay1 2213	Text im Skalenstreifen für **Dienstag**
	.vcTXERibDay2 2214	Text im Skalenstreifen für **Mittwoch**
	.vcTXERibDay3 2215	Text im Skalenstreifen für **Donnerstag**
	.vcTXERibDay4 2216	Text im Skalenstreifen für **Freitag**
	.vcTXERibDay5 2217	Text im Skalenstreifen für **Samstag**
	.vcTXERibDay6 2218	Text im Skalenstreifen für **Sonntag**
	.vcTXERibMon0 2200	Text im Skalenstreifen für **Januar**
	.vcTXERibMon1 2201	Text im Skalenstreifen für **Februar**
	.vcTXERibMon10 2210	Text im Skalenstreifen für **November**
	.vcTXERibMon11 2211	Text im Skalenstreifen für **Dezember**
	.vcTXERibMon2 2202	Text im Skalenstreifen für **März**
	.vcTXERibMon3 2203	Text im Skalenstreifen für **April**
	.vcTXERibMon4 2204	Text im Skalenstreifen für **Mai**
	.vcTXERibMon5 2205	Text im Skalenstreifen für **Juni**
	.vcTXERibMon6 2206	Text im Skalenstreifen für **Juli**
	.vcTXERibMon7 2207	Text im Skalenstreifen für **August**
	.vcTXERibMon8 2208	Text im Skalenstreifen für **September**
	.vcTXERibMon9 2209	Text im Skalenstreifen für **Oktober**
	.vcTXERiboClock 2224	Text im Skalenstreifen für **Uhr**
	.vcTXERibPM 2226	Text im Skalenstreifen für **nachmittags**
	.vcTXERibQuar0 2219	Text im Skalenstreifen für **1. Quartal**
	.vcTXERibQuar1 2220	Text im Skalenstreifen für **2. Quartal**
	.vcTXERibQuar2 2221	Text im Skalenstreifen für **3. Quartal**
	.vcTXERibQuar3 2222	Text im Skalenstreifen für **4. Quartal**
⇦ textEntry	System.String	Texteintrag, der den Standardtext ersetzt
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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.



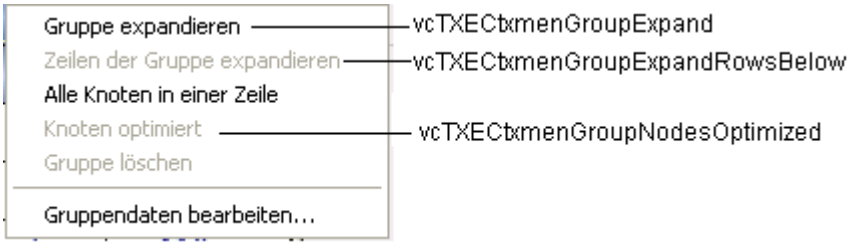
Konstanten des Kontextmenüs für das Diagramm



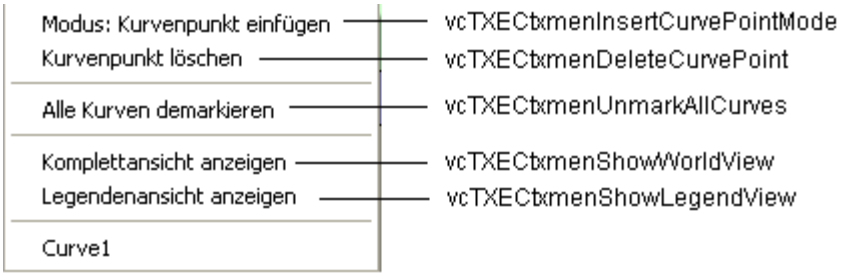
Konstanten des Kontextmenüs für Knoten



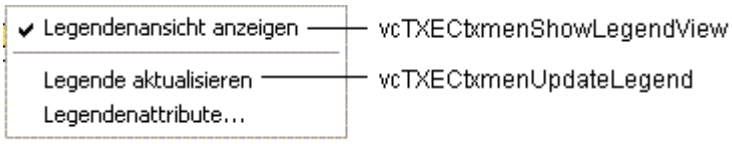
Konstanten des Kontextmenüs für Gruppen bei expandierten Gruppen



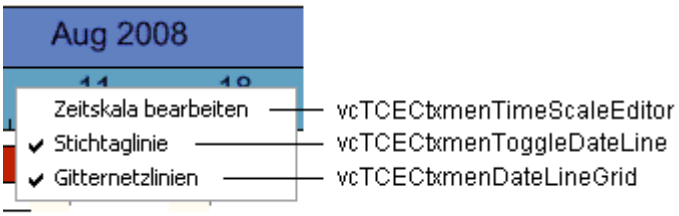
Konstanten des Kontextmenüs für Gruppen bei kollabierten Gruppen



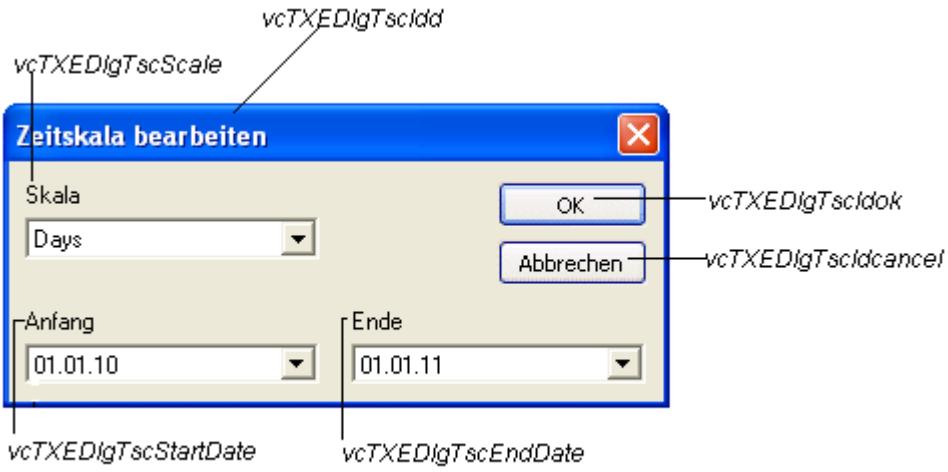
Konstanten des Kontextmenüs für Histogramme



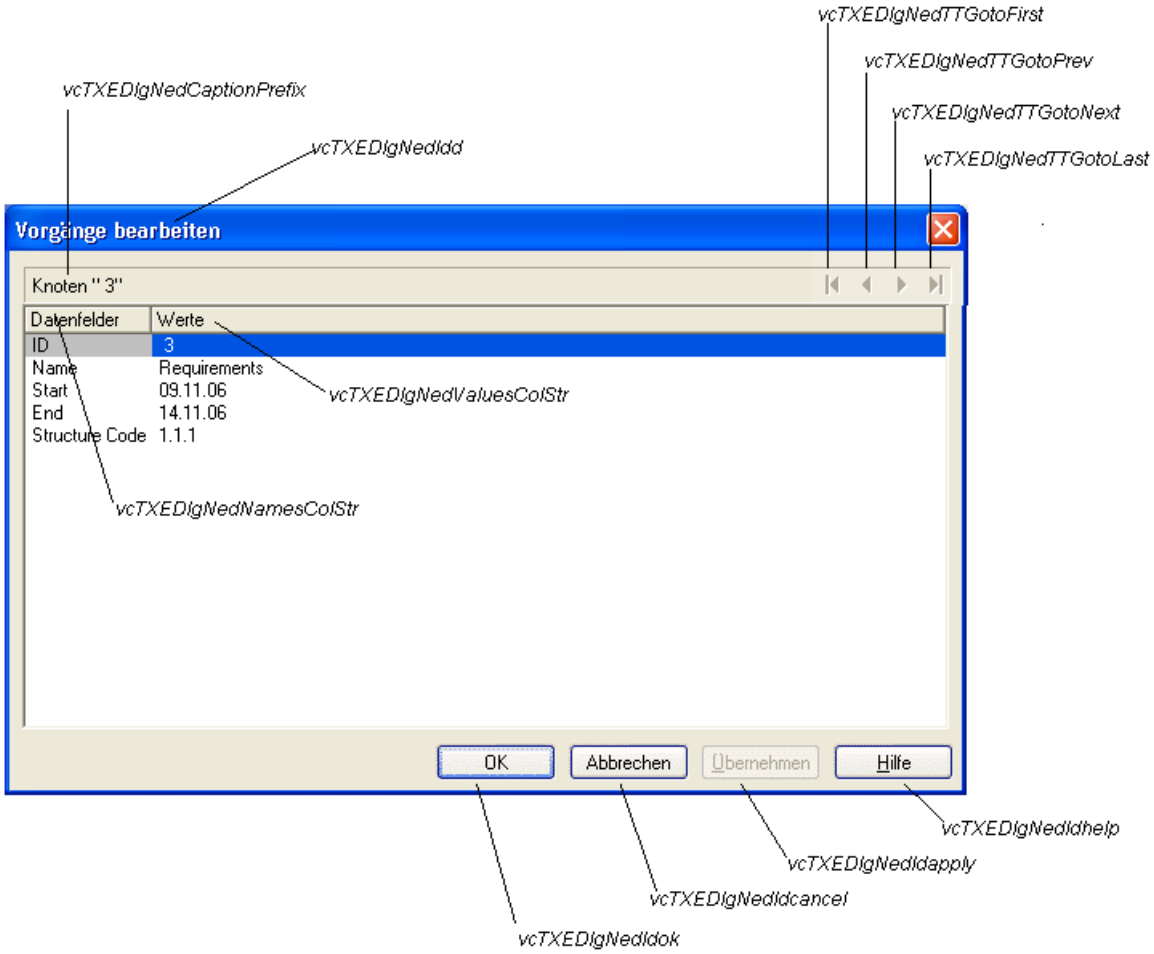
Konstanten des Kontextmenüs für die Legende



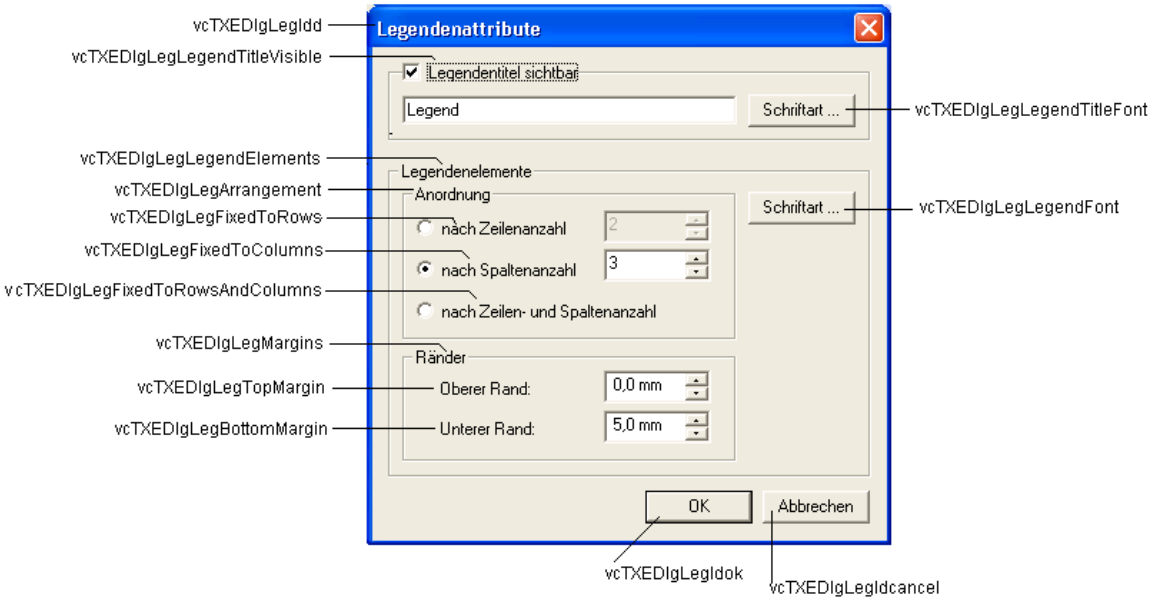
Konstanten des Kontextmenüs für die Zeitskala



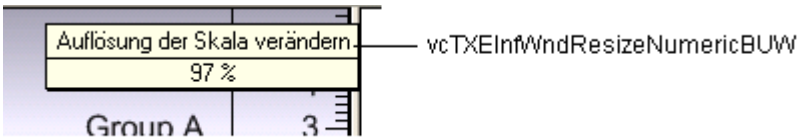
*Konstanten des Dialogs **Zeitskala bearbeiten***



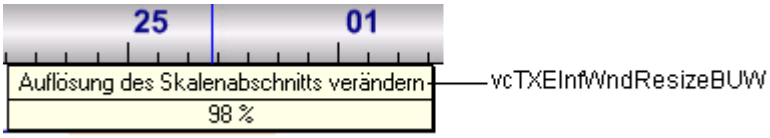
*Konstanten der Dialoge **Vorgänge bearbeiten**, **Verbindung bearbeiten** und **Gruppierung bearbeiten**, hier am Beispiel des Dialogs **Vorgänge bearbeiten**dargestellt*



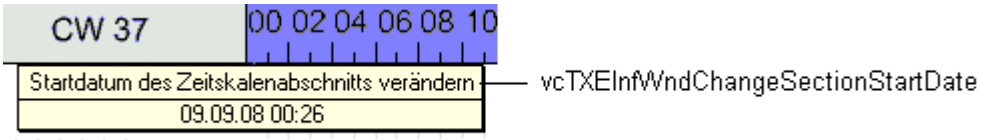
*Konstanten des Dialogs **Legendenattribute***



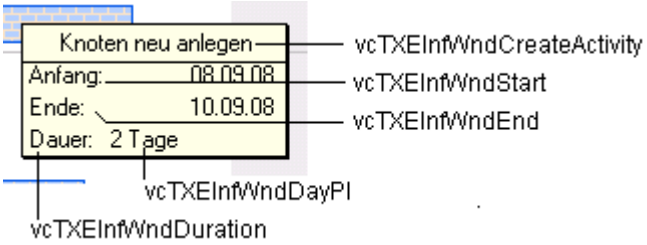
*Konstante des Tooltip-Textes, der erscheint, wenn die Größe der Grundeinheit (Basic Unit Width) der **numerischen Skala im Histogramm** verändert wird.*



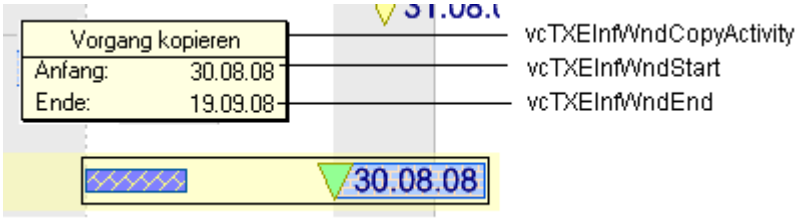
*Konstanten der Tootip-Textes, der erscheint, wenn die **Auflösung des Zeitskalenabschnitts** verändert wird*



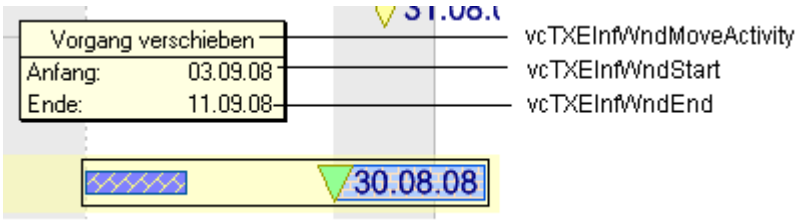
*Konstanten des Tooltip-Textes, der erscheint, wenn das **Anfangsdatum eines Zeitskalenabschnitts** verändert wird*



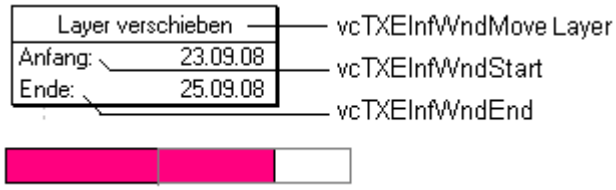
*Konstanten des Tooltip-Textes, der erscheint, wenn ein **neuer Knoten** angelegt wird*



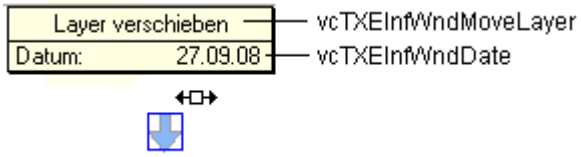
*Konstanten des Tooltip-Textes, der erscheint, wenn ein **Knoten kopiert** wird*



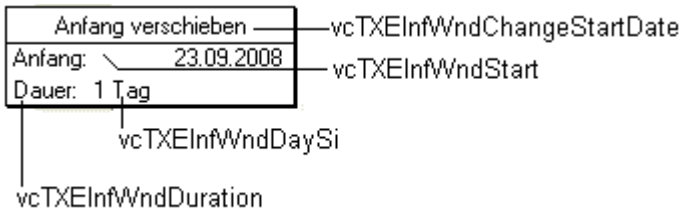
*Konstanten des Tooltip-Textes, der beim **Verschieben eines Knotens** erscheint*



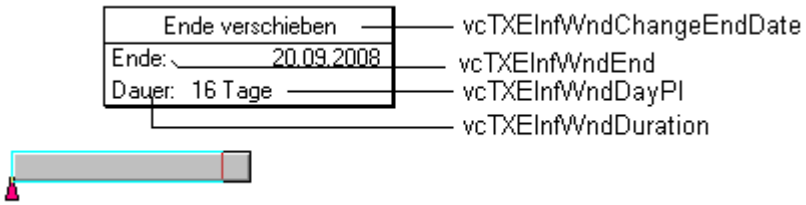
*Konstanten des Tooltip-Textes, der beim **Verschieben eines Layers** erscheint*



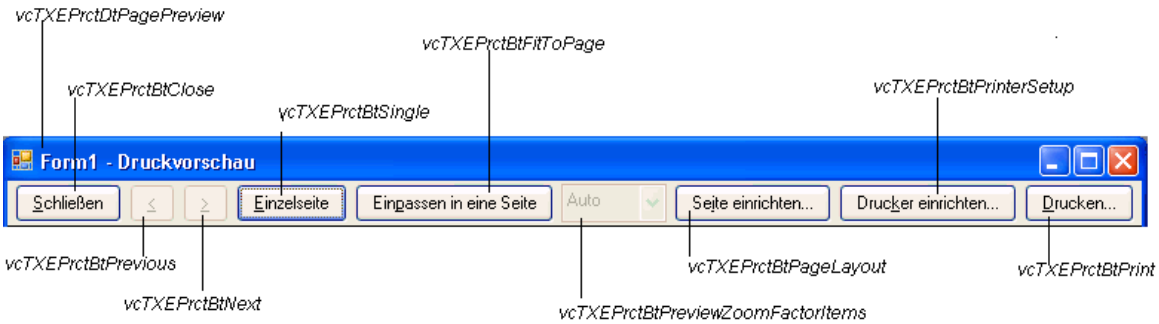
*Konstanten des Tooltip-Textes, der beim **Verschieben eines Symbollayers** erscheint*



*Konstanten des Tooltip-Textes, der erscheint, wenn der **Anfang eines Knotens** verschoben wird*

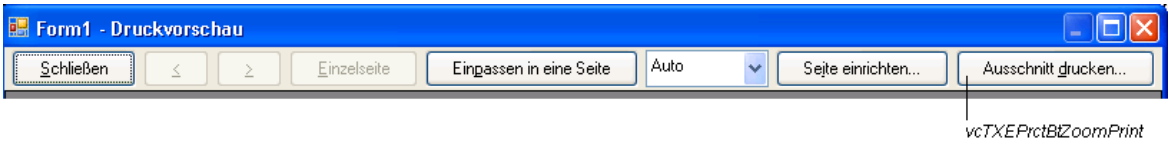


*Konstanten des Tooltip-Textes, der erscheint, wenn das **Ende eines Knotens** verschoben wird*

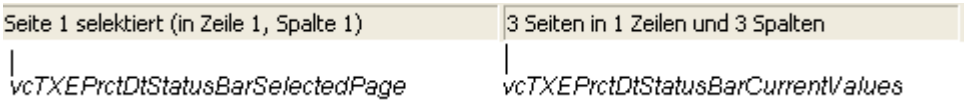


*Konstanten der Tastenbeschriftungen in der **Druckvorschau im Übersichtsmodus***

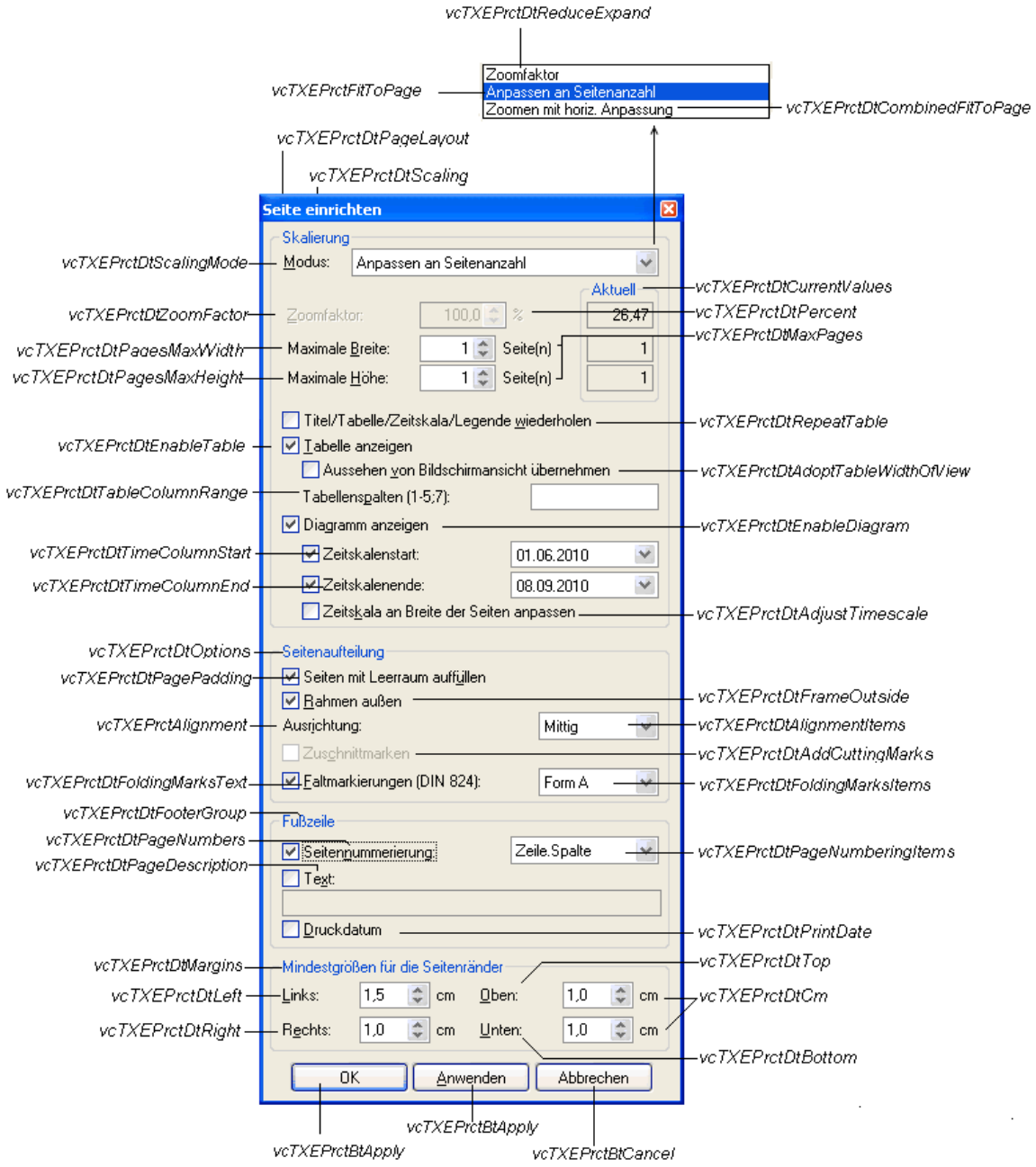
*Konstanten der Tastenbeschriftungen in der **Druckvorschau im Einzelansichtsmodus***



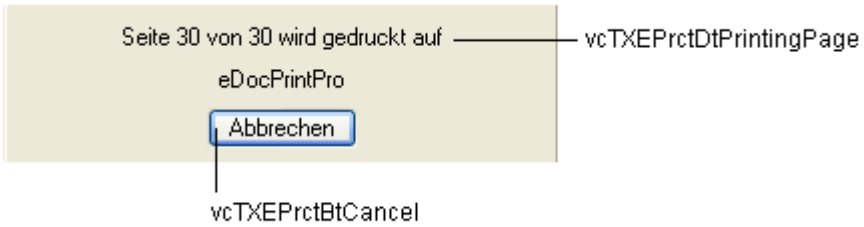
*Konstanten der Tastenbeschriftungen in der **Druckvorschau im Einzelansichtsmodus bei interaktiv ausgewählten Ausschnitt***



*Konstanten der Statuszeile im Dialog **Druckvorschau***



*Konstanten des Dialogs **Seite einrichten***



*Konstanten der Infobox **Drucken***

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.vcTXEPrctBtNext
        e.Text = "Next page"
     Case VcTextEntryIndex.vcTXEPrctBtPrevious
        e.Text = "Previous page"
      End Select
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTextEntrySupplying(object sender,
NETRONIC.XGantt.VcTextEntrySupplyingEventArgs e)
   {
   switch (e.ControlIndex)
      {
      case VcTextEntryIndex.vcTXEPrctBtNext:
         e.Text = "Next page";
         break;
      case VcTextEntryIndex.vcTXEPrctBtPrevious:
         e.Text = "Previous page";
         break;
       }
   }
```

VcTimeScaleEndModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Änderung des Enddatums der angegebenen Zeitsakala abgeschlossen ist.

	Datentyp	Beschreibung
Eigenschaften:		
⇨ newEndDate	System.DateTime	Neues Enddatum

VcTimeScaleLeftClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf die Zeitskala klickt. Das getroffene TimeScale-Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTimeScaleClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTimeScaleClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ timeScale	VcTimeScale	Getroffene Zeitskala
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcTimeScaleLeftClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcTimeScaleClickingEventArgs) Handles
VcGantt1.VcTimeScaleLeftClicking
   VcGantt1.TimeScaleCollection.Active.BackgroundColor = Color.Blue
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTimeScaleLeftClicking(object sender,
NETRONIC.XGantt.VcTimeScaleClickingEventArgs e)
   {
   vcGantt1.TimeScaleCollection.Active.BackgroundColor = Color.LightSteelBlue;
   }
```

VcTimeScaleLeftDoubleClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der linken Maustaste auf die Zeitskala doppelt klickt. Das getroffene TimeScale-Objekt wird zusammen mit der Mausposition (x,y-Koordinaten) als Parameter zurück gegeben. Das integrierte Dialogfeld kann durch Setzen des Rückgabestatus unterdrückt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTimeScaleClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTimeScaleClickingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ timeScale	VcTimeScale	Getroffene Zeitskala
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der **Zeitskala bearbeiten** Dialog erscheint nicht. Der **Zeitskala bearbeiten** Dialog erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTimeScaleLeftDoubleClicking(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcTimeScaleClickingEventArgs) Handles
VcGantt1.VcTimeScaleLeftDoubleClicking
        MsgBox("Show your own dialog")
        e.ReturnStatus = VcReturnStatus.vcRetStatFalse
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTimeScaleLeftDoubleClicking(object sender,
NETRONIC.XGantt.VcTimeScaleClickingEventArgs e)
   {
   MessageBox.Show("Show your own dialog");
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcTimeScaleModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn das Zoomen der angegebenen Zeitsakala abgeschlossen ist.

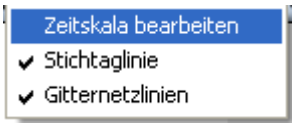
	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ timeScale	VcTimeScale	Veränderte Zeitskala

VcTimeScaleRightClicking

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender mit der rechten Maustaste auf die Zeitskala klickt. Das getroffene TimeScale-Objekt wird zusammen mit

der Mausposition (x,y-Koordinaten) als Parameter zurückgegeben. Das integrierte Kontextmenü kann durch Setzen des Rückgabestatus unterdrückt werden. Sie können so an der entsprechenden Position Ihr gewünschtes Kontextmenü anzeigen.



Oben: integriertes Kontextmenü

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTimeScaleClickingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTimeScaleClickingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ timeScale	VcTimeScale	Getroffene Zeitskala
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatNoPopup 4 .vcRetStatOK 1	 Das Kontextmenü wird unterdrückt. Das Kontext-Menu erscheint.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTimeScaleRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcTimeScaleClickingEventArgs) Handles
VcGantt1.VcTimeScaleRightClicking
        PopupMenu.Show(VcGantt1, New Point(e.X, e.Y))
        e.ReturnStatus = NETRONIC.XGantt.VcReturnStatus.vcRetStatNoPopup
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTimeScaleRightClicking(object sender,
NETRONIC.XGantt.VcTimeScaleClickingEventArgs e)
   {
   PopupMenu.Show(vcGantt1, new Point (e.X, e.Y));
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   }
```

VcTimeScaleSectionRescaled

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender einen Zeitskalenabschnitt interaktiv skaliert hat. Die Zeitskala, der Zeitskalenabschnitt und die neue **BasicUnitWidth** werden als Parameter übergeben.

	Datentyp	Beschreibung
Eigenschaften:		
⇨ timeScale	VcTimeScale	Zeitskala
⇨ sectionIndex	System.Int16	Index des Skalenabschnitts
⇨ newBasicUnitWidth	System.Int32	Neue Breite der Grundeinheit

VcTimeScaleSectionRescaledEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender einen Zeitskalenabschnitt interaktiv skaliert hat. Die Zeitskala, der Zeitskalenabschnitt und die neue **BasicUnitWidth** werden als Parameter übergeben.

	Datentyp	Beschreibung
Eigenschaften:		
⇨ timeScale	VcTimeScale	Zeitskala
⇨ sectionIndex	System.Int16	Index des Skalenabschnitts
⇨ newBasicUnitWidth	System.Double	Neue Breite der Grundeinheit

VcTimeScaleSectionRescaling

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender einen Abschnitt der Zeitskala interaktiv skaliert. Die Zeitskala, die Nummer des Zeitskalenabschnitts und die neue Grundeinheit (**BasicUnitWidth**) werden als Parameter zurückgegeben, so dass Sie beispielsweise die Zulässigkeit der Skalierung prüfen können. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTimeScaleSectionRescalingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTimeScaleSectionRescalingEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ timeScale	VcTimeScale	Zeitskala
⇨ sectionIndex	System.Int16	Index des Skalenabschnitts
⇨ newBasicUnitWidth	System.Int32	Neue Breite der Grundeinheit
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Der Zeitskalenabschnitt wird nicht verändert. Der Zeitskalenabschnitt wird verändert.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTimeScaleSectionRescaling(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcTimeScaleSectionRescalingEventArgs) Handles
VcGantt1.VcTimeScaleSectionRescaling
        If e.NewBasicUnitWidth <= 1000 Then
            MsgBox("New basic unit width: " + e.NewBasicUnitWidth)
        Else
            MsgBox("The maximum basic unit width is 1000")
            e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        End If
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTimeScaleSectionRescaling(object sender,
NETRONIC.XGantt.VcTimeScaleSectionRescalingEventArgs e)
   {
   if (e.NewBasicUnitWidth <= 1000)
      MessageBox.Show("New basic unit width: " + e.NewBasicUnitWidth);
   else
      {
      MessageBox.Show("The maximum basic unit width is 1000");
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      }
   }
```

VcTimeScaleSectionRescalingEx

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender einen Abschnitt der Zeitskala interaktiv skaliert. Die Zeitskala, die Nummer des Zeitskalenabschnitts und die neue Grundeinheit (**BasicUnitWidth**) werden als Parameter zurückgegeben, so dass Sie beispielsweise die Zulässigkeit der Skalierung prüfen können. Durch Setzen des Rückgabestatus können Sie die Änderung verhindern.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTimeScaleSectionRescalingExEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTimeScaleSectionRescalingExEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ timeScale	VcTimeScale	Zeitskala
⇨ sectionIndex	System.Int16	Index des Skalenabschnitts
⇨ newBasicUnitWidth	System.Double	Neue Breite der Grundeinheit
⇔ returnStatus	VcReturnStatus	Rückgabestatus

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTimeScaleSectionRescalingEx(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcTimeScaleSectionRescalingExEventArgs) Handles
VcGantt1.VcTimeScaleSectionRescalingEx
        If e.NewBasicUnitWidth <= 1000 Then
            MsgBox("New basic unit width: " + e.NewBasicUnitWidth)
        Else
            MsgBox("The maximum basic unit width is 1000")
            e.ReturnStatus = VcReturnStatus.vcRetStatFalse
        End If
    End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTimeScaleSectionRescalingEx(object sender,
NETRONIC.XGantt.VcTimeScaleSectionRescalingExEventArgs e)
   {
   if (e.NewBasicUnitWidth <= 1000)
      MessageBox.Show("New basic unit width: " + e.NewBasicUnitWidth);
   else
      {
      MessageBox.Show("The maximum basic unit width is 1000");
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
      }
   }
```

VcTimeScaleSectionStartModifying

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn der Anwender den Anfang eines Zeitskalenabschnitts interaktiv verändert. Die Zeitskala, die Nummer des Zeitskalenabschnitts und das neue Anfangsdatum 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 **VcTimeScaleSectionStartModified**.

Durch Setzen des Rückgabestatus kann die Änderung verhindert werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcTimeScaleSectionStartModifyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcTimeScaleSectionStartModifyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ timeScale	VcTimeScale	Zeitskala
⇨ sectionIndex	System.Int16	Index des Zeitskalenabschnitts
⇨ newStartDate	System.DateTime	Datum
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte: .vcRetStatFalse 0 .vcRetStatOK 1	 Die Veränderung wird rückgängig gemacht. Die Veränderung wird durchgeführt.

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcTimeScaleSectionStartModifying(ByVal sender As Object,
ByVal e As NETRONIC.XGantt.VcTimeScaleSectionStartModifyingEventArgs) Handles
VcGantt1.VcTimeScaleSectionStartModifying
   If MsgBox("Do you want to change the start of section No. " +
e.SectionIndex.ToString() + " to " + e.NewStartDate + "?", MsgBoxStyle.OKCancel)
= MsgBoxResult.Cancel Then
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse
   End If
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcTimeScaleSectionStartModifying(object sender,
NETRONIC.XGantt.VcTimeScaleSectionStartModifyingEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to change the start of
section No " + e.SectionIndex.ToString() + " to " + e.NewStartDate + " ?",
"Changing numeric scale", MessageBoxButtons.OKCancel);
   if (retVal ==  DialogResult.Cancel)
      e.ReturnStatus = VcReturnStatus.vcRetStatFalse;
   }
```

VcTimeScaleStartModified

Ereignis von VcGantt

Dieses Ereignis tritt ein, wenn die Änderung des Anfangsdatums der angegebenen Zeitsakala abgeschlossen ist.

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ newStartDate	System.DateTime	Neues Startdatum

VcToolTipTextSupplying

Ereignis von VcGantt

Dieses Ereignis tritt nur auf, wenn Sie die VcGantt-Eigenschaft **ToolTip-TextSupplyingEventEnabled** auf **True** gesetzt haben haben bzw. auf der Eigenschaftenseite **Allgemeines** das Kontrollkästchen **VcToolTipSupplying events** aktiviert haben. Sie können mit Hilfe dieses Ereignisses Daten des getroffenen Objektes als Tooltip anzeigen. Das Ereignis tritt auf, sobald der Cursor auf ein VcGantt-Objekt bewegt wird. Es gibt das getroffene Objekt, den Objekttyp und die Koordinaten des Cursors zurück. Durch Setzen des Return-Status auf **vcRetStatFalse** können Sie den Tooltip an dieser Stelle unterdrücken.

Bei einem Kalendergitter wird ein Tooltip-Text nur dann angefordert, wenn das Gitter identifizierbar ist, d.h. die Kalendergitter-Eigenschaft **Identifiable** auf **True** gesetzt wurde.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcToolTipTextSupplyingEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcToolTipTextSupplyingEventArgs-Objektes

	Datentyp	**Beschreibung**
Eigenschaften:		
⇨ hitObject	VcObject	Getroffenes Objekt
⇨ hitObjectType	VcObjectType	Typ des getroffenen Objektes
	Mögliche Werte:	
	.vcObjTypeBox 15	Objekttyp **Box**
	.vcObjTypeCalendarGrid 18	Objekttyp **Kalendergitter**
	.vcObjTypeCurve 12	Objekttyp **Kurve**
	.vcObjTypeDateLine 9	Objekttyp **Stichtaglinie**
	.vcObjTypeGroup 7	Objekttyp **Gruppe**
	.vcObjTypeGroupInDiagram 11	Objekttyp **Gruppe im Knotenbereich**
	.vcObjTypeGroupInTable 7	Objekttyp **Gruppe im Tabellenbereich**
	.vcObjTypeHistogram 13	Objekttyp **Histogramm**
	.vcObjTypeLayer 8	Objekttyp **Layer**
	.vcObjTypeLinkCollection 3	Objekttyp **LinkCollection**
	.vcObjTypeNodeInDiagram 2	Objekttyp **Knoten im Knotenbereich**
	.vcObjTypeNodeInLegend 17	Objekttyp **Knoten im Legendenbereich**
	.vcObjTypeNodeInTable 1	Objekttyp **Knoten im Tabellenbereich**
	.vcObjTypeNone 0	kein Objekt
	.vcObjTypeNumericScale 10	Objekttyp **Werteskala**
	.vcObjTypeSummaryNode 14	Objekttyp **Summenbalken**
	.vcObjTypeTable 4	Objekttyp **Tabelle**
	.vcObjTypeTableCaption 5	Objekttyp **Tabellenüberschrift**
	.vcObjTypeTimeScale 6	Objekttyp **Zeitskala**
⇨ x	System.Int32	X-Koordinate des Mauszeigers
⇨ y	System.Int32	Y-Koordinate des Mauszeigers
⇦ toolTipText	System.String	Anzuzeigender Text, ASP-Editionen: unbeschränkte Länge Übrige Editionen: maximal 1024 Zeichen lang
⇔ returnStatus	VcReturnStatus	Rückgabestatus
	Mögliche Werte:	
	.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 VcGantt1_VcToolTipTextSupplying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcToolTipTextSupplyingEventArgs) Handles
VcGantt1.VcToolTipTextSupplying

        Dim node As VcNode
        If Convert.ToString(e.HitObject) = "NETRONIC.XGantt.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 vcGantt1_VcToolTipTextSupplying(object sender,
NETRONIC.XGantt.VcToolTipTextSupplyingEventArgs e)
   {
   VcNode node;
   if (e.HitObject.ToString() == "NETRONIC.XGantt.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;
         }
      }
   }
```

VcViewComponentsSizeModified

Ereignis von VcGantt

Dieses Ereignis tritt auf, wenn zur Laufzeit die Größe eines grafischen Grundelements des VARCHART-Windows-Forms-Steuerelements (Zeitskala, Diagramm, Histogramm, Tabelle, Tabellenüberschrift usw.) verändert wurde. Sie können nun über die Schnittstelle darauf reagieren, indem Sie die Position und Größe aller grafischen Grundelemente abfragen. Dabei ist folgendes zu beachten:

1. Die Position bezieht sich auf den Nullpunkt des VARCHART-XGantt-Steuerelements.

2. Die Werte werden in Pixel zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcViewComponentsSizeModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcViewComponentsSizeModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇨ (no parameter)		

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcViewComponentsSizeModified(ByVal sender As Object, ByVal
e As NETRONIC.XGantt.VcViewComponentsSizeModifiedEventArgs) Handles
VcGantt1.VcViewComponentsSizeModified

   Dim x As Integer
   Dim y As Integer
   Dim width As Integer
   Dim height As Integer

VcGantt1.GetViewComponentSize(VcComponentType.vcHistogramVerScaleComponent, x,
y, width, height)

   ' plus 6 because of the sash
   TextBox1.Top = VcGantt1.Top + y + 6
   TextBox1.Left = VcGantt1.Left + x
   ' minus 25 because of the numeric scale
    TextBox1.Width = Width - 25
   ' minus 6 because of the sash
   TextBox1.Height = Height - 6
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcViewComponentsSizeModified(object sender,
NETRONIC.XGantt.VcViewComponentsSizeModifiedEventArgs e)
   {
   int x = 0;
   int y = 0;
   int width = 0;
   int height = 0;
  vcGantt1.GetViewComponentSize(VcComponentType.vcHistogramVerScaleComponent,
ref x, ref y, ref width, ref height);
   //plus 6 because of the sash
   textBox1.Top = vcGantt1.Top + y + 6;
   extBox1.Left = vcGantt1.Left + x;
   //minus 25 because of the numeric scale
   textBox1.Width = Width - 25;
   //minus 6 because of the sash
   textBox1.Height = Height - 6;
   }
```

VcWorldViewClosed

Ereignis von VcGantt

Dieses Ereignis wird aufgerufen, wenn das Popup-Fenster der Komplett-ansicht geschlossen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcWorldViewClosedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcWorldViewClosedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇦ (no parameter)		

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcWorldViewClosed(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcWorldViewClosedEventArgs) Handles VcGantt1.VcWorldViewClosed
   MsgBox("Do you want to close the worldview window?", MsgBoxStyle.OKCancel)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcWorldViewClosed(object sender,
NETRONIC.XGantt.VcWorldViewClosedEventArgs e)
   {
   DialogResult retVal = MessageBox.Show("Do you want to close the worldview
window?", "Closing worldview window", MessageBoxButtons.OKCancel);
   }
```

VcZoomFactorModified

Ereignis von VcGantt

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
Parameter:		
⇨ sender	VcGantt	Verweis auf das ereignisauslösende Objekt
⇨ e	VcZoomFactorModifiedEventArgs	Ereignisspezifisches Objekt

Eigenschaften des VcZoomFactorModifiedEventArgs-Objektes

	Datentyp	Beschreibung
Eigenschaften:		
⇦ (no parameter)		

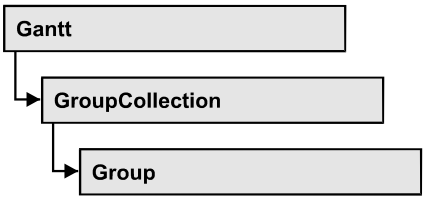
Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcZoomFactorModified(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcZoomFactorModifiedEventArgs) Handles
VcGantt1.VcZoomFactorModified
   MsgBox("Zoomfactor: " + VcGantt1.ZoomFactor)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcZoomFactorModified(object sender,
NETRONIC.XGantt.VcZoomFactorModifiedEventArgs e)
   {
   MessageBox.Show("Zoomfactor: " + vcGantt1.ZoomFactor.ToString());
   }
```

7.33 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

- BodyCollapsed
- DataField
- GroupingLevel
- GroupInvisible
- ID
- Marked
- Name
- NodeCollection
- NodesAndGroupsBelowCollapsed
- NodesInHeader
- NodesOverlaid
- SubGroups
- SuperGroup
- Visible

Methoden

- DataRecord
- Delete
- RelatedDataRecord
- ReOptimizeNodes
- Update

Eigenschaften

BodyCollapsed

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob eine Gruppe kollabiert (True) oder expandiert (False) ist. Diese Eigenschaft kann nur beim Modus Clusterung verwendet werden (GroupMode = vcGMClustering). Das Setzen kann auch interaktiv erfolgen, wenn die Eigenschaft VcGantt.GroupInteractionsAllowed eingeschaltet ist.

	Datentyp	Beschreibung
Parameter:		
⇨ Rückgabewert	System.Boolean	Gruppe kollabiert/expandiert
Eigenschaftswert	System.Boolean	Gruppe kollabiert/expandiert

Code-Beispiel VB.NET

```
body.Collapsed = True
```

Code-Beispiel C#

```
body.Collapsed = true;
```

DataField

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie den Inhalt eines Datenfeldes für den Datensatz der Gruppe festlegen oder erfragen. Der Gruppendatensatz ist eine Kopie des Datensatzes des ersten Vorganges, der in die Gruppe eingefügt wurde. Das Datenfeld wird über seinen Index angesprochen. Anschließend aktualisieren Sie die Gruppe über die Methode **Update**.

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
Parameter:		
⇨ index	System.Int16	Index des Datenfelds
Eigenschaftswert	Void	

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodecltn As VcNodeCollection
Dim node As VcNode

groupCltn = VcGantt1.GroupCollection
For Each group In groupCltn
   nodecltn = group.NodeCollection
   For Each node In nodecltn
      If node.DataField(3) > group.DataField(3) Then
          group.DataField(3) = node.DataField(3)
      End If
   Next
group.Update()
Next
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
foreach (VcGroup group in groupCltn)
   {
   VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
   foreach (VcNode node in nodeCltn)
      {
      if (node.get_DataField(3) > group.get_DataField(3))
         group.set_DataField(3,node.get_DataField(3));
      }
   group.Update();
   }
```

GroupingLevel

Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie die Gruppierebene der Gruppe bei mehrstufiger Gruppierung erfragen. Es sind maximal 25 Gruppierungsebenen möglich.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Gruppierebene der Gruppe

Code-Beispiel VB.NET

```
Dim group As VcGroup
Dim subGroup As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
node = nodeCltn.FirstNode
group = node.SuperGroup
If group.GroupingLevel > 0 Then
    subGroup = group.SuperGroup
End If
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
VcGroup  group = node.SuperGroup;
VcGroup subGroup;

if (group.GroupingLevel > 0)
   subGroup = group.SuperGroup;
```

GroupInvisible

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob diese Gruppe angezeigt wird. Der Standardwert ist der für die Gruppierungsebene festgelegte Wert.

	Datentyp	Beschreibung

ID

Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie die ID einer Gruppe erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Gruppen-ID

Code-Beispiel VB.NET

```
Code-Beispiel VB.NET
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupID As String
groupCltn = VcGantt1.GroupCollection
group = groupCltn.FirstGroup
groupID = group.ID

MsgBox (group.ID)
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
string groupID = group.ID;
MessageBox.Show(group.ID);
```

Marked

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob eine Gruppe markiert ist.

	Datentyp	Beschreibung

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
groupCltn.SelectGroups(VcSelectionType.vcSelected)

For Each group In groupCltn
         group.Marked = False
 Next
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
groupCltn.SelectGroups(VcSelectionType.vcAll);

foreach (VcGroup group in groupCltn)
   {
         Group.Marked = false;
    }
```

Name

Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie den Namen der Gruppe (= Wert des Gruppierfeldes GroupField) erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Gruppenname

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim groupName As String

groupCltn = VcGantt1.GroupCollection
group = groupCltn.FirstGroup
groupName = group.Name
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.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
Eigenschaftswert	VcNodeCollection	NodeCollection Objekt

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection

groupCltn = VcGantt1.GroupCollection
group = groupCltn.FirstGroup
nodeCltn = group.NodeCollection
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
VcNodeCollection nodeCltn = group.NodeCollection;
```

NodesAndGroupsBelowCollapsed

Eigenschaft von VcGroup

Diese Eigenschaft wirkt nur auf der 1. bis (n-1). Ebene bei mehrstufiger Gruppierung (n Ebenen). Wenn für die Gruppe alle Knoten in einer Zeile dargestellt werden, bewirkt diese Eigenschaft (True), dass nur die vorhandenen Untergruppen in der gewählten Gruppe ausgeblendet werden. Wenn Sie stattdessen diese Gruppe mit der Eigenschaft **Collapsed** kollabieren, so werden zusätzlich die Vorgänge in der Gruppe, die zu keiner Untergruppe gehören, überlappend dargestellt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Zeilen unterhalb der obersten sind/sind nicht kollabiert

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")

group.NodesAndGroupsBelowCollapsed = True
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
group.NodesAndGroupsBelowCollapsed = true;
```

NodesInHeader

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Knotenobjekte der Gruppe alle in derselben Zeile liegen (True) oder nicht (False).

	Datentyp	Beschreibung

NodesOverlaid

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das Knotenlayout optimiert ist (False) oder ob Knoten überlappen (True).

	Datentyp	Beschreibung
Parameter:		
⇦ Rückgabewert	System.Boolean	Das Knotenlayout ist/ist nicht optimal
Eigenschaftswert	System.Boolean	Das Knotenlayout ist/ist nicht optimal

SubGroups

Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie bei mehrstufiger Gruppierung Untergruppen erfragen, die in einem GroupCollection-Objekt zurückgeliefert werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcGroupCollection	GoupCollection-Objekt, das die Untergruppen enthält

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim subGroupCltn As VcGroupCollection

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
subGroupCltn = group.SubGroups
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
VcGroupCollection subGroupCltn = group.SubGroups;
```

SuperGroup

Nur-Lese-Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie bei mehrstufiger Gruppierung die Obergruppe der aktuellen Gruppe erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcGroup	Obergruppe

Code-Beispiel VB.NET

```
Dim group As VcGroup
Dim subGroup As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
node = nodeCltn.FirstNode
group = node.SuperGroup
If group.GroupingLevel > 0 Then
   superGroup = group.SuperGroup
End If
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
VcGroup group = node.SuperGroup;
VcGroup superGroup;

if (group.GroupingLevel > 0)
   superGroup = group.SuperGroup;
```

Visible

Eigenschaft von VcGroup

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob die aktuelle Gruppe sichtbar (True) oder unsichtbar (False) sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppe sichtbar/unsichtbar

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
group.Visible = True
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");

group.Visible = true;
```

Methoden

DataRecord

Methode von VcGroup

Mit dieser Eigenschaft können Sie die Gruppe 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 VcGroup

Mit dieser Methode können Sie eine Gruppe löschen. Das Löschen einer Gruppe ist nur möglich, wenn kein Knoten mehr in der Gruppe ist. Andernfalls müssen Sie zuvor alle Vorgänge aus der Gruppe löschen.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Gruppe erfolgreich/nicht erfolgreich gelöscht

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
nodeCltn = group.NodeCollection
For Each node In nodeCltn
    node.Delete()
Next
group.Delete()
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
VcNodeCollection nodeCltn = group.NodeCollection;

foreach (VcNode node in nodeCltn)
   {
   node.Delete();
   }
group.Delete();
```

RelatedDataRecord

Methode von VcGroup

Mit dieser Eigenschaft können Sie einen Datensatz aus einer verknüpften Tabelle erfragen, der dem Datensatz der Gruppendatentabelle zugeordnet ist. Der im Parameter übergebene Index bezeichnet das Feld im Datensatz, in dem der Schlüssel des zugeordneten Datensatzes steht.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes, das den Schlüssel enthält
Rückgabewert	VcDataRecord	Zurückgegebener zugeordneter Datensatz

ReOptimizeNodes

Methode von VcGroup

Ist die Eigenschaft **VcGantt.GroupOptimizationOnInteractionsEnabled** ausgeschaltet und die Knoten der Gruppe sind grundsätzlich optimiert

dargestellt, dann kann hierüber nach einer vorangegangenen Interaktion eine erneute Optimierung der Gruppe erreicht werden.

	Datentyp	Beschreibung
Rückgabewert	Void	

Update

Methode von VcGroup

Mit dieser Methode aktualisieren Sie eine Gruppe nach der Veränderung eines Datenfeldes mit der Eigenschaft **DataField**.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Gruppe erfolgreich/nicht erfolgreich aktualisiert

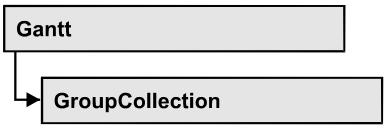
Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("A")
nodeCltn = group.NodeCollection

group.DataField(3) = nodeCltn.FirstNode.DataField(3)
For Each node In nodeCltn
   If node.DataField(3) > group.DataField(3) Then
       group.DataField(3) = node.DataField(3)
   End If
Next
group.Update()
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.GroupByName("A");
VcNodeCollection nodeCltn = group.NodeCollection;

group.set_DataField(3, nodeCltn.FirstNode().get_DataField(3));
foreach(VcNode node in nodeCltn)
   {
   if (node.get_DataField(3) > group.get_DataField(3))
      group.set_DataField(3, node.get_DataField(3));
   }
group.Update();
```

7.34 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
- SelectGroups

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 = VcGantt1.GroupCollection
numberOfGroups = groupCltn.Count
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.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 = VcGantt1.GroupCollection
group = groupCltn.FirstGroup
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.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. Existiert keine Gruppe unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ Rückgabewert	VcGroup	Gruppe
⇨ groupName	System.String	Gruppenbezeichnung
Rückgabewert	VcGroup	Gruppe

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.GroupByName("Group A")
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.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
Rückgabewert	VcGroup	Folgegruppe

Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection
Dim group As VcGroup

groupCltn = VcGantt1.GroupCollection
group = groupCltn.FirstGroup
While Not group Is Nothing
   ListBox1.Items.Add(group.Name)
   group = groupCltn.NextGroup
End While
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
VcGroup group = groupCltn.FirstGroup();
while (group != null)
   {
   listBox1.Items.Add(group.Name);
   group = groupCltn.NextGroup();
   }
```

SelectGroups

Methode von VcGroupCollection

Mit dieser Methode können Sie festlegen, welche Gruppen im GroupCollection-Objekt verfügbar sein sollen.

	Datentyp	**Beschreibung**
Parameter:		
⇨ groupSelType	VcGroupSelectionType	Auszuwählender Gruppentyp
	Mögliche Werte: .vcAllGroups 0 .vcCollapsedGroups 1 .vcExpandedGroups 2 .vcInvisibleGroups 5 .vcSelectedGroups 3 .vcVisibleGroups 4	 alle Gruppen ausgewählt kollabierte Gruppen ausgewählt expandierte Gruppen ausgewählt unsichtbare Gruppen ausgewählt ausgewählte Gruppen ausgewählt sichtbare Gruppen ausgewählt
Rückgabewert	System.Int32	Anzahl der ausgewählten Gruppen

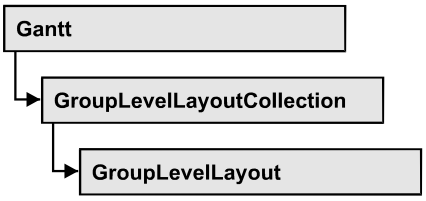
Code-Beispiel VB.NET

```
Dim groupCltn As VcGroupCollection

groupCltn = VcGantt1.GroupCollection
groupCltn.SelectGroups(VcGroupSelectionType.vcAllGroups)
```

Code-Beispiel C#

```
VcGroupCollection groupCltn = vcGantt1.GroupCollection;
groupCltn.SelectGroups(VcGroupSelectionType.vcAllGroups);
```

7.35 VcGroupLevelLayout



Ein Objekt vom Typ VcGroupLevelLayout legt Inhalt und Erscheinungsbild von Gruppierungsebenen fest. Dazu dienen der Name der Gruppierungsebene, Ebenennummer, das Feld, nach dem gruppiert wird, Sortierung und Sortierreihenfolge sowie verschiedene Optionen zum Design von Kalender- und Liniengittern sowie Trennlinien.

Eigenschaften

- AutoCollapseGroups
- AutoExpandTargetGroup
- BodiesCollapsed
- BodiesCollapsedDataFieldIndex
- BodiesCollapsedMapName
- CalendarGridName
- CalendarGridsVisible
- CalendarGridsWithChildGroups
- CalendarNameDataFieldIndex
- DateLineGridName
- DateLineGridsVisible
- DateLineGridsWithChildGroups
- DateLineName
- DateLinesVisible
- DateLinesWithChildGroups
- GroupDataFieldIndex
- GroupNodesVisible
- GroupsInvisible
- GroupsInvisibleCollapsedMapName
- GroupsInvisibleDataFieldIndex
- Level
- ModificationsAllowed
- MovingGroupsVerticallyViaDiagramAllowed
- MovingGroupsVerticallyViaTableAllowed
- Name
- NodesInHeaders

- NodesOverlaid
- OptimizedNodesSortDataFieldIndex
- OptimizedNodesSortOrder
- OverlaidNodesSortDataFieldIndex
- OverlaidNodesSortOrder
- PagebreakMode
- RestoreAutoCollapsedGroups
- RestoreAutoExpandedGroups
- RowBackColorAsARGB
- RowBackColorDataFieldIndex
- RowBackColorMapName
- RowPattern
- RowPatternColorAsARGB
- RowPatternColorDataFieldIndex
- RowPatternColorMapName
- RowPatternDataFieldIndex
- RowPatternMapName
- SeparationLineColor
- SeparationLineColorDataFieldIndex
- SeparationLineColorMapName
- SeparationLinesVisible
- SeparationLinesVisibleAtTop
- SeparationLineThickness
- SeparationLineType
- SortDataFieldIndex
- SortOrder
- Specification
- SummaryBarsVisible
- Visible

Eigenschaften

AutoCollapseGroups

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Gruppierungsebenen-Layout die Gruppen bei Interaktionen automatisch kollabiert werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppen werden/werden nicht bei Interaktionen automatisch kollabiert

AutoExpandTargetGroup

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Gruppierungsebenen-Layout Zielgruppen bei Interaktionen automatisch expandiert werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Zielgruppen werden/werden nicht bei Interaktionen automatisch expandiert

BodiesCollapsed

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob die Gruppen dieser Ebene kollabiert (True) oder expandiert (False) sind.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppe kollabiert/expandiert

BodiesCollapsedDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes für die kollabierten Rümpfen dieser Gruppierungsebene setzen oder erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes mit den kollabierten Rümpfen dieser Gruppierungsebene.

BodiesCollapsedMapName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Zuordnungstabelle für den kollabierten Gruppenkörper auf dieser Gruppenebene setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **BodiesCollapsedDataField-Index** -1 angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle für die kollabierten Rümpfe

CalendarGridName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen des Kalendergitters für dieses Gruppierebenenlayout erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalendergitters

CalendarGridsVisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob arbeitsfreie Zeiten farblich und/oder durch ein Muster hervorgehoben werden sollen. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Arbeitsfreie Zeiten werden/werden nicht hervorgehoben

CalendarGridsWithChildGroups

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen, ob auch für Untergruppen ein Kalendergitter anzeigt wird. Die Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kalendergitter für Untergruppen wird/wird nicht angezeigt

CalendarNameDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, das den Namen des für die Gruppierungsebene zu verwendenden Kalenders enthält. Diese Eigenschaft kann auch auf der Eigenschaftenseite **Kalendergitter** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das den Namen des zu verwendenden Kalenders enthält

DateLineGridName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen des Terminliniengitters für dieses Gruppierungsebenenlayout erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Terminliniengitters

DateLineGridsVisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob vertikale Rasterlinien angezeigt werden. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Rasterlinien werden/werden nicht angezeigt

DateLineGridsWithChildGroups

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen, ob auch für Untergruppen ein Terminliniengitter anzeigt wird. Die Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Terminliniengitter für Untergruppen wird/wird nicht angezeigt

DateLineName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen der Stichtalinie für dieses Gruppierungsebenenlayout erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Stichtaglinie

DateLinesVisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Stichtaglinien angezeigt werden sollen. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Stichtaglinien werden/werden nicht angezeigt

DateLinesWithChildGroups

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Stichtaglinien gruppenübergreifend angezeigt werden sollen. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Stichtaglinie für Untergruppen wird/wird nicht angezeigt

GroupDataFieldIndex

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index, der für die Gruppierung dieses VcGrouLevelLayout-Objekt verwendet wird, festlegen oder erfragen

	Datentyp	Beschreibung
Eigenschaftswert	SystemInt.32	Gruppierungsindex dieses VcGroupLevelLayout-Objektes

GroupNodesVisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Knoten dieser Gruppierungsebene angezeigt werden. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppenknoten sind/sind nicht sichtbar

GroupsInvisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Gruppen dieser Gruppierungsebene angezeigt werden. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppen sichtbar/unsichtbar

GroupsInvisibleCollapsedMapName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Zuordnungstabelle für die nicht sichtbaren Gruppen auf dieser Gruppenebene setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **BodiesCollapsedDataFieldIndex** -1 angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung

GroupsInvisibleDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes für die nicht sichtbaren Gruppen dieser Gruppierungsebene setzen oder erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	SystemInt.32	Datenfeldindex

Level

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Gruppierebene dieses Gruppierebenenlayouts erfragen. Es sind maximal 25 Gruppierungsebenen möglich.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Gruppierebene des Gruppierebenenlayouts

ModificationsAllowed

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft stellen Sie ein, ob der Anwender expandierte Gruppen dieser Ebene kollabieren oder kollabierte Gruppen wieder expandieren kann. Der Anwender kann dann die Gruppen durch einen Doppelklick auf die Gruppenüberschrift im Tabellenteil oder durch einen Klick auf das Minus- bzw. Plus-Zeichen neben der Gruppenüberschrift oder über das Kontextmenü für Gruppen kollabieren bzw. expandieren.

Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Änderungen erlaubt (True)/ nicht erlaubt (False)

Code-Beispiel VB.NET

```
VcGroupLevelLayout.ModificationsAllowed(0) = False
```

MovingGroupsVerticallyViaDiagramAllowed

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das vertikale Verschieben von Gruppen im Diagramm zugelassen ist. Sie können diese Eigenschaft auch im Dialog **Gruppierung** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Vertikales Verschieben von Gruppen im Diagramm zugelassen/nicht zugelassen **Standardwert:** True

MovingGroupsVerticallyViaTableAllowed

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das vertikale Verschieben von Gruppen in der Tabelle zugelassen ist. Sie können diese Eigenschaft auch im Dialog **Gruppierung** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Vertikales Verschieben von Gruppen in Tabelle zugelassen/nicht zugelassen **Standardwert:** true

Name

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen des Gruppierungsebenenlayouts erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Gruppierungsebene

NodesInHeaders

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Knotenobjekte der Gruppe auf dieser Gruppierungsebene alle in derselben Zeile liegen (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Alle Knoten befinden sich/befinden sich nicht in derselben Zeile

NodesOverlaid

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob auf dieser Gruppierungsebene das Knotenlayout optimiert ist (False) oder ob Knoten überlappen (True).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Das Knotenlayout ist/ist nicht optimal

Code-Beispiel VB.NET

```
group.LevelLayout.NodesOverlaid = True
```

Code-Beispiel C#

```
group.LevelLayout.NodesOverlaid = true;
```

OptimizedNodesSortDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, das das Sortierkriterium (die Zeichnungspriorität) für die Darstellung mehrere Knoten in einer Zeile enthält. Das Setzen dieser Eigenschaft ist nur sinnvoll, wenn die Eigenschaft **NodesArranged-Optimized** auf **True** gesetzt ist. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das Sortierkriterium enthält

OptimizedNodesSortOrder

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Sortierrichtung für das Sortierkriterium festlegen oder erfragen, das durch die Eigenschaft **OptimizedNodesSortDataFieldIndex** ausgewählt wurde. Das Setzen dieser Eigenschaft ist nur sinnvoll, wenn die Eigenschaft **NodesArranged-Optimized** auf **True** gesetzt ist. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodesSortingOrder	Sortierrichtung **Standardwert:** vcAscending

OverlaidNodesSortDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes festlegen oder erfragen, das das Sortierkriterium (die Zeichnungspriorität) für die Darstellung mehrere Knoten in einer Zeile enthält. Das Setzen dieser Eigenschaft ist nur sinnvoll, wenn die Eigenschaft **NodesArranged-Optimized** auf **False** gesetzt ist. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das Sortierkriterium enthält

OverlaidNodesSortOrder

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Sortierrichtung für das Sortierkriterium festlegen oder erfragen, das durch die Eigenschaft **Overlaid-NodesSortDataFieldIndex** ausgewählt wurde. Das Setzen dieser Eigenschaft ist nur sinnvoll, wenn die Eigenschaft **NodesArranged-Optimized** auf **False** gesetzt ist. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeSortingOrder	Sortierrichtung **Standardwert:** vcAscending
	Mögliche Werte: .vcAscending 1 .vcDescending 2	 aufsteigende Abfolge absteigende Abfolge

PagebreakMode

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob und wann Seitenumbrüche nach Gruppen durchgeführt werden. Diese Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcPagebreakMode	Art des Seitenumbruchs **Standardwert:** vcPagebreakNone
	Mögliche Werte: .vcPagebreakAfterEachGroup 1 .vcPagebreakNone 0 .vcPagebreakOnPageFull 2	 Seitenumbruch nach jeder Gruppe Kein Seitenumbruch Seitenumbruch, wenn Folgegruppe nicht mehr komplett auf die Seite passt

RestoreAutoCollapsedGroups

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Gruppierungsebenen-Layout automatisch kollabierte Gruppen bei Interaktionen automatisch wieder hergestellt werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Automatisch kollabierte Gruppen werden/werden nicht bei Interaktionen automatisch wieder hergestellt

RestoreAutoExpandedGroups

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Gruppierungsebenen-Layout automatisch expandierte Gruppen bei Interaktionen automatisch wieder hergestellt werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Automatisch expandierte Gruppen werden/werden nicht bei Interaktionen automatisch wieder hergestellt

RowBackColorAsARGB

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Hintergrundfarbe der Gruppentitelzeile für diese Gruppierungsebene setzen oder erfragen. Die Standard-Farbe ist weiß.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255})

RowBackColorDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **RowBackColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

RowBackColorMapName

Eigenschaft von VcGroupLevelLayout

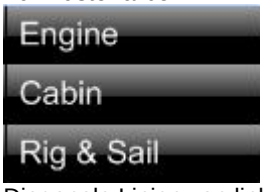
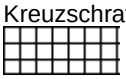
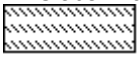
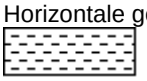
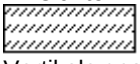
Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **RowBackColorDataFieldIndex** angegeben ist, wird die Hintergrundfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **RowBackColor** ausgegeben.

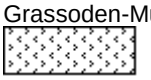
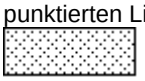
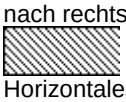
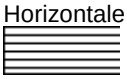
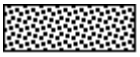
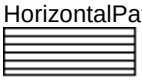
	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

RowPattern

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie für den Hintergrund des Gruppentitelzeile dieser Gruppierungsebene ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte:	
	.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 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 

RowPatternColorAsARGB

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Musterfarbe der Gruppentitelzeile dieser Gruppierungsebene festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil (ARGB-Wert) im Zahlenbereich von 0..255. 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.

S. auch **set/getRowBackColorAsARGB**.

Wenn in der Eigenschaft **RowPatternColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Musterfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255})

RowPatternColorDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **RowPatternColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

RowPatternColorMapName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **RowPatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Gruppentitelzeile aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **RowPatternColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

RowPatternDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **RowPatternMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

RowPatternMapName

Eigenschaft von VcGroupLevelLayout

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 Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **RowPatternDataFieldIndex** angegeben sind, wird das Muster des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird das Muster aus der Eigenschaft **RowPattern** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Musterzuordnungstabelle

SeparationLineColor

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Farbe der Trennlinien für die Gruppierungsebenen festlegen.

Sie können diese Eigenschaft auch im Dialog **Gruppierung**, Bereich **Gruppierung**, Feld **Trennlinie** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Farbwert ({0...255},{0...255},{0...255})

SeparationLineColorDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Farbzuordnungstabelle in der Eigenschaft **SeparationLineColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

SeparationLineColorMapName

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle für die Trennlinienfarbe setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **SeparationLineColorDataFieldIndex** -1 angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

SeparationLinesVisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob bei eingestellter Gruppierung Trennlinien zwischen den Gruppierungsebenen dargestellt werden sollen oder nicht. Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Gruppierung** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Trennlinien werden angezeigt/nicht angezeigt

SeparationLinesVisibleAtTop

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob bei eingestellter Gruppierung die Trennlinien oberhalb der Gruppen verschiedener Gruppierungsebenen dargestellt werden sollen (oder unterhalb). Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Gruppenweise** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Trennlinien oben werden angezeigt/nicht angezeigt

SeparationLineThickness

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Linienstärke einer Trennlinie zwischen Gruppierungsebenen 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 Dialog **Gruppierung**, Bereich **Gruppierung**, Feld **Trennlinie** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000} Werte in 1/100 mm

SeparationLineType

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Linienart einer Trennlinie zwischen Gruppierungsebenen erfragen oder festlegen.

Sie können diese Eigenschaft auch im Dialog **Gruppierung**, Bereich **Gruppierung**, Feld **Trennlinie** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLineType **Mögliche Werte:** .vcDashed 4 .vcDashed 4 .vcDashedDotted 5 .vcDashedDotted 5 .vcDotted 3 .vcDotted 3 .vcLineType0 100	Typ der Trennlinien der Hierarchieebenen Linientyp **gestrichelt** Linientyp **gestrichelt** Linientyp **gestrichelt-gepunktet** Linientyp **gestrichelt-gepunktet** Linientyp **gepunktet** Linientyp **gepunktet** 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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

SortDataFieldIndex

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, nach dem Gruppen auf dieser Ebene sortiert werden sollen. Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

	Datentyp	**Beschreibung**
Parameter:		
⇨ sortlevel	System.Int32	Sortierebene
Eigenschaftswert	System.Int32	Index des Datenfeldes, das das Sortierkriterium enthält

SortOrder

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft legen Sie fest, ob die Gruppen auf- oder absteigend sortiert werden. Das Feld, nach dem die Gruppen sortiert werden, legen Sie mit der Eigenschaft **SortDataFieldIndex** fest. Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

	Datentyp	Beschreibung
Parameter:		
⇨ sortLevel	System.Int32	Sortierebene
Eigenschaftswert	VcNodesSortingOrder	Sortierrichtung **Standardwert:** vcAscending

Specification

Nur-Lese-Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie die Spezifikation des Layouts der Gruppierungsebene 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 Layouts der Gruppierungsebene mit der Methode **VcGroupLevelLayout.-AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Layouts der Gruppierungsebene

SummaryBarsVisible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Summenbalken angezeigt werden oder nicht.

Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Gruppierung** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Summenbalken sichtbar (True)/ unsichtbar (False)

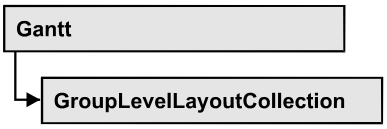
Visible

Eigenschaft von VcGroupLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob die aktuelle Gruppierebene sichtbar (True) oder unsichtbar (False) sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppierebene sichtbar/unsichtbar

7.36 VcGroupLevelLayoutCollection



In einem Objekt des Typs VcGroupLevelLayoutCollection sind alle verfügbaren Layout-Objekte der Gruppierungsebenen zusammengefasst, sofern Knoten gruppiert worden sind. Über **For Each groupLevelLayout In GroupLevelLayoutCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Layout-Ojekte zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **GroupLevelLayout-ByName** und **GroupLevelLayoutByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Layouts kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Layouts.

Eigenschaften

- Count

Methoden

- Add
- AddBySpecification
- Copy
- FirstGroupLevelLayout
- GetEnumerator
- GroupLevelLayoutByIndex
- GroupLevelLayoutByName
- NextGroupLevelLayout
- Remove
- Update

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcGroupLevelLayoutCollection

Mit dieser Eigenschaft kann die Anzahl der Gruppierungsebenenlayouts in der Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der GroupLevelLayouts

Methoden

Add

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie ein neues Gruppierungsebenenlayout in der GroupLevelLayoutCollection anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Gruppierungsebenenlayoutobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ groupLevelLayoutName	System.String	Name des GroupLevelLayouts
Rückgabewert	VcGroupLevelLayout	Neues GroupLevelLayout-Objekt

AddBySpecification

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie ein Gruppierungsebenenlayout über eine Gruppierungsebenenlayout-Spezifikation erzeugen. Dies dient der Persistenz von Gruppierungsebenenlayout-Objekten. Die Spezifikation einer Gruppierungsebene kann erfragt und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Gruppierungsebenenlayout mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ Specification	System.String	GroupLevelLayout-Spezifikation
Rückgabewert	VcGroupLevelLayout	Neues GroupLevelLayout-Objekt

Copy

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie ein Gruppierungsebenenlayout kopieren. Wenn das Gruppierungsebenenlayout mit dem angegebenen Namen existiert und der Name des neuen Gruppierungsebenenlayouts noch nicht verwendet wird, wird das neue Gruppierungsebenenlayoutobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ groupLevelLayoutName	System.String	Name des zu kopierenden GroupLevelLayouts
⇨ newGroupLevelLayoutName	System.String	Name des neuen GroupLevelLayouts
Rückgabewert	VcGroupLevelLayout	GroupLevelLayoutobjekt

FirstGroupLevelLayout

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie auf das erste Gruppierungsebenenlayout der Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextGroupLevelLayout** über die nachfolgenden Gruppierungsebenenlayouts zu iterieren. Existiert kein Gruppierungsebenenlayout in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcGroupLevelLayout	Erstes GroupLevelLayout

GetEnumerator

Methode von VcGroupLevelLayoutCollection

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 Terminlinien-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	System.Object	Referenzobjekt

GroupLevelLayoutByIndex

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie auf eine bestimmtes Gruppierungsebenenlayout über seinen Index zugreifen. Existiert kein Gruppierungsebenenlayout unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des GroupLevelLayouts
Rückgabewert	VcGroupLevelLayout	Ermitteltes GroupLevelLayoutobjekt

GroupLevelLayoutByName

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie unter Verwendung des Namens des Gruppierungsebenenlayouts auf eine bestimmtes Gruppierungsebenenlayout zugreifen. Existiert kein GroupLevelLayout-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ groupLevelLayoutName	System.String	Name des GroupLevelLayouts
Rückgabewert	VcGroupLevelLayout	GroupLevelLayout

NextGroupLevelLayout

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Gruppierungsebenenlayouts der Auflistung zugreifen, nachdem Sie mit der Methode **FirstGroupLevelLayout** den Initialwert erfasst haben. Sind alle Levels durchlaufen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcGroupLevelLayout	Nachfolgendes GroupLevelLayout

Remove

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie ein Gruppierungsebenenlayout löschen. Wenn das Gruppierungsebenenlayout noch in irgendeinem anderen Objekt benutzt wird, kann sie nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
Parameter:		
⇨ groupLevelLayoutName	System.String	Name des GroupLevelLayouts
Rückgabewert	System.Boolean	GroupLevelLayout gelöscht (True)/nicht gelöscht (False)

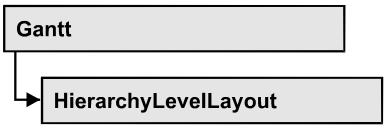
Update

Methode von VcGroupLevelLayoutCollection

Mit dieser Methode können Sie die Darstellung aller Objekte, die durch die verwendeten Gruppierungsebenenlayouts bestimmt werden, aktualisieren. Wenn Sie diese Methode nicht aufrufen, werden die Änderungen der Gruppierungsebenenlayouts zur Laufzeit nicht ausgeführt. Sie sollten diese Methode erst am Ende des Codes zur Festlegung der Gruppierungsebenenlayouts und der Level-Auflistung verwenden, damit die Aktualisierung nicht schon ausgeführt wird, bevor alle Festlegungen der Gruppierungsebenenlayouts ausgeführt worden sind.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

7.37 VcHierarchyLevelLayout



Ein Objekt vom Typ **VcHierarchyLevelLayout** legt Inhalt und Erscheinungsbild einer hierarchischen Anordnung von Knoten fest.

Eigenschaften

- AutoCollapseGroups
- AutoExpandTargetGroup
- BodiesCollapsed
- BodiesCollapsedDataFieldIndex
- BodiesCollapsedMapName
- HierarchyDataFieldIndex
- LevelMaximumForPagebreaks
- NodeSeparationLinesVisible
- NodesInHeaders
- NodesOverlaid
- PagebreakMode
- RestoreAutoCollapsedGroups
- RestoreAutoExpandedGroups
- SeparationLineColor
- SeparationLinesVisible
- SeparationLineThickness
- SeparationLineType
- SummaryBarsVisible

Eigenschaften

AutoCollapseGroups

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Hierarchiebenen-Layout die Gruppen bei Interaktionen automatisch kollabiert werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppen werden/werden nicht bei Interaktionen automatisch kollabiert

AutoExpandTargetGroup

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Hierarchiebenen-Layout Zielgruppen bei Interaktionen automatisch expandiert werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Zielgruppen werden/werden nicht bei Interaktionen automatisch expandiert

BodiesCollapsed

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob alle Gruppen kollabiert sind (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Gruppe kollabiert/expandiert

BodiesCollapsedDataFieldIndex

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes für die kollabierten Rümpfen dieser Hierarchieeebene setzen oder erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes mit den kollabierten Rümpfen dieser Hierarchiebene

BodiesCollapsedMapName

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Zuordnungstabelle für den kollabierten Gruppenkörper auf dieser Hierarchieebene setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **BodiesCollapsedDataField-Index** -1 angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle für die kollabierten Rümpfe

HierarchyDataFieldIndex

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, nach dem Vorgänge in diesem **VcGroupLevelLayout**-Objekt hierarchisch angeordnet werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, nach dem Vorgänge hierarchisch angeordnet werden sollen

LevelMaximumForPagebreaks

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, bis zu welcher Hierarchieebene Seitenumbrüche ausgeführt werden.

Ist der Standardwert -1 gesetzt, werden für alle Ebenen Seitenumbrüche durchgeführt.

	Datentyp	Beschreibung

NodeSeparationLinesVisible

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Trennlinien zwischen Knoten angezeigt werden oder nicht.

Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Knoten** festgelegt werden.

	Datentyp	Beschreibung

NodesInHeaders

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Knotenobjekte der Gruppe auf dieser Hierarchieebene alle in derselben Zeile liegen (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Alle Knoten befinden sich/befinden sich nicht in derselben Zeile

NodesOverlaid

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob auf dieser Hierarchieebene das Knotenlayout optimiert ist (False) oder ob Knoten überlappen (True).

	Datentyp	Beschreibung
Parameter:		
⇦ Rückgabewert	System.Boolean	Das Knotenlayout ist/ist nicht optimal
Eigenschaftswert	System.Boolean	Das Knotenlayout ist/ist nicht optimal

PagebreakMode

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob und wann Seitenumbrüche nach Gruppen durchgeführt werden. Diese Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcPagebreakMode	Art des Seitenumbruchs **Standardwert:** vcPagebreakNone
	Mögliche Werte: .vcPagebreakAfterEachGroup 1 .vcPagebreakNone 0 .vcPagebreakOnPageFull 2	 Seitenumbruch nach jeder Gruppe Kein Seitenumbruch Seitenumbruch, wenn Folgegruppe nicht mehr komplett auf die Seite passt

RestoreAutoCollapsedGroups

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Hierarchiebenen-Layout automatisch kollabierte Gruppen bei Interaktionen automatisch wieder hergestellt werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Automatisch kollabierte Gruppen werden/werden nicht bei Interaktionen automatisch wieder hergestellt

RestoreAutoExpandedGroups

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob im Hierarchiebenen-Layout automatisch expandierte Gruppen bei Interaktionen automatisch wieder hergestellt werden sollen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Automatisch expandierte Gruppen werden/werden nicht bei Interaktionen automatisch wieder hergestellt

SeparationLineColor

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie die Farbe der Trennlinien für die Hierarchieebenen festlegen.

Sie können diese Eigenschaft auch im **Hierarchie**-Bereich des Dialogs **Gruppierung** festlegen, indem Sie bei **Trennlinie** auf ... klicken.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Farbwert ({0...255},{0...255},{0...255})

SeparationLinesVisible

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob bei eingestellter Hierarchie Trennlinien zwischen den Hierarchieebenen dargestellt werden sollen oder nicht.

Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Hierarchie** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Trennlinien werden angezeigt/nicht angezeigt

SeparationLineThickness

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie die Linienstärke einer Trennlinie zwischen Hierarchieebenen 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 **Hierarchie**-Bereich des Dialogs **Gruppierung** festlegen, indem Sie bei **Trennlinie** auf ··· klicken.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm

SeparationLineType

Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie die Linienart einer Trennlinie zwischen Hierarchieebnenen erfragen oder festlegen.

Sie können diese Eigenschaft auch im **Hierarchie**-Bereich des Dialogs **Gruppierung** festlegen, indem Sie bei **Trennlinie** auf ··· klicken.

	Datentyp	Beschreibung
Eigenschaftswert	LineTypeEnum	Typ der Trennlinien der Hierarchieebenen

SummaryBarsVisible

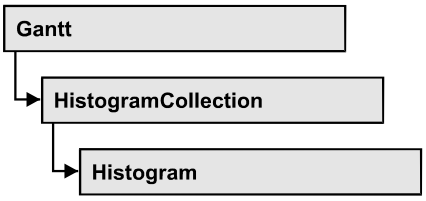
Eigenschaft von VcHierarchyLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Summenbalken angezeigt werden oder nicht.

Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Hierarchie** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Summenbalken sichtbar (True)/ unsichtbar (False)

7.38 VcHistogram



Ein Objekt vom Typ VcHistogram ist ein Element des Objektes **VcHistogramCollection** und bezeichnet ein Histogramm, das sinnvollerweise mit Auslastungs- und Kapazitätskurven zu dem darüber liegenden Gantt-Diagramm bestückt wird. Sie können hier eine Skala anlegen und Kurven definieren, die aus verschiedenen Quellen generiert werden können.

Eigenschaften

- CalendarGridsVisible
- CalendarName
- CurveCollection
- Name
- NominalScaleMaximum
- NominalScaleMinimum
- NumericScaleCollection
- RowBackColorAsARGB
- RowPattern
- RowPatternColorAsARGB
- Visible

Methoden

- FitRangeIntoView
- GetActualScaleValues
- GetCurrentYValues
- PutInOrderAfter
- ScrollToValue

Eigenschaften

CalendarGridsVisible

Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob arbeitsfreie Zeiten farblich und/oder durch ein Muster hervorgehoben werden sollen. Sie können diese Eigenschaft auch im Dialog **Histogramme verwalten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Arbeitsfreie Zeiten werden/werden nicht hervorgehoben

CalendarName

Nur-Lese-Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie dem Histogramm einen Kalender zuweisen. Der Kalender beinhaltet das abzubildende Zeitmuster für die Gitterlinien und wird mit Hilfe seines Namens spezifiziert.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Kalendernamen übergibt

CurveCollection

Nur-Lese-Eigenschaft von VcHistogram

Mit dieser Eigenschaft haben Sie Zugriff auf die Kurvenauflistung, in der alle zur Verfügung stehenden Kurven zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcCurveCollection	CurveCollection-Objekt

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram
Dim curveCltn As VcCurveCollection

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
curveCltn = histogram.CurveCollection
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
VcCurveCollection curveCltn = histogram.CurveCollection;
```

Name

Nur-Lese-Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie den Namen eines Histogramms erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Histogramms

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.Active
MsgBox(histogram.Name)
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram  histogram = histogramCltn.Active;
MessageBox.Show(histogram.Name);
```

NominalScaleMaximum

Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie den Maximalwert der numerischen Skala des Histogramms beim Starten des Programms setzen. Falls die y-Werte der Histogrammkurven diesen Maximalwert überschreiten, wird die numerische Skala an die y-Werte der Histogrammkurven angepasst.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Maximaler y-Wert

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.NominalScaleMaximum (20)
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
histogram.NominalScaleMaximum(20);
```

NominalScaleMinimum

Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie den Minimalwert der numerischen Skala des Histogramms setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Minimaler y-Wert

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.NominalScaleMinimum (2)
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
histogram.NominalScaleMinimum(2);
```

NumericScaleCollection

Nur-Lese-Eigenschaft von VcHistogram

Diese Eigenschaft ermöglicht den Zugriff auf das NumericScaleCollection-Objekt, in dem alle zur Verfügung stehenden numerischen Skalen zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcNumericScaleCollection	NumericScaleCollection-Objekt

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram
Dim numericScaleCltn As VcNumericScaleCollection

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
numericScaleCltn = histogram.NumericScaleCollection
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
VcNumericScaleCollection numericScaleCltn = histogram.NumericScaleCollection;
```

RowBackColorAsARGB

Nur-Lese-Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie die Hintergrundfarbe für das Histogramm setzen oder erfragen. Diese Option können Sie auch im Dialog "Histogramme verwalten festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255})

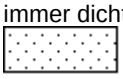
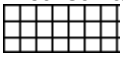
Code-Beispiel VB.NET

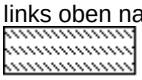
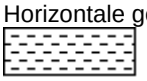
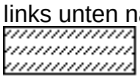
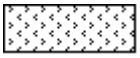
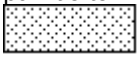
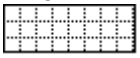
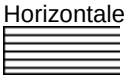
```
VcHistogram.RowBackColor = RGB(255, 0, 0)
```

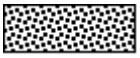
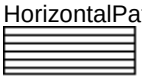
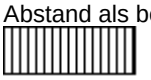
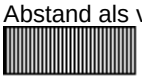
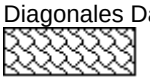
RowPattern

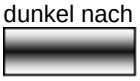
Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie für den Hintergrund des Histogramms ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte: .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 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 

RowPatternColorAsARGB

Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie die Musterfarbe des Histogramms festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil (ARGB-Wert) im Zahlenbereich von 0..255. 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.

Tipp:> Der Hintergrund im Bereich der numerischen Skala wird erst dann sichtbar, wenn deren Streifenhintergrund transparent ist, ansonsten bleibt immer der Streifenhintergrund sichtbar.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255})

Visible

Eigenschaft von VcHistogram

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das Histogramm sichtbar ist.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Histogramm sichtbar (True)/ nicht sichtbar (False)

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.Active
histogram.Visible = True
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.Active;
histogram.Visible = true;
```

Methoden

FitRangeIntoView

Methode von VcHistogram

Mit dieser Methode wird ein bestimmter Teilbereich der numerischen Skala in das Histogramm eingepasst und damit sichtbar gemacht. Die Skalierung der Skala wird dabei entsprechend verändert. Mit den Parametern startValue und **endValue** werden Anfang und Ende des Bereiches festgelegt. Der Parameter **gapAsNoOfTimeUnits** wird nicht

verwendet. Um passende Bereichsgrenzen aus den vorhandenen Kurven abzuleiten s. **GetCurrentYValues(...)**

Für das Einpassen von Histogrammen in ein Fenster s. **VcGantt.FitHistogramsIntoView**.

	Datentyp	Beschreibung
Parameter:		
⇨ startValue	System.Int32	Anfangswert des einzupassenden Bereiches
⇨ endValue	System.Int32	Endwert des einzupassenden Bereiches
⇨ gapAsNoOfTimeUnits	System.Int32	Parameter wird nicht verwendet
Rückgabewert	System.Boolean	Bereich wurde erfolgreich/nicht erfolgreich eingepasst.

GetActualScaleValues

Methode von VcHistogram

Mit dieser Methode können Sie die tatsächlichen minimalen und maximalen Werte der numerischen Skala im Histogramm ermitteln.

	Datentyp	Beschreibung
Parameter:		
⇦ minimumValue	System.Int32	Kleinster tatsächlicher Wert der numerischen Skala
⇦ maximumValue	System.Int32	Größter tatsächlicher Wert der numerischen Skala
Rückgabewert	System.Boolean	Extremwerte wurden erfolgreich (True) / nicht erfolgreich (False) ermittelt.

GetCurrentYValues

Methode von VcHistogram

Mit dieser Methode können Sie den minimalen und maximalen Y-Wert aller Kurven des Histogramms ermitteln. Diese Abfrage kann dazu dienen, z. B. den sichtbaren Bereich der numerischen Skala entsprechend anzupassen (s. **FitRangeIntoView**)

	Datentyp	Beschreibung
Parameter:		
⇦ minValue	System.Int32	Kleinster Y-Wert aller Kurven.

⇦ maxValue	System.Int32	Größter Y-Wert aller Kurven.
Rückgabewert	System.Boolean	Extremwerte wurden erfolgreich (True) / nicht erfolgreich (False) ermittelt.

PutInOrderAfter

Methode von VcHistogram

Mit dieser Methode können Sie das Histogramm in der Auflistung aller Histogramme hinter das durch den Namen angegebene setzen. Wenn kein Name übergeben wird, wird das Histogramm an die erste Stelle gesetzt. Die Reihenfolge der Histogramme in der Auflistung entscheidet darüber, in welcher Reihenfolge sie dargestellt werden.

	Datentyp	**Beschreibung**
Parameter:		
⇨ refName	System.String	Name des Histogramms, hinter das das aktuelle Histogramm eingeordnet werden soll.
Rückgabewert	Void	

Code-Beispiel VB.NET

```
Dim histgrCltn As VcHistogramCollection
Dim histgr1 As VcHistogram
Dim histgr2 As VcHistogram

histgrCltn = VcGantt1.HistogramCollection()
histgr1 = histgrCltn.Add("histgr1")
histgr2 = histgrCltn.Add("histgr2")
histgr1.PutInOrderAfter("histgr2")
histgrCltn.Update()
```

Code-Beispiel C#

```
VcHistogramCollection histgrCltn = vcGantt1.HistogramCollection;
VcHistogram histgr1 = histgrCltn.Add("histgr1");
VcHistogram histgr2 = histgrCltn.Add("histgr2");
histgr1.PutInOrderAfter("histgr2");
histgrCltn.Update();
```

ScrollToValue

Methode von VcHistogram

Mit dieser Methode können Sie zu einem bestimmten y-Wert im Histogramm scrollen und festlegen, ob dieser Wert oben, unten oder mittig auf dem Bildschirm erscheinen soll.

	Datentyp	Beschreibung
Parameter:		
⇨ value	System.Int32	y-Wert, zu dem gescrollt werden soll
⇨ verAlignment	VcVerticalAlignment	Vertikale Ausrichtung
	Mögliche Werte: .vcBottomAligned 2 .vcTopAligned 1 .vcVerCenterAligned -1	 bündig unten bündig oben vertikal mittig
Rückgabewert	System.Boolean	Scrollen erfolgreich/nicht erfolgreich durchgeführt

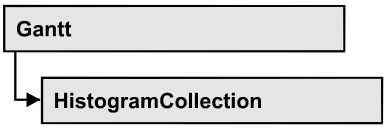
Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("HISTOGRAM_1")
histogram.ScrollToValue(7, VcVerticalAlignment.vcVerCenterAligned)
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("HISTOGRAM_1");
histogram.ScrollToValue(7, VcVerticalAlignment.vcVerCenterAligned);
```

7.39 VcHistogramCollection



Im HistogramCollection-Objekt sind alle Histogramme automatisch zusammengefasst. Über **For Each histogram In HistogramCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Histogramme zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaft **HistogramByName**. Die Anzahl der im Auflistungsobjekt vorhandenen Histogramme kann über die Eigenschaft **Count** erfragt werden.

Eigenschaften

- Active
- Count

Methoden

- CreateHistogram
- Delete
- FirstHistogram
- GetEnumerator
- HistogramByIndex
- HistogramByName
- NextHistogram

Eigenschaften

Active

Eigenschaft von VcHistogramCollection

Mit dieser Eigenschaft können Sie den Namen des aktuell verwendeten Histogramms setzen oder erfragen.

Das aktive Histogramm kann durchaus NOTHING sein, wenn bislang noch nichts im Histogrammbereich gemacht wurde. Ein Histogramm wird erst beispielsweise durch die Markierung einer Kurve aktiviert.

	Datentyp	Beschreibung
Eigenschaftswert	VcHistogram	Aktuell verwendetes Histogramm

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.Active
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.Active;
```

Count

Nur-Lese-Eigenschaft von VcHistogramCollection

Mit dieser Eigenschaft kann die Anzahl der Histogramme in der Histogrammauflistung abgefragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Histogramme

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim numberOfHistograms As Integer

histogramCltn = VcGantt1.HistogramCollection
numberOfHistograms = histogramCltn.Count
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
int numberOfHistograms = histogramCltn.Count;
```

Methoden

CreateHistogram

Methode von VcHistogramCollection

Mit dieser Methode können Sie ein Histogrammobjekt anlegen. Dieses Histogramm ist eine Kopie des letzten zuvor angelegten und enthält somit auch dessen Kurven. Es gehört automatisch der Histogrammauflistung an.

	Datentyp	Beschreibung
Parameter:		
⇨ histogramName	System.String	Name des anzulegenden Histogramms
Rückgabewert	VcHistogram	Angelegtes Histogramm

Code-Beispiel VB.NET

```
Dim histoggramCltn As VcHistogramCollection
Dim newHistogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
newHistogram = histogramCltn.CreateHistogram("resourceHistogram")
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram newHistogram = histogramCltn.CreateHistogram("resourceHistogram");
```

Delete

Methode von VcHistogramCollection

Mit dieser Methode können Sie ein Histogramm löschen.

	Datentyp	Beschreibung

FirstHistogram

Methode von VcHistogramCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Histogramm des HistogramCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextHistogram** über die nachfolgenden Histogramme zu iterieren. Existiert kein Histogramm im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcHistogram	Erstes Histogramm

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.FirstHistogram
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.FirstHistogram();
```

GetEnumerator

Methode von VcHistogramCollection

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 Histogramme iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

HistogramByIndex

Methode von VcHistogramCollection

Mit dieser Methode können Sie auf ein einzelnes Histogramm über seinen Index zugreifen. Existiert kein Histogramm an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des Histogramms
Rückgabewert	VcHistogram	Ermitteltes Histogrammobjekt

HistogramByName

Methode von VcHistogramCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Histogramm zugreifen. Existiert kein Histogramm unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ histogramName	System.String	Name des Histogramms

Rückgabewert	VcHistogram	Histogramm

Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.HistogramByName("Histogram_2")
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.HistogramByName("Histogram_2");
```

NextHistogram

Methode von VcHistogramCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Histogramme des HistogramCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstHistogram** den Initialwert erfasst haben. Sind alle Histogramme durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	**Beschreibung**
Rückgabewert	VcHistogram	Nachfolgendes Histogramm

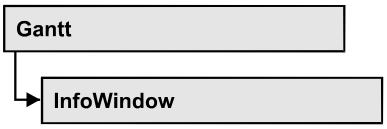
Code-Beispiel VB.NET

```
Dim histogramCltn As VcHistogramCollection
Dim histogram As VcHistogram

histogramCltn = VcGantt1.HistogramCollection
histogram = histogramCltn.FirstHistogram

While Not histogram Is Nothing
   ListBox1.Items.Add(histogram.Name)
   histogram = histogramCltn.NextHistogram
End While
```

Code-Beispiel C#

```
VcHistogramCollection histogramCltn = vcGantt1.HistogramCollection;
VcHistogram histogram = histogramCltn.FirstHistogram();

while (histogram != null)
   {
   listBox1.Items.Add(histogram.Name);
   histogram = histogramCltn.NextHistogram();
   }
```

7.40 VcInfoWindow



Ein Objekt vom Typ VcInfoWindow bezeichnet das Informationsfenster eines Knotens, das in einem Balkenplan beim Anlegen oder Verändern des Knotens erscheint.

Eigenschaften

- OutputFormatForCenterDate
- OutputFormatForDuration
- OutputFormatForEndDate
- OutputFormatForStartDate
- ReferenceDate
- UseReferenceDate
- Visible

Eigenschaften

OutputFormatForCenterDate

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie das Ausgabeformat des in der Mitte eines Layers befindlichen Datums (z.B. Symbol-Layer) im Informationsfenster eines Knotens einstellen oder erfragen. Für das Datumsformat 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 "Uhr" (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 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.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.String	Zeichenkette, die den Code für das zu verwendende Format enthält; bei einer leeren Zeichenkette wird das Output-Format des Gantt-Objektes verwendet (s. VcGantt.DateOutputFormat).

OutputFormatForDuration

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie das Ausgabeformat der Dauer im Informationsfenster eines Knotens einstellen oder erfragen. Für das Datumsformat stehen folgende Kürzel zur Verfügung:

hh: Stunde zweistellig im 24-Stunden-Format: 00-23

mm: Minute zweistellig: 00-59

ss: Sekunde zweistellig: 00-59

xC/XC: *Um dieses Format nutzen zu können, muss eine spezielle Einstellung in der Konfigurationsdatei vorgenommen werden. Bitte kontaktieren Sie bei Bedarf NETRONIC*. Sie können die Dauer als maximal zehngliedrige, einfache Aufwärtszählung gestalten, z.B. "07:16:00". Dieses Beispiel zeigt eine Dauer von 7 Stunden, 16 Minuten und 0 Sekunden. Die Notation dazu ist: **xC22:C11:C00**. Es sollen je 2 Stellen für die Glieder 2...0 vorgesehen werden. Die Trennzeichen sind variabel und könnten durch andere ersetzt werden. "x" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, aber kein "+"Symbol, falls er positiv ist. "X" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, und ebenso ein "+"Symbol, falls er positiv ist.

Hinweis Bitte setzen Sie 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.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Code für das zu verwendende Format enthält; bei einer leeren Zeichenkette wird das Output-Format des Gantt-Objektes verwendet (s. VcGantt.DateOutputFormat).

OutputFormatForEndDate

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie Mit dieser Eigenschaft können Sie das Ausgabeformat des Layer-Enddatums im Informationsfenster eines Knotens

einstellen oder erfragen. Für das Datumsformat 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 "Uhr" (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 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.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Code für das zu verwendende Format enthält; bei einer leeren Zeichenkette wird das Output-Format des Gantt-Objektes verwendet (s. VcGantt.DateOutputFormat).

OutputFormatForStartDate

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie das Ausgabeformat des Layer-Startdatums im Informationsfenster eines Knotens einstellen oder erfragen. Für das Datumsformat 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 "Uhr" (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 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.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Code für das zu verwendende Format enthält; bei einer leeren Zeichenkette wird das Output-Format des Gantt-Objektes verwendet (s. VcGantt.DateOutputFormat).

ReferenceDate

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie das Referenzdatum erfragen oder festlegen

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Referenzdatum

UseReferenceDate

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Informationsfenster ein Referenzdatum verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Das Informationsfenster verwendet (True) / verwendet kein (False) Referenzdatum **Standardwert:** False

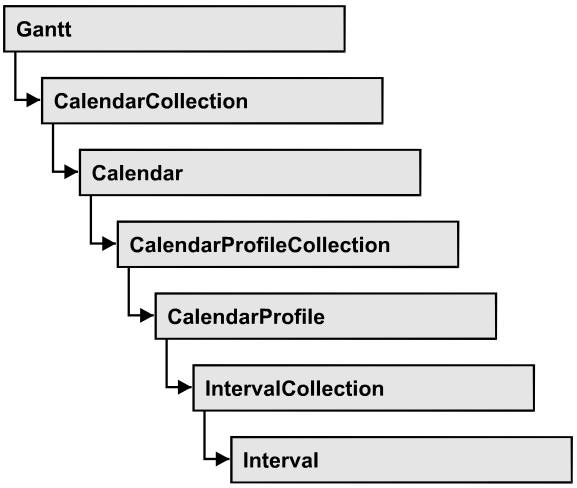
Visible

Eigenschaft von VcInfoWindow

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Informationsfenster sichtbar ist.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Informationsfenster sichtbar/unsichtbar **Standardwert:** True

7.41 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 Kalenderpofile 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:

vcCalendar-Interval	vcYearProfile-Interval	vcWeekProfile-Interval	vcDayProfile-Interval	vcShift-Interval
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 v**cSeconds** ohne Terminbezug darstellt. Beispiel: 4 Stunden

Eigenschaften

- BackgroundColor
- CalendarProfileName
- DayInEndMonth
- DayInStartMonth
- Duration
- EndDateTime
- EndMonth
- EndTime
- EndWeekday
- LineColor
- LineThickness
- LineType
- Name
- Pattern
- PatternColor
- Specification
- StartDateTime
- StartMonth
- StartTime
- StartWeekday
- Text
- TimeUnit
- Type
- UseGraphicalAttributes

Methoden

- PutInOrderAfter

Eigenschaften

BackgroundColor

Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie die Hintergrundfarbe für das Kalendergitter des Intervalls erfragen oder festlegen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil (ARGB-Wert) im Zahlenbereich von 0..255. 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.

Die Hintergrundfarbe kann auch im Dialog **Intervalle verwalten** festgelegt werden.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.DrawingColor	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255}) **Standardwert:** &hFFD8D8D8 (gray)

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

Duration

Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie die Dauer des Intervalls setzen oder erfragen. Dies gilt nur für Kalenderprofile vom Typ **vcShiftProfile**. Die Dauer kann auch im Dialog **Intervalle verwalten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Letzter Wochentag des Intervalls

EndDateTime

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	Endedatum 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 **Mögliche Werte:** .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 **April** **August** **Dezember** **Februar** **Januar** **Juli** **Juni** **März** **Mai** **November** **Oktober** **September**

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
Eigenschaftswert	VcWeekday **Mögliche Werte:** .vcFriday 5 .vcMonday 1 .vcSaturday 6 .vcSunday 7 .vcThursday 4 .vcTuesday 2 .vcWednesday 3	Letzter Wochentag des Intervalls Wochentag **Freitag** Wochentag **Montag** Wochentag **Samstag** Wochentag **Sonntag** Wochentag **Donnerstag** Wochentag **Dienstag** Wochentag **Mittwoch**

LineColor

Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie dieLinienfarbe eines Intervalls erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Intervalle verwalten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

LineThickness

Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie die Linienstärke für das Kalendergitter des Intervalls 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 Dialog <bIntervalle verwalten festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the dialog

LineType

Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie den Linientyp für das Kalendergitter des Intervalls erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Intervalle verwalten** festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcLineType **Mögliche Werte:** .vcDashed 4 .vcDashed 4 .vcDashedDotted 5 .vcDashedDotted 5 .vcDotted 3 .vcDotted 3 .vcLineType0 100	Linientyp ({0...255},{0...255},{0...255}) Linientyp **gestrichelt** Linientyp **gestrichelt** Linientyp **gestrichelt-gepunktet** Linientyp **gestrichelt-gepunktet** Linientyp **gepunktet** Linientyp **gepunktet** 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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

Name

Nur-Lese-Eigenschaft von VcInterval

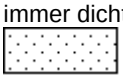
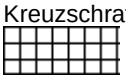
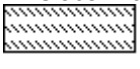
Mit dieser Eigenschaft können Sie den Namen eines Intervalls erfragen. Die Eigenschaft kann auch im Dialog **Intervalle verwalten** eingestellt werden.

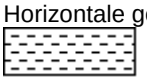
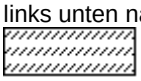
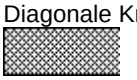
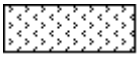
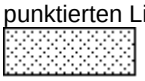
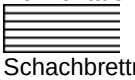
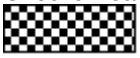
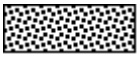
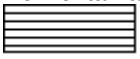
	Datentyp	**Beschreibung**
Eigenschaftswert	System.String	Name des Intervalls

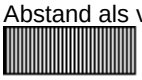
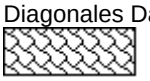
Pattern

Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie das Muster für das Kalendergitter des Intervalls festlegen oder erfragen. Das Muster kann auch im Dialog **Intervalle verwalten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp **Standardwert:** As defined in the dialog
	Mögliche Werte:	
	.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 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 

PatternColor

Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie die Musterfarbe für das Kalendergitter des Intervalls 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.

Die Musterfarbe kann auch im Dialog **Intervalle verwalten** festgelegt werden.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

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
Eigenschaftswert	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 **Mögliche Werte:** .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 **April** **August** **Dezember** **Februar** **Januar** **Juli** **Juni** **März** **Mai** **November** **Oktober** **September**

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
Eigenschaftswert	VcWeekday **Mögliche Werte:** .vcFriday 5 .vcMonday 1 .vcSaturday 6 .vcSunday 7 .vcThursday 4 .vcTuesday 2 .vcWednesday 3	Erster Wochentag des Intervalls Wochentag **Freitag** Wochentag **Montag** Wochentag **Samstag** Wochentag **Sonntag** Wochentag **Donnerstag** Wochentag **Dienstag** Wochentag **Mittwoch**

Text

Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie den Text festlegen oder erfragen, der im Zeitstreifen erscheinen soll. Dies gilt nur für Kalenderprofile vom Typ **vcShiftProfile**. Der Text kann auch im Dialog **Intervalle verwalten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Text für den Zeitstreifen

TimeUnit

Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie die Zeiteinheit für das Intervall setzen oder erfragen. Dies gilt nur für Kalenderprofile vom Typ **vcVariableProfile**. Die Zeiteinheit kann auch im Dialog **Intervalle verwalten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcTimeUnit	Zeiteinheit **Standardwert:** vcDay
	Mögliche Werte: .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	 Zeiteinheit **Tag** Zeiteinheit **Stunde** Zeiteinheit **Minute** Zeiteinheit **Sekunde**

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
Eigenschaftswert	VcIntervalType	Typ des Intervalls
	Mögliche Werte: .vcCalendarInterval 139 .vcDayProfileInterval 4 .vcIntervalProfileInterval 5 .vcWeekProfileInterval 3 .vcYearProfileInterval 2	

UseGraphicalAttributes

Nur-Lese-Eigenschaft von VcInterval

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob die für dieses Intervall gesetzten grafischen Attribute verwendet werden oder nicht. Sollen sie verwendet werden, muss die Eigenschaft **VcCalendarGrid.UseGraphicalAttributesOfIntervals** auf **True** gesetzt werden.

Die Option kann auch im Dialog **Intervalle verwalten** (zu erreichen über den Dialog **Kalenderprofile verwalten**) eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Grafische Attribute des Intervalls werden(True)/werden nicht dargestellt (False)

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.

	Datentyp	Beschreibung
Parameter:		
refName	System.String	Name des Intervalls, hinter das das aktuelle Intervall gesetzt werden soll
Rückgabewert	Void	

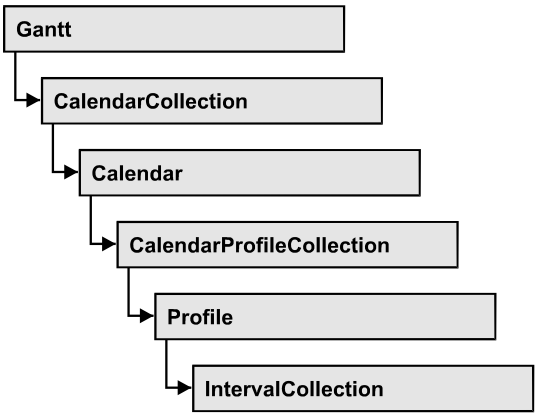
Code-Beispiel VB.NET

```
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();
```

7.42 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
Parameter:		
⇨ intervalName	System.String	Name des Intervalls
Rückgabewert	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
Parameter:		
⇨ Specification	System.String	Intervallspezifikation
Rückgabewert	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
Parameter:		
⇨ intervalName	System.String	Name des zu kopierenden Intervalls
⇨ newIntervalName	System.String	Name des neuen Intervalls
Rückgabewert	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
Rückgabewert	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
Parameter:		
⇨ Index	System.Int16	Index des Intervalls
Rückgabewert	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
Parameter:		
⇨ intervalName	System.String	Name des Intervallobjekts
Rückgabewert	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
Rückgabewert	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
Parameter:		
⇨ intervalName	System.String	Name des Intervalls
Rückgabewert	System.Boolean	Intervall gelöscht (True)/nicht gelöscht (False)

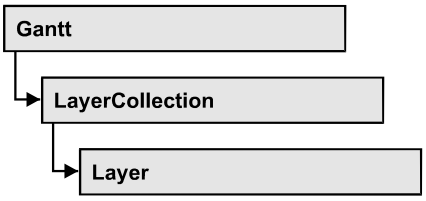
Update

Methode von VcIntervalCollection

Mit dieser Methode können Sie eine Intervall-Collection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

7.43 VcLayer



Ein Layer definiert die grafische Darstellung von einem Termin (Symbollayer) oder einem Terminpaar (Rechtecklayer oder Linienlayer) in einem Knoten. Ein Layer kann durch viele Attribute - wie Form, Farbe, Höhe, Offset sowie Art und Inhalt von Beschriftungsfeldern - konfiguriert werden.

Eigenschaften

- BackgroundColor
- BackgroundColorDataFieldIndex
- BackgroundColorMapName
- CompletionDataFieldIndex
- DurationDataFieldIndex
- EndDataFieldIndex
- EndSnapTarget
- FilterName
- Format
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- Height
- HeightDataFieldIndex
- HeightMapName
- HorizontalOffset
- LabelSizeDependence
- LegendText
- LineColor
- LineColorDataFieldIndex
- LineColorMapName
- MaximumEndDataFieldIndex
- MinimumStartDataFieldIndex
- Movable
- Name
- NonWorkIntervalBackgroundColor

- NonWorkIntervalBackgroundColorDataFieldIndex
- NonWorkIntervalBackgroundColorMapName
- NonWorkIntervalLineColor
- NonWorkIntervalLineColorDataFieldIndex
- NonWorkIntervalLineColorMapName
- NonWorkIntervalLineThickness
- NonWorkIntervalLineType
- NonWorkIntervalPattern
- NonWorkIntervalPatternColor
- NonWorkIntervalPatternColorDataFieldIndex
- NonWorkIntervalPatternColorMapName
- NonWorkIntervalPatternDataFieldIndex
- NonWorkIntervalPatternMapName
- NonWorkIntervalShape
- ObjectDrawEventsEnabled
- Pattern
- PatternColor
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternDataFieldIndex
- PatternMapName
- Shape
- Sizeable
- Specification
- StartDataFieldIndex
- StartSnapTarget
- ThreeDEffect
- UsedAsOverlapLayer
- VerticalOffset
- VerticalOffsetDataFieldIndex
- VerticalOffsetMapName
- Visible
- VisibleInLegend

Methoden

- CalculateCurrentWidth
- PutInOrderAfter

Eigenschaften

BackgroundColor

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Layers 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.

Wenn in der Eigenschaft **BackgroundColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Hintergrundfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

BackgroundColorDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **BackgroundColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")

VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapColor")

map.Type = VcMapType.vcColorMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = System.Drawing.Color.Green
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = System.Drawing.Color.Red
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.BackgroundColorMapName = "MapColor"
layer.BackgroundColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Red");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapColor");
map.Type = VcMapType.vcColorMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Green";
mapEntry.Color = System.Drawing.Color.Green;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Red";
mapEntry.Color = System.Drawing.Color.Red;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.BackgroundColorMapName = "MapColor";
layer.BackgroundColorDataFieldIndex = 5;
vcGantt1.LayerCollection.Update()
```

BackgroundColorMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **BackgroundColorDataFieldIndex** angegeben ist, wird die Hintergrundfarbe des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **BackgroundColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Red")

VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapColor")

map.Type = VcMapType.vcColorMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Green"
mapEntry.Color = System.Drawing.Color.Green
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Red"
mapEntry.Color = System.Drawing.Color.Red
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.BackgroundColorMapName = "MapColor"
layer.BackgroundColorDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Red");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapColor");
map.Type = VcMapType.vcColorMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Green";
mapEntry.Color = System.Drawing.Color.Green;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Red";
mapEntry.Color = System.Drawing.Color.Red;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.BackgroundColorMapName = "MapColor";
layer.BackgroundColorDataFieldIndex = 5;
vcGantt1.LayerCollection.Update()
```

CompletionDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie das Datenfeld setzen oder erfragen, das den prozentualen Fertigstellungsgrad des Layers enthält.

Der durch den Layer dargestellte Endtermin wird aus dem Start-Termin-Feld, dem Endtermin-Feld bzw. der Dauer und dem Fertigstellungsgrad berechnet. Die dem Vorgang zu Grunde liegenden Daten bleiben unverändert.

Für Symbol- und Bitmaplayer ist diese Eigenschaft nicht verfügbar.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Index des Datenfeldes, das den Fertigstellungsgrad enthält

DurationDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie das Datenfeld setzen oder erfragen, das die Dauer des Layers enthält.

Die Einheit der Dauer wird in Abhängigkeit von der Einstellung der Zeiteinheit auf der Eigenschaftenseite **Allgemeines** interpretiert.

Für Symbol- und Bitmaplayer ist diese Eigenschaft nicht verfügbar.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes, das die Dauer enthält

EndDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie das Datenfeld setzen oder erfragen, das den Endtermin des Layers enthält, z. B. Frühestes Ende, Spätestes Ende, Geplantes Ende.

Ein Rechteck- oder Linien-Layer wird durch die Angabe eines Start-Termin-Feldes und eines End-Termin-Feldes oder eines Start-Termin-Feldes und einer Dauer definiert. Sind sowohl das End-Termin-Feld als auch die Dauer spezifiziert, so hat die Dauer Vorrang vor dem End-Termin-Feld. Bei Interaktionen wird dann aber nicht nur das Dauer-Feld aktualisiert, sondern auch das End-Termin-Feld.

Für Symbol- und Bitmaplayer ist diese Eigenschaft nicht verfügbar.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes, das den Endtermin enthält

EndSnapTarget

Nur-Lese-Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Endtermin dieses Layers als Einrastziel definiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Endtermin des Layers wird/wird nicht als Einrastziel definiert.

FilterName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Filter setzen oder erfragen, der bestimmt, unter welcher Bedingung dieser Layer verwendet werden soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Filtername

Format

Nur-Lese-Eigenschaft von VcLayer

Mit dieser Methode können Sie das Format des Layers erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLayerFormat	Layerformat

GraphicsFileName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Grafikdatei setzen oder erfragen, deren Inhalt in dem Layer ausgegeben wird. Der Name muss eine gültige Grafikdatei bezeichnen. 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.

Für die Anzeige der Grafikdatei muss unabhängig vom hier gesetzten Format die Eigenschaft **LayerShape** auf **vcBitmapLayer** gesetzt werden.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.String	Name der Grafikdatei

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;03.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapGraphic")

map.Type = VcMapType.vcGraphicsFileMap
mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.Shape = VcLayerShape.vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.03.14;;8;Pic1.bmp");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapGraphic");
map.Type = VcMapType.vcGraphicsFileMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.GraphicsFileName = @"c:\Pic1.bmp";
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.Shape = VcLayerShape.vcBitmapLayer;
layer.GraphicsFileName = "Pic1.bmp";
layer.GraphicsFileNameMapName = "MapGraphic";
layer.GraphicsFileNameDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

GraphicsFileNameDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **GraphicsFileNameMap-Name** benötigt wird. Ist ein gültiger Datenfeldindex angegeben und keine Zuordnungstabelle, dann wird der Grafikdateiname direkt aus dem Inhalt des angegebenen Datenfelds entnommen.

Damit die Grafik angezeigt wird, muss die Eigenschaft **LayerShape** auf **vcBitmapLayer** gesetzt sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;03.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapGraphic")

map.Type = VcMapType.vcGraphicsFileMap
mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.Shape = VcLayerShape.vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.03.14;;8;Pic1.bmp");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapGraphic");
map.Type = VcMapType.vcGraphicsFileMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.GraphicsFileName = @"c:\Pic1.bmp";
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.Shape = VcLayerShape.vcBitmapLayer;
layer.GraphicsFileName = "Pic1.bmp";
layer.GraphicsFileNameMapName = "MapGraphic";
layer.GraphicsFileNameDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

GraphicsFileNameMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie 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.

Damit die Grafik angezeigt wird, muss die Eigenschaft **LayerShape** auf **vcBitmapLayer** gesetzt sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Grafik-Zuordnungstabelle

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;03.01.14;;8;Pic1.bmp")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapGraphic")

map.Type = VcMapType.vcGraphicsFileMap
mapEntry = map.CreateEntry
mapEntry.GraphicsFileName = "c:\Pic1.bmp"
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.Shape = VcLayerShape.vcBitmapLayer
layer.GraphicsFileName = "c:\Pic1.bmp"
layer.GraphicsFileNameMapName = "MapGraphic"
layer.GraphicsFileNameDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.03.14;;8;Pic1.bmp");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapGraphic");
map.Type = VcMapType.vcGraphicsFileMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.GraphicsFileName = @"c:\Pic1.bmp";
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.Shape = VcLayerShape.vcBitmapLayer;
layer.GraphicsFileName = "Pic1.bmp";
layer.GraphicsFileNameMapName = "MapGraphic";
layer.GraphicsFileNameDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

Height

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Höhe des Layers festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Höhe in 1/100 mm

HeightDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **HeightMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

Die Setzung dieser Eigenschaft wird erst wirksam, nachdem die Layer-Auflistung mit der Methode **Vc.LayerCollection.Update()** aktualisiert wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.FirstLayer
mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcMillimeterMap)
map = mapCltn.FirstMap
layer.HeightMapName = map.Name
ayer.HeightDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata", "LayerHeight")
 VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcLayer layer = vcGantt1.LayerCollection.FirstLayer();
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcMillimeterMap);
VcMap map = mapCltn.FirstMap();
layer.HeightMapName = map.Name;
layer.HeightDataFieldIndex = vcGantt1.DetectFieldIndex("Maindata",
"LayerHeight");
vcGantt1.LayerCollection.Update();
```

HeightMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Millimeter-Zuordnungstabelle (Typ vcMillimeterMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Millimeter-Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **HeightDataFieldIndex** angegeben ist, wird die Höhe des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Höhe aus der Eigenschaft **Height** ausgegeben.

Die Setzung dieser Eigenschaft wird erst wirksam, nachdem die Layer-Auflistung mit der Methode **vcGantt1.LayerCollection.Update()** aktualisiert wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Millimeter-Zuordnungstabelle

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.FirstLayer
mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcMillimeterMap)
map = mapCltn.FirstMap
layer.HeightMapName = map.Name
ayer.HeightDataFieldIndex = VcGantt1.DetectFieldIndex("Maindata", "LayerHeight")
 VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcLayer layer = vcGantt1.LayerCollection.FirstLayer();
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcMillimeterMap);
VcMap map = mapCltn.FirstMap();
layer.HeightMapName = map.Name;
layer.HeightDataFieldIndex = vcGantt1.DetectFieldIndex("Maindata",
"LayerHeight");
vcGantt1.LayerCollection.Update();
```

HorizontalOffset

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den horizontalen Offset des Layers festlegen oder erfragen. Dies gilt nur für Symbol- und Bitmap-Layern, bei allen anderen Layerformen hat die Einstellung des Offsets keine Auswirkung.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Horizontaler Offset in % -50 ... 50

LabelSizeDependence

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob und wie die Größe des Beschriftungsfeldes von der Größe des Layers abhängig sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	VcLabelSizeDependence	Abhängigkeit des Beschriftungsfeldes von der Layer-Größe
	Mögliche Werte: .vcFixedToBar 1 .vcTextHeightAndWidthIndependent 79 .vcTextHeightIndependent 39 .vcTextWidthIndependent 40	 durch Layergröße beschränkt unabhängig von Texthöhe und -breite unabhängig von Texthöhe unabhängig von Textbreite

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start End")
layer.LabelSizeDependence = VcLabelSizeDependence.vcFixedToBar
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.LayerByName("Start-End");
layer.LabelSizeDependence = VcLabelSizeDependence.vcFixedToBar;
```

LegendText

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie einem Layer einen Text zuweisen oder erfragen, der in der Legende für den jeweiligen Layer angezeigt wird. Steht hier "", dann wird der Layername (Eigenschaft **Name**) angezeigt.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.String	Legendentext des Layers **Standardwert:** " " (content of the property **Name**)

LineColor

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Farbe der (Rand)linie des Layers festlegen oder erfragen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

LineColorDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Farbzuordnungstabelle in der Eigenschaft

LineColorMapName benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

LineColorMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle 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 Farbzuordnungstabelle

MaximumEndDataFieldIndex

Eigenschaft von VcLayer

Wird diese Eigenschaft auf einen gültigen Feldindex gesetzt, dann gilt das im angesprochenen Datenfeld hinterlegte Datum mit Uhrzeit als obere Grenze für die Endzeit des Layers beim interaktiven Verschieben des Layers bzw. Knotens.

Diese Eigenschaft kann auch im Dialog **Layer-Bearbeiten** festgelegt werden.

	Datentyp	Beschreibung

MinimumStartDataFieldIndex

Nur-Lese-Eigenschaft von VcLayer

Wird diese Eigenschaft auf einen gültigen Feldindex gesetzt, dann gilt das im angesprochenen Datenfeld hinterlegte Datum mit Uhrzeit als untere Grenze

für die Startzeit des Layers beim interaktiven Verschieben des Layers bzw. Knotens.

Diese Eigenschaft kann auch im Dialog **Layer-Bearbeiten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex für früheste Startzeit **Standardwert:** -1

Movable

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Layer interaktiv verschiebbar sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Verschiebbar (True)/ nicht verschiebbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.LayerByName("layer1")
layer.Movable = False
```

Code-Beispiel C#

```
VcLayer VcLayer = vcGantt1.LayerCollection.LayerByName("layer1");
layer.Movable = false;
```

Name

Nur-Lese-Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen eines Layer-Objekts erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Layers

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
For Each layer In layerCltn
   ListBox1.Items.Add(layer.Name)
Next
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
foreach(VcLayer layer in layerCltn)
   listBox1.Items.Add(layer.Name);
```

NonWorkIntervalBackgroundColor

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Layers 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.

Wenn in der Eigenschaft **NonWorkIntervalBackgroundColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Hintergrundfarbe in Abhängigkeit von den Daten.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

NonWorkIntervalBackgroundColorDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **NonWorkIntervalBackgroundColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int16	Datenfeldindex

NonWorkIntervalBackgroundColorMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **BackgroundColorDataFieldIndex** angegeben ist, wird die Hintergrundfarbe des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **NonWorkIntervalBackgroundColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

NonWorkIntervalLineColor

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Farbe der (Rand)linie des Layers festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

NonWorkIntervalLineColorDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Farbzuordnungstabelle in der Eigenschaft **NonWorkIntervalLineColorMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

NonWorkIntervalLineColorMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle für die Linienfarbe setzen oder erfragen. Wird hier "" oder bei der Eigenschaft **NonWorkIntervalLineColorDataFieldIndex -1** angegeben, dann wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

NonWorkIntervalLineThickness

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Stärke der (Rand)linie des Layers festlegen oder erfragen.

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
Eigenschaftswert	System.Int16	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined in the dialog

NonWorkIntervalLineType

Eigenschaft von VcLayer

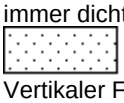
Mit dieser Eigenschaft können Sie den Typ der (Rand)linie des Layers festlegen oder erfragen.

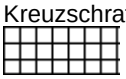
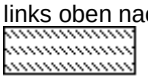
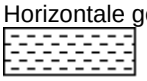
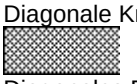
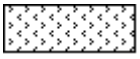
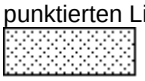
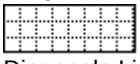
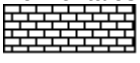
	Datentyp	Beschreibung

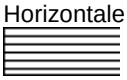
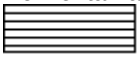
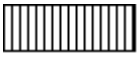
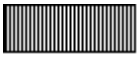
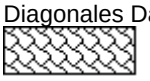
NonWorkIntervalPattern

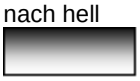
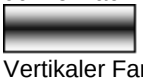
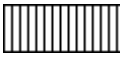
Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie das Muster des Layers festlegen oder erfragen. Wenn in der Eigenschaft **NonWorkIntervalPatternMapName** eine Zuordnungstabelle angegeben ist, steuert diese das Muster in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp **Standardwert:** As defined in the dialog
	Mögliche Werte: .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 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 

NonWorkIntervalPatternColor

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Musterfarbe des Layers 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.

Wenn in der Eigenschaft **NonWorkIntervalPatternColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Musterfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

NonWorkIntervalPatternColorDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **NonWorkIntervalPatternColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

NonWorkIntervalPatternColorMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **NonWorkIntervalPatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **NonWorkIntervalPatternColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

NonWorkIntervalPatternDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **NonWorkIntervalPattern-MapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

NonWorkIntervalPatternMapName

Eigenschaft von VcLayer

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 Farbzuordnungstabelle 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

NonWorkIntervalShape

Eigenschaft von VcLayer

Mit dieser Eigenschaft legen Sie die Form für die Darstellung arbeitsfreier Zeiten in Rechtecklayern fest. Die Auswahl kann auch im Dialog **Layer bearbeiten** getroffen werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcNonWorkIntervalShape	Form für arbeitsfreie Zeiten in Rechtecklayern
	Mögliche Werte:	
	.vcEmptyArea 2	arbeitsfreie Zeiten werden als leerer Bereich dargestellt
	.vcLine 1	arbeitsfreie Zeiten werden als Linie dargestellt
	.vcNo 0	arbeitsfreie Zeiten werden nicht dargestellt
	.vcRectangle 112	arbeitsfreie Zeiten werden als Rechteck dargestellt

ObjectDrawEventsEnabled

Eigenschaft von VcLayer

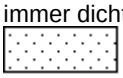
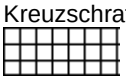
Wenn diese Eigenschaft auf **True** gesetzt wird, können bei Knoten, die mit diesem Layer gezeichnet werden oder bei Skalenbezeichnungen die Ereignisse **VcObjectDrawn** und **VcObjectDrawing** auftreten.

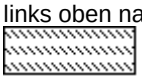
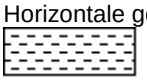
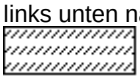
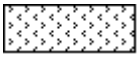
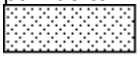
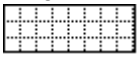
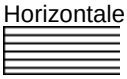
	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	ObjectDraw-Ereignisse ein- (True) oder ausgeschaltet (False) **Standardwert:** False

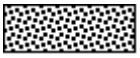
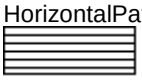
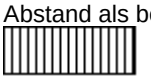
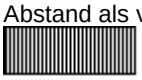
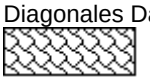
Pattern

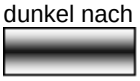
Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie das Muster des Layers festlegen oder erfragen. Wenn in der Eigenschaft **PatternMapName** eine Zuordnungstabelle angegeben ist, steuert diese das Muster in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp **Standardwert:** As defined in the dialog
	Mögliche Werte: .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 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 

PatternColor

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Musterfarbe des Layers 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.

Wenn in der Eigenschaft **PatternColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Musterfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

PatternColorDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

PatternColorMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle 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 Farbzuordnungstabelle

PatternDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Horizontal")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapPattern")

map.Type = VcMapType.vcPatternMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Diagonal"
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Horizontal"
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternMapName = "MapPattern"
layer.PatternDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

Code-Beispiel C#

```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Horizontal");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapPattern");
map.Type = VcMapType.vcPatternMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Diagonal";
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Horizontal";
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.PatternMapName = "MapPattern";
layer.PatternDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

PatternMapName

Eigenschaft von VcLayer

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 Farbzuordnungstabelle 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
Parameter:		
⇨ Rückgabewert	System.String	Name der Musterzuordnungstabelle
Eigenschaftswert	System.String	Name der Musterzuordnungstabelle

Code-Beispiel VB.NET

```
Dim dataTable As VcDataTable
Dim dataRecCltn As VcDataRecordCollection
Dim dataRec1 As VcDataRecord
Dim layer As VcLayer
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

dataTable = VcGantt1.DataTableCollection.DataTableByName("Maindata")
dataRecCltn = dataTable.DataRecordCollection
dataRec1 = dataRecCltn.Add("1;Node 1;01.01.14;;8;Horizontal")
VcGantt1.EndLoading()

mapCltn = VcGantt1.MapCollection
map = mapCltn.Add("MapPattern")

map.Type = VcMapType.vcPatternMap
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Diagonal"
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern
mapEntry = map.CreateEntry
mapEntry.DataFieldValue = "Horizontal"
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern
mapCltn.Update()

layer = VcGantt1.LayerCollection.LayerByIndex(0)
layer.PatternMapName = "MapPattern"
layer.PatternDataFieldIndex = 5
VcGantt1.LayerCollection.Update()
```

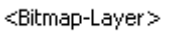
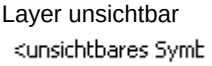
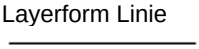
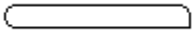
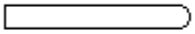
Code-Beispiel C#

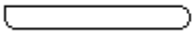
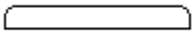
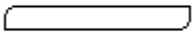
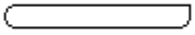
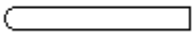
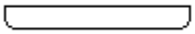
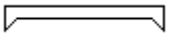
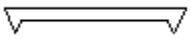
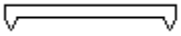
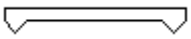
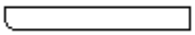
```
VcDataTable dataTable =
vcGantt1.DataTableCollection.DataTableByName("Maindata");
VcDataRecordCollection dataRecCltn = dataTable.DataRecordCollection;

VcDataRecord dataRec1 = dataRecCltn.Add("1;Node 1;02.01.14;;8;Horizontal");
vcGantt1.EndLoading();

VcMapCollection mapCltn = vcGantt1.MapCollection;
VcMap map = mapCltn.Add("MapPattern");
map.Type = VcMapType.vcPatternMap;

VcMapEntry mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Diagonal";
mapEntry.Pattern = VcFillPattern.vcFDiagonalPattern;
mapEntry = map.CreateEntry();
mapEntry.DataFieldValue = "Horizontal";
mapEntry.Pattern = VcFillPattern.vcHorizontalPattern;
mapCltn.Update();

VcLayer layer = vcGantt1.LayerCollection.LayerByIndex(0);
layer.PatternMapName = "MapPattern";
layer.PatternDataFieldIndex = 5;
vcGantt1.LayerCollection.Update();
```

Shape

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Form des Layers festlegen oder erfragen. Bei den unten abgebildeten Symbolen sind schwarze Flächen durch zugewiesene Farben ersetzbar (s. **BackgroundColorAsARGB**, **Pattern** und **PatternColor**).

	Datentyp	**Beschreibung**
Eigenschaftswert	VcLayerShape	Layerform
	Mögliche Werte:	
	.vcAllRoundedRectangleLayer 61441	Alle Ecken gerundet 
	.vcBitmapLayer 103007	Layerform Bitmap 
	.vcInvisibleSymbolLayer 101000	Layer unsichtbar 
	.vcLineLayer 2	Layerform Linie 
	.vcNAndSERoundedRectangleLayer 45057	Linke und rechte obere Ecke sowie rechte untere Ecke gerundet 
	.vcNAndSWRoundedRectangleLayer 28673	Linke und rechte obere Ecke sowie linke untere Ecke gerundet 
	.vcNEAndSERoundedRectangleLayer 40961	Obere und untere rechte Ecke gerundet 

.vcNEAndSRoundedRectangleLayer 57345	Linke und rechte untere Ecke sowie obere rechte Ecke gerundet 
.vcNEAndSWRoundedRectangleLayer 24577	Linke untere und rechte obere Ecke gerundet 
.vcNERoundedRectangleLayer 8193	Obere rechte Ecke gerundet 
.vcNRoundedRectangleLayer 12289	Obere rechte und obere linke Ecke gerundet 
.vcNWAndSERoundedRectangleLayer 36865	Obere linke und untere rechte Ecke gerundet 
.vcNWAndSRoundedRectangleLayer 52349	Untere linke und rechte Ecke sowie obere linke Ecke gerundet 
.vcNWAndSWRoundedRectangleLayer 20481	Untere und obere linke Ecke gerundet 
.vcNWRoundedRectangleLayer 4097	Obere linke Ecke gerundet 
.vcRectangleLayer 1	
.vcSERoundedRectangleLayer 32769	Untere rechte Ecke gerundet 
.vcSRoundedRectangleLayer 49153	Untere rechte und untere linke Ecke gerundet 
.vcSummaryBar1 1858	Summenbalken 
.vcSummaryBar2 1859	Summenbalken 
.vcSummaryBar3 1860	Summenbalken 
.vcSummaryBar4 1861	Summenbalken 
.vcSWRoundedRectangleLayer 16385	Untere linke Ecke gerundet 
.vcSymbolLayer1 101001	Pfeil nach unten 
.vcSymbolLayer10 101010	Quadrat 
.vcSymbolLayer11 101032	Kreis 
.vcSymbolLayer12 101033	Pfeil nach unten in Kreis 
.vcSymbolLayer13 101034	Dreieck im Kreis, Spitze nach unten 
.vcSymbolLayer14 101035	Spitze rechte Klammer im Kreis 

.vcSymbolLayer15 101036	Schmales Dreieck im Kreis, Spitze oben 
.vcSymbolLayer16 101037	Dreieck im Kreis, Spitze rechts 
.vcSymbolLayer17 101038	Dreieck im Kreis, Spitze links 
.vcSymbolLayer18 101039	Auf die Spitze gestelltes Quadrat im Kreis 
.vcSymbolLayer19 101040	Zwei schmale Dreiecke im Kreis, horizontal, Spitzen zur Mitte 
.vcSymbolLayer2 101002	Dreieck, Spitze nach unten 
.vcSymbolLayer20 101041	Schmales Dreieck im Kreis, Spitze nach unten 
.vcSymbolLayer21 101042	Quadrat in Kreis 
.vcSymbolLayer22 103001	Kreis 
.vcSymbolLayer23 102031	Pfeil nach oben 
.vcSymbolLayer24 102034	Dreieck, Spitze nach oben 
.vcSymbolLayer25 102016	Linke spitze Klammer 
.vcSymbolLayer26 102051	Pfeil nach oben im Kreis 
.vcSymbolLayer27 102054	Dreieck im Kreis, Spitze nach oben 
.vcSymbolLayer3 101003	Rechte spitze Klammer 
.vcSymbolLayer4 101004	Schmales Dreieck, Spitze oben 
.vcSymbolLayer5 101005	Dreieck, Spitze nach rechts 
.vcSymbolLayer6 101006	Dreieck, Spitze nach links 
.vcSymbolLayer7 101007	Auf die Spitze gestelltes Quadrat 
.vcSymbolLayer8 101008	Zwei schmale Dreiecke, horizontal, Spitzen zur Mitte 

.vcSymbolLayer9 101009	Schmales Dreieck, Spitze nach unten 
.vcTriangleBottomLeftLayer 1566	Dreiecklayer; Spitze links 
.vcTriangleBottomRightLayer 1564	Dreiecklayer; Spitze rechts 

Sizeable

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Layerbreite interaktiv veränderbar sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	VcLayerSizeability **Mögliche Werte:** .vcSizeableLeft 1 .vcSizeableLeftRight 3 .vcSizeableNone 0 .vcSizeableRight 2	Art der Veränderbarkeit der Layerbreite **Standardwert:** True

Code-Beispiel VB.NET

```
Dim layer As VcLayer

layer = vcGantt1.LayerCollection.LayerByName("layer1")
layer.Sizeable = VcLayerSizeability.vcSizeableLeftRight
```

Code-Beispiel C#

```
VcLayer VcLayer = VcGantt1.LayerCollection.LayerByName("layer1");
layer.Sizeable = VcLayerSizeability.vcSizeableLeftRight;
```

Specification

Nur-Lese-Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie die Spezifikation dieses Layers 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 Layers mit der Methode **VcLayeCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Layerspezifikation

StartDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie das Datenfeld setzen oder erfragen, das den Anfangstermin des Layers enthält, z. B. Frühester Anfang, Spätester Anfang, Geplanter Start.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes, das den Anfangstermin enthält

StartSnapTarget

Nur-Lese-Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Starttermin dieses Layers als Einrastziel definiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Starttermin des Layers wird/wird nicht als Einrastziel definiert.

ThreeDEffect

Eigenschaft von VcLayer

Mit dieser Eigenschaft wird festgelegt oder erfragt, ob der Layer einen 3D-Effekt erhält oder nicht.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	3D-Effekt eingeschaltet (True)/ausgeschaltet (False) **Standardwert:** False

UsedAsOverlapLayer

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Layer als Overlap-Layer verwendet werden soll. Overlap-Layer treten in Erscheinung, wenn eine Überlappung (und damit die Verdeckung) von Knoten anzuzeigen ist und werden mit dem Ausmaß der Überlappung größer oder kleiner. Vgl. dazu auch die Eigenschaften **VcGantt.OverlapLayerEnabled** und **VcGantt.OverlapLayerName**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**True**: Layer wird als überlappungslayer verwendet; **False**: Layer wird nicht als Überlappungslayer verwendet **Standardwert:** False

Code-Beispiel VB.NET

```
Dim layer As VcLayer

layer = VcGantt1.LayerCollection.LayerByName("layer1")
layer.UsedAsOverlapLayer = False
```

Code-Beispiel C#

```
VcLayer VcLayer = vcGantt1.LayerCollection.LayerByName("layer1");
layer.UsedAsOverlapLayer = false;
```

VerticalOffset

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den vertikalen Offset des Layers festlegen oder erfragen. Wenn in der Eigenschaft **VerticalOffsetMapName** eine Zuordnungstabelle angegeben ist, steuert diese den vertikalen Offset in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Vertikaler Offset in 1/100 mm

VerticalOffsetDataFieldIndex

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **VerticalOffsetMapName**

benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

VerticalOffsetMapName

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie den Namen einer Millimeter-Zuordnungstabelle (Typ vcMillimeterMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Millimeter-Zuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **VerticalOffsetDataFieldIndex** angegeben ist, wird der vertikale Offset des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird der vertikale Offset aus der Eigenschaft **VerticalOffset** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Millimeter-Zuordnungstabelle

Visible

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Layer sichtbar sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Layer sichtbar/nicht sichtbar **Standardwert:** True

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start End")
layer.Visible = False
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.LayerByName("Start-End");
layer.Visible = false;
```

VisibleInLegend

Eigenschaft von VcLayer

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob das Layer-Objekt in der Legende sichtbar ist. Diese Eigenschaft kann auch im Dialog **Balkenaussehen verwalten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Layer in Legende sichtbar (True)/nicht sichtbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start End")
layer.VisibleInLegend = False
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.LayerByName("Start-End");
layer.VisibleInLegend = false;
```

Methoden

CalculateCurrentWidth

Methode von VcLayer

Mit dieser Methode können Sie die aktuelle Breite des Layers in 1/100 mm ermitteln, der zur angegebenen Layerdefinition am angegebenen Knoten gehört. Wenn kein Layer an diesem Knoten zur angegebenen Layerdefinition sichtbar ist z.B. aufgrund von Filterbedingungen, dann wird **-1** zurückgegeben.

	Datentyp	Beschreibung
Parameter:		
⇨ node	VcNode	Knoten, in dessen Layer-Definition der Layer gesucht wird.
Rückgabewert	System.Int32	Breite des Layers in 1/100 mm

PutInOrderAfter

Methode von VcLayer

Mit dieser Methode können Sie den Layer in der Auflistung aller Layer hinter den durch den Namen angegebenen setzen. Wenn als Name "" übergeben wird, wird der Layer an die erste Stelle gesetzt. Die Reihenfolge der Layer in der Auflistung entscheidet darüber, in welcher Reihenfolge sie dargestellt werden.

	Datentyp	Beschreibung
Parameter:		
⇦ refName	System.String	Name des Layers, hinter den der aktuelle Layer eingeordnet werden soll.
Rückgabewert	Void	

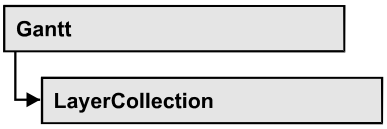
Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer1 As VcLayer
Dim layer2 As VcLayer

layerCltn = VcGantt1.LayerCollection()
layer1 = layerCltn.Add("layer1")
layer2 = layerCltn.Add("layer2")
layer1.PutInOrderAfter("layer2")
layerCltn.Update()
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer1 = layerCltn.Add("layer1");
VcLayer layer2 = layerCltn.Add("layer2");
layer1.PutInOrderAfter("layer2");
layerCltn.Update();
```

7.44 VcLayerCollection



In einem Objekt vom Typ VcLayerCollection sind automatisch alle verfügbarenLayer zusammengefasst. Über **For Each layer In Layer-Collection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Layer zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **LayerByName** und **LayerByIndex**. Die Anzahl der im Collection-Objekt vorhandenen Layer kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Layern.

Eigenschaften

- Count

Methoden

- Add
- AddBySpecification
- Copy
- FirstLayer
- GetEnumerator
- LayerByIndex
- LayerByName
- NextLayer
- Remove
- Update

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcLayerCollection

Mit dieser Eigenschaft kann die Anzahl der Layer in der Layer-Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Layer

Code-Beispiel VB.NET

```
Dim numberOfLayers As Integer

numberOfLayers = VcGantt1.LayerCollection.Count
```

Code-Beispiel C#

```
int numberOfLayers = vcGantt1.LayerCollection.Count;
```

Methoden

Add

Methode von VcLayerCollection

Mit dieser Methode können Sie einen neuen Layer in der LayerCollection anlegen. Wenn der Name noch nicht verwendet wird, wird das neue Layerobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ layerName	System.String	Name des Layers
Rückgabewert	VcLayer	Neues Layerobjekt

Code-Beispiel VB.NET

```
newlayer = VcGantt1.LayerCollection.Add("test1")
```

Code-Beispiel C#

```
VcLayer newLayer = vcGantt1.LayerCollection.Add("test1");
```

AddBySpecification

Methode von VcLayerCollection

Mit dieser Methode können Sie einen Layer über eine Layer-Spezifikation erzeugen. Dies dient der Persistenz von Layerobjekten. Die Spezifikation eines Layers kann erfragt (siehe VcLayer-Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann der gleiche Layer mit der

wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ specification	System.String	Layerspezifikation
Rückgabewert	VcLayer	Neues Layerobjekt

Copy

Methode von VcLayerCollection

Mit dieser Methode können Sie einen Layer kopieren. Wenn der Layer mit dem angegebenen Namen existiert und der Name des neuen Layers noch nicht verwendet wird, wird das neue Layerobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ layerName	System.String	Name des zu kopierenden Layers
⇨ newLayerName	System.String	Name des neuen Layers
Rückgabewert	VcLayer	Layerobjekt

FirstLayer

Methode von VcLayerCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. den ersten Layer des LayerCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextLayer** über die nachfolgenden Layer zu iterieren. Existiert kein Layer im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLayer	Erster Layer

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.FirstLayer
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.FirstLayer();
```

GetEnumerator

Methode von VcLayerCollection

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 Layer-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

LayerByIndex

Methode von VcLayerCollection

Mit dieser Methode können Sie auf einen einzelnen Layer über ihren Index zugreifen. Existiert kein Layer an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des Layers
Rückgabewert	VcLayer	Ermitteltes Layerobjekt

LayerByName

Methode von VcLayerCollection

Mit dieser Methode können Sie unter Verwendung des Layernamens auf einen bestimmten Layer zugreifen. Existiert kein Layer unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ layerName	System.String	Name des Layers
Rückgabewert	VcLayer	Layer

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.LayerByName("Start-End")
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.LayerByName("Start-End");
```

NextLayer

Methode von VcLayerCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Layer der LayerCollection zugreifen, nachdem Sie mit der Methode **FirstLayer** den Initialwert erfasst haben. Sind alle Layer durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLayer	Folgelayer

Code-Beispiel VB.NET

```
Dim layerCltn As VcLayerCollection
Dim layer As VcLayer

layerCltn = VcGantt1.LayerCollection
layer = layerCltn.FirstLayer
While Not layer Is Nothing
   ListBox1.Items.Add(layer.Name)
   layer = layerCltn.NextLayer
End While
```

Code-Beispiel C#

```
VcLayerCollection layerCltn = vcGantt1.LayerCollection;
VcLayer layer = layerCltn.FirstLayer();

while (layer != null)
   {
   listBox1.Items.Add(layer.Name);
   layer = layerCltn.NextLayer();
   }
```

Remove

Methode von VcLayerCollection

Mit dieser Methode können Sie einen Layer löschen. Wenn der Layer noch in irgendeinem anderen Objekt benutzt wird, kann er nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
Parameter:		
⇨ layerName	System.String	Name des Layers
Rückgabewert	System.Boolean	Layer gelöscht (True)/nicht gelöscht (False)

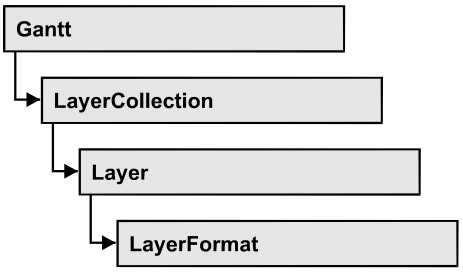
Update

Methode von VcLayerCollection

Mit dieser Methode können Sie eine LayerCollection aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

7.45 VcLayerFormat



Ein Layerformat definiert die Beschriftung eines Layers. Über **For Each formatfield In LayerFormat** können Sie in einer Schleife auf alle Layer zugreifen.

Eigenschaften

- FormatField
- FormatFieldCount

Methoden

- CopyFormatField
- GetEnumerator
- RemoveFormatField

Eigenschaften

FormatField

Nur-Lese-Eigenschaft von VcLayerFormat

Mit dieser Eigenschaft können Sie ein VcLayerFormatField-Objekt per Index ansprechen. Der Index muss im Bereich von 0 bis FormatFieldCount-1 liegen.

Die Eigenschaft FormatField ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_FormatField (index, pvn) und get_FormatField (index) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
index	System.Int16	Index des Layerformatfeldes 0 ... FormatFieldCount-1
Eigenschaftswert	VcLayerFormatField	Layerformatfeld

FormatFieldCount

Nur-Lese-Eigenschaft von VcLayerFormat

Diese Eigenschaft gibt die Anzahl der Felder eines Layerformats an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Felder im Layerformat

Methoden

CopyFormatField

Methode von VcLayerFormat

Mit dieser Methode können Sie ein Layerformatfeld kopieren. Das neue VcLayerFormatField-Objekt wird zurückgegeben. Es erhält den nächsten, noch nicht vergebenen Index.

	Datentyp	Beschreibung
Parameter:		
⇨ position	VcFormatFieldPosition	Position des neuen Layerformatfeldes
	Mögliche Werte:	
	.vcAbove 1	oberhalb
	.vcBelow 3	unterhalb
	.vcLeftOf 0	links von
	.vcOutsideAbove 9	außerhalb, oberhalb
	.vcOutsideBelow 11	außerhalb, unterhalb
	.vcOutsideLeftOf 8	außerhalb, links von
	.vcOutsideRightOf 12	außerhalb, rechts von
	.vcRightOf 4	rechts von
⇨ refIndex	System.Int16	Index des Referenz-Layerformatfeldes
Rückgabewert	VcLayerFormatField	Layerformatfeld-Objekt

GetEnumerator

Methode von VcLayerFormat

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 Layer-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

RemoveFormatField

Methode von VcLayerFormat

Mit dieser Methode können Sie ein Layerformatfeld über den angegebenen Index löschen. Anschließend wird ggf. der Index aller Layerformatfelder neu festgesetzt, so dass sie wieder fortlaufend numeriert sind.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des zu löschenden Layerformatfeldes

7.46 VcLayerFormatField

Ein Objekt vom Typ **VcLayerFormatField** stellt ein Layerformatfeld, also ein Feld eines VcLayerFormat-Objekts, dar. Ein Layerformatfeld besitzt im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, unter dem es im Layerformat untergebracht ist.

Eigenschaften

- Alignment
- BottomMargin
- ConstantText
- FormatName
- Index
- LeftMargin
- MinimumWidth
- Priority
- RightMargin
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TextLineCount
- TextLineCountDataFieldIndex
- TextLineCountMapName
- TopMargin
- TruncatedTextSuppressed

Methoden

- CalculateLineCount

Eigenschaften

Alignment

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Layerformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFormatFieldAlignment **Mögliche Werte:** .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Ausrichtung des Feldinhalts unten unten links unten rechts unten mittig links rechts oben oben links oben rechts

BottomMargin

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des unteren Randes des Layerformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Layerformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung

ConstantText

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie einen konstanten Text in dem Layerformatfeld ausgeben, wenn die Eigenschaft **TextDataFieldIndex** auf **-1** gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Konstanter Text

FormatName

Nur-Lese-Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Namen des Layerformats erfragen, zu dem dieses Layerformatfeld gehört.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Layerformats

Index

Nur-Lese-Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Index des Layerformatfelds im zugehörigen Layerformat erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Layerformatfeldes

LeftMargin

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des linken Randes des Layerformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Layerformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung

MinimumWidth

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die minimale Breite des Layerformatfeldes in mm erfragen oder festlegen, wenn die Größe des Beschriftungsfeldes dies zulässt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Minimale Breite des Layerformatfeldes in mm 0 ... 99

Priority

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Priorität des Layerformatfeldes erfragen oder festlegen. Über die Priorität können Sie die Aufteilung des vorhandenen Platzes beeinflussen. Felder mit höherer Priorität erhalten bevorzugt den Platz, den sie benötigen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Priorität des Layerformatfeldes {-9...9}

RightMargin

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des rechten Randes des Layerformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Layerformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung

TextDataFieldIndex

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Index des Datenfelds, dessen Inhalt in dem Layerformatfeld dargestellt werden soll, erfragen oder setzen. Falls der Index den Wert **-1** besitzt, wird stattdessen der Inhalt der Eigenschaft **ConstantText** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

TextFont

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Schriftart des Layerformatfeldes festlegen oder erfragen. Wenn in der Eigenschaft **TextFontMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Schriftart in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftart des Layerformatfelds

TextFontColor

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Schriftfarbe des Layerformatfeldes festlegen oder erfragen. Wenn über die Eigenschaft **TextFontColorMap-Name** eine Zuordnungstabelle angegeben ist, steuert diese die Schriftfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Schriftfarbe des Layerformatfelds **Standardwert:** -1

TextFontColorDataFieldIndex

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Schriftfarbe-Zuordnungstabelle in Verbindung mit der Eigenschaft **TextFontColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

TextFontColorMapName

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) für die Schriftfarbe setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in Verbindung mit der Eigenschaft **TextFontColorDataFieldIndex** angegeben ist, wird die Schriftfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **TextFontColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Schriftfarben-Zuordnungstabelle

TextFontDataFieldIndex

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **TextFontMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

TextFontMapName

Eigenschaft von VcLayerFormatField

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 Verbindung mit 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

TextLineCount

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Zeilenanzahl erfragen oder setzen, wenn die Größe des Beschriftungsfeldes dies zulässt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Zeilenzahl

TextLineCountDataFieldIndex

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **TextLineCountMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

TextLineCountMapName

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie den Namen einer numerischen Zuordnungstabelle für die Anzahl von Textzeilen setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn der Name einer Zuordnungstabelle und zusätzlich ein Datenfeldindex in Verbindung mit der Eigenschaft **TextLineCountDataFieldIndex** angegeben ist, wird die Anzahl der Textzeilen aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Zeilenzahl aus der Eigenschaft **TextLineCount** verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der numerischen Zuordnungstabelle

TopMargin

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des oberen Randes des Layerformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Layerformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung

TruncatedTextSuppressed

Eigenschaft von VcLayerFormatField

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob ein Text, der nicht ganz in das Layerformatfeld passt, unterdrückt oder abgeschnitten werden soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft (True)/ nicht in Kraft (False)

Methoden

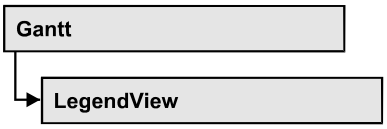
CalculateLineCount

Methode von VcLayerFormatField

Gilt nur für außenliegende Felder eines Layers: Mit dieser Methode können Sie die Anzahl der Zeilen im Layerformatfeld des angegebenen Knotens ermitteln, die der Text bei der aktuellen Größe des Feldes und der Schriftgröße einnehmen wird. Bei innenliegenden Feldern wird -1 zurückgegeben. Das Ergebnis dieser Methode kann zur Weiterverarbeitung in ein Datenfeld des Knotens geschrieben werden, um die Anzahl angezeigter Zeilen im gleichen Formatfeld zu steuern (S. Dialog **Layerformat bearbeiten -> Zeilenanzahl**)

	Datentyp	**Beschreibung**
Parameter:		
⇨ node	VcNode	Knoten
Rückgabewert	System.Int32	Ermittelte Zeilenzahl

7.47 VcLegendView



Ein Objekt vom Typ **VcLegendView** bezeichnet das Legendenansicht-Fenster.

Eigenschaften

- Border
- BorderColor
- 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
Eigenschaftswert	System.Boolean	Rahmen um die Legendenansicht (True)/kein Rahmen um die Legendenansicht (False) **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Mode = VcLegendViewMode.vcNotFixed
VcGantt1.LegendView.Border = True
```

Code-Beispiel C#

```
vcGantt1.LegendView.Mode = VcLegendViewMode.vcNotFixed;
vcGantt1.LegendView.Border = true;
```

BorderColor

Eigenschaft von VcLegendView

Diese Eigenschaft setzt/liefert die des ggf. sichtbaren Rahmens.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** 0,0,0

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
Eigenschaftswert	System.Int32	Höhe der Legendenansicht **Standardwert:** 100

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Height = 100
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche Höhe der Legendenansicht {0, ...}

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Height = 300
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Linke Position der Legendenansicht **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Left = 200
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche linke Position der Legendenansicht {0, ...}

Code-Beispiel VB.NET

```
VcGantt1.LegendView.LeftActualValue = 150
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	VcLegendViewScrollBarMode	Scrollbarmodus **Standardwert:** NoScrollBar **Mögliche Werte:** .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

```
VcGantt1.LegendView.ScrollBarMode = vcAutomaticScrollbar
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Obere Position der Legendenansicht **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Top = 20
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche obere Position der Legendenansicht {0, ...}

Code-Beispiel VB.NET

```
VcGantt1.LegendView.TopActualValue = 40
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Boolean	Legendenansicht sichtbar (True)/unsichtbar (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Visible = True
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Horizontale Ausdehnung der Legendenansicht **Standardwert:** 100

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Width = 200
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche horizontale Ausdehnung der Legendenansicht {0, ...}

Code-Beispiel VB.NET

```
VcGantt1.LegendView.WidthActualValue = 600
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	VcLegendViewWindowMode	Modus der Gesamtansicht **Standardwert:** vcPopupWindow
	Mögliche Werte:	
	.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.

	.vcPopupWindow 6	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 **Schließen**-Schaltfläche in der Titelleiste ausgeschaltet werden.

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Mode = VcLegendViewMode.vcNotFixed
```

Code-Beispiel C#

```
vcGantt1.LegendView.Mode = VcLegendViewMode.vcNotFixed;
```

Methoden

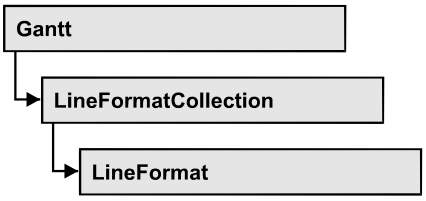
Update

Methode von VcLegendView

Mit dieser Methode wird die Legende aktualisiert.

	Datentyp	Beschreibung

7.48 VcLineFormat



Ein Objekt vom Typ VcLineFormat legt Inhalt und Erscheinungsbild von Linien fest, z.B. in einem Terminliniengitter.

Eigenschaften

- FormatField
- FormatFieldCount
- Name
- Specification

Methoden

- CopyFormatField
- RemoveFormatField

Eigenschaften

FormatField

Nur-Lese-Eigenschaft von VcLineFormat

Mit dieser Eigenschaft können Sie ein VcLineFormatField-Objekt per Index erfragen. Der Index muss im Bereich von 0 bis FormatFieldCount-1 liegen.

	Datentyp	Beschreibung
Parameter: index	System.Int16	Index des Linienformatfeldes
Eigenschaftswert	VcNodeFormatField	Linienformatfeld

FormatFieldCount

Nur-Lese-Eigenschaft von VcLineFormat

Mit dieser Eigenschaft können Sie die Anzahl der Felder eines Linienformats ermitteln.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Felder im Linienformat

Code-Beispiel VB.NET

```
Dim lineFormat As VcLineFormat

lineFormat = VcGantt1.LineFormatCollection.FirstFormat
MsgBox(lineFormat.FormatFieldCount)
```

Code-Beispiel C#

```
VcLineFormat nodeFormat = vcGantt1.LineFormatCollection.FirstFormat();
MessageBox.Show(lineFormat.FormatFieldCount.ToString());
```

Name

Eigenschaft von VcLineFormat

Mit dieser Eigenschaft können Sie den Namen des Linienformats erfragen oder setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Linienformatname

Code-Beispiel VB.NET

```
Dim lineFormat As VcLineFormat

lineFormat = VcGantt1.LineFormatCollection.FirstFormat
MsgBox(lineFormat.Name)
```

Code-Beispiel C#

```
VcLineFormat lineFormat = vcGantt1.LineFormatCollection.FirstFormat();
MessageBox.Show(lineFormat.Name);
```

Specification

Nur-Lese-Eigenschaft von VcLineFormat

Mit dieser Eigenschaft können Sie die Spezifikation dieses Linienformats 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 Linienformats mit der Methode **VcLineFormatCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Linienformats

Methoden

CopyFormatField

Methode von VcLineFormat

Mit dieser Methode können Sie ein Linienformatfeld kopieren. Das neue VcNodeFormatField-Objekt wird zurückgegeben. Es erhält den nächsten, noch nicht vergebenen Index.

	Datentyp	Beschreibung
Parameter:		
⇨ position	VcFormatFieldPosition **Mögliche Werte:** .vcAbove 1 .vcBelow 3 .vcLeftOf 0 .vcOutsideAbove 9 .vcOutsideBelow 11 .vcOutsideLeftOf 8 .vcOutsideRightOf 12 .vcRightOf 4	Position des neuen Linienformatfeldes oberhalb unterhalb links von außerhalb, oberhalb außerhalb, unterhalb außerhalb, links von außerhalb, rechts von rechts von
⇨ refIndex	System.Int16	Index des Referenz-Linienformatfeldes
Rückgabewert	VcNodeFormatField	Linienformatfeld-Objekt

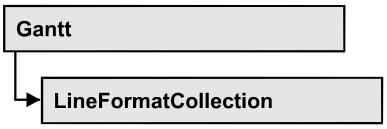
RemoveFormatField

Methode von VcLineFormat

Mit dieser Methode können Sie ein Linienformatfeld über den angegebenen Index löschen. Anschließend wird ggf. der Index aller Linienformatfelder neu festgesetzt, so dass sie wieder fortlaufend numeriert sind.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des zu löschenden Linienformatfeldes

7.49 VcLineFormatCollection



In einem Objekt vom Typ VcLineFormatCollection sind alle verfügbaren Linienformate zusammengefasst. Über **For Each lineFormat InLineFormatCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Formate zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **LineFormatByName** und **LineFormatByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Linienformate kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add**, **Copy** und **Remove** ermöglichen das Hinzufügen, Kopieren und Löschen von Linienformaten.

Eigenschaften

- Count

Methoden

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

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcLineFormatCollection

Mit dieser Eigenschaft können Sie die Anzahl der Linienformatobjekte in der Linienformat-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Linienformate

Code-Beispiel VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection
Dim numberOfLineformats As Integer

lineFormatCltn = VcGantt1.LineFormatCollection
numberOfLineformats = lineFormatCltn.Count
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
int numberOfLineformats = lineFormatCltn.Count;
```

Methoden

Add

Methode von VcLineFormatCollection

Mit dieser Methode können Sie ein neues Linienformat in der LinienFormat-Auflistung anlegen. Wenn der Name noch nicht verwendet wird, wird das neue Linienformat-Objekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ formatName	System.String	Name des Linienformats
Rückgabewert	VcLineFormat	Neues Linienformatobjekt

Code-Beispiel VB.NET

```
Dim newLineFormat = VcGantt1.LineFormatCollection.Add("lineFormat1")
```

Code-Beispiel C#

```
newLineFormat = vcGantt1.LineFormatCollection.Add("lineFormat1");
```

AddBySpecification

Methode von VcLineFormatCollection

Mit dieser Methode können Sie ein Linienformat über eine Linienformat-Spezifikation erzeugen. Dies dient der Persistenz von Linienformat-Objekten. Die Spezifikation eines Linienformats kann erfragt (siehe VcLineFormat-

Eigenschaft **Specification**) und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Linienformat mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ formatSpecification	System.String	Linienformatspezifikation
Rückgabewert	VcLineFormat	Neues Linienformatobjekt

Copy

Methode von VcLineFormatCollection

Mit dieser Methode können Sie ein Linienformat kopieren. Wenn das Linienformat mit dem angegebenen Namen existiert und der Name des neuen Linienformats noch nicht verwendet wird, wird das neue Linienformatobjekt zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ newFormatName	System.String	Name des neuen Linienformats
Rückgabewert	VcLineFormat	Linienformatobjekt

Code-Beispiel VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection
Dim lineFormat As VcLineFormat

lineFormatCltn = VcGantt1.LineFormatCollection
lineFormat = lineFormatCltn.Copy("CurrentLineFormat", "NewLineFormat")
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
VcLineFormat lineFormat = lineFormatCltn.Copy("CurrentLineFormat",
"NewLineFormat");
```

FirstFormat

Methode von VcLineFormatCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Linienformat der Linienformat-Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextFormat** über die nachfolgenden Linienformate zu iterieren. Existiert kein Linienformat im

LinienFormatCollection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLineFormat	Erstes Linienformat

Code-Beispiel VB.NET

```
Dim format As VcLineFormat

format = VcGantt1.LineFormatCollection.FirstFormat
```

Code-Beispiel C#

```
VcLineFormat format = vcGantt1.LineFormatCollection.FirstFormat();
```

FormatByIndex

Methode von VcLineFormatCollection

Mit dieser Methode können Sie auf ein einzelnes Linienformat über ihren Index zugreifen. Existiert kein LinienFormat-Objekt an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcLineFormat	Ermitteltes Linienformat-Objekt

Code-Beispiel VB.NET

```
Dim formatLineCltn As VcLineFormatCollection
Dim formatLine As VcLineFormat

formatLineCltn = VcGantt1.LineFormatCollection
formatLine = formatLineCltn.FormatByIndex(2)
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
VcLineFormat format = lineFormatCltn.FormatByIndex(2);
```

FormatByName

Methode von VcLineFormatCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Linienformat zugreifen. Existiert kein Linienformat unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ formatName	System.String	Name des Linienformats
Rückgabewert	VcLinexFormat	Linienformat

Code-Beispiel VB.NET

```
Dim formatLineCltn As VcLineFormatCollection
Dim formatLine As VcLineFormat

formatLineCltn = VcGantt1.LineFormatCollection
formatLine = formatLineCltn.FormatByName("Standard")
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
VcLineFormat format = lineFormatCltn.FormatByName("Standard");
```

NextFormat

Methode von VcLineFormatCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Linienformate des Linienformat-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	VcLineFormat	Folgendes Linienformat

Code-Beispiel VB.NET

```
Dim formatLineCltn As VcLineFormatCollection
Dim formatLine As VcLineFormat

formatLineCltn = VcGantt1.LineFormatCollection
formatLine = formatLineCltn.FirstFormat

While Not formatLine Is Nothing
   ListLine1.Items.Add(formatLine.Name)
   formatLine = formatLineCltn.NextFormat
End While
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
VcLineFormat lineFormat = lineFormatCltn.FirstFormat();

while (lineFormat != null)
   {
   ListLine.Items.Add(lineFormat.Name);
  lineFormat = lineFormatCltn.NextFormat();
   }
```

Remove

Methode von VcLineFormatCollection

Mit dieser Methode können Sie ein Linienformat löschen. Wenn das Linienformat noch in irgendeinem anderen Objekt benutzt wird, kann es nicht gelöscht werden. In diesem Fall wird **False** zurückgegeben, sonst **True**.

	Datentyp	Beschreibung
Parameter:		
⇨ FormatName	System.String	Name des Linienformats
Rückgabewert	System.Boolean	Linienformat gelöscht (True) / nicht gelöscht (False)

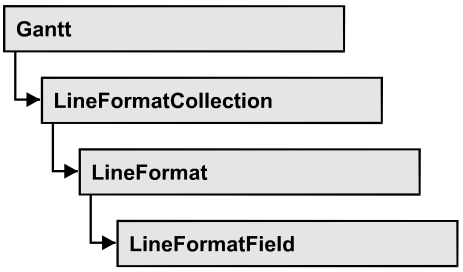
Code-Beispiel VB.NET

```
Dim lineFormatCltn As VcLineFormatCollection
Dim lineFormat As VcLineFormat

lineFormatCltn = VcGantt1.LineFormatCollection
lineFormat = lineFormatCltn.FormatByIndex(1)
lineFormatCltn.Remove(lineFormat.Name)
```

Code-Beispiel C#

```
VcLineFormatCollection lineFormatCltn = vcGantt1.LineFormatCollection;
VcLineFormat lineFormat = lineFormatCltn.FormatByIndex(1);
lineFormatCltn.Remove(lineFormat.Name);
```

7.50 VcLineFormatField



Ein Objekt vom Typ VcLineFormatField stellt ein Linienformatfeld, also ein Feld eines VcLineFormat-Objekts dar. Ein Linienformatfeld besitzt im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, unter dem es im Linienformat untergebracht ist.

Eigenschaften

- Alignment
- ConstantText
- DateOutputFormat
- FormatName
- Index
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName
- TextDataFieldIndex
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TextLineCount

Eigenschaften

Alignment

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Linienformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFormatFieldAlignment **Mögliche Werte:** .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Ausrichtung des Feldinhalts unten unten links unten rechts unten mittig links rechts oben oben links oben rechts

ConstantText

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie einen konstanten Text in dem Linienformatfeld ausgeben, falls der Typ des Linienformatfeldes auf **vcFFTText** und falls die Eigenschaft **TextDataFieldIndex** auf -1 gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Konstanter Text

DateOutputFormat

Eigenschaft von VcLineFormatField

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.

	Datentyp	Beschreibung
Eigenschaftswert	System.String {DMYhms:;/}	Datum

Code-Beispiel VB.NET

```
VcGantt1.DateOutputFormat = "DD.MM.YY"
```

Code-Beispiel C#

```
vcGantt1.DateOutputFormat = "DD.MM.YY";
```

FormatName

Nur-Lese-Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Namen des Linienformats erfragen, zu dem dieses Tabellenformatfeld gehört.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Linienformats

Index

Nur-Lese-Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Index des Linienformatfelds im zugehörigen Linienformat erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Linienformatfeldes

PatternBackgroundColorAsARGB

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Linienformatfeldes 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 Linienformats besitzen soll.

Wenn in der Eigenschaft **PatternBackgroundColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Hintergrundfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** -1

PatternBackgroundColorDataFieldIndex

Eigenschaft von VcLineFormatField

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.Int16	Datenfeldindex

PatternBackgroundColorMapName

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn Sie diesen Name für eine Farbzuordnungstabelle und zusätzlich einen Datenfeldindex über die Eigenschaft **PatternBackgroundColorDataFieldIndex** angeben, wird die Hintergrundfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Wert aus der Zuordnungstabelle zu, kommt die Hintergrundfarbe aus der Eigenschaft **BackgroundColor** zur Anwendung.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

PatternColorAsARGB

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie die Musterfarbe desLinienformatfeldes 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 Linienformats besitzen soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Musterfarbe des Linienformatfeldes

PatternColorMapName

Eigenschaft von VcLineFormatField

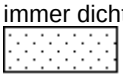
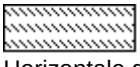
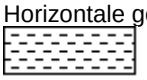
Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **PatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Kalendergitters aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **PatternColor** ausgegeben.

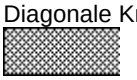
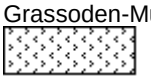
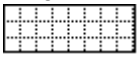
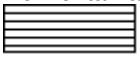
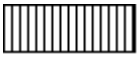
	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

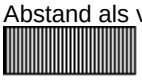
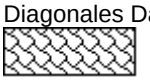
PatternEx

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie für den Hintergrund des Linienformatfeldes ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte:	
	.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 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 

PatternExDataFieldIndex

Nur-Lese-Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternExMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

PatternExMapName

Nur-Lese-Eigenschaft von VcLineFormatField

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
Eigenschaftswert	System.String	Name der Musterzuordnungstabelle

TextDataFieldIndex

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Index des Datenfelds, dessen Inhalt in dem Linienformatfeld dargestellt werden soll, erfragen oder setzen, sofern es sich um ein Feld des Datentyps **vcFFTText** handelt. Falls Sie den Index auf -1 setzen, wird stattdessen der Inhalt der Eigenschaft **ConstantText** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

TextFontColor

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie die Schriftfarbe des Linienformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFTText** 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 {True}	Schriftfarbe des Linienformatfelds

TextFontColorDataFieldIndex

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Zuordnungstabelle für Schriftfarbe in der Eigenschaft **TextFontColorMapName** benötigt wird. Wenn Sie hier -1 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16 {True}	Datenfeldindex

TextFontColorMapName

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) für die Schriftfarbe setzen oder erfragen, falls der Typ des Formatfeldes auf **vcFFTText** gesetzt wurde. Wird für den Namen "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn der Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **TextFontColorDataFieldIndex** angegeben ist, wird die Schriftfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Eintrag der Zuordnungstabelle zu, wird die Hintergrundfarbe verwendet, die über die Eigenschaft **TextFontColor** gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Schriftfarben-Zuordnungstabelle

TextFontDataFieldIndex

Eigenschaft von VcLineFormatField

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 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

TextFontMapName

Eigenschaft von VcLineFormatField

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

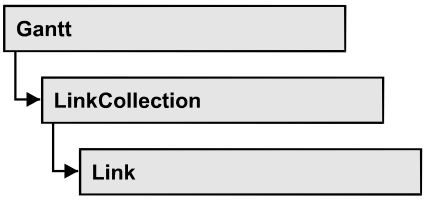
TextLineCount

Eigenschaft von VcLineFormatField

Mit dieser Eigenschaft können Sie die Zeilenanzahl erfragen oder setzen, wenn die Größe des Beschriftungsfeldes dies zulässt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Zeilenzahl

7.51 VcLink



Ein Verbindungsobjekt stellt die logische und grafische Verbindung zwischen zwei Knoten dar. Ob Verbindungen dargestellt werden, kann auf der Eigenschaftenseite **Verbindungen** im Kontrollkästchen **Verbindungen anzeigen** festgelegt werden. Auch wenn sie nicht angezeigt werden, werden sie für die Terminrechnung (Schedule) verwendet.

Eigenschaften

- AllData
- DataField
- ID
- 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
Eigenschaftswert	System.String	Alle Daten der Verbindung

Code-Beispiel VB.NET

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

linkCltn = VcGantt1.LinkCollection
link = linkCltn.FirstLink
allDataOfLink = link.AllData
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.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
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes
Eigenschaftswert	System.Object	Inhalt des Datenfelds

Code-Beispiel VB.NET

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

linkCltn = VcGantt1.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 = vcGantt1.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
Eigenschaftswert	System.String	Verbindungs-ID

PredecessorNode

Nur-Lese-Eigenschaft von VcLink

Mit dieser Eigenschaft kann der Vorgängerknoten einer Verbindung identifiziert werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcNode	Vorgängerknoten

Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

linkCltn = VcGantt1.LinkCollection
link = linkCltn.FirstLink
node = link.PredecessorNode
nodeName = node.DataField(1)
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.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
Eigenschaftswert	VcNode	Nachfolgerknoten

Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink
Dim node As VcNode
Dim nodeName As String

linkCltn = VcGantt1.LinkCollection
link = linkCltn.FirstLink
node = link.SuccessorNode
nodeName = node.DataField(1)
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.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 VcGantt1_VcLinksRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcLinksClickingEventArgs) Handles VcGantt1.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 vcGantt1_VcLinksRightClicking(object sender,
NETRONIC.XGantt.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
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes, das den Schlüssel enthält
Rückgabewert	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
Rückgabewert	System.Boolean	Verbindung erfolgreich /nicht erfolgreich aktualisiert

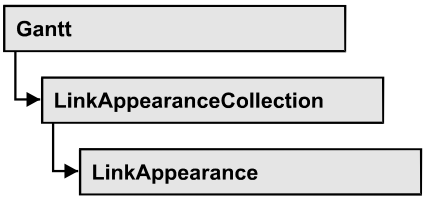
Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
Dim link As VcLink

linkCltn = VcGantt1.LinkCollection
link = linkCltn.FirstLink
link.DataField(2) = 10
link.Update()
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.LinkCollection;
VcLink link = linkCltn.FirstLink();

link.set_DataField(2, 10);
link.Update();
```

7.52 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 Tabelle voreingestellt werden.

Eigenschaften

- FilterName
- LineColor
- LineThickness
- LineType
- Name
- PredecessorLayerName
- PredecessorPortSymbol
- RoutingType
- SuccessorLayerName
- SuccessorPortSymbol
- Visible

Methoden

- PutInOrderAfter

Eigenschaften

FilterName

Nur-Lese-Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie den Filter erfragen, der für ein bestimmtes Verbindungsaussehen 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 = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
filterOfLinkApp = linkAppearance.FilterName
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance =
linkAppearanceCltn.LinkAppearanceByName("Blue");
string filterOfLinkApp = linkAppearance.FilterName;
```

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-Farbwerte

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineColor = Color.Blue
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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
Eigenschaftswert	System.Int32	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000}: Werte in 1/100 mm **Standardwert:** As defined on property page

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
linkAppearance.LineThickness = 4
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcLineType	Linientyp **Standardwert:** vcSolid
	Mögliche Werte:	
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashed 4	Linientyp **gestrichelt**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDashedDotted 5	Linientyp **gestrichelt-gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.vcDotted 3	Linientyp **gepunktet**
	.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
.vcSolid 2	Linientyp **durchgezogen**
.vcSolid 2	Linientyp **durchgezogen**

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Blue")
linkAppearance.LineType = 5
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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**
Eigenschaftswert	System.String	Name des Verbindugnsaussehens

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
Dim nameLinkApp As String

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
nameLinkApp = linkAppearance.Name
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
string nameLinkApp = linkAppearance.Name;
```

PredecessorLayerName

Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie vorgeben oder erfragen, von welchem Layer des jeweiligen Vorgängerknotens eine Verbindung, die mit dieser VcLinkAppearance verwendet wird, abgehen soll. Bei Eingabe von "" (Default) geht die Verbindung vom ersten sichtbaren Layer des jeweiligen Vorgängerknotes ab.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Layer-Namen übergibt

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
Eigenschaftswert	VcLinkPredecessorPortSymbol	Symbol am Vorgängerknoten **Standardwert:** vcLPSNone

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSDoubleSemiCircle
```

Code-Beispiel C#

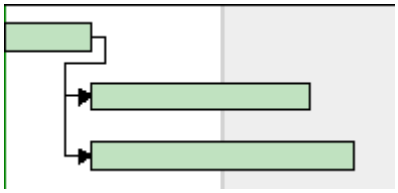
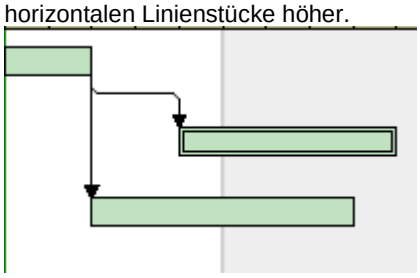
```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.PredecessorPortSymbol =
VcLinkPredecessorPortSymbol.vcLPSFilledDoubleSemiCircle;
```

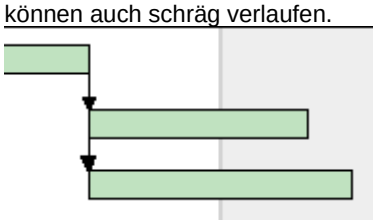
RoutingType

Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob die Verbindungen im Diagramm nur waagerecht 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
Eigenschaftswert	VcRoutingType	Routentyp **Standardwert:** vcLRTOrthogonal
	Mögliche Werte:	
	.vcLRTOrthogonal 1	Verbindungen verlaufen nur waagerecht und senkrecht und zeigen eine orthogonale Form. 
	.vcLRTOrthogonalDistinguishable 2	Verbindungen verlaufen nur waagerecht und senkrecht und zeigen eine orthogonale Form. Durch die Darstellung von entsprechenden Schrägen bei Knicken kann man die Verbindungen voneinander unterscheiden und ihre Richtung besser verfolgen. Das Diagramm wird dabei entsprechend der Anzahl der entstehenden horizontalen Linienstücke höher. 

.vcLRTStraightLined 4	Verbindungen zeigen eine gerade Form und können auch schräg verlaufen. 

SuccessorLayerName

Eigenschaft von VcLinkAppearance

Mit dieser Eigenschaft können Sie vorgeben oder erfragen, zu welchem Layer des jeweiligen Vorgängerknotens eine Verbindung, die mit dieser VcLinkAppearance verwendet wird, führen soll. Bei Eingabe von "" (Default) führt die Verbindung zum ersten sichtbaren Layer des jeweiligen Nachfolgerknotes.

Diese Eigenschaft kann auch auf der Eigenschaftenseite **Verbindungen** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die den Layer-Namen übergibt

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.

	Datentyp	Beschreibung
Eigenschaftswert	VcLinkSuccessorPortSymbol	Symbol am Nachfolgerknoten **Standardwert:** vcLSSNone

Code-Beispiel VB.NET

```
VcLinkAppearanceCollection linkAppearanceCltn =
VcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.FirstLinkAppearance();
linkAppearance.SuccessorPortSymbol =
VcLinkSuccessorPortSymbol.vcLSSFilledDoubleArrow;
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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
Eigenschaftswert	System.Boolean	Eigenschaft in Kraft/nicht in Kraft **Standardwert:** True

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
linkAppearance.Visible = False
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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
Parameter:		
refLinkAppearanceName	System.String	Name des Verbindungsaussehens, hinter das das aktuelle Verbindungsaussehen gesetzt werden soll

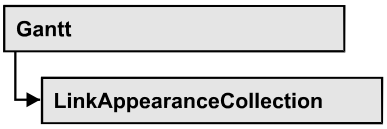
Code-Beispiel VB.NET

```
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();
```

7.53 VcLinkAppearanceCollection



In einem Objekt vom Typ **VcLinkAppearanceCollection** sind automatisch alle definierten LinkAppearance-Objekte zusammengefasst. Über **For Each linkAppearance InLinkAppearanceCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Verbindungsaussehen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **LinkAppearanceByName** und **LinkAppearanceByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Verbindungsaussehen 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
Eigenschaftswert	System.Int32	Anzahl der LinkAppearance-Objekte

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim numberOfLinkAppearance As Integer

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
numberOfLinkAppearance = linkAppearanceCltn.Count
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.LinkAppearanceCollection;
int numberOfLinkAppearance = linkAppearanceCltn.Count;
```

Methoden

Add

Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie ein neues Verbindungsaussehen in der Link-Appearance-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
Parameter:		
⇨ newName	System.String	Name des LinkAppearance-Objekts
Rückgabewert	VcLinkAppearance	Neues LinkAppearance-Objekt

Code-Beispiel VB.NET

```
newLinkAppearance = VcGantt1.LinkAppearanceCollection.Add("linkapp1")
```

Code-Beispiel C#

```
newLinkAppearance = vcGantt1.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
Parameter:		
⇨ linkAppearanceSpecification	System.String	Verbindungsaussehen-Spezifikation
Rückgabewert	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
Parameter:		
⇨ fromName	System.String	Name des zu kopierenden Verbindungsaussehens
⇨ newName	System.String	Name des neuen Verbindungsaussehens
Rückgabewert	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 = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.FirstLinkAppearance
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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 Verbindungsaussehen-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
Parameter:		
⇨ index	System.Int16	Index des LinkAppearance-Objektes
Rückgabewert	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
Parameter:		
⇨ linkAppearanceName	System.String	Name des LinkAppearance-Objektes
Rückgabewert	VcLinkAppearance	LinkAppearance-Objekt

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance
linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByName("Standard")
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.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
Rückgabewert	VcLinkAppearance	Nachfolgendes LinkAppearance-Objekt

Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.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 =
vcGantt1.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
Parameter:		
⇨ name	System.String	Name des Verbindungsaussehens
Rückgabewert	System.Boolean	Verbindungsaussehen gelöscht (True)/nicht gelöscht (False)

Update

Methode von VcLinkAppearanceCollection

Mit dieser Methode können Sie eine die Auflistung von Verbindungsaussehen aktualisieren, nachdem Sie sie verändert haben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Auflistung erfolgreich /nicht erfolgreich aktualisiert

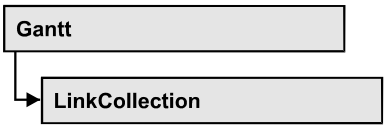
Code-Beispiel VB.NET

```
Dim linkAppearanceCltn As VcLinkAppearanceCollection
Dim linkAppearance As VcLinkAppearance

linkAppearanceCltn = VcGantt1.LinkAppearanceCollection
linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0)
linkAppearanceCltn.Remove(linkAppearance.Name)
linkAppearanceCltn.Update()
```

Code-Beispiel C#

```
VcLinkAppearanceCollection linkAppearanceCltn =
vcGantt1.LinkAppearanceCollection;
VcLinkAppearance linkAppearance = linkAppearanceCltn.LinkAppearanceByIndex(0);
linkAppearanceCltn.Remove(linkAppearance.Name);
linkAppearanceCltn.Update();
```

7.54 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 = VcGantt1.LinkCollection
numberOfLinks = linkCltn.Count
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.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 = VcGantt1.LinkCollection
link = linkCltn.FirstLink
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.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 = VcGantt1.LinkCollection
link = linkCltn.FirstLink
While Not link Is Nothing
   ListBox1.Items.Add(link.AllData)
   link = linkCltn.NextLink
End While
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.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
Parameter:		
⇨ selectionType	VcSelectionType	Auszuwählende Verbindungen
	Mögliche Werte:	
	.vcAll 0	Alle Objekte im Diagramm werden ausgewählt.
	.vcAllLinksCausingCycles 7	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.

	.vcAllLinksInCycles 6	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	Alle sichtbaren Objekte werden ausgewählt.
	.vcSelected 2	Alle markierten Objekte werden ausgewählt.
Rückgabewert	System.Int32	Anzahl ausgewählter Verbindungen

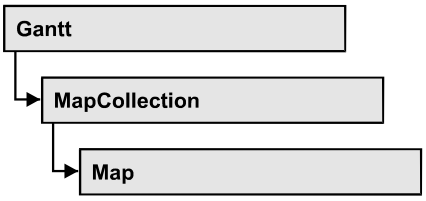
Code-Beispiel VB.NET

```
Dim linkCltn As VcLinkCollection
linkCltn = VcGantt1.LinkCollection
linkCltn.SelectGroups (vcAllMarked)
```

Code-Beispiel C#

```
VcLinkCollection linkCltn = vcGantt1.LinkCollection;
linkCltn.SelectGroups (vcAllMarked);
```

7.55 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
- GetEnumerator
- 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
Eigenschaftswert	System.Int32	Anzahl der Mapeinträge

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim numberOfEntries As Integer

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
numberOfEntries = map.Count
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.MapByName("Map1");
int numberOfEntries = map.Count;
```

GetEnumerator

Nur-Lese-Eigenschaft von VcMap

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 Einträge iterieren.

	Datentyp	Beschreibung
Eigenschaftswert	VcObject	Referenzobjekt

Name

Nur-Lese-Eigenschaft von VcMap

Mit dieser Eigenschaft können Sie den Namen der Zuordnungstabelle (Map) abfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuordnungstabelle

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapName As String

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
mapName = map.Name
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Eigenschaftswert	System.String	Spezifikation der Zuordnungstabelle

Code-Beispiel VB.NET

```
Dim boxCltn As VcBoxCollection
Dim box As VcBox

boxCltn = VcGantt1.BoxCollection
box = boxCltn.FirstBox
MsgBox(box.Specification)
```

Code-Beispiel C#

```
VcBoxCollection boxCltn = vcGantt1.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**
Eigenschaftswert	VcMapType **Mögliche Werte:** .vcAnyMap 0 .vcColorMap 1 .vcFontMap 8 .vcGraphicsFileMap 7 .vcMillimeterMap 9 .vcNumberMap 10 .vcPatternMap 3 .vcTextMap 6	Typ der Zuordnungstabelle **Beliebig** (nur zum Selektieren verwendet) **Farben** **Schriften** **Grafikdatei** **Millimeter** **Zahlen** **Schraffuren** **Text**

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
map.Type = VcMapType.vcPatternMap
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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 Zu-

ordnungstabelle 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 = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.CreateEntry
mapCltn.Update
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Parameter:		
⇨ mapEntry	VcMapEntry	Eintrag der Map
Rückgabewert	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 = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
map.DeleteEntry(mapEntry)
mapCltn.Update
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Rückgabewert	VcMapEntry	Erster Map-Eintrag

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)

map = mapCltn.FirstMap
mapEntry = map.FirstMapEntry
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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

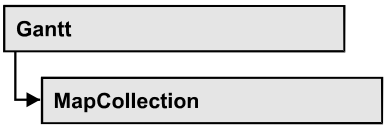
Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGantt1.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 = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcAnyMap);
VcMap map = mapCltn.FirstMap();
VcMapEntry mapEntry = map.FirstMapEntry()
while (mapEntry != null)
   {
   listBox1.Items.Add(mapEntry.LegendText);
   mapEntry= map.NextMapEntry();
   }
```

7.56 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 = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
numberOfMaps = mapCltn.Count
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Parameter:		
⇨ mapName	System.String	Name der Zuordnungstabelle
Rückgabewert	VcMap	Neues Zuordnungstabellenobjekt

Code-Beispiel VB.NET

```
newMap = VcGantt1.MapCollection.Add("Map1")
```

Code-Beispiel C#

```
VcMap newMap = vcGantt1.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
Parameter:		
⇨ specification	System.String	Zuordnungstabellenspezifikation
Rückgabewert	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
Parameter:		
⇨ mapName	System.String	Name der zu kopierenden Zuordnungstabelle
⇨ newMapName	System.String	Name der neuen Zuordnungstabelle
Rückgabewert	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 = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.FirstMap
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Parameter:		
⇨ index	System.Int16	Index der Zuordnungstabelle
Rückgabewert	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
Parameter:		
⇨ mapName	System.String	Name der Map
Rückgabewert	VcMap	Map

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Rückgabewert	VcMap	Nachfolgende Map

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGantt1.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 = vcGantt1.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
Parameter:		
⇨ mapName	System.String	Name der Zuordnungstabelle
Rückgabewert	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
Parameter:		
⇨ selectionType	VcMapType	Auszuwählender Map-Typ
	Mögliche Werte: .vcAnyMap 0 .vcColorMap 1 .vcFontMap 8 .vcGraphicsFileMap 7 .vcMillimeterMap 9 .vcNumberMap 10 .vcPatternMap 3 .vcTextMap 6	**Beliebig** (nur zum Selektieren verwendet) **Farben** **Schriften** **Grafikdatei** **Millimeter** **Zahlen** **Schraffuren** **Text**
Rückgabewert	System.Int32	Anzahl der Ausgewählten Maps

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Aktualisierung erfolgt (True)/ nicht erfolgt (False)

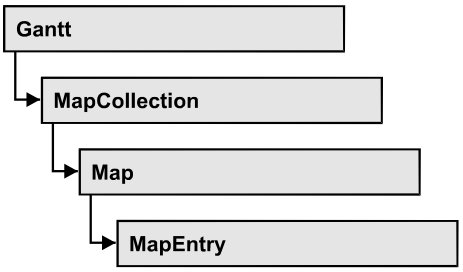
Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry

mapCltn = VcGantt1.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 = vcGantt1.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();
```

7.57 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
- LegendText
- Millimeter
- 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
Eigenschaftswert	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 = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcColorMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
colorOfMapEntry = mapEntry.Color
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Eigenschaftswert	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 = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcAnyMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry

dataFieldValue = mapEntry.DataFieldValue
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Eigenschaftswert	VcFontBody	Schriftgrad
	Mögliche Werte:	
	.vcBold 2	fett
	.vcBoldItalic 4	fett und kursiv
	.vcItalic 3	kursiv
	.vcRegular 1	normal

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontBodyOfMapEntry As VcFontBody

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontBodyOfMapEntry = VcFontBody.vcBold
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Eigenschaftswert	System.String	Schriftart

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontNameOfMapEntry As String

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontNameOfMapEntry = "Arial"
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Eigenschaftswert	System.Int32	Schriftgröße

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim fontSizeOfMapEntry As Integer

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
fontSizeOfMapEntry = 14
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.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
Eigenschaftswert	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 = VcGantt1.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 = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcGraphicsFileMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();

String exeName = Environment.GetCommandLineArgs()[0];
mapEntry.GraphicsFileName = System.IO.Path.GetDirectoryName(exeName)+
@"\..\Bitmaps\picture1.bmp";
```

LegendText

Eigenschaft von VcMapEntry

Mit dieser Eigenschaft können Sie den Legendentext für den Zuordnungstabelleneintrag erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Legendentext

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim legendOfMapEntry As String

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcFontMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
legendOfMapEntry = "1. activity"
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcFontMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
string legendOfMapEntry = "1. activity";
```

Millimeter

Eigenschaft von VcMapEntry

für Millimeter-Zuordnungstabellen: Mit dieser Eigenschaft können Sie den Millimeter-Wert des Zuordnungstabelleneintrags erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Wert in 1/100

Code-Beispiel VB.NET

```
Dim mapCltn As VcMapCollection
Dim map As VcMap
Dim mapEntry As VcMapEntry
Dim millimeterOfMapEntry As Integer

mapCltn = VcGantt1.MapCollection
mapCltn.SelectMaps(VcMapType.vcMillimeterMap)
map = mapCltn.MapByName("Map1")
mapEntry = map.FirstMapEntry
millimeterOfMapEntry = 3
```

Code-Beispiel C#

```
VcMapCollection mapCltn = vcGantt1.MapCollection;
mapCltn.SelectMaps(VcMapType.vcMillimeterMap);
VcMap map = mapCltn.MapByName("Map1");
VcMapEntry mapEntry = map.FirstMapEntry();
int millimeterOfMapEntry = 3;
```

Number

Eigenschaft von VcMapEntry

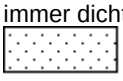
Für numerische Zuordnungstabellen: Mit dieser Eigenschaft können Sie eine Zahl als Zuordnungstabelleneintrag festlegen oder erfragen.

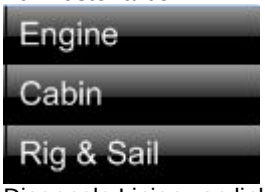
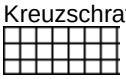
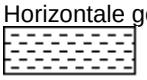
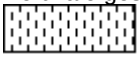
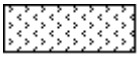
	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Numerischer Wert

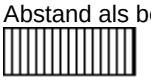
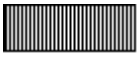
Pattern

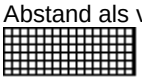
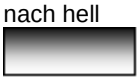
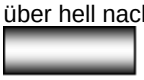
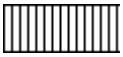
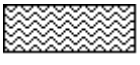
Eigenschaft von VcMapEntry

Für Schraffuren-Zuordnungstabellen (vcPatternMap): mit dieser Eigenschaft können Sie den Schraffurtyp des Zuordnungstabelleneintrags erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern **Mögliche Werte:** .vc05PercentPattern... vc90PercentPattern 01 - 11	Mustertyp 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 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 

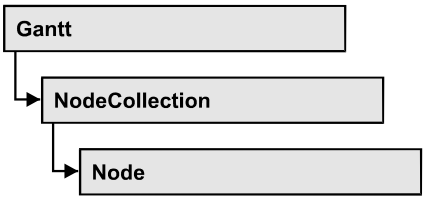
Code-Beispiel VB.NET

```
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;
```

7.58 VcNode



Ein Knoten ist ein Grundelement eines Balkendiagramms. Knoten lassen sich über Verbindungen zu einer Struktur verknüpfen. Das Aussehen eines Knotens wird über die Layer bestimmt, deren Filter auf den Knoten zutreffen. Eingefügt werden Knoten über die VcGantt-Methoden **InsertNodeRecord** bzw. **Open** oder interaktiv.

Eigenschaften

- AllData
- DataField
- ID
- IncomingLinks
- Marked
- OutgoingLinks
- SnapTargetMode
- SnapTargetMode
- SuperGroup
- UpdateBehaviorName

Methoden

- DataRecord
- Delete
- GetPositionInView
- NodeRowInView
- OutlineIndent
- OutlineOutdent
- RelatedDataRecord
- SetPositionInView
- 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
Eigenschaftswert	System.String	Alle Daten des Datensatzes

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeModifying(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeModifyingEventArgs) Handles VcGantt1.VcNodeModifying
   Dim allDataOfNode As String
   e.ReturnStatus = VcReturnStatus.vcRetStatFalse

   allDataOfNode = e.Node.AllData
   MsgBox(allDataOfNode)
End Sub
```

Code-Beispiel C#

```
private void vcGantt1_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
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes
Eigenschaftswert	System.Object	Inhalt des Datenfeldes

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.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 vcGantt1_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
Eigenschaftswert	System.String	Knoten-ID

Code-Beispiel VB.NET

```
VcNode node = VcGantt1.NodeCollection.FirstNode()

MsgBox (node.ID)
```

Code-Beispiel C#

```
VcNode node = vcGantt1.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
Eigenschaftswert	VcLinkCollection	LinkCollection-Objekt

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.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 vcGantt1_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
Eigenschaftswert	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 = VcGantt1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcAll)

For Each node In nodeCltn
   linkCltn = node.IncomingLinks
   For Each link In linkCltn
      predecessor = link.PredecessorNode
      predecessor.Marked = True
   Next
Next
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcAll);
VcNode predecessorNode;
VcLinkCollection linkCltn;
foreach (VcNode node in nodeCltn)
   {
   linkCltn = node.IncomingLinks;
   foreach (VcLink link in linkCltn)
      {
      predecessorNode = link.PredecessorNode;
      predecessorNode.Marked = true;
      }
   }
```

OutgoingLinks

Nur-Lese-Eigenschaft von VcNode

Mit dieser Eigenschaft haben Sie Zugriff auf alle Verbindungen, die von einem Knoten ausgehen.

	Datentyp	Beschreibung
Eigenschaftswert	VcLinkCollection	LinkCollection-Objekt

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.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 vcGantt1_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;
   }
```

SnapTargetMode

Nur-Lese-Eigenschaft von VcNode

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob dieser Knoten automatisch oder manuell als mögliches Einrstziel selektiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeSnapTargetMode	Selektionsmodus für den Knoten beim Verschieben mit Einrastzielen **Standardwert:** vcNSTMAutomatically

SnapTargetMode

Nur-Lese-Eigenschaft von VcNode

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob dieser Knoten automatisch oder manuell als mögliches Einrstziel selektiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	VcNodeSnapTargetMode	Selektionsmodus für den Knoten beim Verschieben mit Einrastzielen **Standardwert:** vcNSTMAutomatically

SuperGroup

Nur-Lese-Eigenschaft von VcNode

Mit dieser Eigenschaft können Sie bei gruppierter Darstellung die Gruppe erfragen, der der aktuelle Knoten angehört.

	Datentyp	Beschreibung
Eigenschaftswert	VcGroup	Gruppe, der der Knoten angehört

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.VcNodeRightClicking

   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup

   Dim group As VcGroup
   group = e.Node.SuperGroup
   Label1.Text = "Group: " + group.Name

End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNodeRightClicking(object sender,
NETRONIC.XGantt.VcNodeClickingEventArgs e)
   {
   e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
   VcGroup group = e.Node.SuperGroup;
   label1.Text = "Group: " + group.Name;
   }
```

UpdateBehaviorName

Eigenschaft von VcNode

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

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
Rückgabewert	System.Boolean	Knoten erfolgreich/nicht erfolgreich gelöscht

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNodeRightClicking(ByVal sender As Object, ByVal e As
NETRONIC.XGantt.VcNodeClickingEventArgs) Handles VcGantt1.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 vcGantt1_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();
      e.ReturnStatus = VcReturnStatus.vcRetStatNoPopup;
      }
   }
```

GetPositionInView

Methode von VcNode

Mit dieser Methode können Sie erfragen, an welcher Position im sichtbaren Bereich des Diagramms der Knoten liegt.

	Datentyp	Beschreibung
Parameter:		
viewReferencePoint	VcViewReferencePoint	Referenzpunkt (des Diagramms)
	Mögliche Werte:	
	.vcVRPBottomCenter 28	unten mittig
	.vcVRPBottomLeft 27	unten links

	.vcVRPBottomRight 29 .vcVRPCenterCenter 25 .vcVRPCenterLeft 24 .vcVRPCenterRight 26 .vcVRPTopCenter 22 .vcVRPTopLeft 21 .vcVRPTopRight 23	unten rechts mittig mittig mittig links mittig rechts oben mittig oben links oben rechts
nodeReferencePoint	VcNodeReferencePoint **Mögliche Werte:** .vcNRPBottomCenter 28 .vcNRPBottomLeft 27 .vcNRPBottomRight 29 .vcNRPCenterCenter 25 .vcNRPCenterLeft 24 .vcNRPCenterRight 26 .vcNRPTopCenter 22 .vcNRPTopLeft 21 .vcNRPTopRight 23	Knotenreferenzpunkt unten mittig unten links unten rechts mittig mittig mittig links mittig rechts oben mittig oben links oben rechts
⇦ xOffset	System.Int32	x-Wert des Offsets (= des Abstands des Knotenreferenzpunkts vom Referenzpunkt) in Pixeln
⇦ yOffset	System.Int32	y-Wert des Offsets in Pixeln
Rückgabewert	Void	

NodeRowInView

Methode von VcNode

Mit dieser Methode können Sie erfragen, ob die Zeile, in der sich der Knoten befindet, im sichtbaren Bereich des Diagramms ist.

	Datentyp	**Beschreibung**
Rückgabewert	System.Boolean	Zeile befindet sich/befindet sich nicht im sichtbaren Bereich des Diagramms

Code-Beispiel VB.NET

```
Dim node As VcNode

node = VcGantt1.GetNodeByID(15)
If Not node.NodeRowInView Then
   VcGantt1.ScrollToNodeLine(node, VcVerticalAlignment.vcVerCenterAligned)
End If
```

Code-Beispiel C#

```
VcNode node = vcGantt1.GetNodeByID(2);
if (node.NodeRowInView() == false)
   vcGantt1.ScrollToNodeLine(node, VcVerticalAlignment.vcVerCenterAligned);
```

OutlineIndent

Methode von VcNode

Mit dieser Methode kann man im hierarchisch gruppierten Diagramm Knoten tiefer stufen. Dabei wird der Knoten im Tabellenteil nach rechts gerückt, bleibt aber in seiner bisherigen Zeile. Diese Methode entspricht dem Menüpunkt **Tiefer stufen** im Kontextmenü für Knoten.

Der Rückgabewert gibt an, ob die Methode erfolgreich war. Das Tieferstufen ist beispielsweise für Knoten, die bereits auf der tiefsten Ebene liegen, nicht möglich.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Methode erfolgreich (True)/ Methode nicht erfolgreich (False)

Code-Beispiel VB.NET

```
Private Sub CommandOutlineIndent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode   'Function returns selected node
    MsgBox anode.OutlineIndent
    Set anode = Nothing
End Sub
```

Code-Beispiel C#

```
Private Sub CommandOutlineIndent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode   'Function returns selected node
    MsgBox anode.OutlineIndent
    Set anode = Nothing
End Sub
```

OutlineOutdent

Methode von VcNode

Mit dieser Methode kann man im hierarchisch gruppierten Diagramm Knoten höher stufen. Dabei wird der Knoten im Tabellenteil nach links gerückt, bleibt aber in seiner bisherigen Zeile. Diese Methode entspricht dem Menüpunkt **Höher stufen** im Kontextmenü für Knoten.

Der Rückgabewert gibt an, ob die Methode erfolgreich war. Das Höherstufen ist beispielsweise für Knoten, die bereits auf der höchsten Ebene liegen, nicht möglich.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Methode erfolgreich (True)/ Methode nicht erfolgreich (False)

Code-Beispiel VB.NET

```
Private Sub CommandOutlineOutdent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode   'Function returns selected node
    MsgBox anode.OutlineOutdent
    Set anode = Nothing
End Sub
```

Code-Beispiel C#

```
Private Sub CommandOutlineOutdent_Click()
    Dim anode As VcNode
    Set anode = getSelectedNode   'Function returns selected node
    MsgBox anode.OutlineOutdent
    Set anode = Nothing
End Sub
```

RelatedDataRecord

Methode von VcNode

Mit dieser Eigenschaft können Sie einen Datensatz aus einer verknüpften Tabelle erfragen, der dem Datensatz der Knotendatentabelle zugeordnet ist. Der im Parameter übergebene Index bezeichnet das Feld im Datensatz, in dem der Schlüssel des zugeordneten Datensatzes steht.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des Datenfeldes, das den Schlüssel enthält
Rückgabewert	VcDataRecord	Zurückgegebener zugeordneter Datensatz

SetPositionInView

Methode von VcNode

Mit dieser Methode können Sie festlegen, dass so gescrollt werden soll, dass der Knoten an einer bestimmten Position im sichtbaren Bereich des Diagramms liegt. Diese Position können Sie angeben, indem Sie den Abstandsvektor (x,y) zwischen einem bestimmten Knotenreferenzpunkt und einem bestimmten Referenzpunkt des Diagramms angeben.

	Datentyp	Beschreibung
Parameter:		
viewReferencePoint	VcViewReferencePoint	Referenzpunkt (des Diagramms)
	Mögliche Werte:	
	.vcVRPBottomCenter 28	unten mittig
	.vcVRPBottomLeft 27	unten links
	.vcVRPBottomRight 29	unten rechts
	.vcVRPCenterCenter 25	mittig mittig
	.vcVRPCenterLeft 24	mittig links
	.vcVRPCenterRight 26	mittig rechts
	.vcVRPTopCenter 22	oben mittig
	.vcVRPTopLeft 21	oben links
	.vcVRPTopRight 23	oben rechts
nodeReferencePoint	VcNodeReferencePoint	Knotenreferenzpunkt
	Mögliche Werte:	
	.vcNRPBottomCenter 28	unten mittig
	.vcNRPBottomLeft 27	unten links
	.vcNRPBottomRight 29	unten rechts
	.vcNRPCenterCenter 25	mittig mittig
	.vcNRPCenterLeft 24	mittig links
	.vcNRPCenterRight 26	mittig rechts
	.vcNRPTopCenter 22	oben mittig
	.vcNRPTopLeft 21	oben links
	.vcNRPTopRight 23	oben rechts
⇦ xOffset	System.Int32	x-Wert des Offsets (= des Abstands des Knotenreferenzpunkts vom Referenzpunkt) in Pixeln
⇦ yOffset	System.Int32	y-Wert des Offsets in Pixeln
Rückgabewert	Void	

Code-Beispiel VB.NET

```
' scroll the diagram so that the vector between the bottom right corner of the
node and the bottom right corner of the diagram is (-10, -10)
Dim node As VcNode

node.SetPositionInView(VcViewReferencePoint.vcVRPBottomRight,
VcNodeReferencePoint.vcNRPBottomRight, -10, -10)
```

Code-Beispiel C#

```
// scroll the diagram so that the vector between the bottom right corner of the
node and the bottom right corner of the diagram is (-10, -10)
VcNode node;
node.SetPositionInView(VcViewReferencePoint.vcVRPBottomRight,
VcNodeReferencePoint.vcNRPBottomRight, -10, -10);
```

Update

Methode von VcNode

Nachdem Sie ein oder mehrere Datenfelder eines Knotens mit der Eigenschaft **DataField** verändert haben, aktualisieren Sie die Grafik mit **Update**.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	Knoten erfolgreich/nicht erfolgreich aktualisiert

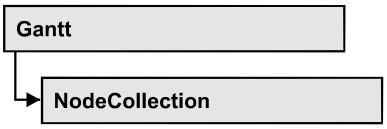
Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
node = nodeCltn.FirstNode

node.DataField(12) = "Group A"
node.Update()
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
VcNode node = nodeCltn.FirstNode();
node.set_DataField(12, "Group A");
node.Update();
```

7.59 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 InNode-Collection** 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 = VcGantt1.NodeCollection
MsgBox("Number of nodes: " + nodeCltn.Count)
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.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 = VcGantt1.NodeCollection
node = nodeCltn.FirstNode
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.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 = VcGantt1.NodeCollection
node = nodeCltn.FirstNode
While Not node Is Nothing
   node.Marked = False
   node = nodeCltn.NextNode
End While
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.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
Parameter:		
⇨ selType	VcSelectionType	Auszuwählende Knoten
	Mögliche Werte:	
	.vcAll 0	Alle Objekte im Diagramm werden ausgewählt.
	.vcAllLinksCausingCycles 7	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.
	.vcAllLinksInCycles 6	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 .vcSelected 2	Alle sichtbaren Objekte werden ausgewählt. Alle markierten Objekte werden ausgewählt.
Rückgabewert	System.Int32	Anzahl ausgewählter Knoten

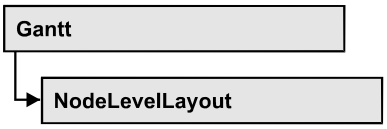
Code-Beispiel VB.NET

```
Dim nodeCltn As VcNodeCollection
Dim node As VcNode

nodeCltn = VcGantt1.NodeCollection
nodeCltn.SelectNodes(VcSelectionType.vcSelected)
```

Code-Beispiel C#

```
VcNodeCollection nodeCltn = vcGantt1.NodeCollection;
nodeCltn.SelectNodes(VcSelectionType.vcSelected);
```

7.60 VcNodeLevelLayout



Ein Objekt vom Typ VcNodeLevelLayout legt die Art der Sortierung von Knoten sowie das Erscheinungsbild der Knotenzeilen fest:

Eigenschaften

- CalendarGridName
- CalendarGridsVisible
- DateLineName
- DateLinesVisible
- RowBackgroundColorAsARGB
- RowBackgroundColorDataFieldIndex
- RowBackgroundColorMapName
- RowPattern
- RowPatternColorAsARGB
- RowPatternColorDataFieldIndex
- RowPatternColorMapName
- RowPatternDataFieldIndex
- RowPatternMapName
- SeparationLineColor
- SeparationLineInterval
- SeparationLinesVisible
- SeparationLinesVisibleAtTop
- SeparationLineThickness
- SeparationLineType
- SortDataFieldIndex
- SortOrder

Eigenschaften

CalendarGridName

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Namen des Kalendergitters für die Vorgänge setzen oder erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Knoten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalendergitters

CalendarGridsVisible

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob Kalendergitter dargestellt werden sollen oder nicht. Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Arbeitsfreie Zeiten werden/werden nicht hervorgehoben

DateLineName

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Namen der Stichtalinie für dieses Knotenebenenlayout erfragen. Diese Eigenschaft kann auch im Dialog **Gruppierung** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Stichtaglinie

DateLinesVisible

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob Stichtaglinien angezeigt werden sollen. Sie können diese Eigenschaft auch im Dialog **Gruppierung** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Stichtaglinien werden/werden nicht angezeigt

RowBackgroundColorAsARGB

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie die Hintergrundfarbe der Knotenzeile für diese Gruppierungsebene setzen oder erfragen. Die Standard-Farbe ist weiß.

	Datentyp	Beschreibung

RowBackgroundColorDataFieldIndex

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **RowBackColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung

RowBackgroundColorMapName

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft

RowBackColorDataFieldIndex angegeben ist, wird die Hintergrundfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Hintergrundfarbe aus der Eigenschaft **RowBackColor** ausgegeben.

	Datentyp	Beschreibung

RowPattern

Nur-Lese-Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie für den Hintergrund der Knotenzeilen dieser Gruppierungsebene ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	FillPatternEnum	Mustertyp

RowPatternColorAsARGB

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie die Musterfarbe der Knotenzeilen dieser Gruppierungsebene festlegen oder erfragen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil (ARGB-Wert) im Zahlenbereich von 0..255. 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.

S. auch **set/getRowBackColorAsARGB**.

Wenn in der Eigenschaft **RowPatternColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Musterfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	ARGB-Farbwerte ({0...255},{0...255},{0...255},{0...255})

RowPatternColorDataFieldIndex

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **RowPatternColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

RowPatternColorMapName

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **RowPatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Gruppentitelzeile aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **RowPatternColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

RowPatternDataFieldIndex

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **RowPatternMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Datenfeldindex

RowPatternMapName

Eigenschaft von VcNodeLevelLayout

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 Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **RowPatternDataFieldIndex** angegeben sind, wird das Muster des Layers aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird das Muster aus der Eigenschaft **RowPattern** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Musterzuordnungstabelle

SeparationLineColor

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie die Farbe der Trennlinien für die Gruppierungsebenen festlegen.

Sie können diese Eigenschaft auch im Dialog **Gruppierung**, Bereich **Knoten**, Feld **Trennlinie** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Farbwert ({0...255},{0...255},{0...255})

SeparationLineInterval

Nur-Lese-Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie setzen oder erfragen nach wie vielen Vorgängen eine Trennlinie gezogen werden soll.

	Datentyp	Beschreibung

SeparationLinesVisible

Nur-Lese-Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob Trennlinien zwischen den Vorgängen dargestellt werden sollen oder nicht. Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Trennlinien werden angezeigt/nicht angezeigt

SeparationLinesVisibleAtTop

Nur-Lese-Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie einstellen oder erfragen, ob Trennlinien oberhalb von Vorgängen dargestellt werden sollen oder unterhalb. Diese Eigenschaft kann auch im Dialog **Gruppierung**, Bereich **Knoten** festgelegt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Trennlinien oben werden angezeigt/nicht angezeigt

SeparationLineThickness

Nur-Lese-Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie die Linienstärke einer Trennlinie zwischen Vorgängen 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.

Sie können diese Eigenschaft auch im Dialog **Gruppierung**, Bereich **Knoten**, Feld **Trennlinie** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Linienstärke LineType {1...4}: Werte in Pixeln LineType {5...1000} Werte in 1/100 mm

SeparationLineType

Nur-Lese-Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie die Linienart einer Trennlinie zwischen Vorgängen erfragen oder festlegen.

Sie können diese Eigenschaft auch im Dialog **Gruppierung**, Bereich **Knoten**, Feld **Trennlinie** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	LineTypeEnum	Typ der Trennlinien der Hierarchieebenen

SortDataFieldIndex

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft können Sie den Index des Datenfeldes setzen oder erfragen, nach dem Knoten in diesem VcGroupLevelLayout-Objekt sortiert werden sollen.

	Datentyp	Beschreibung
Parameter:		
⇨ sortlevel	System.Int32	Sortierebene
Eigenschaftswert	System.Int32	Feld, nach dem sortiert wird **Standardwert:** vcAscending

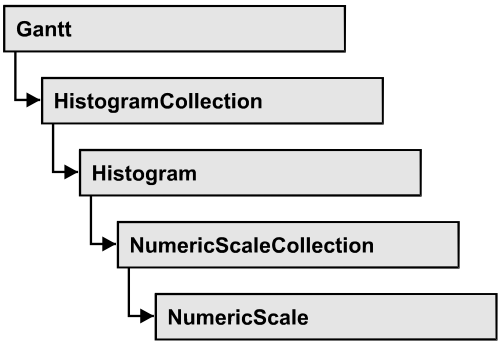
SortOrder

Eigenschaft von VcNodeLevelLayout

Mit dieser Eigenschaft legen Sie fest, ob die Vorgänge auf- oder absteigend sortiert werden. Das Feld, nach dem die Vorgänge sortiert werden, legen Sie mit der Eigenschaft **SortDataFieldIndex** fest. Diese Eigenschaft können Sie auch im Dialog **Gruppierung** festlegen.

	Datentyp	Beschreibung
Parameter:		
⇨ sortLevel	System.Int32	Sortierebene
Eigenschaftswert	SortOrderEnum	aufsteigende oder absteigende Sortierreihenfolge **Standardwert:** vcAscending

7.61 VcNumericScale



Ein Objekt vom Typ VcNumericScale stellt die Skala der vertikalen Achse von Histogrammen dar.

Eigenschaften

- DoubleOutputFormat
- Font
- FontColor
- Histogram
- LineColor
- MajorTicks
- MajorTicksEx
- MinorTicks
- MinorTicksEx
- Name
- PatternBackgroundColorAsARGB
- PatternColorAsARGB
- PatternEx
- ThreeDEffect
- TickColor
- Title
- Unit
- UnitEx
- UnitLabel
- UnitWidth
- UpdateBehaviorName

Eigenschaften

DoubleOutputFormat

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie das Ausgabeformat von Zahlen als double-Wert in der numerischen Skala 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
Eigenschaftswert	System.String	Zeichenfolge, die das Double-Format beschreibt, z.B. "I,DDDD ppm"

Code-Beispiel VB.NET

```
VcGantt1.DoubleOutputFormat = "I,DDDD ppm"
```

Code-Beispiel C#

```
vcGantt1.DoubleOutputFormat = "$I,III.DD";
```

Font

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie alle Schriftattribute der numerischen Skala erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftattribute der numerischen Skala

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale
Dim newFont As Font

histogram = VcGantt1.HistogramCollection.FirstHistogram()
numericScale = histogram.NumericScaleCollection.FirstNumericScale()
newFont = New Font("Times New Roman", 14, FontStyle.Italic)
numericScale.Font = newFont
```

Code-Beispiel C#

```
VcHistogram histogram = vcGantt1.HistogramCollection.FirstHistogram();
VcNumericScale numericScale =
histogram.NumericScaleCollection.FirstNumericScale()
Font newFont = new Font("Times New Roman", 14, FontStyle.Italic);
numericScale.Font = newFont;
```

FontColor

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Schriftfarbe der numerischen Skala erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** RGB (0,0,0)

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.FontColor = Color.Blue
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.FontColor = Color.LightSteelBlue;
```

Histogram

Nur-Lese-Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie erfragen, zu welchem Histogramm eine numerische Skala gehört.

	Datentyp	Beschreibung
Eigenschaftswert	VcHistogram	Histogrammobjekt

Code-Beispiel VB.NET

```
Private Sub VcGantt1_VcNumericScaleLeftDoubleClicking(ByVal sender As
System.Object, ByVal e As NETRONIC.XGantt.VcNumericScaleClickingEventArgs)
Handles VcGantt1.VcNumericScaleLeftDoubleClicking

   MessageBox.Show("Clicked on numeric scale of the histogram " +
e.NumericScale.Histogram.Name)

End Sub
```

Code-Beispiel C#

```
private void vcGantt1_VcNumericScaleLeftDoubleClicking(object sender,
VcNumericScaleClickingEventArgs e)
   {
   MessageBox.Show("Clicked on numeric scale of the histogram " +
e.NumericScale.Histogram.Name);
   }
```

LineColor

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Markierungsfarbe aller numerischer Skalenstreifen von Histogrammen festlegen oder erfragen.

Gesetzt wird die Randlinienfarbe für **alle** numerischen Skalenstreifen, zurückgegeben wird die Linienfarbe des **ersten numerischen Skalaenstreifens>.**

	Datentyp	Beschreibung
Eigenschaftswert	**System.Drawing.Color**	**RGB-Farbwerte** **({0...255},{0...255},{0...255})**

MajorTicks

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Einheiten eine Hauptmarkierung erscheint. Eine Hauptmarkierung ist eine beschriftete Markierung auf der numerischen Achse. Die Anzahl der Einheiten können Sie auch im Dialog Histogramm bearbeiten festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Einheiten zwischen zwei Hauptmarkierungen {1...32767}

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MajorTicks = 4
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.MajorTicks = 4;
```

MajorTicksEx

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Einheiten eine Hauptmarkierung erscheint. Im Vergleich zur Eigenschaft **MajorTicks** können hier auch Fließkomma-Zahlen eingesetzt werden. Die Anzahl der Einheiten können Sie auch im Dialog **Histogramm bearbeiten** festlegen.

S. auch **set/getMinorTicks**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Anzahl der Einheiten zwischen zwei Hauptmarkierungen

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MajorTicks = 4
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.MajorTicks = 4;
```

MinorTicks

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Einheiten eine Nebenmarkierung (ein kleinerer, unbeschrifteter Teilstrich) erscheint. Die Anzahl der Einheiten können Sie im Dialog **Histogramm bearbeiten** festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Einheiten zwischen zwei Nebenmarkierungen {1...32767}

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MinorTicks = 2
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.MinorTicks = 2;
```

MinorTicksEx

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Einheiten eine Nebenmarkierung (ein kleinerer, unbeschrifteter Teilstrich) erscheint. Im Vergleich zur Eigenschaft **MinorTicks** können hier auch Fließkomma-Zahlen eingesetzt werden. Die Anzahl der Einheiten können Sie

im Dialog **Histogramm bearbeiten** festlegen. S. auch **set/getMinorTicksEx** und **set/getMajorTicks**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Anzahl der Einheiten zwischen zwei Nebenmarkierungen

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.MinorTicks = 2
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.MinorTicks = 2;
```

Name

Nur-Lese-Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie den Namen einer numerischen Skala eines Histogramms erfragen. Sie können diesen Namen im Dialog **Histogramm bearbeiten** auswählen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der numerischen Skala

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
MsgBox("Active numeric Scale: " + numericScale.Name)
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
MessageBox.Show("Active numeric scale: " + numericScale.Name);
```

PatternBackgroundColorAsARGB

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Hintergrundfarbe der numerischen Skala erfragen oder festlegen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil (ARGB-Wert) im Zahlenbereich von 0..255. 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.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** -1

PatternColorAsARGB

Eigenschaft von VcNumericScale

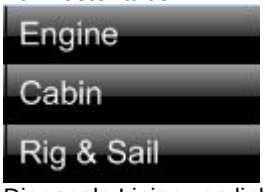
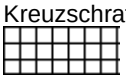
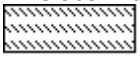
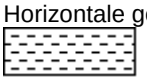
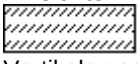
Mit dieser Eigenschaft können Sie die Musterfarbe der numerischen Skala erfragen oder festlegen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil im Zahlenbereich von 0..255 (ARGB-Wert). Ein Alpha-Wert von 0 bedeutet vollständige Transparenz, während der 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.

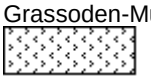
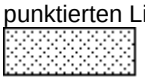
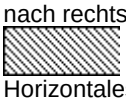
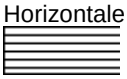
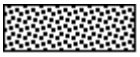
	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** -1

PatternEx

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie für den Hintergrund der numerischen Skala ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte:	
	.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 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 

ThreeDEffect

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die numerische Skala mit 3D-Effekt dargestellt wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	3D-Effekt eingeschaltet (True)/ausgeschaltet (False) **Standardwert:** False

Code-Beispiel VB.NET

```
Dim histogram As VcHistogram
Dim numericScale As VcNumericScale

histogram = VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1")
numericScale = histogram.NumericScaleCollection.Active
numericScale.ThreeDEffect = True
```

Code-Beispiel C#

```
VcHistogram histogram =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1");
VcNumericScale numericScale = histogram.NumericScaleCollection.Active;
numericScale.ThreeDEffect = true;
```

TickColor

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Farbe aller Randlinien der numerischen Skalen von Histogrammen festlegen oder erfragen.

Gesetzt wird die Randlinienfarbe für **alle** numerischen Skalen, zurückgegeben wird die Linienfarbe der **ersten numerischen Skala**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** 0,0,0

Title

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie einen Titel für die numerische Skala erfragen oder festlegen. Der Skalenstreifen, der den Titel zeigt, muss vom Typ **textuell** sein. Skalen und Skalenstreifen werden über den Dialog **Histogramm bearbeiten** angelegt, der von der Eigenschaftenseite **Layout** aus aufrufbar ist.

	Datentyp	Beschreibung
Parameter:		
⇨ position	VcNumericAnnotationPosition **Mögliche Werte:** .vc10PercentFromTop 4 .vc30PercentFromTop 3 .vc50PercentFromTop 2 .vc70PercentFromTop 1 .vc90PercentFromTop 0	Position des Titels der numerischen Skala 10% der Skalenlänge vom oberen Rand entfernt 30% der Skalenlänge vom oberen Rand entfernt 50% der Skalenlänge vom oberen Rand entfernt 70% der Skalenlänge vom oberen Rand entfernt 90% der Skalenlänge vom oberen Rand entfernt
Eigenschaftswert	System.String	Titel der numerischen Skala

Code-Beispiel VB.NET

```
' Title positioned at 50% downward from top
numericScale.Title(VcNumericAnnotationPosition.vc50PercentFromTop) = "1350
Loops"
```

Code-Beispiel C#

```
// Title positioned at 50% downward from top
numericScale.set_Title(VcNumericAnnotationPosition.vc50PercentFromTop, "1350
Loops");
```

Unit

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Einheiten der numerischen Skala erfragen oder festlegen. Die Einheit ist eine ganze Zahl. S. auch **set/getUnit-Width**. Diese Eigenschaft kann auch im Dialog **Histogramm bearbeiten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Einheit

UnitEx

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Grundeinheit der numerischen Skala als double-Wert einstellen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Einheit

Code-Beispiel VB.NET

```
Dim numCol As VcNumericScaleCollection
Dim numScale As VcNumericScale

Set numCol = VcGantt1.HistogramCollection.FirstHistogram.NumericScaleCollection
Set numScale = numCol.FirstNumericScale
numScale.UnitEx = numScale.UnitEx / 2
```

Code-Beispiel C#

```
VcNumericScaleCollection numColl =
vcGantt1.HistogramCollection.FirstHistogram().NumericScaleCollection;
VcNumericScale numScale = numColl.FirstNumericScale();
numScale.UnitEx = numScale.UnitEx / 2;
```

UnitLabel

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Beschriftung der Einheiten der numerischen Skala erfragen oder festlegen. Diese Beschriftung erscheint mittig am oberen Rand der numerischen Skala.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Bezeichnung der Einheit

Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
activeNumericScale = numericScaleCltn.Active
activeNumericScale.UnitLabel = "Hours"
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
VcNumericScale activeNumericScale = numericScaleCltn.Active;
activeNumericScale.UnitLabel = "Hours";
```

UnitWidth

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie die Breite der auf der numerischen Skala verwendeten Einheiten erfragen oder festlegen (in 1/100 mm). Siehe auch **set/getUnit**. Diese Eigenschaft kann auch im Dialog **Histogramm bearbeiten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Breite der Einheiten (1/100 mm)

Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
activeNumericScale = numericScaleCltn.Active
activeNumericScale.UnitWidth = 200
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
VcNumericScale activeNumericScale = numericScaleCltn.Active;
activeNumericScale.UnitWidth = 200;
```

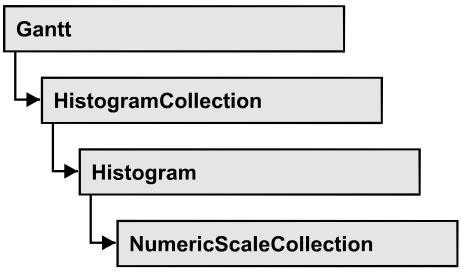
UpdateBehaviorName

Eigenschaft von VcNumericScale

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

7.62 VcNumericScaleCollection



Ein Objekt des Typs VcNumericScaleCollection beinhaltet alle bestehenden numerischen Skalen. Über **For Each numericScale In NumericScale-Collection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle numerischen Skalen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **NumericScaleByName** und **NumericScaleByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Skalen kann über die Eigenschaft **Count** erfragt werden. Die aktuelle numerische Skala kann durch die Eigenschaft **Active** gesetzt oder erfragt werden.

Eigenschaften

- Active
- Count

Methoden

- FirstNumericScale
- GetEnumerator
- NextNumericScale
- NumericScaleByIndex
- NumericScaleByName

Eigenschaften

Active

Eigenschaft von VcNumericScaleCollection

Mit dieser Eigenschaft kann die aktuell für die Darstellung verwendete numerische Skala erfragt oder gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcNumericScale	Aktuell verwendete numerische Skala

Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim activeNumericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
activeNumericScale = numericScaleCltn.Active
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
VcNumericScale activeNumericScale = numericScaleCltn.Active;
```

Count

Eigenschaft von VcNumericScaleCollection

Mit dieser Eigenschaft kann die Anzahl der numerischen Skalen in der NumericScale-Auflistung abgefragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der numerischen Skalen

Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numberOfNumericScales As Integer

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numberOfNumericScales = numericScaleCltn.Count
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
int numberOfNumericScale = numericScaleCltn.Count;
```

Methoden

FirstNumericScale

Methode von VcNumericScaleCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste numerische Skala des NumericScaleCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextNumericScale** über die nachfolgenden numerischen Skalen zu iterieren. Existiert keine numerische Skala im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcNumericScale	Erste numerische Skala

Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numericScale = numericScaleCltn.FirstNumericScale
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
numericScale = numericScaleCltn.FirstNumericScale();
```

GetEnumerator

Methode von VcNumericScaleCollection

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 Numerischen Skalen iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

NextNumericScale

Methode von VcNumericScaleCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden numerischen Skalen des NumericScaleCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstNumericScale** den Initialwert erfasst haben. Sind alle numerischen Skalen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcNumericScale	Nachfolgende numerische Skala

Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection
Dim numericScale As VcNumericScale

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numericScale = numericScaleCltn.FirstNumericScale

While Not numericScale Is Nothing
   ListBox1.Items.Add(numericScale.Name)
   numericScale = numericScaleCltn.NextNumericScale
End While
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
numericScale = numericScaleCltn.FirstNumericScale();
while (numericScale != null)
   {
   listBox1.Items.Add(numericScale.Name);
   numericScale = numericScaleCltn.NextNumericScale();
   }
```

NumericScaleByIndex

Methode von VcNumericScaleCollection

Mit dieser Methode können Sie auf eine einzelne numerische Skala über ihren Index zugreifen. Existiert keine Skala an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index der Numerischen Skala
Rückgabewert	VcNumericScale	Ermitteltes Skalenobjekt

NumericScaleByName

Methode von VcNumericScaleCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf eine bestimmte numerische Skala zugreifen. Existiert keine numerische Skala unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ numericScaleName	System.String	Name der numerischen Skala
Rückgabewert	VcNumericScale	Numerische Skala

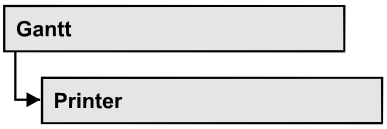
Code-Beispiel VB.NET

```
Dim numericScaleCltn As VcNumericScaleCollection

numericScaleCltn =
VcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on
numericScaleCltn.Active = numericScaleCltn.NumericScaleByName("STEP1")
```

Code-Beispiel C#

```
VcNumericScaleCollection numericScaleCltn =
vcGantt1.HistogramCollection.HistogramByName("HISTOGRAM_1").NumericScaleCollecti
on;
numericScaleCltn.Active = numericScaleCltn.NumericScaleByName("STEP1");
```

7.63 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 Diagram verteilt werden soll, sowie Vergrößerungsfaktor, Druckausrichtung, Hoch- bzw. Querformat, Papierformat und Farbmodus festlegen.

Eigenschaften

- AbsoluteBottomMarginInInches
- AbsoluteLeftMarginInCM
- AbsoluteLeftMarginInInches
- AbsoluteRightMarginInCM
- AbsoluteRightMarginInInches
- AbsoluteTopMarginInCM
- AbsoluteTopMarginInInches
- Alignment
- AllBorderBoxesShownOnCombinedControls
- CombiningControlsEnabled
- CurrentHorizontalPagesCount
- CurrentVerticalPagesCount
- CurrentZoomFactor
- CuttingMarks
- DateFormat
- DefaultPrinterName
- DiagramEnabled
- DiagramEnabled
- DocumentName
- FitToPage
- FoldingMarksType
- MarginsShownInInches
- MaxHorizontalPagesCount
- MaxVerticalPagesCount
- Orientation

- PageDescription
- PageDescriptionString
- PageFrame
- PageNumberMode
- PageNumbers
- PagePaddingEnabled
- PaperSize
- PrintDate
- PrinterName
- PrintPreviewWithFirstPage
- ReOptimizeNodesInGroupsEnabled
- ScalingMode
- TableColumnRanges
- TableTimeScaleOnAllPages
- TableWidthAdoptionFromViewOnScreen
- TimeColumnEndDate
- TimeColumnStartDate
- TimeScaleAdjustment
- 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 **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Code-Beispiel C#

```
vcGantt1.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 **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Double	Absolute Breite des linken Seitenrandes in Zoll **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Double	Breite des rechten Seitenrandes in cm **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Double	Absolute Breite des rechten Seitenrandes in Zoll **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Double	Höhe des oberen Seitenrandes in cm **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Double	Absolute Höhe des oberen Seitenrandes in Zoll **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.Printer.AbsoluteBottomMarginInInches = 0.5   ' 0.5 inches
```

Code-Beispiel C#

```
vcGantt1.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 **TableTimeScaleOnAllPages** eingeschaltet ist. In allen anderen Fällen wird die Gesamtgrafik zentriert ausgerichtet.

	Datentyp	Beschreibung
Eigenschaftswert	VcPrinterAlignment	Ausrichtung des Ausdrucks auf der Seite **Standardwert:** vcPCenterCenter
	Mögliche Werte:	
	.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

```
VcGantt1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft
```

Code-Beispiel C#

```
vcGantt1.Printer.Alignment = VcPrinterAlignment.vcPTopLeft;
```

AllBorderBoxesShownOnCombinedControls

Eigenschaft von VcPrinter

Wenn diese Eigenschaft auf "True" gesetzt wird, werden alle Borderboxen auch beim kombinierten Druck mehrerer XGantt-Diagramme mit ausgedruckt. Ist "False" eingestellt, bleiben die definierten Borderboxen unberücksichtigt. Siehe auch die Objekte **VcBorderArea** und <!VcBorderBox.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Borderboxen werden (True)/werden nicht (False) beim kombinierten Druck berücksichtigt

Code-Beispiel VB.NET

```
VcGantt1.AllBorderBoxesShownOnCombinedControls = True
```

Code-Beispiel C#

```
vcGantt1.AllBorderBoxesShownOnCombinedControls = True;
```

CombiningControlsEnabled

Nur-Lese-Eigenschaft von VcPrinter

Wenn diese Eigenschaft auf **True** gesetzt ist, werden alle XGantt-Steuerelemente von einer Form beim Exportieren sowie beim Drucken und in der Druckvorschau nach ihrer relativen vertikalen Position untereinander angeordnet. Dadurch können mehrere Diagramme auf einmal ausgegeben werden.

Hinweis: Beim kombinierten Druck werden die Eigenschaften **RepeatTableTimeScale** und **TimeScaleAdjustment** nicht berücksichtigt und ihr Wert als "False" angenommen. Auch die Eigenschaft **VcPrinter.FoldingMarksType** wird nicht berücksichtigt und ihr Wert als "vcFMTNone" angenommen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	XGantt-Steuerelemente einer Form werden (True) / werden nicht (False) untereinander angeordnet **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.CombiningControlsEnabled = True
```

Code-Beispiel C#

```
vcGantt1.Printer.CombiningControlsEnabled = true;
```

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 **Current-VerticalPagesCount** 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 Zoomfaktor in Prozent für den Skalierungsmodus **vcFitToPageCount** 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 **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.CuttingMarks = True
```

Code-Beispiel C#

```
vcGantt1.Printer.CuttingMarks = true;
```

DateFormat

Eigenschaft von VcPrinter

Mit dieser Eigenschaft kann das Datumsformat bestimmt werden, das in den DatePicker-Dialogelementen des Dialogs **Seite einrichten** verwendet werden soll. Der leere String steht dabei für das Standard-Datumsformat TS. Folgende Kürzel stehen 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.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.String	Datumsformat für Seite einrichten Dialog **Standardwert:** " "

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

DiagramEnabled

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie festlegen, ob das Diagramm (Zeitskala und Balken) mit ausgedruckt werden soll oder nicht.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Diagramm wird (True) / wird nicht (False) gedruckt **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.Printer.DiagramEnabled = True
```

Code-Beispiel C#

```
vcGantt1.Printer.DiagramEnabled = true;
```

DiagramEnabled

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie festlegen, ob das Diagramm (Zeitskala und Balken) mit ausgedruckt werden soll oder nicht.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Diagramm wird (True) / wird nicht (False) gedruckt **Standardwert:** True

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
Eigenschaftswert	System.String	Dokumentenname **Standardwert:** " "

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
Eigenschaftswert	System.Boolean	Diagramm wird auf eine definierte Anzahl von Seiten verteilt/wird in der voreingestellten Größe ausgegeben.

Code-Beispiel VB.NET

```
VcGantt1.Printer.FitToPage = True
```

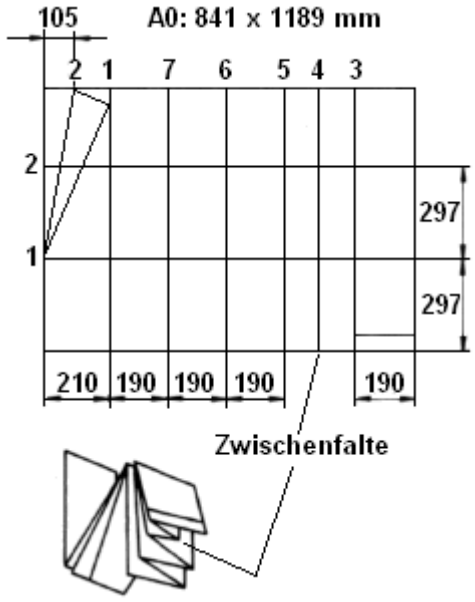
Code-Beispiel C#

```
vcGantt1.Printer.FitToPage = true;
```

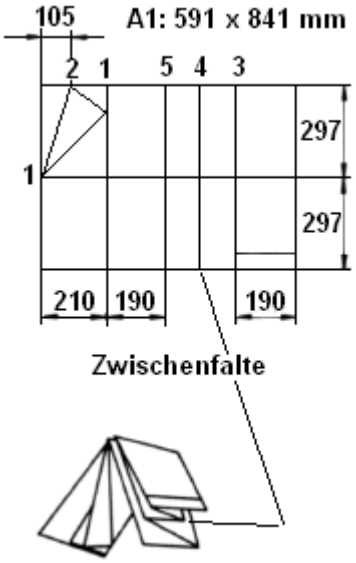
FoldingMarksType

Eigenschaft von VcPrinter

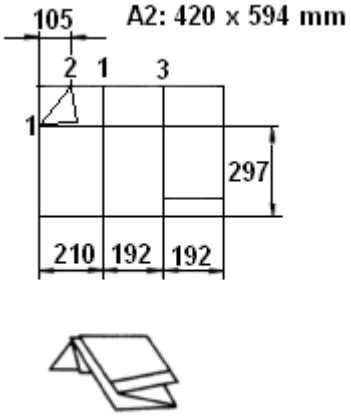
Mit dieser Eigenschaft können Sie folgende Faltmarkierungen nach DIN 824 für den Ausdruck festlegen oder erfragen. Diese ermöglichen das standardisierte Falten für DIN-A-Blattgrößen:



Faltung des DIN-A-0 Formats

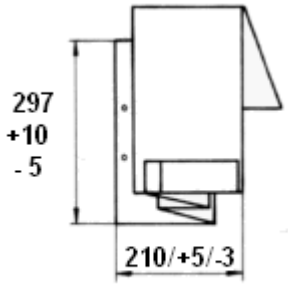
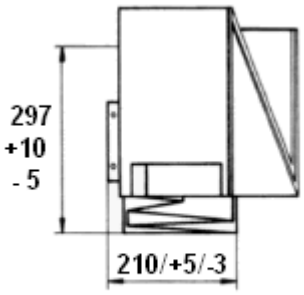
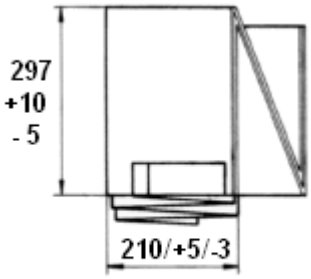


Faltung des DIN-A-1 Formats



Faltung des DIN-A-2 Formats

	Datentyp	**Beschreibung**
Eigenschaftswert	VcFoldingMarksType **Mögliche Werte:**	Faltmarkierungen **Standardwert:** vcFMTNone

.vcFMTDIN824FormA 65	Ausgabe von Faltmarkierungen nach DIN824-A: Die gefaltete Zeichnung kann gelocht und ohne Heftstreifen abgeheftet werden.  **Faltung nach DIN 824-A**
.vcFMTDIN824FormB 66	Ausgabe von Faltmarkierungen nach DIN824-B: Die gefaltete Zeichnung kann gelocht und mittels Heftstreifens abgeheftet werden.  **Faltung nach DIN 824-B**
.vcFMTDIN824FormC 67	Ausgabe von Faltmarkierungen nach DIN824-C: Die gefaltete Zeichnung wird nicht gelocht, sondern in eine Sichthülle gelegt.  **Faltung nach DIN 824-C**
.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 **Seite einrichten** in Zoll (True)/ in cm (False) **Standardwert:** 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 **vcFitToPage-Count** oder **vcZoomWithHorizontalFit** festgelegt haben. S. auch Eigenschaft **MaxVerticalPagesCount**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Maximale Anzahl Seiten in horizontaler Richtung **Standardwert:** 1

Code-Beispiel VB.NET

```
VcGantt1.Printer.MaxHorizontalPagesCount = 4
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Maximale Anzahl Seiten in vertikaler Richtung **Standardwert:** 1

Code-Beispiel VB.NET

```
VcGantt1.Printer.MaxVerticalPagesCount = 4
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	VcOrientation	Ausrichtung **Standardwert:** VcPortrait
	Mögliche Werte: .vcLandscape 42 .vcPortrait 41	 Querformat Hochformat

Code-Beispiel VB.NET

```
VcGantt1.Printer.Orientation = VcOrientation.vcLandscape
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Boolean	Seitenbeschriftung wird (True) / wird nicht gedruckt (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.PageDescription = True
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.String	Seitenbeschriftung **Standardwert:** Empty string ""

Code-Beispiel VB.NET

```
VcGantt1.Printer.PageDescriptionString = "Gantt-Graphics"
```

Code-Beispiel C#

```
vcGantt1.Printer.PageDescriptionString = "Gantt-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
Eigenschaftswert	System.Boolean	Seitenrahmen wird dargestellt (True) / wird nicht dargestellt (False). **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.Printer.PageFrame = True
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	VcPageNumberMode	Art der Seitennumerierung **Standardwert:** vcPRowColumn
	Mögliche Werte: .vcPageNOfM 1597 .vcPRowColumn 1596	 "Seite N von M Seiten" "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
VcGantt1.ShowPrintPreviewDialog()
```

Code-Beispiel C#

```
VcPrinter printer = vcGantt1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PageNumberMode = VcPageNumberMode.vcPageNOfM;
printer.PageNumbers = true;
printer.FitToPage = false;
vcGantt1.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
Eigenschaftswert	System.Boolean	Seitenzahlen werden (True) / werden nicht (False) ausgegeben **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.PageNumbers = True
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Boolean	Zwischenraum zwischen Diagramm und Boxen für Legende/Titel wird (True) / wird nicht (False) ausgegeben **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.Printer.PagePaddingEnabled = True
```

Code-Beispiel C#

```
vcGantt1.Printer.PagePaddingEnabled = true;
```

PaperSize

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie die zu verwendende Papiergröße festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcPaperSize **Mögliche Werte:** .vcDIN_A2 66 .vcDIN_A3 8 .vcDIN_A4 9 .vcISO_C 24 .vcISO_D 25 .vcISO_E 26 .vcUS_LEGAL 5 .vcUS_LETTER 1	Papiergröße DIN A2 DIN A3 DIN A4 ISO C ISO D ISO E US LEGAL US LETTER

Code-Beispiel VB.NET

```
VcGantt1.Printer.PaperSize = VcPaperSize.vcDIN_A3
```

Code-Beispiel C#

```
vcGantt1.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

```
VcGantt1.Printer.PrintDate = True
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	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

VcGantt1.ShowPrintPreviewDialog()
```

Code-Beispiel C#

```
VcPrinter printer = vcGantt1.Printer;
printer.Orientation = VcOrientation.vcLandscape;
printer.PrintPreviewWithFirstPage = true;
printer.FitToPage = false;

vcGantt1.ShowPrintPreviewDialog();
```

ReOptimizeNodesInGroupsEnabled

Eigenschaft von VcPrinter

Wenn die Eigenschaft **TimeScaleAdjustment** aktiviert ist, können hierüber optimiert dargestellte Gruppen für die Druckvorschau bzw. den Druck automatisch neu optimiert werden. Dies ist nur notwendig, wenn einzelne Layer einen außenliegenden Text aufweisen. Die automatische Optimierung ist äusserst zeitaufwändig und kann zu stark verlängerten Reaktionszeiten in der Druckvorschau führen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Bei gleichzeitig aktivierter Eigenschaft **TimeScaleAdjustment**: Optimierte Gruppen werden (True)/werden nicht (False) automatisch für Druck oder Druckvorschau neu optimiert **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.ReOptimizeNodesInGroupsEnabled = True
```

Code-Beispiel C#

```
vcGantt1.Printer.ReOptimizeNodesInGroupsEnabled = true;
```

ScalingMode

Nur-Lese-Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie den Skalierungsmodus beim Drucken setzen oder erfragen. Wenn der Skalierungsmodus auf **vcZoomFactor** gesetzt ist, bestimmt der Wert in der Eigenschaft **ZoomFaktor** die Größe des Ausdrucks. Wenn er auf **vcFitToPageCount** gesetzt ist, sind die Werte in **MaxHorizontalPagesCount** und **MaxVerticalPagesCount** ausschlaggebend. Bei **vcZoomWithHorizontalFit** sind dies **ZoomFactor** und **MaxHorizontalPagesCount**, das heißt ein fester Zoomfaktor mit einer festen Anzahl an Druckseiten in der Breite, die im Druck durch Stauchen oder Strecken der Zeitskala eingehalten wird. Bei den ersten beiden Skalierungsmodi kann das vollständige Füllen der Druckseiten über die Eigenschaft **AdjustTimeScale** erreicht werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcScalingMode	
	Mögliche Werte:	
	.vcFitToPageCount 1	Skalierungsmodus "Anpassung an Seitenzahl"
	.vcZoomFactor 0	Skalierungsmodus "Zoomfaktor".
	.vcZoomWithHorizontalFit 2	Skalierungsmodus "Kombiniertes Anpassen"

TableColumnRanges

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie Spalten der Tabelle auswählen, die im Druck angezeigt werden sollen. Dazu können Sie wie z.B. bei Microsoft Word die anzuzeigenden Spalten einzeln oder in Bereichen durch Komma

oder Semikolon getrennt angeben. Beispiel: "1;5-7;3" selektiert die Spalten 1, 3 und 5 bis 7. Die Angabe "0", ein einfaches Komma oder ein Semikolon führen dazu, dass keine Spalten angezeigt werden. Der Standardwert -1 bewirkt, dass alle Spalten ausgedruckt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Anzahl der zu druckenden Tabellenspalten **Standardwert:** empty string

Code-Beispiel VB.NET

```
VcGantt1.Printer.TableColumnRanges = "1;5-7;3"
```

Code-Beispiel C#

```
vcGantt1.TableColumnRanges = "1;5-7;3";
```

TableTimeScaleOnAllPages

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob Titel, Legende, Tabelle und Zeitskala auf jeder Seite erscheinen sollen (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Titel, Legende, Tabelle und Zeiskala werden auf jeder Seite wiederholt (True)/ Titel, Legende, Tabelle und Zeiskala werden nur einmal ausgegeben und ggf. beim Seitenumbruch durchtrennt (False)

Code-Beispiel VB.NET

```
VcGantt1.Printer.TableTimeScaleOnAllPages = True
```

Code-Beispiel C#

```
vcGantt1.Printer.TableTimeScaleOnAllPages = true;
```

TableWidthAdoptionFromViewOnScreen

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob bei der Druckvorschau und beim Druck die aktuell auf dem Bildschirm sichtbare Breite der Tabelle übernommen wird.

Diese Eigenschaft kann auch zur Laufzeit im Dialog "Seite einrichten" eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Die aktuell auf dem Bildschirm sichtbare Breite der Tabelle wird (True) / wird nicht (False) für Druckvorschau und Druck übernommen **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.TableWidthAdoptionFromViewOnScreen = True
```

TimeColumnEndDate

Eigenschaft von VcPrinter

Mit dieser Eigenschaft kann der Endtermin des zu druckenden Zeitraums festgelegt werden. Dabei ist es nur möglich, den Zeitraum gegenüber dem am Bildschirm gezeigten einzuschränken, d.h. nur ein früheres Enddatum als das über die Eigenschaft **VcGantt.TimeScaleEnd** gesetzte führt zu einer Änderung des Drucks.

Diese Eigenschaft kann auch zur Laufzeit im Dialog **Seite einrichten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Enddatum des Zeitraumes für den Ausdruck **Standardwert:** System.DateTime.MaxValue

TimeColumnStartDate

Eigenschaft von VcPrinter

Mit dieser Eigenschaft kann der Starttermin des zu druckenden Zeitraums festgelegt oder erfragt werden. Dabei ist es nur möglich, den Zeitraum gegenüber dem am Bildschirm gezeigten einzuschränken, d.h. nur ein späteres Startdatum als das über die Eigenschaft **VcGantt.TimeScaleStart** gesetzte führt zu einer Änderung des Ausdrucks.

Diese Eigenschaft kann auch zur Laufzeit im Dialog **Seite einrichten** eingestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Startdatum des Zeitraumes für den Ausdruck **Standardwert:** System.DateTime.MinValue

TimeScaleAdjustment

Eigenschaft von VcPrinter

Mit dieser Eigenschaft kann der Platz auf den Druckseiten besser ausgenutzt werden:

- Bei einer Skalierung über eine feste Anzahl Seiten: Der Zoomfaktor wird so berechnet, dass die eingestellte Anzahl Seiten in der Höhe voll bedruckt werden. Gleichzeitig wird die Zeitskala so gestaucht oder gestreckt, dass die eingestellte Anzahl Seiten in der Breite voll ausgenutzt wird.

- Bei einer Skalierung über einen Zoomfaktor: Die Zeitskala wird so gestaucht oder gestreckt, dass die eingestellte Anzahl Seiten in der Breite voll ausgenutzt wird.

	Datentyp	Beschreibung
Parameter: ⇔ Rückgabewert	System.Boolean	Zeitskala wird angepasst
Eigenschaftswert	SystemInt.32	Zeitskala wird angepasst **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.Printer.TimeScaleAdjustment = True
```

Code-Beispiel C#

```
vcGantt1.TimeScaleAdjustment = 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

```
VcGantt1.Printer.AbsoluteTopMarginInCM = 1.5   ' 2 cm
```

Code-Beispiel C#

```
vcGantt1.Printer.AbsoluteTopMarginInCM = 1.5;   // 1.5 cm
```

ZoomFactorAsDouble

Eigenschaft von VcPrinter

Mit dieser Eigenschaft können Sie den Zoomfaktor in Prozent für den Skalierungsmodus **vcZoomFactor** bzw. **vcZoomWithHorizontalFit** 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

```
VcGantt1.Printer.ZoomFactor = 150
```

Code-Beispiel C#

```
vcGantt1.Printer.ZoomFactor = 150;
```

7.64 VcRect

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 VcGantt1_VcInPlaceEditorShowing(ByVal sender As Object, ByVal e
As NETRONIC.XGantt.VcInPlaceEditorShowingEventArgs) Handles
VcGantt1.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 + VcGantt1.Left
                    TextBox1.Top = e.FldRectVisible.Top + VcGantt1.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 + VcGantt1.Left
                    DateTimePicker1.Top = e.FldRectVisible.Top + VcGantt1.Top
                    DateTimePicker1.Value = node.DataField(0)
                    DateTimePicker1.Visible = True
                    DateTimePicker1.Focus()
                Case 13     'Employee
                    ComboBox1.Left = e.FldRectVisible.Left + VcGantt1.Left
                    ComboBox1.Top = e.FldRectVisible.Top + VcGantt1.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 vcGantt1_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 + vcGantt1.Left;
           textBox1.Top = e.FldRectVisible.Top + vcGantt1.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 + vcGantt1.Left;
           dateTimePicker1.Top = e.FldRectVisible.Top + vcGantt1.Top;
           dateTimePicker1.Value = Convert.ToDateTime(node.get_DataField(0));
           dateTimePicker1.Visible = true;
           dateTimePicker1.Focus();
            break;
         case 13: //Employee
           comboBox1.Left = e.FldRectVisible.Left + vcGantt1.Left;
            comboBox1.Top = e.FldRectVisible.Top + vcGantt1.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
Eigenschaftswert	System.Int32	Position des rechten Rands des Rechtecks

Top

Eigenschaft von VcRect

Diese Eigenschaft gibt die obere Position des Rechteckobjekts an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Position des oberen Rands des Rechtecks

Code-Beispiel VB.NET

```
DateTimePicker1.Top = e.FldRectVisible.Top + VcGantt1.Top
```

Code-Beispiel C#

```
dateTimePicker1.Top = e.FldRectVisible.Top + vcGantt1.Top;
```

Width

Nur-Lese-Eigenschaft von VcRect

Diese Eigenschaft gibt die Breite des Rechteckobjekts an.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Breite des Rechtecks

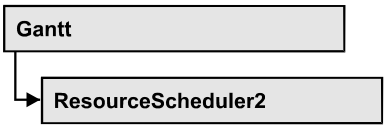
Code-Beispiel VB.NET

```
Text1.Width = fldRectVisible.Width
```

Code-Beispiel C#

```
textBox1.Width = e.FldRectVisible.Width;
```

7.65 VcResourceScheduler2



Der neue ResourceScheduler2 stellt eine wesentliche Erweiterung des ResourceScheduler1 aus Version 3.1 dar. Es werden die für eine Ressourcenplanung benötigten Objektarten nun in eigenen Datentabellen erwartet, was erst mit VARCHART XGantt 4.0 möglich wird. Im ResourceScheduler1 waren dagegen Aufgaben, Operationen, Zuweisungen und Ressourcen implizit in der Maindata-Tabelle zu definieren.

Zur Verfügung stehen die folgenden Objektarten, die jeweils in einer eigenen Datentabelle erwartet werden (bei Ressourcen können es bis zu 25 sein):

- **Aufgaben** (Tasks): Diese Objekte bilden eine Gruppierung von Operationen (siehe unten) und halten elementare Eigenschaften wie das Freigabe- und das Fälligkeitsdatum sowie Prioriät und Aufgabenmenge.
- **Operationen**: Diese Objekte sind über Zuweisungen (siehe unten) auf Ressourcen (siehe unten) einplanbar und erhalten durch die Ressourcenplanung als Ergebnis den Ausführungszeitraum. Operationen besitzen eine definierte Reihenfolge in der zugehörigen Aufgabe und können als bereits begonnen gekennzeichnet werden. Außerdem ist es möglich, mehrere einander ausschließende Operationsfolgen (sogenannte Routen) als Alternativen vorzugeben. Alle Operationen der von der Ressourcenplanung gewählten Route werden gemeinsam eingeplant.
- **Ressourcen**: Diese Objekte besitzen als Hauptmerkmal eine Kapazitätskurve und nach der Ausführung einer Ressourcenplanung auch eine Belegungskurve. Außerdem können Ressourcen für eine auf ihr eingeplanten Operation zeitdauerbestimmend sein, wovon augenblicklich einer Operation immer eine zugewiesen sein muss, um die Operation einplanen zu können. Daneben können einer Operation beliebig viele weitere Arbeits- und Material-Ressourcen zugewiesen sein. Weiteres Wesensmerkmal einer zeitdauerbestimmenden Ressource ist die Fähigkeit zur ein- oder mehrstufigen Gruppierung, wobei eine solche Ressource auch mehreren Gruppen gleichzeitig angehören kann.
- **Zuweisungen**: Diese Objekte sind die eigentliche Verknüpfung zwischen Operationen und Ressourcen, wobei hier ein Übersetzungsfaktor für die Aufgabenmenge angegeben werden kann. Im Falle von zeitdauerbestimmenden Ressourcengruppen werden Zuweisungen von der Ressourcenplanung entsprechend als solche gekennzeichnet und konkrete Ersatz-

zuweisungen zu der ausgewählten Einzelressource erzeugt, so dass die Einplanung nachvollziehbar und im XGantt darstellbar wird.

- **Verbindungen**: Diese Objekte beschreiben die Reihenfolge von Aufgaben, das heißt, vorangehende Aufgaben müssen beendet sein, bevor nachfolgende Aufgaben begonnen werden dürfen.

Eigenschaften

- AssignmentDataTableName
- AssignmentIsResultFieldIndex
- AssignmentIsVisibleFieldIndex
- AssignmentLoadOrConsumptionPerItemFieldIndex
- AssignmentMaximumLoadFieldIndex
- AssignmentMinimumLoadFieldIndex
- AssignmentMinimumMaximumLoadType
- AssignmentOperationIDFieldIndex
- AssignmentResourceIDFieldIndex
- AssignmentResourceSelectionStrategyFieldIndex
- BaseCalendarUsageForSupplementTimes
- BaseTimeUnit
- BaseTimeUnitsPerStep
- DataRecordEventsEnabled
- DefaultOperationMaximumInterruptionTime
- DefaultResourceCalendarName
- FullUsageOfPlanningUnitsEnabled
- LinkDataTableName
- LinkDurationFieldIndex
- LinkPredecessorOperationIDFieldIndex
- LinkPredecessorTaskIDFieldIndex
- LinkSuccessorOperationIDFieldIndex
- LinkSuccessorTaskIDFieldIndex
- OperationDataTableName
- OperationLoadPerItemFieldIndex
- OperationMaximumInterruptionTimeFieldIndex
- OperationMinimumSupplementTimeFieldIndex
- OperationOverlapQuantityFieldIndex
- OperationPostLoadFieldIndex
- OperationPostOffsetFieldIndex
- OperationPreparationLoadFieldIndex
- OperationPreparationOffsetFieldIndex

- OperationResultEndDateFieldIndex
- OperationResultPostEndDateFieldIndex
- OperationResultPreparationStartDateFieldIndex
- OperationResultProcessingTimeFieldIndex
- OperationResultSelectedTimingResourceIDFieldIndex
- OperationResultStartDateFieldIndex
- OperationResultStatusFieldIndex
- OperationRouteFieldIndex
- OperationSequenceNumberFieldIndex
- OperationStartLockDateFieldIndex
- OperationTaskIDFieldIndex
- OperationWorkInProcessFieldIndex
- PlanningEndDate
- PlanningStartDate
- PlanningStrategy
- ResourceCalendarNameFieldIndex
- ResourceCapacityType
- ResourceCapacityTypeFieldIndex
- ResourceConstraintTypeFieldIndex
- ResourceDataTableName
- ResourceEfficiencyFieldIndex
- ResourceGroupDataTableName
- ResourceGroupIDFieldIndex
- ResourceNameFieldIndex
- ResourceResultLoadCurveNamePrefix
- ResourceResultStockCurveNamePrefix
- ResourceSelectionStrategy
- ResourceType
- ResultProcessingStepCount
- TaskDataTableName
- TaskDueDateFieldIndex
- TaskPlanningStrategyFieldIndex
- TaskPriorityFieldIndex
- TaskQuantityFieldIndex
- TaskReleaseDateFieldIndex
- TaskResultEndDateFieldIndex
- TaskResultPostEndDateFieldIndex
- TaskResultPreparationStartDateFieldIndex
- TaskResultProcessingStepFieldIndex
- TaskResultProcessingTimeFieldIndex

- TaskResultRouteFieldIndex
- TaskResultStartDateFieldIndex
- ToleranceTimeOnASAPDueDates
- ToleranceTimeOnJITReleaseDates
- ToleranceTimeOnStartLockDates
- WorkInProcessType
- WritingDebugFilesEnabled

Methoden

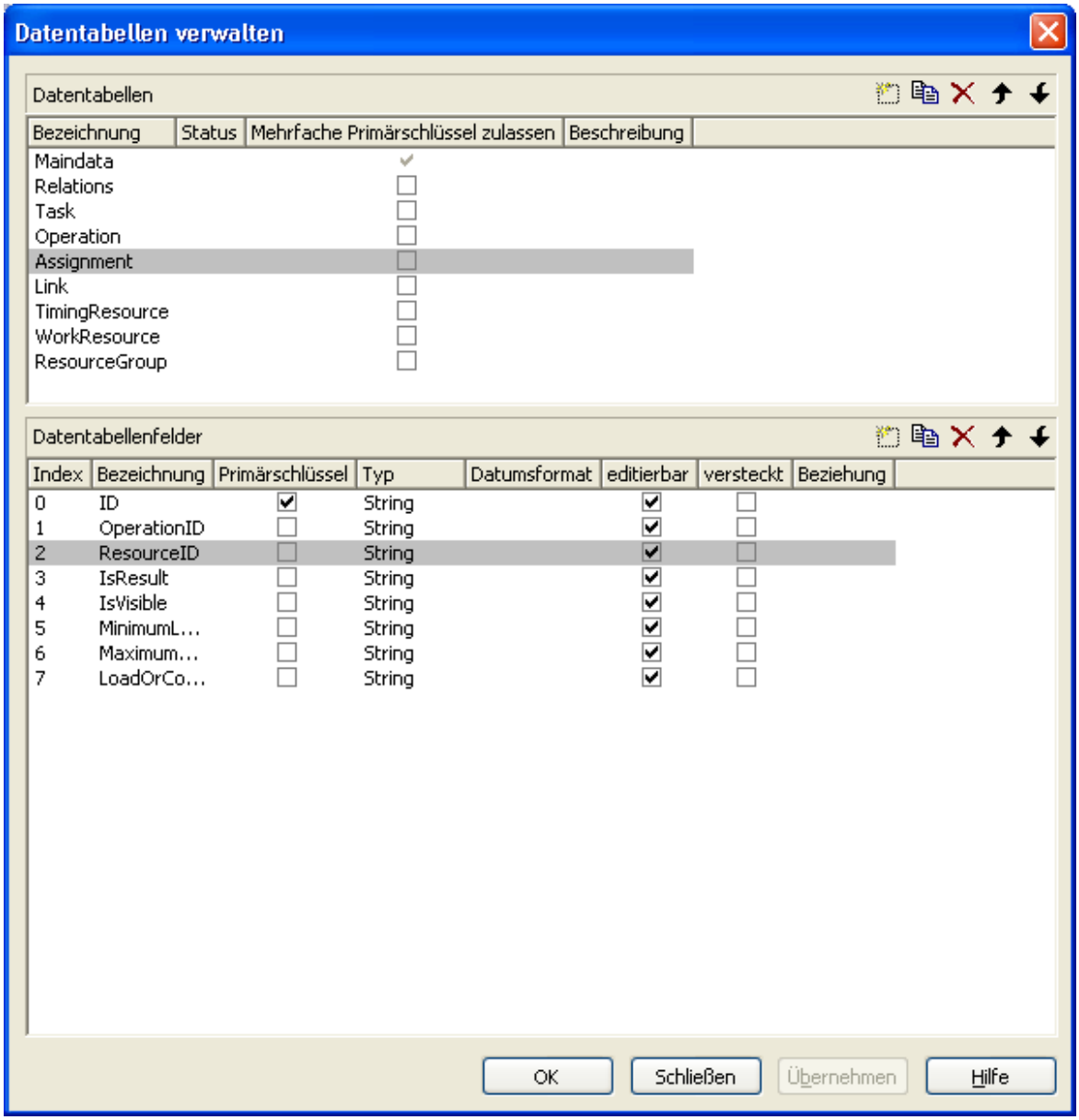
- DetermineIDOfFirstOperationByTaskID
- DetermineIDOfLastOperationByTaskID
- Process

Eigenschaften

AssignmentDataTableName

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können sie den Namen der Zuweisungstabelle setzen oder erfragen. Sie ist die Datentabelle für die Zuweisungen von Operationen zu Ressourcen, deren Name obligat zu setzen ist.



	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zuweisungsdatentabelle **Standardwert:** Empty string

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentDataTableName = "Assignment"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentDataTableName("Assignment");
```

AssignmentIsResultFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in der Zuweisungstabelle setzen oder erfragen, in welches VARCHART XGantt einträgt, ob der Datensatz als Ergebnis der Ressourcenplanung erzeugt wurde. In der Abbildung zu **AssignmentDataTableName** ist der Index beispielsweise 3. Das Setzen dieser Eigenschaft ist optional. Die Ressourcenplanung erzeugt Zuweisungen nur, wenn in den vorgegebenen Zuweisungen beim Start Zuweisungen auf Ressourcengruppen vorhanden sind. Bei Gruppen wird eine Zuweisung auf die aus der Gruppe ermittelte Ressource generiert und bei dieser das entsprechende Feld auf den Wert 1 gesetzt. In alle von der Anwendung vorgegebenen Zuweisungen sollte in diesem Feld nichts oder den Wert 0 eingetragen sein.

Die Benutzung eines solchen Feldes ermöglicht das mehrmalige Aufrufen mit dem jeweils gleichen Ergebnis, was der Anwendung das manuelle Zurückändern der Zuweisungen auf den Ursprungszustand erspart. Die Ressourcenplanung verwendet erzeugte Zuweisungen immer wieder, um unnötiges Löschen und Erzeugen von Datensätzen zu vermeiden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme der Werte zur Kennzeichnung der von der Ressourcenplanung erzeugten Datensätze vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentIsResultFieldIndex = 3
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentIsResultFieldIndex = 3;
```

AssignmentIsVisibleFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft setzen oder erfragen Sie den Index eines Datenfeldes der Zuweisungstabelle, in welches die Ressourcenplanung einträgt, ob die Zuweisung im Gantt-Graphen sichtbar gemacht werden sollten. In der

Abbildung zu **AssignmentDataTableName** ist der Index beispielsweise 4. Das Feld ist nützlich, wenn Sie z. B. vor dem Planungsdurchlauf im Gantt-Graphen die Zuweisungen auf die Ressourcengruppen anzeigen möchten und nach dem Durchlauf die Zuweisungen auf die ermittelten Einzelressourcen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme der Werte zur Sichtbarkeit vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentIsVisibleFieldIndex = 4
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentIsVisibleFieldIndex = 4;
```

AssignmentLoadOrConsumptionPerItemFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft setzen oder erfragen Sie den Index eines Datenfeldes der Zuweisungstabelle, in dem ein Wert pro Stück (s. Eigenschaft **TaskQuantityFieldIndex**) festgelegt werden kann. Ein Wert kann nur für Zuweisungen an Arbeits- oder Materialressourcen gesetzt werden. Enthält der Index den Wert -1, wird 1 angenommen. Steht im Datenfeld des Datensatzes kein gültiger Wert, wird 0 angenommen. Wenn das Datenfeld vom Typ **String** ist, können Sie im Datenfeld auch eine Fließkommazahl angeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme des Wertes vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentLoadOrConsumptionPerItemFieldIndex = 7
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentLoadOrConsumptionPerItemFieldIndex = 7;
```

AssignmentMaximumLoadFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Zuweisungstabelle, das für den maximalen Auslastungsgrenzwert einer Ressource vorgesehen ist. In der Abbildung zu **AssignmentDataTableName** ist der Index beispielsweise 6.

Ein derartiger Höchstwert kann nur für Zuweisungen an zeitdauerbestimmende Ressourcen gesetzt werden. Das Datenfeld selbst enthält relative Werte in % von {0...100}, wobei ein leeres Feld sowie der Wert 0 ebenfalls als 100 interpretiert werden.

Werte zwischen 1 und 99 im Datenfeld setzen die Eigenschaften **FullUsageOfPlanningUnitsEnabled** und **OperationMaximumInterruptionTimeFieldIndex** außer Kraft.

Beachten Sie bitte auch **AssignmentMinimumLoadFieldIndex**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme des Höchstwertes zur Begrenzung der Ressourcenauslastung vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentMaximumLoadFieldIndex = 6
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentMaximumLoadFieldIndex = 6;
```

AssignmentMinimumLoadFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Zuweisungstabelle, das für den minimalen Auslastungsgrenzwert einer Ressource vorgesehen ist. In der Abbildung zu **AssignmentDataTableName** ist der Index beispielsweise 5.

Nur zeitdauerbestimmenden Ressourcen kann ein derartiger Tiefstwert zugewiesen werden. Das Datenfeld selbst enthält relative Werte in % von {0...100}. Beachten Sie bitte auch **AssignmentMaximumLoadFieldIndex**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme des Mindestwertes zur Begrenzung der Ressourcenauslastung vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentMinimumLoadFieldIndex = 5
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentMinimumLoadFieldIndex = 5;
```

AssignmentMinimumMaximumLoadType

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie bestimmen oder erfragen, ob die Werte, die in den Datenfeldern mit den über die Eigenschaften **AssignmentMinimumLoadFieldIndex** und **AssignmentMaximumLoadFieldIndex** gesetzten Indizes bereitgestellt werden, relativ zur Ressourcenkapazität sind oder absolut. Absolute Werte sind z. B. hilfreich, wenn die zugewiesene Ressource ein Team mit einer nicht konstanten Anzahl Personen ist und die Zuweisung nicht das ganze Team belegen soll.

	Datentyp	Beschreibung
Eigenschaftswert	VcResourceSchedulingMinimumMaximumLoadType	Feldwerte absolut/relativ zur Ressourcenkapazität **Standardwert:** vcResSchedPercentageValues
	Mögliche Werte: .vcResSchedAbsoluteValues 2 .vcResSchedPercentageValues 0	 Datenfeldwerte absolut zur Ressourcenkapazität Datenfeldwerte relativ zur Ressourcenkapazität

AssignmentOperationIDFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index des Datenfeldes in der Zuweisungstabelle setzen oder erfragen, das die ID einer Operation enthält. In der Abbildung zu **AssignmentDataTableName** ist der Index beispielsweise 1. Diese Eigenschaft muss auf einen Wert ungleich -1 gesetzt werden, bevor die Methode **Process** aufgerufen wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme der Operation-ID vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentOperationIDFieldIndex = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.AssignmentOperationIDFieldIndex = 1;
```

AssignmentResourceIDFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser indizierten Eigenschaft können Sie den Index eines Datenfeldes in der Zuweisungstabelle setzen oder erfragen, das IDs für Ressourcen enthält. In der Abbildung zu **AssignmentDataTableName** ist der Index beispielsweise 2.

Der als Parameter übergebene Ressourcentabellenindex bezeichnet eine von 25 möglichen Ressourcentabellen. Die davon verwendeten Tabellen werden über die indizierte Eigenschaft **ResourceDataTableName** definiert.

	Datentyp	Beschreibung
Parameter:		
⇨ resourceTableIndex	System.Int16	Index einer Ressourcendatentabelle entsprechend der Zuweisung durch die Eigenschaft **ResourceDataTableName** {0...24}
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdatentabelle, das für die Aufnahme von Ressourcen-IDs vorgesehen ist. {-1...AnzahlDatenfelderInZuweisungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Zuweisungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.AssignmentResourceIDFieldIndex(0) = 2
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_AssignmentResourceIDFieldIndex(0,2);
```

AssignmentResourceSelectionStrategyFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Zuweisungstabelle, das für die jeweilige Zuweisung zu einer Ressourcengruppe eine eigene Ressourcenauswahlstrategie bestimmt. Ist dieses Datenfeld an einer Ressource leer oder ist die Eigenschaft auf -1 gesetzt, dann gilt defaultmäßig der Wert der allgemeinen Eigenschaft **ResourceSelectionStrategy** (siehe dort).

Das Datenfeld kann folgende Werte enthalten:

0: enspricht vcResSchedRSSequential

1: enspricht vcResSchedRSLeastLoaded

2: enspricht vcResSchedRSMostLoaded

3: enspricht vcResSchedRSHighestEfficiency

7: enspricht vcResSchedRSFirstAvailable

Bei den Werten 1 und 2 (LeastLoaded bzw. MostLoaded) wird bei der Einplanung der Ressourcen eine laufende Summe der Belegungen aufaddiert, anhand derer die Entscheidung für die mindest- bzw. meistbelegte Ressource getroffen wird. Daher ist es nicht ausgeschlossen, dass bei stark differierenden Planungzeiträumen von Aufgaben oder bei Nutzung beider Planungsstrategien das Ergebnis nicht zufriedenstellend ausfällt.

Bei dem Wert 7 (FirstAvailable) ist die Entscheidung nur von der zuerst verfügbaren zeitdauergebenden Ressource abhängig. Weitere Zuweisungen der gleichen Operation werden nicht berücksichtigt, so dass bei Verwendung von Material- und Arbeitsressourcen ein nicht zufriedenstellendes Ergebnis entstehen kann.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Zuweisungsdaten-tabelle, das für die Aufnahme der Daten zur Planungsstrategie vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

BaseCalendarUsageForSupplementTimes

Eigenschaft von VcResourceScheduler2

Wenn diese Eigenschaft auf **false** gesetzt ist, wird für die Mindest-Ergänzungszeiten (die indirekt über die Eigenschaft **VcResource-Scheduler2.OperationMinimumSupplementTimeFieldIndex** definiert werden) kein Kalender verwendet, d.h. die angegebene Zeit wird direkt verwendet (Praxisbeispiel: geeignet für die Trocknung produzierter Teile). Wenn diese Eigenschaft auf **true** gesetzt ist, wird der Basiskalender des Gantt-Objektes verwendet, d.h. die Ergänzungszeit wird als Arbeitszeit abgearbeitet, die im Basiskalender definiert wurde (Praxisbeispiel: geeignet für einen Transport produzierter Teile).

Bitte sehen Sie ergänzend auch **VcResourceScheduler2.Operation-MinimumSupplementTimeFieldIndex**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**true**: Der Basiskalender des Gantt-Objekts wird verwendet. **false**: Die angegebene Zeit wird direkt verwendet. **Standardwert:** false

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.BaseCalendarUsageForSupplementTimes = True
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.BaseCalendarUsageForSupplementTimes = true;
```

BaseTimeUnit

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft könnenSie eine von der Eigenschaft **VcGantt.TimeUnit** abweichende Basiszeiteinheit für die Ressourcenplanung festlegen oder erfragen. Werte in den Kapazitäts-, Belegungs- und Lagerbestandskurven sind immer auf die hier angegebene Basiszeiteinheit bezogen.

	Datentyp	Beschreibung
Eigenschaftswert	VcTimeUnit	Zeiteinheit **Standardwert:** Value, which was set during design time by vcGantt.TimeUnit. If no setting was made, the value is **vcDay**.
	Mögliche Werte: .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	 Zeiteinheit **Tag** Zeiteinheit **Stunde** Zeiteinheit **Minute** Zeiteinheit **Sekunde**

Code-Beispiel VB.NET

```
VcResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute
VcResourceScheduler2.BaseTimeUnitsPerStep = 15
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute;
vcGantt1.ResourceScheduler2.BaseTimeUnitsPerStep = 15;
```

BaseTimeUnitsPerStep

Nur-Lese-Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie die Schrittweite für die Ressourcenplanung festlegen oder erfragen. Je höher dieser Wert ist, umso schneller, aber auch umso grober wird die Berechnung. Die Zahl bezeichnet ein Vielfaches der über **VcResourceScheduler2.BaseTimeUnit** festgelegten Grundeinheit.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Zeiteinheiten pro Schritt **Standardwert:** 1

Code-Beispiel VB.NET

```
VcResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute
VcResourceScheduler2.BaseTimeUnitsPerStep = 30
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.BaseTimeUnit = VcTimeUnit.vcMinute;
vcGantt1.ResourceScheduler2.BaseTimeUnitsPerStep = 30;
```

DataRecordEventsEnabled

Eigenschaft von VcResourceScheduler2

Wenn diese Eigenschaft auf **true** gesetzt wird, dann werden die Ereignisse aufgerufen, die Datenveränderungen während der Process-Methode anzeigen, d.h. DataRecordModifying, DataRecordModified, DataRecordCreating, DataRecordCreated, DataRecordDeleting und DataRecordDeleted.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**true**: Ereignisse werden ausgelöst. **false**: Ereignisse werden nicht ausgelöst. **Standardwert:** false

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.DataRecordEventsEnabled = True
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.DataRecordEventsEnabled = true;
```

DefaultOperationMaximumInterruptionTime

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie einen Wert für die maximale Zeitdauer einer Unterbrechung der Verarbeitung von Operationen setzen oder erfragen. Der Wert ist eine Zahl, die in Basiszeiteinheiten angegeben wird (s. Eigenschaft **BaseTimeUnit**). Sie wird wirksam, wenn die Eigenschaft **OperationMaximumInterruptionTimeFieldIndex** auf -1 gesetzt ist oder der aus einem Datensatz gelesene Wert der Operationstabelle 0 ist oder das Feld leer ist. Wird der Wert auf 0 gesetzt, ist keine Unterbrechung erlaubt.

Diese Eigenschaft wird durch die Herabsetzung der maximalen Auslastung auf weniger als 100% (s. Eigenschaft **AssignmentMaximumLoadField-Index**) außer Kraft gesetzt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Basiszeiteinheiten **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.DefaultOperationMaximumInterruptionTime = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.DefaultOperationMaximumInterruptionTime = 1;
```

DefaultResourceCalendarName

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Namen für einen Standardkalender vorgeben, der verwendet wird, wenn über die Eigenschaften **VcResource-Scheduler2.ResourceCalendarNameFieldIndex** und **VcResource-Scheduler2.ResourceNameFieldIndex** kein ressourcenspezifischer Kalender gefunden wurde. Wird die Eigenschaft nicht gesetzt, greift die Ressource auf den Standard-Kalender des XGantt-Objektes zurück (siehe **VcCalendarCollection.Active**).

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Kalenders **Standardwert:** Empty String

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.DefaultResourceCalendarName = ""
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.DefaultResourceCalendarName = "";
```

FullUsageOfPlanningUnitsEnabled

Eigenschaft von VcResourceScheduler2

Wenn diese Eigenschaft auf **True** gesetzt wird, dann kann in der ersten und/oder letzten Planungszeiteinheit der Belegung einer Ressource durch eine Aufgabe eine weitere Aufgabe enden bzw. beginnen. Auf diese Weise können vorhandene Restkapazitäten genutzt werden. Bei **False** werden Restkapazitäten dagegen nicht genutzt.

Die Eigenschaft beeinflusst nur den Beginn der ersten Operation einer Aufgabe. Auf die Operationen innerhalb einer Aufgabe hat diese Eigenschaft keine Auswirkung.

Diese Eigenschaft wird durch die Herabsetzung der maximalen Auslastung auf weniger als 100% (s. Eigenschaft **AssignmentMaximumLoadFieldIndex**) außer Kraft gesetzt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**true**: Vorhandene Restkapazitäten werden genutzt. **false**: Vorhandene Restkapazitäten werden nicht genutzt. **Standardwert:** true

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.FullUsageOfPlanningUnitsEnabled = True
```

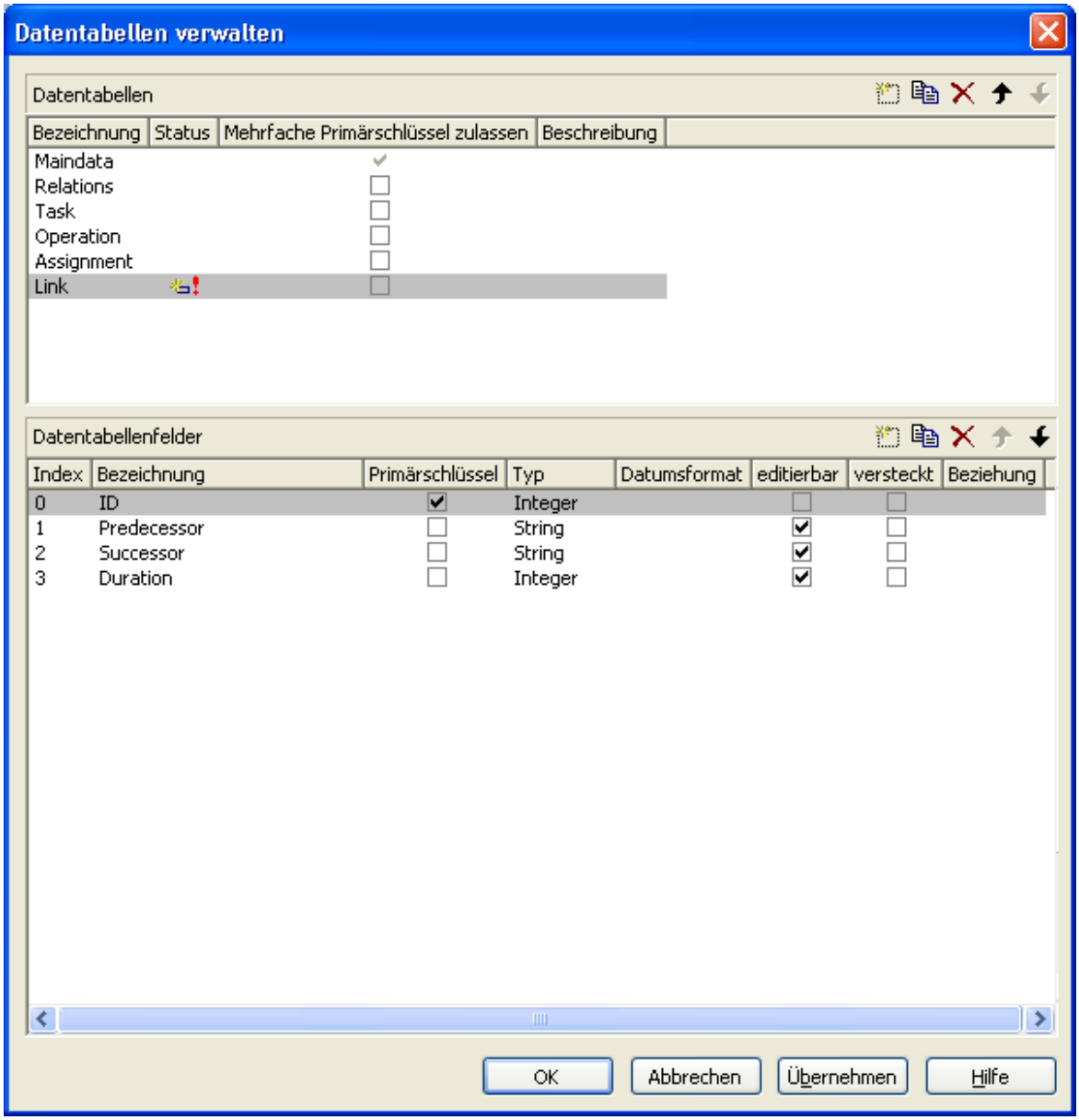
Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.FullUsageOfPlanningUnitsEnabled = true;
```

LinkDataTableName

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Namen der Verbindungsdatentabelle setzen oder erfragen. Wenn Sie diesen Namen nicht setzen, werden Verbindungen bei der Ressourcenplanung nicht berücksichtigt.



	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Verbindungstabelle **Standardwert:** Empty string

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.NodeDataTableName = "Node"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.LinkDataTableName("Link");
```

LinkDurationFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index des Datenfeldes in der Verbindungstabelle setzen oder erfragen, in dem ein minimaler zeitlicher Abstand zwischen Vorgänger und Nachfolger abgelegt werden kann. Dieser zeitliche Abstand kann auch negativ sein. Einheiten: Basiszeiteinheiten (s. Methode BaseTimeUnit). In der Abbildung zu **LinkDataTableName** ist der Index beispielsweise 3.

Als Begrenzung kann der Nachfolger bei Planungsstrategie ASAP nicht früher anfangen als der Vorgänger und bei Planungsstrategie JIT der Vorgänger nicht später aufhören als der Nachfolger.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Verbindungsdatentabelle, das für die Aufnahme der Dauer vorgesehen ist. {-1...AnzahlDatenfelderInVerbindungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Verbindungsdatentabelle zugeordnet ist. **Standardwert:** -1

LinkPredecessorOperationIDFieldIndex

Nur-Lese-Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in der Verbindungstabelle setzen oder erfragen, das die ID der Vorgängeroperation enthält. Da das ResourceScheduling-Modul nur Aufgaben (Tasks) miteinander verbinden kann, erleichtert diese Eigenschaft die Verwendung der Verbindungen in XGantt, die zur Zeit nur zwischen Operationen dargestellt werden. Intern wird daher immer eine Verbindung zwischen den Tasks der durch die ID angegebenen Operationen erzeugt.

Bei Verwendung einer Verbindungstabelle ist diese Eigenschaft verbindlich auf einen Wert ungleich -1 zu setzen, wenn nicht die VcResourceScheduler2-Eigenschaft **LinkPredecessorTaskIDFieldIndex** verwendet wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Verbindungsdaten-tabelle, das für die Aufnahme von IDs von Vorgängeroperationen vorgesehen ist. {-1...AnzahlDatenfelderInVerbindungstabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Verbindungstabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.LinkPredecessorOperationIDFieldIndex = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.LinkPredecessorOperationIDFieldIndex = 1;
```

LinkPredecessorTaskIDFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in der Verbindungstabelle setzen oder erfragen, das die ID der Vorgänger-Aufgabe enthält. In der Abbildung zu **LinkDataTableName** ist der Index beispielsweise 1.

Bei Verwendung einer Verbindungstabelle ist diese Eigenschaft verbindlich auf einen Wert ungleich -1 zu setzen, wenn nicht die VcResourceScheduler2-Eigenschaft **LinkPredecessorOperationIDFieldIndex** verwendet wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Verbindungsdaten-tabelle, das für die Aufnahme von IDs von Vorgängeraufgaben vorgesehen ist. {-1...AnzahlDatenfelderInVerbindungsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Verbindungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.NodePredecessorTaskIDFieldIndex = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.LinkPredecessorTaskIDFieldIndex = 1;
```

LinkSuccessorOperationIDFieldIndex

Nur-Lese-Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in der Verbindungstabelle setzen oder erfragen, das die ID der Nachfolgeroperation enthält. Da das ResourceScheduling-Modul nur Aufgaben (Tasks) miteinander verbinden kann, erleichtert diese Eigenschaft die Verwendung der Verbindungen in XGantt, die zur Zeit nur zwischen Operationen dargestellt werden. Intern wird daher immer eine Verbindung zwischen den Tasks der durch die ID angegebenen Operationen erzeugt.

Bei Verwendung einer Verbindungstabelle ist diese Eigenschaft verbindlich auf einen Wert ungleich -1 zu setzen, wenn nicht die VcResourceScheduler2-Eigenschaft **LinkSuccessorTaskIDFieldIndex** verwendet wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Verbindungsdaten-tabelle, das für die Aufnahme von IDs von Nachfolgeoperationen vorgesehen ist. {-1...AnzahlDatenfelderInVerbindungstabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Verbindungstabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.LinkSuccessorOperationIDFieldIndex = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.LinkSuccessorOperationIDFieldIndex = 1;
```

LinkSuccessorTaskIDFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in der Verbindungstabelle setzen oder erfragen, das die ID der Nachfolger-Aufgabe enthält. In der Abbildung zu **LinkDataTableName** ist der Index beispielsweise 2.

Bei Verwendung einer Verbindungstabelle ist diese Eigenschaft verbindlich auf einen Wert ungleich -1 zu setzen, wenn nicht die VcResourceScheduler2-Eigenschaft **LinkSuccessorOperationIDFieldIndex** verwendet wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Verbindungsdaten-tabelle, das für die Aufnahme von Ids von Nachfolgeraufgaben vorgesehen ist. {-1...AnzahlDatenfelderInVerbindungsdatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Verbindungsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.NodeSuccessorTaskIDFieldIndex = 2
```

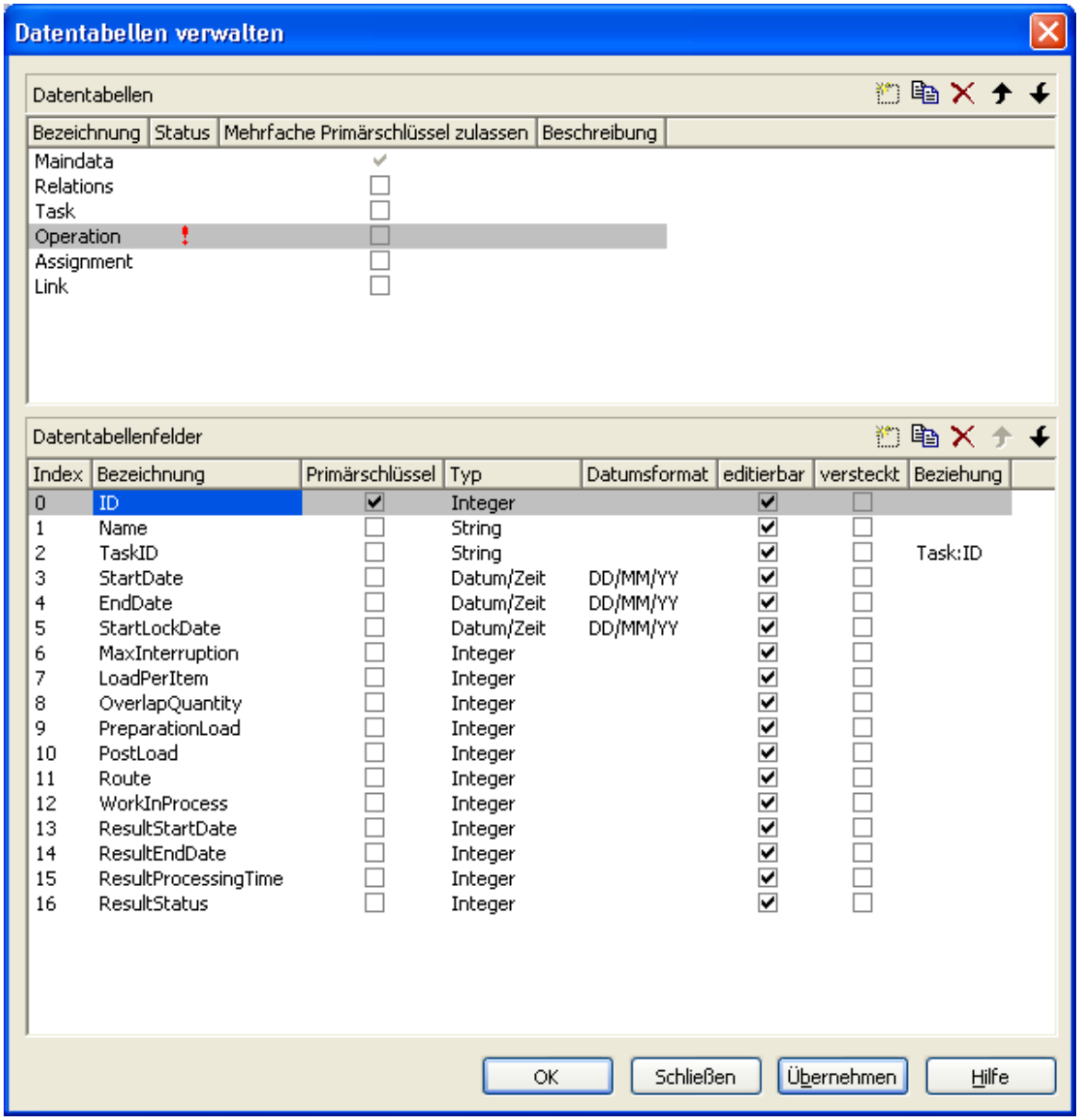
Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.LinkSuccessorTaskIDFieldIndex = 2;
```

OperationDataTableName

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Namen der Operationstabelle setzen oder erfragen. Sie ist die Datentabelle für Operationen, deren Name obligat zu setzen ist.



	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Operationsdatentabelle **Standardwert:** Empty string

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationDataTableName = "Operation"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationDataTableName("Operation");
```

OperationLoadPerItemFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft setzen oder erfragen Sie den Index eines Datenfelds in der Operations-Datentabelle, das die Belegung der zeitdauergebenden Ressourcen pro Stück bezeichnet. Für die Gesamtbelegung auf der zeitdauergebenden Ressource wird dieser Wert mit der in der Aufgabe angegebenen Anzahl multipliziert. Wenn in einem Datenfeld ein ungültiger Wert steht oder diese Eigenschaft auf -1 gesetzt ist, dann wird der Wert zu 0 angenommen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdatentabelle, das für die Aufnahme der Auslastung vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationLoadPerItemFieldIndex = 10
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationLoadPerItemFieldIndex = 10;
```

OperationMaximumInterruptionTimeFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, das für einen Wert vorgesehen ist, der die maximale Zeitdauer bezeichnet, die diese Operation während ihrer Verarbeitungszeit unterbrochen werden darf. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 9.

Eine "Unterbrechung" ist eine aktionsfreie Zeit auf einer voll belegten und einer Operation zugewiesenen Ressource. Sie unterscheidet sich von einer "Pause" dadurch, dass sie nicht durch eine vordefinierte Nicht-Arbeitszeit verursacht wird.

Der Feldinhalt ist eine Zahl, die in Basiszeiteinheiten angegeben wird (s. Eigenschaft **BaseTimeUnit**).

Ist diese Eigenschaft auf -1 gesetzt oder der Wert im jeweiligen Datensatz leer oder 0, dann wird der Wert verwendet, der durch die Eigenschaft

DefaultOperationMaximumInterruptionTime gesetzt wurde. Ist auch diese 0, dann ist keine Unterbrechung erlaubt. Ist der Wert im jeweiligen Datensatz < 0, dann ist ebenfalls keine Unterbrechung erlaubt, auch wenn die Eigenschaft **DefaultOperationMaximumInterruptionTime** einen Wert ungleich 0 hat.

Diese Eigenschaft wird durch die Herabsetzung der maximalen Auslastung auf weniger als 100% (s. Eigenschaft **AssignmentMaximumLoadField-Index**) außer Kraft gesetzt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der maximalen Unterbrechungszeit vorgesehen ist. {-...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationMaximumInterruptionTimeFieldIndex = 9
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationMaximumInterruptionTimeFieldIndex = 9;
```

OperationMinimumSupplementTimeFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem eine Mindest-Ergänzungszeit für jede Operation definiert ist. Für diese Zeitspanne wird auf den der Operation zugewiesenen Ressourcen keine Belegung vorgenommen, d.h. diese Zeitspanne kann z. B. für Warte- oder Transport-zeiten genutzt werden, die unabhängig von den zugewiesenen Ressourcen auftreten.

Der Feldinhalt ist eine Zahl, die in Basiszeiteinheiten angegeben wird (s. Eigenschaft **BaseTimeUnit**). In der Abbildung zu **OperationDataTable-Name** ist der Index beispielsweise 7.

Bitte sehen Sie ergänzend auch **OperationMaximumSupplementLoad-FieldIndex** sowie **OperationPreparationLoadFieldIndex** und **Operation-PostLoadFieldIndex**

BaseCalendarUsageForSupplementTimes.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der Mindest-Ergänzungszeit vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationMinimumSupplementTimeFieldIndex = 7
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationMinimumSupplementTimeFieldIndex = 7;
```

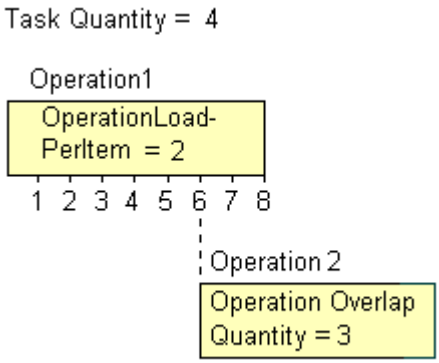
OperationOverlapQuantityFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft setzen oder erfragen sie den Index eines Datenfeldes der Operationstabelle, in dem die 'Überlappungsmenge' einer Operation angegeben werden kann. Mit dieser Eigenschaft können Sie überlappende Ressourcenbelegungen aufeinanderfolgender Operationen erreichen. Dies ist sinnvoll, wenn die nachfolgende Operation nicht auf die vollständige Fertigstellung der vorhergehenden warten muss.

Die im Datenfeld angegebene Menge bezieht sich auf die fertiggestellte Menge der Aufgabe (s. Eigenschaft **TaskQuantityFieldIndex**), ab der die nachfolgende Operation überlappend gestartet werden kann. Es handelt sich um die Angabe einer Mindestmenge, d.h. nach wievielen Einheiten die Überlappung frühestens starten darf, bei optional späterem Start.

Im Beispiel unten ist der Overlap-Wert=3, der sich auf die Aufgabenmenge 4 bezieht. Wenn von diesen 4 Einheiten 3 durch die Operation1 abgearbeitet worden sind, darf Operation2 starten. Wenn Sie einen Mengenfaktor (OperationLoadPerItem) für die Operation1 gesetzt haben (im Bild ist dieser =2), wird dieser mit dem Overlap-Wert multipliziert: 3*2=6. Daher startet Operation2 erst, wenn Operation1 den Wert 6 erreicht hat:



Beispiel-Szenario: Es sollen 4 Kerzenleuchter mit jeweils 3 Kerzen produziert werden. Je 2 Leuchter mit 6 Kerzen werden zusammen verpackt. Nachdem über Operation1 final 6 Kerzen produziert worden sind, kann die Operation2 mit der Verpackung beginnen.

Wird in dem Index nichts oder ein Wert = 0 angegeben, dann findet keine Überlappung mit der Vorgängeroperation statt; bei -1 startet die Operation gleichzeitig mit der Vorgängeroperation.

Ist gleichzeitig eine Vorbereitungszeit definiert, wird diese in der Überlappung berücksichtigt, so dass dann für die Überlappungsmenge ggf. diese Vorbereitungszeit geteilt durch die Auslastung pro Stück der Operation (s. Eigenschaft **OperationLoadPerItemFieldIndex**) hinzuaddiert werden muss. Diese Eigenschaft sollte nicht zusammen mit der Eigenschaft **Resource-EfficienyFieldIndex** verwendet werden. Auch kann es Probleme bei der gleichzeitigen Verwendung von **AssigmentMaximumLoadFieldIndex** geben.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der "Überlappungsmenge" vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationstabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationOverlapQuantityFieldIndex = 11
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationOverlapQuantityFieldIndex = 11;
```

OperationPostLoadFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem eine Nachlaufzeit für jede Operation definiert ist. Für diese Zeitspanne wird auf den der Operation zugewiesenen Ressourcen eine Belegung vorgenommen.

Der Feldinhalt ist eine Zahl, die die erforderliche Kapazität darstellt. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 13.

Bitte sehen Sie dazu auch **OperationPreparationTimeFieldIndex**, **OperationMaximumInterruptionTimeFieldIndex** und **OperationMinimumSupplementTimeFieldIndex**.

Ressourcenunabhängige Nachlaufzeiten können über die Eigenschaft **OperationPostOffsetFieldIndex** definiert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdatentabelle, das für die Aufnahme der Nachlaufzeit vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationPostLoadFieldIndex = 13
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationPostLoadFieldIndex = 13;
```

OperationPostOffsetFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem eine Nachlaufzeit für jede Operation definiert ist. Enthält dieses Feld positive ganze Zahlen (in der Einheit der gültigen Basiszeiteinheit), so ist die Nachlaufzeit der Operationen von den Ressourcen unabhängig.

Wenn der Index den Wert -1 enhält, gibt es keine Nachlaufzeiten. Dies gilt ebenso für Operationen, bei denen der Index auf ein Datenfeld verweist, das bei der jeweiligen Operation keine gültige Zahl oder eine 0 enthält.

Bitte sehen Sie dazu auch **OperationPreparationOffsetFieldIndex**.

Ressourcenabhängige Nachlaufzeiten können über die Eigenschaft **OperationPostLoadFieldIndex** definiert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdatentabelle, mit dem bestimmt wird, ob die Nachlaufzeit einer Operation ressourcenunabhängig sein soll. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass keine Nachlaufzeiten existieren. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationPostOffsetFieldIndex = 8
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationPostOffsetFieldIndex = 8;
```

OperationPreparationLoadFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem eine Vorbereitungszeit für jede Operation definiert ist. Für diese Zeitspanne wird auf den der Operation zugewiesenen Ressourcen eine Belegung vorgenommen.

Der Feldinhalt ist eine Zahl, die die erforderliche Kapazität darstellt. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 12.

Bitte sehen Sie ergänzend auch **OperationPostLoadFieldIndex**, **Operation-MaximumInterruptionTimeFieldIndex** und **OperationMinimumSupplementTimeFieldIndex**.

Ressourcenunabhängige Nachlaufzeiten können über die Eigenschaft **OperationPreparationOffsetFieldIndex**definiert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der Vorbereitungszeit vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationPreparationLoadFieldIndex = 12
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationPreparationLoadFieldIndex = 12;
```

OperationPreparationOffsetFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem eine Vorlaufzeit für jede Operation definiert ist. Enthält dieses Feld positive ganze Zahlen (in der Einheit der gültigen Basiszeiteinheit), so ist die Vorlaufzeit der Operationen von den Ressourcen unabhängig.

Wenn der Index den Wert -1 enhält, gibt es keine Vorlaufzeiten. Dies gilt ebenso für Operationen, bei denen der Index auf ein Datenfeld verweist, das bei der jeweiligen Operation keine gültige Zahl oder eine 0 enthält.

Bitte sehen Sie ergänzend auch **OperationPostOffsetFieldIndex**.

Ressourcenabhängige Nachlaufzeiten können über die Eigenschaft **OperationPreparationLoadFieldIndex**definiert werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, mit dem bestimmt wird, ob die Vorlaufzeit einer Operation ressourcenunabhängig sein soll. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass keine Vorlaufzeiten existieren. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationPreparationOffsetFieldIndex = 8
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationPreparationOffsetFieldIndex = 8;
```

OperationResultEndDateFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft setzen oder erfragen Sie den Index eines Datenfeldes der Operationstabelle, in das das berechnete Enddatum für die jeweilige Operation eingetragen wird. In der Abbildung zu **OperationDataTable-Name** ist der Index beispielsweise 17.

Um sinnvolle Ergebnisse aus der Ressourcenplanung zu erhalten, müssen mindestens zwei der drei Eigenschaften **OperationResultStartDateField-Index**, **OperationResultProcessingTimeFieldIndex** und **OperationResult-EndDateFieldIndex** ungleich -1 sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des Enddatums vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultEndDateFieldIndex = 17
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationResultEndDateFieldIndex = 17;
```

OperationResultPostEndDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in das nach einer erfolgten Planung die Endezeit der Nachlaufphase einer Operation abgelegt wird. Wenn die Nachlaufzeit 0 ist, dann ist dieses Datum identisch mit dem Wert in dem Datenfeld, das über die Eigenschaft **OperationResult-EndDateFieldIndex** adressiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des Enddatums der Nachlaufphase vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultPostEndDateFieldIndex = 15
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationPostEndDateFieldIndex = 15;
```

OperationResultPreparationStartDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in das nach einer erfolgten Planung die Startzeit der Vorbereitungsphase einer Operation abgelegt wird. Wenn die Vorbereitungszeit 0 ist, dann ist dieses Datum identisch mit dem Wert in dem Datenfeld, das über die Eigenschaft **OperationResultStartDateFieldIndex** adressiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des Startdatums der Vorbereitungsphase vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10;
```

OperationResultProcessingTimeFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft setzen oder erfragen Sie den Index eines Datenfeldes der Operationstabelle, in das die berechneten Verarbeitungszeitraum für die jeweilige Operation eingetragen wird. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise18 .

Um sinnvolle Ergebnisse aus der Ressourcenplanung zu erhalten, müssen mindestens zwei der drei Eigenschaften **OperationResultStartDateFieldIndex**, **OperationResultProcessingTimeFieldIndex** und **OperationResultEndDateFieldIndex** ungleich -1 sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdatentabelle, das für die Aufnahme des Verarbeitungszeitraums vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultProcessingTimeFieldIndex = 18
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationResultProcessingTimeFieldIndex = 18;
```

OperationResultSelectedTimingResourceIDFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in das nach einer erfolgten Planung die vom Modul ausgewählte ID der zeitdauerbestimmenden Ressource abgelegt wird. Auf diese Art kann in der Tabelle oder einer Layerbeschriftung eine zugewiesene Ressource einfach grafisch dargestellt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der ID der zeitdauer-bestimmenden Ressource vorgesehen ist, die vom Modul bei der Berechnung ausgewählt wird. {0...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.OperationResultSelectedTimingResourceFieldIndex = 8
```

Code-Beispiel C#

```
vcGantt1.OperationResultSelectedTimingResourceFieldIndex = 8;
```

OperationResultStartDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in das das berechnete Startdatum für die jeweilige Operation eingetragen wird. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 16.

Um sinnvolle Ergebnisse aus der Ressourcenplanung zu erhalten, müssen mindestens zwei der drei Eigenschaften **OperationResultStartDateField-Index**, **OperationResultProcessingTimeFieldIndex** und **OperationResult-EndDateFieldIndex** ungleich -1 sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des Startdatums vorgesehen ist. {-1...AnzahlDatensätzeInOperationenTabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultStartDateFieldIndex = 16
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationResultStartDateFieldIndex = 16;
```

OperationResultStatusFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, das einen Fehler- oder einen Warnstatus aufnimmt. In der Abbildung zu **Operation-DataTableName** ist der Index beispielsweise 19.

Mögliche Werte, die von der Ressourenplanung hier eingetragen werden:

0: die Operation wurde fehlerfrei eingeplant

1: die Operation wurde nicht eingeplant, weil die Ressourcenplanung eine andere Route der zugehörigen Aufgabe ausgewählt hat. Der Fall kann nur auftreten, wenn die Eigenschaft **OperationRouteFieldIndex** auf einen Wert ungleich -1 gesetzt wurde.

1000: die Operation wurde nicht eingeplant

1001: die Operation wurde nicht eingeplant und ist die Operation einer Aufgabe, die verursacht hat, dass die gesamte Aufgabe nicht einplanbar war. Die Gründe dafür können vielfältig sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des Fehlerstatus vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultStatusFieldIndex = 19
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationResultStatusFieldIndex = 19;
```

OperationRouteFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in der Operationstabelle setzen oder erfragen, über dessen Werte Operationen einer

bestimmten Route zugewiesen werden. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 14.

Operationen mit gleichem Inhalt in diesem Feld gehören der gleichen Route an. Der Inhalt des Felds bezeichnet damit den Namen einer Route.

Hiermit können Alternativen für die Ausführung einer Aufgabe (Task) beschrieben werden, das heißt, die Ressourcenplanung prüft alle Routen für die Ausführung einer Aufgabe und wählt eine Route aus. So können Sie mehrere alternative Operationsketten für eine Aufgabe definieren. Nicht mehr als 10 Routen können pro Aufgabe definiert werden. Die Route wird in der Reihenfolge des Auftretens der Routen in den Operationen ausgewählt.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdatentabelle, das für die Aufnahme des Namens der Route vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationRouteFieldIndex = 14
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationRouteFieldIndex = 14;
```

OperationSequenceNumberFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index des Datenfeldes in der Operationstabelle setzen oder erfragen, über dessen Werte die Reihenfolge der Operationen innerhalb einer Aufgabe (Task) festgelegt wird. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 6.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der Reihenfolge vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationSequenceNumberFieldIndex = 6
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationSequenceNumberFieldIndex = 6;
```

OperationStartLockDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem für jede Operation (Operation) im Falle der Planungsstrategie ASAP (s. Eigenschaft **PlanningStrategy**) ein Startdatum hinterlegt werden kann. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 5.

Wenn das Datenfeld einen gültigen Datumseintrag enthält, dann wird die Aufgabe an dieses Startdatum gebunden und durch die Ressourcenplanung nicht mehr verschoben, was insbesondere für bereits begonnene Aufgaben sinnvoll ist (siehe in diesem Zusammenhang bitte auch die Eigenschaft **OperationWorkInProcessFieldIndex**).

Sie können über die Eigenschaft **ToleranceTimeOnStartLockDates** eine Toleranz festlegen, um die eine Operation später anfangen darf, als über das gebundenes Startdatum festgelegt. Beachten Sie auch, dass Aufgaben mit Operationen, die ein gebundenes Startdatum haben, nicht automatisch bevorzugt eingeplant werden. Wenn Sie dies wünschen, dann müssen Sie die Prioritäten der Aufgaben selbst berechnen (s. Eigenschaft **TaskPriority-FieldIndex**).

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des feststehenden Startdatums vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationStartLockDateFieldIndex = 5
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationStartLockDateFieldIndex = 5;
```

OperationTaskIDFieldIndex

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index des Datenfeldes in der Operationstabelle setzen oder erfragen, in dem die ID der zugehörigen Aufgabe abgelegt ist. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 2.

Für die Einplanung der Operation muss diese Eigenschaft auf einen Wert ungleich -1 gesetzt werden. Mithilfe dieses Datenfelds können einer Aufgabe beliebig viele Operationen zugewiesen werden. Die Reihenfolge, in der die Operationen einer Aufgabe eingeplant werden, hängt vom Wert in dem Datenfeld ab, dessen Index über die Eigenschaft **OperationSequence-NumberDataFieldIndex** eingestellt wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme der ID der Aufgabe vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationTaskIDFieldIndex = 2
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationTaskIDFieldIndex = 2;
```

OperationWorkInProcessFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Operationstabelle, in dem der Fertigstellungsgrad einer Operation in Prozent enthalten ist, also ein Wert zwischen 0 und 100. In der Abbildung zu **OperationDataTableName** ist der Index beispielsweise 15.

Falls der Datenfeldindex auf -1 gesetzt ist oder im Datenfeld kein gültiger Wert gefunden wird, wird für 0% angenommen ("nicht angefangen").

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Operationsdaten-tabelle, das für die Aufnahme des Fertigstellungsgrades vorgesehen ist. {-1...AnzahlDatenfelderInOperationsdatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Operationsdatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationWorkInProcessFieldIndex = 15
```

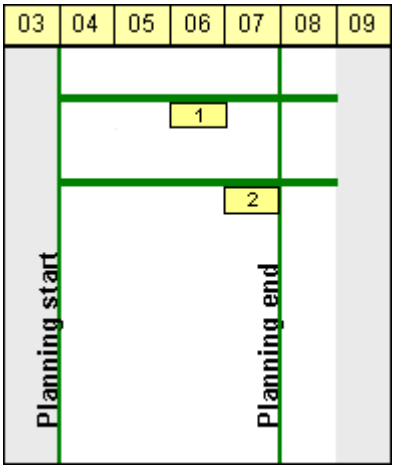
Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationWorkInProcessFieldIndex = 15;
```

PlanningEndDate

Eigenschaft von VcResourceScheduler2

Mit Hilfe dieser Eigenschaft setzen oder erfragen Sie das Enddatum des Planungszeitraums für die Ressourcenplanung. Wenn Sie kein Enddatum setzen, wird das Ende der Zeitskala verwendet (gesetzt durch die Eigenschaft **VcGantt.TimeScaleEnd**). Den Anfang des Zeitraums setzen Sie mit der Eigenschaft **PlanningStartDate**.



Begrenzter Planungszeitraum

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Enddatum der Planungsperiode **Standardwert:** DateTime.MinValue

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.PlanningEndDate = VcGantt1.TimeScaleEnd
```

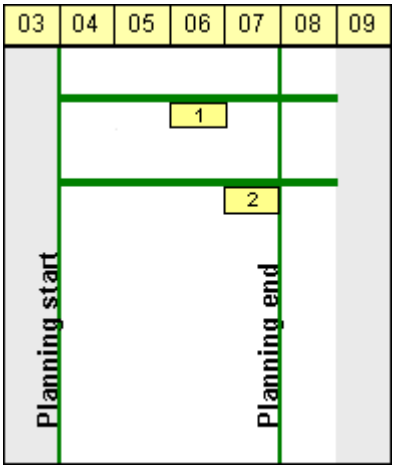
Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.PlanningEndDate = vcGantt1.TimeScaleEnd;
```

PlanningStartDate

Eigenschaft von VcResourceScheduler2

Mit Hilfe dieser Eigenschaft setzen oder erfragen Sie das Startdatum des Planungszeitraums für die Ressourcenplanung. Wenn Sie kein Startdatum setzen, wird der Beginn der Zeitskala verwendet (gesetzt durch die Eigenschaft **VcGantt.TimeScaleStart**). Das Ende des Zeitraums setzen Sie mit der Eigenschaft **PlanningEndDate**.



Begrenzter Planungszeitraum

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Startdatum der Planungsperiode **Standardwert:** DateTime.MinValue

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.PlanningStartDate = VcGantt1.TimeScaleStart
```

Code-Beispiel C#

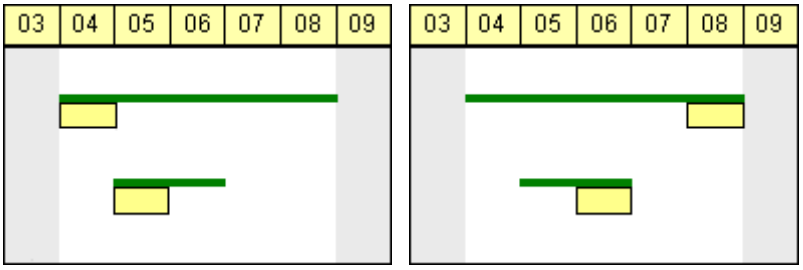
```
vcGantt1.ResourceScheduler2.PlanningStartDate = vcGantt1.TimeScaleStart;
```

PlanningStrategy

Eigenschaft von VcResourceScheduler2

Diese Eigenschaft beschreibt die Planungsstrategie für alle Arbeitsaufträge.

Es gibt zwei mögliche Planungsstrategien: Bei der einen ist die Zielsetzung, Aufgaben so früh wie möglich (as soon as possible, ASAP) abzuarbeiten, um Ressourcen für Folgeaufträge bereit zu halten und einen hohen Durchsatz anzustreben. Die zweite Strategie strebt an, Aufgaben termingerecht (just in time, JIT) abzuarbeiten, um z. B. den Lagerungsaufwand gering zu halten.



Bei der ASAP-Strategie wird das Startdatum möglichst früh gelegt (Bild links), bei der JIT-Strategie wird das Enddatum möglichst spät eingeplant (Bild rechts). Die langen, linienförmigen Balken zeigen den möglichen Ausführungszeitraum des Aufgabe an; die kurzen, rechteckigen stellen die tatsächlich eingeplante Ausführungszeit dar. ASAP-Aufgaben erscheinen daher nach Möglichkeit links, JIT-Aufgaben rechts innerhalb des möglichen Ausführungszeitraums.

Abweichend von der hier eingestellten Planungsstrategie kann durch die Zuweisung eines Datenfeldes bei **TaskPlanningStrategyFieldIndex** für jeden Arbeitsauftrag eine individuelle Einstellung vorgenommen werden. Letztere überschreibt die allgemein gesetzte Strategie.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcResourceSchedulingPlanningStrategy	Planungsstrategie **Standardwert:** vcResSchedPSASAP
	Mögliche Werte:	
	.vcResSchedPSASAP -1	So bald wie möglich (as soon as possible)
	.vcResSchedPSJIT 0	Gerade rechtzeitig (just in time)

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.PlanningStrategy =
VcResourceSchedulingPlanningStrategy.vcResSchedPSASAP
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.PlanningStrategy =
VcResourceSchedulingPlanningStrategy.vcResSchedPSASAP;
```

ResourceCalendarNameFieldIndex

Eigenschaft von VcResourceScheduler2

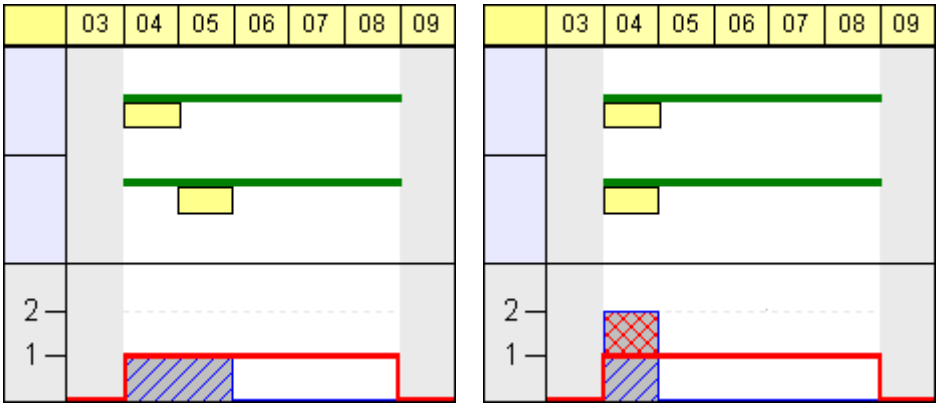
Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Ressourcentabelle, das für die jeweilige Ressource des Typs **TimingResource** oder **WorkResource** den Namen eines Kalenders bestimmt. Ist dieses Datenfeld an einer Ressource leer, enthält einen ungültigen Namen oder ist die Eigenschaft auf -1 gesetzt, dann wird ersatzweise die Bezeichnung der Ressource als Kalendername benutzt, die der Zuordnung durch die Eigenschaft **ResourceNameFieldIndex** entspricht.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Ressourcendaten-tabelle, das für eine Resource vom Typ **Timing Resource** oder **WorkResource** einen Kalendernamen bestimmt. **Standardwert:** -1

ResourceCapacityType

Eigenschaft von VcResourceScheduler2

Diese Eigenschaft beschreibt den Kapazitätstyp für alle Ressourcen, falls dieser nicht vorgangsweise durch **ResourceCapacityTypeFieldIndex** über-schrieben wird.



Bei begrenzter Kapazität werden die Aufgaben nacheinander eingeplant (links); bei unbegrenzter Kapazität können sie zeitgleich eingeplant werden (rechts).

	Datentyp	Beschreibung
Parameter: ⇨ resourceTableIndex	System.Int16	Index der Ressourcendatentabelle. {0...24}.
Eigenschaftswert	VcResourceCapacityType **Mögliche Werte:** .vcResSchedCTFinite -1 .vcResSchedCTInfinite 0	Kapazitätstypen **Standardwert:** vcResSchedCTFinite Begrenzte Kapazitäten Unbegrenzte Kapazitäten

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceCapacityType =
VcResourceSchedulingCapacityType.vcResSchedCTFinite
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ResourceCapacityType =
VcResourceSchedulingCapacityType.vcResSchedCTFinite;
```

ResourceCapacityTypeFieldIndex

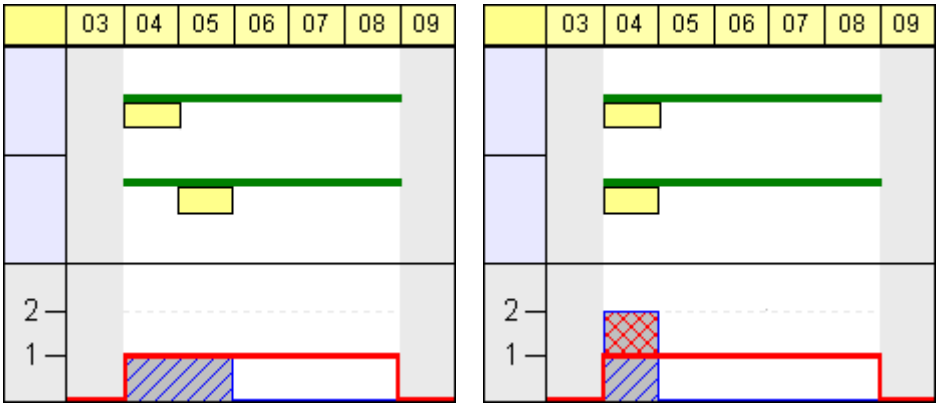
Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Index eines Datenfeldes in einer Ressourcentabelle setzen oder erfragen, in dem der Kapazitätstyp für eine einzelne zeitdauerbestimmende Ressource enthalten ist. In der Abbildung zu **ResourceDataTableName** ist der Index beispielsweise 2.

Der als Parameter übergebene Ressourcentabellenindex bezeichnet eine von 25 möglichen Ressourcentabellen, die verwendet werden soll und über die indizierte Eigenschaft **ResourceDataTableName** definiert wird.

Definierte Werte der Datenfeldinhalte:

1 begrenzte Kapazität 2 unbegrenzte Kapazität



Bei begrenzter Kapazität werden die Arbeitsaufträge nacheinander eingeplant (links); bei unbegrenzter Kapazität können sie zeitgleich eingeplant werden (rechts).

Mit der Eigenschaft **ResourceCapacityType** können Sie den Kapazitätstyp für alle Datensätze einer Ressourcentabelle vorgeben. Diese Vorgabe wird durch die Einzelsetzung hier überschrieben.

Wenn eine Ressource mehreren Gruppen zugehörig ist, muss sie in jeder Gruppe den gleichen Kapazitätstyp besitzen.

	Datentyp	Beschreibung
Parameter:		
⇨ resourceTableIndex	System.Int16	Index der Ressourcendatentabelle {0...24}
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Ressourcendaten-tabelle, das für die Aufnahme des Kapazitätstyps vorgesehen ist. {-1...AnzahlDatenfelderInRessourcendatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Ressourcendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceCapacityTypeFieldIndex(0) = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_ResourceCapacityTypeFieldIndex(0,1);
```

ResourceConstraintTypeFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Ressourcentabelle, in dem eine Bedingung oder Beschränkung einer einzelnen Arbeits- und Materialressource enthalten ist.

Unter den 25 möglichen Ressourcentabellen wird die betroffene durch den als Parameter übergebenen Index adressiert.

Erlaubt sind als Typen die Inhalte 0,1 oder 3 oder ein Leerwert. Die Werte "" oder "1" oder wenn kein Feld vorhanden ist, bedeuten, dass die vorgegebene Kapazität der Resource bindend ist (sogen. "harte" Ressource).

Der Wert "0" bedeutet, dass die vorgegebene Kapazität der Ressource bei Bedarf ignoriert wird, da sie dann in unbegrenzter Kapazität zur Verfügung steht (sogen. "weiche" Ressource).

Der Wert "3" bedeutet, dass die Ressource "hart" ist, aber arbeitsfreie Zeiten berücksichtigt werden, die nicht als Unterbrechung bei der Einplanung der Operation gelten sollen.

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index der Ressourcendatentabelle {0...24}
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Ressourcendaten-tabelle, das für die Aufnahme einer Bedingung vorgesehen ist. {-1...AnzahlDatenfelderInRessourcendatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Ressourcendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceConstraintTypeFieldIndex(0) = 1
```

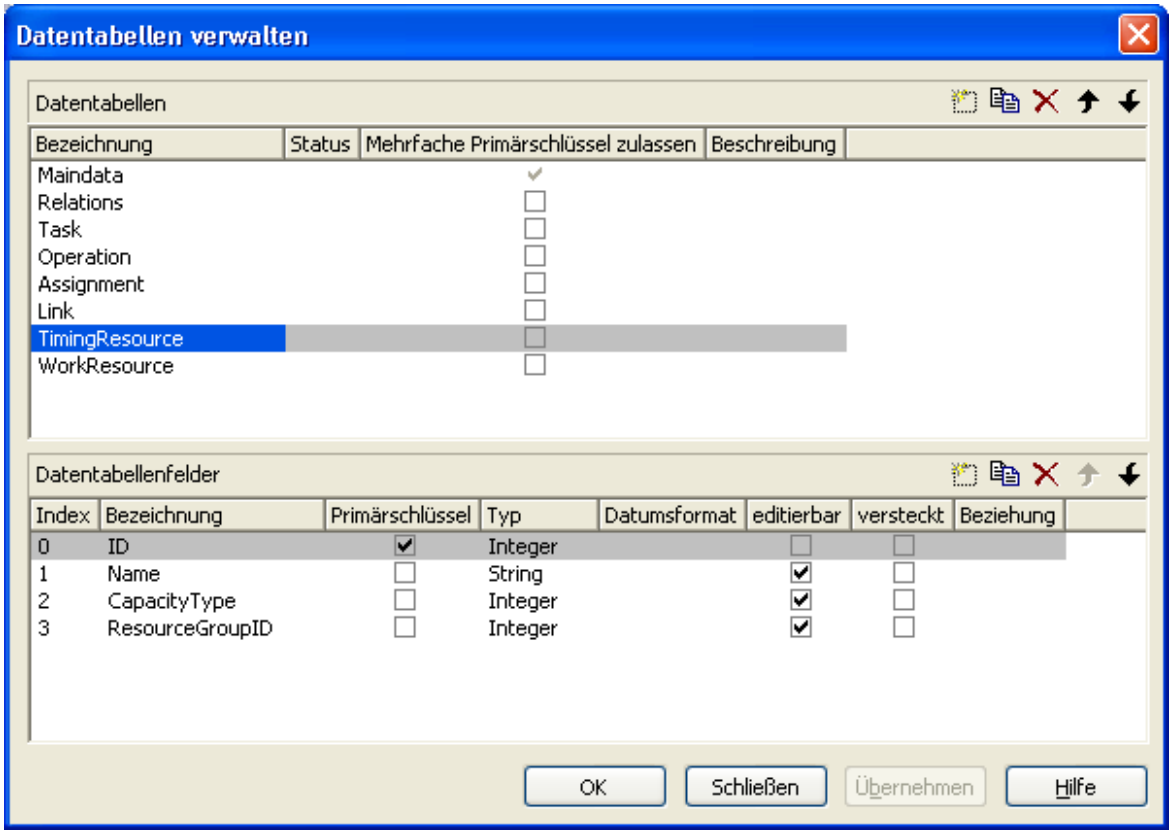
Code-Beispiel C#

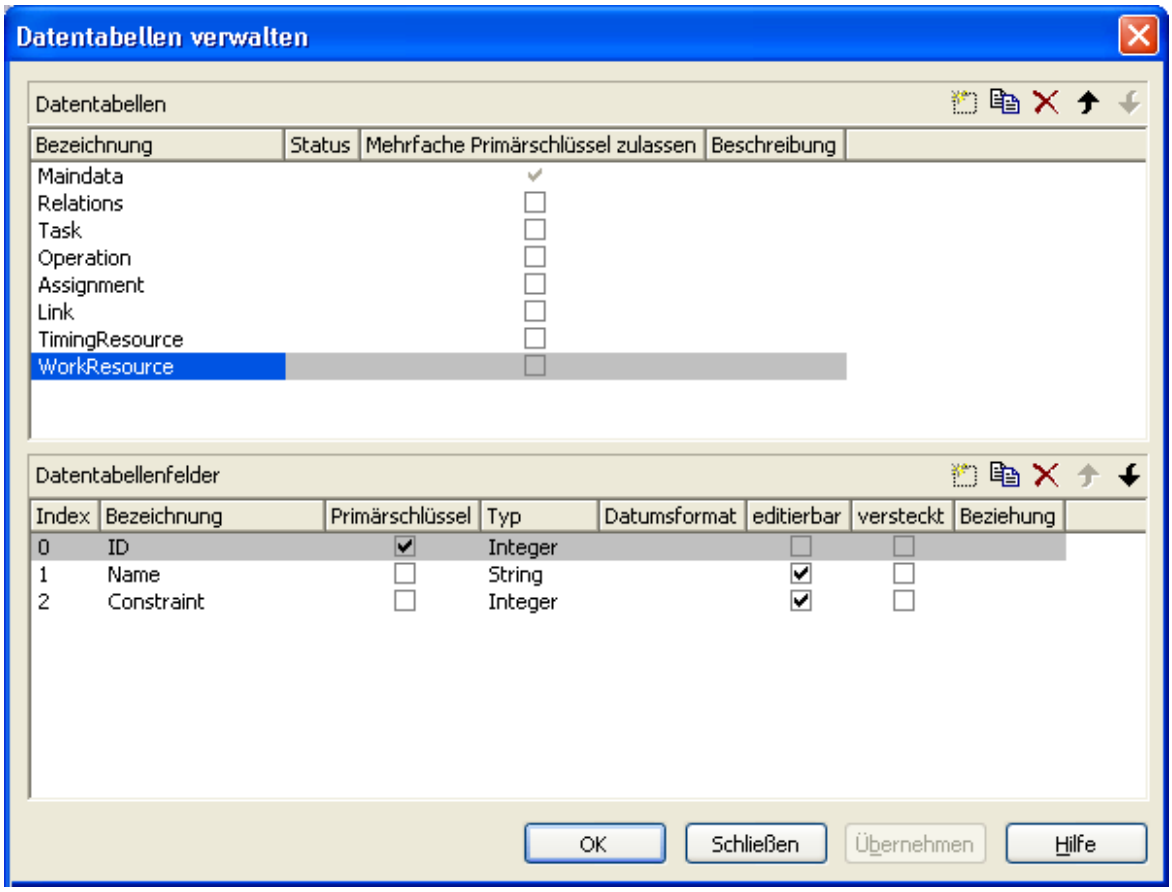
```
vcGantt1.ResourceScheduler2.set_ResourceConstraintTypeFieldIndex(0,1);
```

ResourceDataTableName

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie die Namen von bis zu 25 Ressourcen-tabellen setzen oder erfragen. Der Name der Tabelle mit dem Index 0 ist obligat zu setzen. Die Indizes sind von 0 an aufwärts fortlaufend zu besetzen. Für jede hier angegebene Ressourcentabelle ist auch ein Feld in der Zuweisungstabelle vorzusehen und mit der Eigenschaft **Assignment-ResourceIDFieldIndex** bekannt zu geben.


Datentabellen verwalten
Datentabellen
Bezeichnung
Status
Mehrfache Primärschlüssel zulassen
Beschreibung
Maindata
Relations
Task
Operation
Assignment
Link
TimingResource
WorkResource
Datentabellenfelder
Index
Bezeichnung
Primärschlüssel
Typ
Datumsformat
editierbar
versteckt
Beziehung
0 ID Integer
1 Name String
2 CapacityType Integer
3 ResourceGroupID Integer
OK
Schließen
Übernehmen
Hilfe


Datentabellen verwalten
Datentabellen
Bezeichnung
Status
Mehrfache Primärschlüssel zulassen
Beschreibung
Maindata
Relations
Task
Operation
Assignment
Link
TimingResource
WorkResource
Datentabellenfelder
Index
Bezeichnung
Primärschlüssel
Typ
Datumsformat
editierbar
versteckt
Beziehung
0 ID Integer
1 Name String
2 Constraint Integer
OK
Schließen
Übernehmen
Hilfe

	Datentyp	Beschreibung
Parameter: ⇨ resourceTableIndex	System.Int16	Index der Ressourcendatentabelle {0...24}
Eigenschaftswert	System.String	Name der Datentabelle **Standardwert:** Empty string

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceDataTableName(1) = "Timing Resource"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_ResourceDataTableName(1, "Timing Resource");
```

ResourceEfficiencyFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Ressourcentabelle, das für die jeweilige Ressource des Typs **TimingResource** eine Effizienz in Prozent angibt. Ist dieses Datenfeld an einer Ressource leer oder die Eigenschaft auf -1 gesetzt, dann ist die Effizienz standardmäßig 100. Wenn hingegen ein Wert angegeben wird, dann wird die Gesamtsumme der Belegungen durch eine Zuweisung vor der Einplanung auf dieser Ressource mit dem gegebenen Effizienzwert multipliziert, d.h. bei einer Effizienz unter 100 Prozent dauert eine zugewiesene Operation auf dieser Ressource länger als im Default, während bei Werten über 100 Prozent eine zugewiesene Operation hier schneller abgearbeitet wird als im Default. Interessant ist dies besonders bei der Definition von Ressourcengruppen (s. auch **ResourceSelectionStrategy**), wobei dann die verfügbare Ressource mit der höchsten Effizienz bevorzugt werden kann.

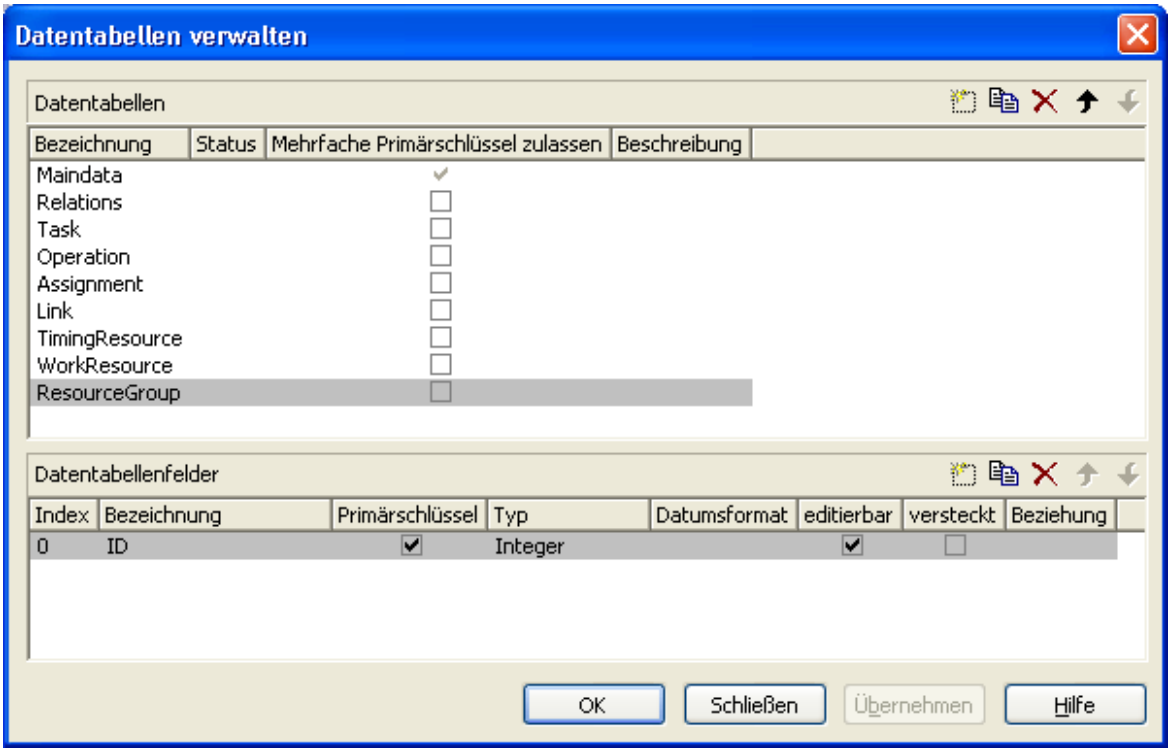
Üblicherweise bewegt sich die Effizienz im Bereich zwischen 1 und 100. Werte über 1.000 werden automatisch auf 1.000 begrenzt. Die Effizienz sollte nicht so hoch gesetzt werden, dass die Belegung einer Ressource durch eine Operation den Wert von 1 unterschreitet.

	Datentyp	Beschreibung
Eigenschaftswert	SystemInt.32	Index des Datenfeldes in der Ressourcendaten-tabelle, das die Effizienz einer Resource vom Typ **Timing Resource** in Prozent angibt.

ResourceGroupDataTableName

Eigenschaft von VcResourceScheduler2

Diese indizierte Eigenschaft bestimmt, in welcher Datentabelle die Ressourcengruppen gefunden werden, deren IDs in den Feldern enthalten sind, die durch die Eigenschaft **ResourceGroupIDFieldIndex** adressiert werden. D.h. für jeden Feldindex, den Sie über die Eigenschaft **ResourceGroupIDFieldIndex** ungleich -1 setzen, müssen Sie über diese Eigenschaft den Namen einer Datentabelle angeben, wobei die gleichen Datentabellen benutzt werden, die auch in der indizierten Eigenschaft **ResourceDataTableName** angegeben werden. Der als Parameter übergebene Ressourcentabellenindex bezeichnet eine von 25 möglichen Ressourcentabellen, die per indizierter Eigenschaft **ResourceDataTableName** zugewiesen werden.



	Datentyp	Beschreibung
Parameter: ⇨ ResourceGroupTableIndex	System.Int16	Index einer Ressourcengruppendatentabelle. {0...24}
Eigenschaftswert	System.String	Name der Ressourcengruppendatentabelle

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceGroupDataTableName(1) = "Printer Resource"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_ResourceGroupDataTableName(1, "Printer
Resource");
```

ResourceGroupIDFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Ressourcentabelle, das für die ID einer Gruppen-Ressource vorgesehen ist. Die Ressource wird durch das Setzen der ID in dem bezeichneten Feld als einer Gruppe zugehörig beschrieben. In der Abbildung zu **ResourceGroup-DataTableName** ist der Index beispielsweise 0. Wird der Feldindex auf -1 gesetzt oder ist das entsprechende Datenfeld der Ressource leer, gehört die Resource keiner Gruppe an. Diese Eigenschaft darf nur bei einer zeitdauer-bestimmenden Ressource gesetzt werden (s. Eigenschaft **ResourceType**).

Der als Parameter übergebene Ressourcentabellenindex bezeichnet eine von 25 möglichen Ressourcentabellen, die über die indizierte Eigenschaft **ResourceDataTableName** zugewiesen werden.

	Datentyp	Beschreibung
Parameter:		
⇨ resourceTableIndex	System.Int16	Index der Ressourcendatentabelle. {0...24}
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Ressourcendaten-tabelle, das für die Aufnahme einer GruppenID vorgesehen ist. {-1...AnzahlDatenfelderInRessourcendatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Ressourcendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceGroupIDFieldIndex(0) = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_ResourceGroupIDFieldIndex(0,1);
```

ResourceNameFieldIndex

Eigenschaft von VcResourceScheduler2

Der als Eigenschaftswert übergebene Index bezeichnet ein Datenfeld in der Ressourcentabelle, das für den Namen einer Ressource vorgesehen ist. In der Abbildung zu **ResourceDataTableName** ist der Index beispielsweise 1.

Der Ressourcenname dient der Ermittlung von Histogramm-, Kurven- und Kalendername. Außerdem wird er benutzt, um bei Verwendung von Ressourcengruppen die Möglichkeit zu erhalten, eine Ressource gleichzeitig mehreren Gruppen zuordnen zu können. Dazu muss der Ressourcenname in mehreren Datensätzen identisch angegeben werden, aber mit verschiedenen IDs der Gruppen-Ressourcen. Wird kein Feldindex angegeben, wird für die Ermittlung von Histogramm-, Kurven- und Kalendername die Ressourcen-ID benutzt.

Der als Parameter übergebene Ressourcentabellenindex bezeichnet eine von 25 möglichen Ressourcentabellen, deren Verwendung über die indizierte Eigenschaft **ResourceDataTableName** gesetzt wird.

	Datentyp	Beschreibung
Parameter: ⇨ resourceTableIndex	System.Int16	Index der Ressourcendatentabelle. {0...24}
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Ressourcendaten-tabelle, das für die Aufnahme des Namens vorgesehen ist. {-1...AnzahlDatenfelderInRessourcendatentabelle - 1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Ressourcendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceNameFieldIndex(0) = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_ResourceNameFieldIndex(0,1);
```

ResourceResultLoadCurveNamePrefix

Eigenschaft von VcResourceScheduler2

Präfix für den Namen einer Kurve, die nach der Ressourcenplanung die Auslastung je zeitdauerbestimmender Ressource und je Arbeitsressource enthalten soll.

Diese Kurven müssen vor dem Aufruf der Methode **Process** bereits definiert sein, damit sie im Histogramm dargestellt werden können. Für den Rest des Kurvennamens wird der Ressourcenname oder die Ressourcen-ID verwendet (s. Eigenschaft **ResourceNameFieldIndex**). Wird eine Kurve nicht gefunden, gehen die Ergebnisse der Auslastung für die jeweilige Ressource verloren.

Bei den Kurven muss außerdem die Eigenschaft **CurveSource** auf **vcSetCurve** gesetzt sein, d.h. sie müssen mittels **VcCurve.SetValues** über die API bestückbar sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die das Präfix enthält **Standardwert:** "Load_"

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceResultLoadCurveNamePrefix = "LoadCurve_"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ResourceResultLoadCurveNamePrefix = "LoadCurve_";
```

ResourceResultStockCurveNamePrefix

Eigenschaft von VcResourceScheduler2

Präfix für den Namen einer Kurve, die nach der Ressourcenplanung den Lagerbestand je Materialressource enthalten soll. Diese Kurven müssen vor dem Aufruf der Methode **Process** bereits definiert sein, damit sie im Histogramm dargestellt werden können. Der Ressourcenname oder die Ressourcen-ID wird für den Rest des Namens verwendet.

Wird eine Kurve nicht gefunden, gehen die Ergebnisse des Lagerbestands für die jeweilige Ressource verloren. Der Lagerbestand ergibt sich aus der Kumulierung von Materialzuflüssen (also durch die Lieferkurve, die von der Applikation vor der Durchführung der Ressourcenplanung bestückt wird) und

dem Verbrauch durch die der Ressource zugewiesenen Operationen über die Zeit.

Bei den Kurven muss außerdem die Eigenschaft **CurveSource** auf **vcSetCurve** gesetzt sein, das heißt, sie müssen mittels **VcCurve.SetValues** über die API bestückbar sein.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Zeichenkette, die das Präfix enthält **Standardwert:** "Stock_"

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceResultStockCurveNamePrefix = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ResourceResultStockCurveNamePrefix = 1;
```

ResourceSelectionStrategy

Eigenschaft von VcResourceScheduler2

Diese Eigenschaft beschreibt die Auswahlstrategie für Ressourcen, die von der Ressourcenplanung aus einer Gruppe ausgewählt werden (gilt daher nur für zeitdauerbestimmende Ressourcen).

	Datentyp	Beschreibung
Eigenschaftswert	VcResourceSchedulingResourceSelectionStrategy	Auswahltypen **Standardwert:** VcResSchedRSSequential
	Mögliche Werte: .vcResSchedRSFirstAvailable 6	Die zum Zeitpunkt der Planung einer Operation am frühesten verfügbare Ressource aus der Gruppe, die die benötigte freie Kapazität aufweist, wird ausgewählt. Bei Verwendung dieser Konstanten ist die Entscheidung nur von der zuerst verfügbaren zeitdauergebenden Ressource abhängig. Wietere Zuwiesungen der gleichen Operation werden nicht berücksichtigt, so dass bei Verwendung von Material- und Arbeitsressourcen ein nicht zufriedenstellendes Ergebnis entstehen kann.

.vcResSchedRSHighestEfficiency 2	Die zum Zeitpunkt der Planung einer Operation effizienteste Ressource aus der Gruppe, die die benötigte freie Kapazität aufweist, wird ausgewählt (nur sinnvoll, wenn die Eigenschaft **ResourceEfficiencyFieldIndex** genutzt wird).
.vcResSchedRSLeastLoaded 0	Die zum Zeitpunkt der Planung einer Operation am geringsten ausgelastete Ressource aus der Gruppe, die die benötigte freie Kapazität aufweist, wird zuerst belegt. Diese Strategie ist sinnvoll, wenn Ressourcen möglichst gleichmäßig ausgelastet werden sollen. Bei Verwendung dieser Konstanten wird bei der Einplanung der Ressourcen eine laufende Summe der Belegungen aufaddiert, anhand derer die Entscheidung für die am geringsten belegte Ressource getroffen wird. Daher ist es nicht ausgeschlossen, dass bei stark differierenden Planungszeiträumen von Aufgaben oder bei Nutzung beider Planungsstrategien das Ergebnis nicht zufriedenstellend ausfällt.
.vcResSchedRSMostLoaded 1	Die zum Zeitpunkt der Planung einer Operation am höchsten usgelastete Ressource aus der Gruppe, die die benötigte freie Kapazität aufweist, wird zuerst belegt. Diese Strategie ist sinnvoll, wenn möglichst wenige Ressourcen möglichst hoch ausgelastet werden sollen. Bei Verwendung dieser Konstanten wird bei der Einplanung der Ressourcen eine laufende Summe der Belegungen aufaddiert, anhand derer die Entscheidung für die am höchsten belegte Ressource getroffen wird. Daher ist es nicht ausgeschlossen, dass bei stark differierenden Planungszeiträumen von Aufgaben oder bei Nutzung beider Planungsstrategien das Ergebnis nicht zufriedenstellend ausfällt.
.vcResSchedRSSequential -1	Es wird versucht, die Ressourcen in vorgegebener Reihenfolge zu benutzen.

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceSelectionStrategy =
VcResourceSchedulingResourceSelectionStrategy.vcResSchedRSLeastLoaded
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ResourceSelectionStrategy =
VcResourceSchedulingResourceSelectionStrategy.vcResSchedRSLeastLoaded;
```

ResourceType

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Typ einer Ressourcentabelle festlegen oder erfragen. Der übergebene Index gibt an, von welcher der 25 möglichen Datentabellen der Typ festgelegt werden soll. Es gibt drei Ressourcentypen:

1. **Zeitbestimmende Ressourcen**

Eine Operation wird genau einer Ressource dieses Typs zugewiesen. Sie bestimmt die Verarbeitungsdauer der Operation. Als Kapazitätstypen sind finit und infinit erlaubt (s. Eigenschaft **ResourceCapacityTypeFieldIndex**). Ressourcen dieses Typs können gruppiert werden (s. Eigenschaften **ResourceGroupDataTableName** und **ResourceGroupIDFieldIndex**). Zudem ist eine Beschränkung der Belegung möglich (s. Eigenschaften **AssignmentMinimumLoadFieldIndex** und **AssignmentMaximumLoad-FieldIndex**). Die zeitbestimmende Ressource benötigt Kapazitätskurven als indirekte Ressourceninformation und verwendet Belegungskurven als Ergebnis (s. Eigenschaften **ResourceNameFieldIndex** und **ResourceResultLoad-CurvePrefix).**

2. Arbeits-Ressourcen

Dieser Ressourcentyp ist durch zwei Merkmale gekennzeichnet. Zum einen kann eine Operation mehreren Ressourcen dieses Typs zugewiesen werden. Genau wie die zeitbestimmende Ressource benötigt auch die Arbeits-Ressource Kapazitätskurven als indirekte Ressourceninformation und verwendet Belegungskurven als Ergebnis (s. Eigenschaften **ResourceName-FieldIndex** und **ResourceResultLoadCurvePrefix).**

3. Materialressourcen

Auch die Materialressource ist durch zwei Merkmale gekennzeichnet. Eine Operation kann mehreren Ressourcen dieses Typs zugewiesen werden. Anders als die anderen beiden Ressourcentypen benötigt die Materialressource Lieferkurven als indirekte Ressourceninformation und verwendet Lagerbestandskurven als Ergebnis (s. Eigenschaften **ResourceName-FieldIndex** und **ResourceResultStockCurvePrefix**).

	Datentyp	Beschreibung
Parameter:		
⇨ resourceTableIndex	System.Int16	Index der Ressourcendatentabelle {0...24}.
Eigenschaftswert	VcResourceSchedulingResourceType	Typ der Ressourcendatentabelle **Standardwert:** vcTiming
	Mögliche Werte: .vcMaterial 1 .vcTiming -1 .vcWork 0	 Der Ressourcentyp ist "Material". Der Ressourcentyp ist "zeitdauerbestimmend". Der Ressourcentyp ist "Arbeit".

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ResourceType(0) =
VcResourceSchedulingResourceType.vcResSchedTiming
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.set_ResourceType(0,
VcResourceSchedulingResourceType.vcResSchedTiming);
```

ResultProcessingStepCount

Eigenschaft von VcResourceScheduler2

Nach einem Berechnungsdurchlauf kann mit dieser Eigenschaft die Anzahl der im Durchlauf geplanten Tasks erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Tasks **Standardwert:** 0

Code-Beispiel VB.NET

```
Dim i As Integer
i = VcGantt1.ResourceScheduler2.ResultProcessingStepCount()
```

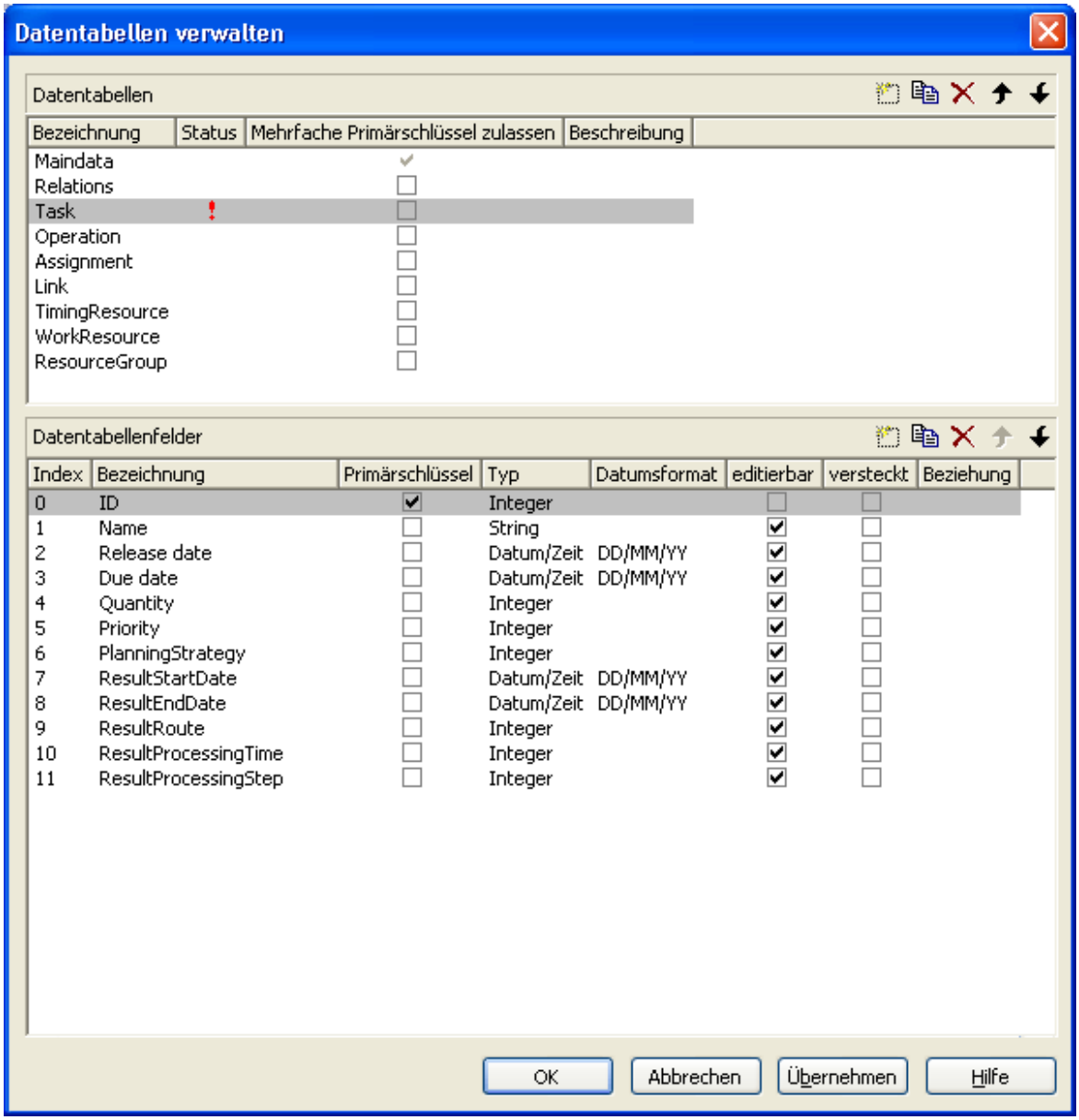
Code-Beispiel C#

```
int i = vcGantt1.ResourceScheduler2.ResultProcessingStepCount;
```

TaskDataTableName

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie den Namen der Aufgabentabelle setzen oder erfragen. Die Eigenschaft muss mit einem gültigen Tabellennamen besetzt werden.



	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Aufgabendatentabelle **Standardwert:** Empty string

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskDataTableName = "Task"
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskDataTableName("Task");
```

TaskDueDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem das Fälligkeitsdatum steht, an dem ein Arbeitsauftrag fertiggestellt sein muss. Falls kein gültiger Wert in dem Datenfeld gefunden wird, wird der von der Eigenschaft **TimeScaleEnd** im VcGantt-Objekt gesetzte Wert angenommen. Falls der entsprechende Arbeitsauftrag eingeplant werden soll, darf diese Eigenschaft während der Ressourcenplanung nicht auf -1 gesetzt sein. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 3.

Fälligkeitsdaten können über die Eigenschaft **ToleranceTimeOnASAPDue-Dates** pauschal mit einer Toleranz versehen werden. Bitte beachten Sie auch, dass Aufgaben mit vergleichsweise nahem Fälligkeitsdatum oder kurzem Zeitraum zwischen Freigabe- und Fälligkeitsdatum nicht automatisch bevorzugt eingeplant werden. Wenn Sie dies wünschen, dann müssen Sie die Prioritäten der Aufgaben selbst berechnen (s. Eigenschaft **TaskPriority-FieldIndex**).

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Fälligkeitsdatums vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 3
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskDueDateFieldIndex = 3;
```

TaskPlanningStrategyFieldIndex

Eigenschaft von VcResourceScheduler2

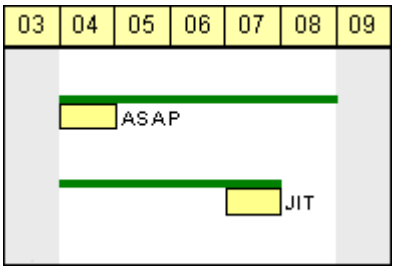
Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem individuell eine Planungsstrategie für eine Aufgabe gesetzt werden kann.

Falls ein Wert nicht gesetzt oder < 1 oder > 2 gesetzt wird, wird der gesetzte Wert aus der Eigenschaft **PlanningStrategy** verwendet. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 6,

Definierte Werte der Datenfeldinhalte {1...2}:

1 - so bald wie möglich (ASAP: as soon as possible)

2 - gerade rechtzeitig (JIT: just in time)



Bei der ASAP-Strategie wird die Aufgabe möglichst früh, bei der JIT-Strategie möglichst spät eingeplant. Die langen, linienförmigen Balken zeigen den möglichen Ausführungszeitraum er Aufgabe an; die kurzen, rechteckigen stellen die tatsächlich eingeplante Ausführungszeit dar. ASAP-Aufgaben erscheinen daher nach Möglichkeit links, JIT-Aufgaben rechts innerhalb des möglichen Ausführungszeitraums.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme der Daten zur Planungsstrategie vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskPlanningStrategyFieldIndex = 6
```

Code-Beispiel C#

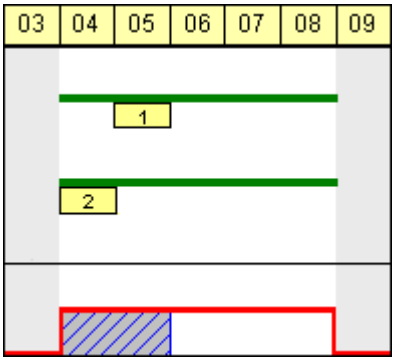
```
vcGantt1.ResourceScheduler2.TaskPlanningStrategyFieldIndex = 6;
```

TaskPriorityFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem für den Arbeitsauftrag eine Priorität hinterlegt werden kann.

Je höher der Wert für die Priorität ist, desto eher wird die Aufgabe in der Reihenfolge des Planungsprozesses berücksichtigt.



Eine Aufgabe mit höherer Priorität (2) wird vor einer Aufgabe mit niedrigerer Priorität (1) eingeplant.

Zur Beachtung: Bei über Verbindungen verknüpften Aufgaben sollte die Priorität mit Bedacht gewählt werden, d.h. bei ASAP sollten üblicherweise Vorgänger maximal die gleiche Priorität haben wie ihre Nachfolger, bei JIT sollten Vorgänger mindestens die gleiche Priorität haben wie ihre Nachfolger. Über die Priorität lassen sich Aufgaben gruppieren. So kann beispielsweise auf einer Maschine bei Gruppierung gleichartiger Tasks auf Vorbereitung und Reinigung zwischen diesen gleichartigen Tasks verzichtet werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme der Priorität vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskPriorityFieldIndex = 5
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskPriorityFieldIndex = 5;
```

TaskQuantityFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem die abzuarbeitende Arbeitsmenge für die Aufgabe steht. Diese Eigenschaft muss auf einen Wert ungleich -1 gesetzt sein.

Die Arbeitsmenge bestimmt indirekt die Dauer, die die Aufgabe zu ihrer Fertigstellung benötigt. Dabei kann die Dauer auch durch die Effizienz bei den Ressourcen (s. **ResourceEfficiencyFieldIndex**) sowie durch Faktoren bei Operationen (s. **OperationLoadPerItemFieldIndex**) und bei Zuweisungen (s. **AssignmentLoadOrConsumptionPerItemFieldIndex**) beeinflusst werden.

Falls kein gültiger Wert in dem Datenfeld gefunden wird, wird eine Arbeitsmenge von 1 angenommen. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 4.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme der Auftragsmenge vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskQuantityFieldIndex = 4
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskQuantityFieldIndex = 4;
```

TaskReleaseDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem das Freigabedatum steht, ab dem ein Arbeitsauftrag eingeplant werden darf. Diese Eigenschaft muss auf einen Wert ungleich -1 gesetzt sein.

Falls kein gültiger Wert in dem Datenfeld gefunden wird, wird der von der Eigenschaft **TimeScaleStart** im VcGantt-Objekt gesetzte Wert angenommen. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 2.

Freigeabedaten können über die Eigenschaft **ToleranceTimeOnJITReleaseDates** pauschal mit einer Toleranz versehen werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Freigabedatums vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskReleaseDateFieldIndex = 2
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2 .TaskReleaseDateFieldIndex = 2;
```

TaskResultEndDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem das ermittelte Enddatum der spätesten zugehörigen, eingeplanten Operation abgelegt wird. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 8.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Enddatums vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultEndDateFieldIndex = 8
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskResultEndDateFieldIndex = 8;
```

TaskResultPostEndDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in das nach einer erfolgten Planung die Endezeit der Nachlaufphase einer Aufgabe abgelegt wird. Wenn die Nachlaufzeit 0 ist, dann ist dieses Datum identisch mit dem

Wert in dem Datenfeld, das über die Eigenschaft **TaskResultEndDateField-Index** adressiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Enddatums der Nachlaufphase vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultPostEndDateFieldIndex = 15
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationPostEndDateFieldIndex = 15;
```

TaskResultPreparationStartDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in das nach einer erfolgten Planung die Startzeit der Vorbereitungsphase einer Aufgabe abgelegt wird. Wenn die Vorbereitungszeit 0 ist, dann ist dieses Datum identisch mit dem Wert in dem Datenfeld, das über die Eigenschaft **Task-ResultStartDateFieldIndex** adressiert wird.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Startdatums der Vorbereitungsphase vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.OperationResultPreparationStartDateFieldIndex = 10;
```

TaskResultProcessingStepFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem die beim vorangegangenen Planungsdurchlauf festgelegte Abfolgenummer der Aufgabe steht. Dieser Wert ist hilfreich um zu erkennen, welche Aufgabe als erste aufgrund von Ressourcenengpässen nicht eingeplant werden konnte.

Die als erste eingeplante Aufgabe erhält die Nummer 0, die folgenden Aufgaben fortlaufend aufsteigende Nummern. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 11.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme der Abfolgenummer vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultProcessingStepFieldIndex = 11
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskResultProcessingStepFieldIndex = 11;
```

TaskResultProcessingTimeFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem die beim vorangegangenen Planungsprozess ermittelte Gesamtverarbeitungszeit aller zu der Aufgabe gehörigen, eingeplanten Operationen abgelegt wird. Es ist die Zeitspanne zwischen dem Startdatum der ersten Operation und dem Enddatum der letzten Operation. Die Angabe erfolgt in der eingestellten BaseTimeUnit. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 10.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme der Verarbeitungszeit vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultProcessingTimeFieldIndex = 10
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskResultProcessingTimeFieldIndex = 10;
```

TaskResultRouteFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem der Name einer Route steht, die während der Planung für die Aufgabe durch die Ressourcenplanung ausgewählt wurde.

Diese Eigenschaft sollte nicht auf -1 gesetzt sein, wenn die Eigenschaft **OperationRouteFieldIndex** verwendet wird. In der Abbildung zu **Task-DataTableName** ist der Index beispielsweise 9.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Namens der Route vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultRouteFieldIndex = 9
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskResultRouteFieldIndex = 9;
```

TaskResultStartDateFieldIndex

Eigenschaft von VcResourceScheduler2

Der Index bezeichnet ein Datenfeld in der Aufgabentabelle, in dem das ermittelte Startdatum der frühesten zugehörigen, eingeplanten Operation abgelegt wird. In der Abbildung zu **TaskDataTableName** ist der Index beispielsweise 7.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Index des Datenfeldes in der Aufgabendatentabelle, das für die Aufnahme des Startdatums vorgesehen ist. {-1...AnzahlDatenfelderInAufgabendatentabelle -1}. Der Wert -1 bedeutet, dass dieser Eigenschaft kein Datenfeld der Aufgabendatentabelle zugeordnet ist. **Standardwert:** -1

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.TaskResultStartDateFieldIndex = 7
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.TaskResultStartDateFieldIndex = 7;
```

ToleranceTimeOnASAPDueDates

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie eine Toleranz in Basiszeiteinheiten (s. Eigenschaft **BaseTimeUnit**) setzen oder erfragen, die sich auf die Fälligkeitsdaten von Aufträgen bezieht und nur bei der Planungsstrategie "As Soon As Possible" (ASAP) wirksam ist.

Beim Planungsprozess werden die Fälligkeitsdaten der Aufgaben um die angegebene Anzahl von Einheiten später gesetzt, womit sich der erlaubte Planungszeitraum für die Aufgabe verlängert. Diese Eigenschaft ist nützlich, um z.B. herauszufinden, ob bei Verlängerung des Planungszeitraums alle Operationen der Aufgaben eingeplant werden können und erspart, die Fälligkeitsdaten der Aufgaben individuell zu ändern und auszuprobieren.

Sehen Sie bitte auch **ToleranceTimeOnJITReleaseDates**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Basiszeiteinheiten {>=0} **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnASAPDueDates = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ToleranceTimeOnASAPDueDates = 1;
```

ToleranceTimeOnJITReleaseDates

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie eine Toleranz in Basiszeiteinheiten (s. Eigenschaft **BaseTimeUnit**) setzen oder erfragen, die sich auf die Freigabedaten von Aufträgen bezieht und nur bei der Planungsstrategie "Just In Time" (JIT) wirksam ist.

Beim Planungsprozess werden die Freigabedaten der Aufgaben um die angegebene Anzahl von Einheiten früher gesetzt, womit sich der erlaubte Planungszeitraum für die Aufgabe verlängert. Diese Eigenschaft ist nützlich, um z.B. herauszufinden, ob bei Verlängerung des Planungszeitraums alle Operationen der Aufgaben eingeplant werden können und erspart, die Freigabedaten der Aufgaben individuell zu ändern.

Sehen Sie bitte auch **ToleranceTimeOnASAPDueDates**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Basiszeiteinheiten {>=0} **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnJITReleaseDates = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ToleranceTimeOnJITReleaseDates = 1;
```

ToleranceTimeOnStartLockDates

Eigenschaft von VcResourceScheduler2

Mit dieser Eigenschaft können Sie eine Toleranz in Basiszeiteinheiten (s. Eigenschaft **BaseTimeUnit**) für ein festes Startdatum einer Operation (s. **OperationStartLockDateFieldIndex**) setzen oder erfragen. Eine Operation kann dadurch um die spezifizierte Anzahl der Einheiten später eingeplant werden als vorgegeben, wenn die zu belegenden Ressourcen zum festgelegten Startdatum nicht verfügbar sind.

.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Basiszeiteinheiten {>=0} **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.ToleranceTimeOnStartLockDates = 1
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.ToleranceTimeOnStartLockDates = 1;
```

WorkInProcessType

Eigenschaft von VcResourceScheduler2

Diese Eigenschaft setzt die Einheit, in der der Fertigstellungsgrad angegeben wird (s. **OperationWorkInProcessFieldIndex**).

	Datentyp	Beschreibung
Eigenschaftswert	VcResourceSchedulingWorkInProcessType	Einheit des Fertigstellungsgrades **Standardwert:** vcResSchedWIPPercentage
	Mögliche Werte:	
	.vcResSchedWIPCompleted 0	Einheit in fertiggestellten Arbeitsmengen
	.vcResSchedWIPPercentage -1	Einheit in Prozentzahlen (0...100)
	.vcResSchedWIPRemaining 1	Einheit in noch zu fertigenden Arbeitsmengen

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.WorkInProcessType =
VcResourceSchedulingWorkInProcessType.vcResSchedWIPCompleted
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.WorkInProcessType =
VcResourceSchedulingWorkInProcessType.vcResSchedWIPCompleted;
```

WritingDebugFilesEnabled

Eigenschaft von VcResourceScheduler2

Wenn diese Eigenschaft auf **true** gesetzt wird, wird in das aktuelle Verzeichnis eine Debug-Datei mit dem Namen **OPS_debug.txt** geschrieben, die für Analysezwecke wichtig sein können.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	**true**: Debug-Dateien können in das aktuelle Verzeichnis geschrieben werden. **false**: Debug-Dateien können nicht in das aktuelle Verzeichnis geschrieben werden. **Standardwert:** false

Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.WritingDebugFilesEnabled = True
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.WritingDebugFilesEnabled = true;
```

Methoden

DetermineIDOfFirstOperationByTaskID

Methode von VcResourceScheduler2

Diese Methode legt über die angegebene TaskID die ID der ersten Operation einer Aufgabe (Task) fest und hilft so dem Entwickler bei der Aktualisierung des Datenfelds einer Verbindung, das die ID der ersten Operation einer Aufgabe enthält.

Weitere Informationen dazu finden Sie auch bei der Beschreibung der beiden VcResourceScheduler-Eigenschaften **LinkPredecessorOperationIDField-Index** und **LinkSuccessorOperationIDFieldIndex**.

	Datentyp	Beschreibung
Parameter:		
⇨ taskID	System.String	ID einer Aufgabe aus der entsprechenden Datentabelle, die zuvor über die VcResourceScheduler2-Eigenschaft **TaskDataTableName** gesetzt wurde.
Rückgabewert	System.String	ID der ersten Operation aus der entsprechenden Datentabelle, die zuvor über die VcResourceScheduler2-Eigenschaft **OperationDataTableName** gesetzt wurde.

DetermineIDOfLastOperationByTaskID

Methode von VcResourceScheduler2

Diese Methode legt über die angegebene TaskID die ID der letzten Operation einer Aufgabe (Task) fest und hilft so dem Entwickler bei der Aktualisierung des Datenfelds einer Verbindung, das die ID der letzten Operation einer Aufgabe enthält.

Weitere Informationen dazu finden Sie auch bei der Beschreibung der beiden VcResourceScheduler-Eigenschaften **LinkPredecessorOperationIDField-Index** und **LinkSuccessorOperationIDFieldIndex**.

	Datentyp	Beschreibung
Parameter:		
taskID	System.String	ID einer Aufgabe aus der entsprechenden Datentabelle, die zuvor über die VcResourceScheduler2-Eigenschaft **TaskDataTableName** gesetzt wurde.
Rückgabewert	System.String	ID der letzten Operation aus der entsprechenden Datentabelle, die zuvor über die VcResourceScheduler2-Eigenschaft **OperationDataTableName** gesetzt wurde.

Process

Methode von VcResourceScheduler2

Mit dieser Methode können Sie die Ressourcenplanung starten, nachdem Sie alle gewünschten Eigenschaften gesetzt haben. Für Fortschrittsmeldungen s. auch Ereignis **ResourceSchedulingProgressing**. Zudem werden Warnungen über **ResourceSchedulingWarning** ausgegeben.

	Datentyp	Beschreibung
Rückgabewert	System.Boolean	**true**: Die Ressourcenplanung wurde fehlerfrei durchgeführt. **false**: Die Ressourcenplanung wurde entweder nicht fehlerfrei durchgeführt oder abgebrochen. Fehler sind - falls so eingestellt - in dem Datenfeld, das über die Eigenschaft **OperationResultStatus-FieldIndex** adressiert wird, für jeden Arbeitsauftrag erfragbar.

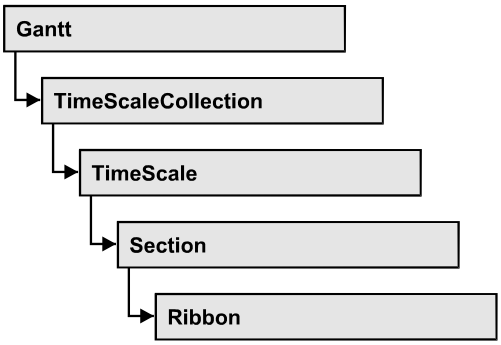
Code-Beispiel VB.NET

```
VcGantt1.ResourceScheduler2.Process()
```

Code-Beispiel C#

```
vcGantt1.ResourceScheduler2.Process();
```

7.66 VcRibbon



Ein Objekt des Typs VcRibbon stellt einen definierten Zeitstreifen in einer Zeitskala mit gleichen Einheiten und einheitlicher Skalierung dar. Bei diesem Objekt können Eigenschaften wie Hintergrundfarbe, Trennlinien, Schrifttyp, -farbe, -größe, -ausrichtung u.a. eingestellt werden.

Eigenschaften

- CalendarName
- DateOutputFormat
- Font
- FontColor
- MajorTicks
- MinorTicks
- ObserveDST
- PatternBackgroundColorAsARGB
- PatternColorAsARGB
- PatternEx
- Position
- ReferenceDate
- TextAlignment
- TickColor
- TickPosition
- Type
- UnitSeparation
- UseReferenceDate

Eigenschaften

CalendarName

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie den Kalendernamen erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Kalendername

DateOutputFormat

Eigenschaft von VcRibbon

Mit dieser Eigenschaft stellen Sie das Ausgabeformat von Terminen für den Zeitstreifen ein. 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 "Uhr" (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

xC/XC: Sie können das Datum als maximal zehngliedrige, einfache Aufwärtszählung ab einem bestimmten Referenzdatum gestalten, z.B. "15:05:07:16:00". Dieses Beispiel datiert 15 Monate, 5 Tage, 7 Stunden, 16 Minuten und 0 Sekunden später als das gesetzte Referenzdatum. Die Notation dazu ist: **xC44:C33:C22:C11:C00**. Es sollen je 2 Stellen für die Glieder 4...0 vorgesehen werden, mit einem vorausgehenden "-" Symbol falls der Wert negativ ist. Die Trennzeichen sind variabel und könnten durch andere ersetzt werden. "x" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, aber kein "+"Symbol, falls er positiv ist. "X" bedeutet: Es soll ein vorausgehendes -Symbol gesetzt werden, falls der Wert negativ ist, und ebenso ein "+"Symbol, falls er positiv ist. Im Dialog **Zeitskalenabschnitt ... bearbeiten** sollten die Felder **Referenzdatum verwenden** und **Hauptmarkierung am Referenzdatum ausrichten** aktiviert sein, ebenso sollte **Serielle Beschriftungen** auf **Keine** gesetzt sein. Das Referenzdatum wird zur Laufzeit über die Methode **VcRibbon.setReferenceDate** gesetzt und überschreibt die Werte des Dialogs.

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.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Datumsformat {DMYhms:;/}

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.DateOutputFormat = "DD.MMM.YYYY hh:mm:ss"
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.DateOutputFormat = "DD.MMM.YYYY hh:mm:ss";
```

Font

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Schriftattribute des Zeitstreifens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftattribute des Zeitstreifens

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon
Dim newFont As Font

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
newFont = New Font("Times New Roman", 14, FontStyle.Italic)
ribbon.Font = newFont
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
Font newFont = new Font("Times New Roman", 14, FontStyle.Italic);
ribbon.Font = newFont;
```

FontColor

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Schriftfarbe des Zeitstreifens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.FontColor = Color.Blue
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.FontColor = Color.LightSteelBlue;
```

MajorTicks

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Zeiteinheiten eine Hauptmarkierung erscheint. Die Zeiteinheit hängt vom Typ des verwendeten Zeitstreifens ab. Die Hauptmarkierungen werden beschriftet, wenn der Platz dafür ausreicht. Diese Eigenschaft können Sie auch im Dialog **Zeitskalenabschnitt bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Einheiten zwischen zwei Hauptmarkierungen

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.MajorTicks = 7
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.MajorTicks = 7;
```

MinorTicks

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie erfragen oder festlegen, nach wie vielen Zeiteinheiten eine Nebenmarkierung erscheint. Die Zeiteinheit hängt vom Typ des verwendeten Zeitstreifens ab. Die Nebenmarkierungen werden nicht beschriftet. Diese Eigenschaft können Sie auch im Dialog **Zeitskalenabschnitt bearbeiten** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl Einheiten zwischen zwei Nebenmarkierungen

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.MinorTicks = 1
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.MinorTicks = 1;
```

ObserveDST

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob für diesen Zeitstreifen die Sommerzeit in der Darstellung berücksichtigt wird oder nicht.

	Datentyp	Beschreibung
Eigenschaftswert	VcRibbonObserveDST **Mögliche Werte:** .vcGODDefault 9999 .vcRODNo 0 .vcRODYes 1	Sommerzeit wird/wird nicht berücksichtigt. Standardeinstellung aus der .INI-Datei wird verwendet Sommerzeit wird nicht berücksichtigt Sommerzeit wird berücksichtigt

PatternBackgroundColorAsARGB

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Zeitstreifens erfragen oder festlegen. Farbwerte haben einen Transparenz- oder Alphawert, einen Rot-, einen Blau- und einen Grünanteil (ARGB-Wert) im Zahlenbereich von 0..255. 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.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.PatternBackgroundColorAsARGB = &h88FF0A06
```

PatternColorAsARGB

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Musterfarbe des Zeitstreifens 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.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

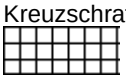
Code-Beispiel VB.NET

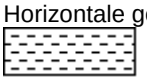
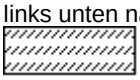
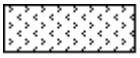
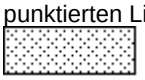
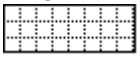
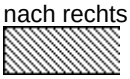
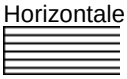
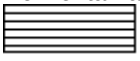
```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

Set timeScale = VcGantt1.TimeScaleCollection.Active
Set ribbon = timeScale.Ribbon(0, 0)
ribbon.PatternColorAsARGB = &h88FF0A06
```

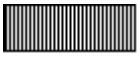
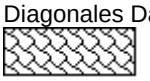
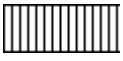
PatternEx

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie für den Hintergrund des Zeitstreifens ein Muster setzen oder erfragen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte:	
	.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 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 

Position

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Position des Zeitstreifens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcRibbonPosition **Mögliche Werte:** .vcRPBottom 2 .vcRPNone 0 .vcRPTop 1	Position des Zeitstreifens unten keine oben

ReferenceDate

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie das Referenzdatum erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Referenzdatum

TextAlignment

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Ausrichtung der Beschriftung der Hauptstriche des Zeitstreifens festlegen oder erfragen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcHorizontalRibbonTextAlignment **Mögliche Werte:** .vcRTAtTickAligned 1039 .vcRTHorCenterAligned -1 .vcRTLeftAligned -3 .vcRTRightAligned -2	Art der Textausrichtung Text an der Hauptmarkierung ausgerichtet Text horizontal mittig zwischen zwei Hauptmarkierungen ausgerichtet Text linksbündig zwischen zwei Hauptmarkierungen ausgerichtet Text rechtsbündig zwischen zwei Hauptmarkierungen ausgerichtet

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.TextAlignment = VcHorizontalRibbonTextAlignment.vcRTLeftAligned
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.TextAlignment = VcHorizontalRibbonTextAlignment.vcRTLeftAligned;
```

TickColor

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Farbe der Markierungen erfragen oder festlegen.

	Datentyp	**Beschreibung**
Eigenschaftswert	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** 0,0,0

TickPosition

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie die Position der Trennstriche erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcRibbonTickPosition **Mögliche Werte:** .vcTPAbove 1044 .vcTPBelow 1045	Position der Markierungen oberhalb unterhalb

Type

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie den Typ des Zeitstreifens festlegen oder erfragen. Zur Verfügung stehen die unten aufgeführten Einheitenstreifen.

	Datentyp	Beschreibung
Eigenschaftswert	VcRibbonType **Mögliche Werte:** .vcDayRibbon 5 .vcFiscalQuarterRibbon 3002 .vcFiscalYearRibbon 3001 .vcHourRibbon 6 .vcMinuteRibbon 7 .vcMonthRibbon 3 .vcQuarterRibbon 10 .vcSecondRibbon 9 .vcShiftRibbon 8 .vcWeekRibbon 4 .vcYearRibbon 1	Typ des Zeitstreifens Zeitstreifen mit Einheit **Tag** Zeitstreifen mit Einheit **Fiskalisches Quartal** Zeitstreifen mit Einheit **Fiskalisches Jahr** Zeitstreifen mit Einheit **Stunde** Zeitstreifen mit Einheit **Minute** Zeitstreifen mit Einheit **Monat** Zeitstreifen mit Einheit **Quartal** Zeitstreifen mit Einheit **Sekunde** Zeitstreifen mit Schichten als Einheit Zeitstreifen mit Einheit **Woche** Zeitstreifen mit Einheit **Jahr**

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.Type = VcRibbonType.vcWeekRibbon
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.Type = VcRibbonType.vcWeekRibbon;
```

UnitSeparation

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie das Aussehen der Hauptmarkierungen (Major Ticks) des Zeitstreifens festlegen oder erfragen. Zur Verfügung stehen Vollstrich, Teilstrich und kein Strich.

	Datentyp	Beschreibung
Eigenschaftswert	VcUnitSeparation **Mögliche Werte:** .vcUSFullLine 4 .vcUSNone 1 .vcUSTick 1035	Aussehen der Hauptmarkierung Trennstrich durchgezogen kein Trennstrich Teiltrennstrich

Code-Beispiel VB.NET

```
Dim timeScale As VcTimeScale
Dim ribbon As VcRibbon

timeScale = VcGantt1.TimeScaleCollection.Active
ribbon = timeScale.Ribbon(0, 0)
ribbon.UnitSeparation = VcUnitSeparation.vcUSTick
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
ribbon.UnitSeparation = VcUnitSeparation.vcUSTick;
```

UseReferenceDate

Eigenschaft von VcRibbon

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob der Zeitstreifen ein Referenzdatum verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Der Zeitstreifen verwendet (True) / verwendet kein (False) Referenzdatum **Standardwert:** False

7.67 VcScheduler

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	SystemInt.32	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	SystemInt.32	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) **Standardwert:** 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	SystemInt.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	SystemInt.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	SystemInt.32	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	SystemInt.32	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	SystemInt.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	SystemInt.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**
Parameter:		
⇨ startDate	System.DateTime	Gewünschtes Anfangsdatum
⇨ endDate	System.DateTime	Gewünschtes Enddatum
Rückgabewert	System.Boolean	Die Projektdaten wurden erfolgreich berechnet (True) / nicht berechnet (False)

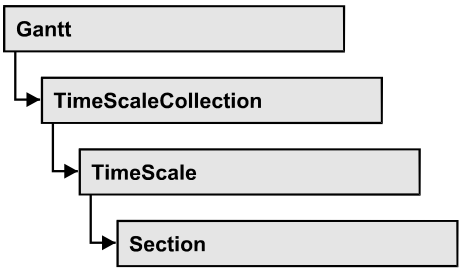
Code-Beispiel VB.NET

```
' Vorwärtsberechung (ASAP)
VcScheduler.ScheduleProject(2.5.2017, newDate(0))

' Rückwärtsberechnung (JIT)
VcScheduler.ScheduleProject(newDate(0), 2.5.2017)
```

Code-Beispiel C#

```
// Vorwärtsberechung (ASAP)
vcScheduler.ScheduleProject(2.5.2017, newDate(0));

// Rückwärtsberechnung (JIT)
vcScheduler.ScheduleProject(newDate(0), 2.5.2017);
```

7.68 VcSection



Ein Objekt vom Typ VcSection stellt einen Abschnitt einer Zeitskala dar.

Eigenschaften

- CalendarGrid
- DateLineGrid
- LineColor
- NonWorkIntervalsCollapsed
- Ribbon
- StartDate
- TimeUnit
- UnitWidth
- UnitWidthEx

Eigenschaften

CalendarGrid

Nur-Lese-Eigenschaft von VcSection

Mit dieser Eigenschaft können Sie eines der im Zeitskalenabschnitt verwendeten Kalenderliniengitter erfragen.

Die Eigenschaft CalendarGrid ist eine indizierte Eigenschaft, die in C# über die Methode get_CalendarGrid (gridIndex) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ gridIndex	System.Int16	Index der Kalenderliniengitters
Eigenschaftswert	VcCalendarGrid	Kalenderliniengitter-Objekt

DateLineGrid

Nur-Lese-Eigenschaft von VcSection

Mit dieser Eigenschaft haben Sie Zugriff auf das VcDateLineGrid-Objekt, mit dem Sie Zeitabschnitte (Tage, Wochen, Monate, etc.) durch vertikale Linien markieren können.

Die Eigenschaft DateLineGrid ist eine indizierte Eigenschaft, die in C# über die Methode get_DateLineGrid (gridIndex) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ gridIndex	System.Int16	Index der Terminliniengitters
Eigenschaftswert	VcDateLineGrid	Terminliniengitter-Objekt

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection
Dim dateLineGrid As VcDateLineGrid

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
dateLineGrid = section.DateLineGrid(0)
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
VcDateLineGrid dateLineGrid = section.get_DateLineGrid(0);
```

LineColor

Nur-Lese-Eigenschaft von VcSection

Mit dieser Eigenschaft können Sie die Farbe der (Rand)linie n **aller** Zeitsskalenabschnitte gemeinsam festlegen oder erfragen. Die Eigenschaft gibt die Linienfarbe des ersten Zeitskalenabschnitts zurück,. Es ist nicht möglich, für jeden Abschnitt eine eigene Farbe zu definieren.

	Datentyp	Beschreibung
Parameter:		
⇨ Rückgabewert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

NonWorkIntervalsCollapsed

Eigenschaft von VcSection

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die arbeitsfreien Bereiche in diesem Zeitskalenabschnitt kollabiert werden sollen. Sie können diese Eigenschaft auch im Unterdialog **Zeitskalenabschnitt bearbeiten** von **Zeitskala festlegen** setzen. Dieser Dialog ist zu erreichen über die Eigenschaftenseite **Objekte**, Schaltfläche **Zeitskalen...**.

Hinweis: Wenn Sie den Eigenschaftswert zur Laufzeit verändern, beachten Sie bitte, dass sich der sichtbare Zeitskalenbereich verschiebt. Möchten Sie sicherstellen, dass Sie links stets das gleiche Bezugsdatum haben, rufen Sie bitte folgende Methode auf:

```
Set_NonWorkIntervalsCollapsed(vcGantt1, true);


private static void Set_NonWorkIntervalsCollapsed(VcGantt gantt, bool collapse)
    {
       DateTime dt_left = new DateTime();
       DateTime dt_right = new DateTime();
       gantt.GetCurrentViewDates(ref dt_left, ref dt_right);


gantt.TimeScaleCollection.Active.get_Section(0).NonWorkIntervalsCollapsed = collapse;


       gantt.ScrollToDate(dt_left, VcHorizontalAlignment.vcLeftAligned, 0);
```

```
}
```

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Arbeitsfreie Bereiche werden/werden nicht kollabiert.

Code-Beispiel VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale
Dim section As VcSection

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.Active
section = timeScale.Section(1)
section.NonWorkIntervalsCollapsed = True
```

Code-Beispiel C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.Active;
VcSection section = timeScale.get_Section(1);
section.NonWorkIntervalsCollapsed = true;
```

Ribbon

Eigenschaft von VcSection

Mit dieser Eigenschaft haben Sie Zugriff auf die einzelnen Zeitstreifen eines Zeitskalenabschnitts.

Die Eigenschaft Ribbon ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_Ribbon (ribbonIndex, pvn) und get_Ribbon (ribbon-Index) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ ribbonIndex	System.Int16	Index des Zeitskalenstreifens
Eigenschaftswert	VcRibbon	Zeitstreifen-Objekt

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection
Dim ribbon As VcRibbon

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
ribbon = section.Ribbon(0)
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
VcRibbon ribbon = section.get_Ribbon(0);
```

StartDate

Eigenschaft von VcSection

Mit dieser Eigenschaft können Sie das Anfangsdatum eines Zeitskalenabschnitts erfragen oder festlegen. Das Anfangsdatum des ersten Zeitskalenabschnitts (Section 0) wird durch den Anfang der Zeitskala festgelegt und darf hier nicht geändert, sondern nur erfragt werden. Sie dürfen auch kein Anfangsdatum wählen, das außerhalb des Bereichs der Zeitskala liegt.

	Datentyp	Beschreibung
Eigenschaftswert	System.DateTime	Anfangsdatum des Zeitskalenabschnittes

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.StartDate = "21.06.14"
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.StartDate = Convert.ToDateTime("21.06.14");
```

TimeUnit

Eigenschaft von VcSection

Mit dieser Eigenschaft können Sie die einem Zeitskalenabschnitt zu Grunde liegende Zeiteinheit erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	VcTimeUnit **Mögliche Werte:** .vcDay 5 .vcHour 6 .vcMinute 7 .vcSecond 8	Zeiteinheit des Zeitskalenabschnitts Zeiteinheit **Tag** Zeiteinheit **Stunde** Zeiteinheit **Minute** Zeiteinheit **Sekunde**

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.TimeUnit = VcTimeUnit.vcHour
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.TimeUnit = VcTimeUnit.vcHour;
```

UnitWidth

Eigenschaft von VcSection

Mit dieser Eigenschaft können Sie die Breite pro Zeiteinheit in 1/100 Millimetern für einen Skalenabschnitt erfragen oder festlegen. Sie können diese Eigenschaft auch im Dialog **Zeitskala festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Breite der Einheiten (1/100 mm)

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.UnitWidth = 660
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.UnitWidth = 660;
```

UnitWidthEx

Eigenschaft von VcSection

Diese Eigenschaft unterscheidet sich von **UnitWidth** lediglich durch den Datentyp **Double** und ist damit genauer als **UnitWidth**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Double	Breite der Einheiten (1/100 mm)

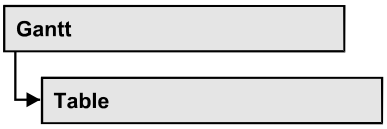
Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
section.UnitWidthEx = 660.0
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
section.UnitWidthEx = 660.0;
```

7.69 VcTable



Ein Objekt des Typs VcTable kontrolliert die grafische Aufbereitung des Tabellenteils der Darstellung: die Spaltenüberschriften, Spaltenbreiten und die verfügbaren Formate.

Eigenschaften

- ColumnTitle
- ColumnWidth
- Name
- NoOfColumns
- Position
- TableFormatCollection
- UpdateBehaviorName
- Visible

Methoden

- IdentifyFormatField
- OptimizeColumnWidth

Eigenschaften

ColumnTitle

Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie für jede Spalte der Tabelle die gewünschte Überschrift setzen. Diese Eigenschaft können Sie auch im Dialog **Tabelle bearbeiten** einstellen.

Die Eigenschaft ColumnTitle ist eine indizierte Eigenschaft, die in C# über die beiden Methoden set_ColumnTitle (colNumber, pvn) und get_Column-Title (colNumber) angesprochen wird.

Hinweis: Der Index beginnt bei 1.

	Datentyp	Beschreibung
Parameter:		
⇨ colNumber	System.Int16	Nummer der Tabellenspalte
Eigenschaftswert	System.String	Spaltentitel

Code-Beispiel VB.NET

```
Dim table As VcTable

table = VcGantt1.LeftTable
table.ColumnTitle(2) = "ID"
```

Code-Beispiel C#

```
VcTable table = vcGantt1.LeftTable;
table.set_ColumnTitle(2, "ID");
```

ColumnWidth

Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie für jede Spalte der Tabelle die gewünschte Breite setzen. Diese Eigenschaft können Sie auch im Dialog **Tabelle bearbeiten** einstellen.

Die Eigenschaft ColumnWidth ist eine Indexed Property, die in C# über die beiden Methoden set_ColumnWidth(colNumber, pvn) und get_ColumnWidth(colNumber) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ colNumber	System.Int16	Nummer der Tabellenspalte
Eigenschaftswert	System.Int32	Spaltenbreite in 1/100 mm

Code-Beispiel VB.NET

```
Dim table As VcTable

table = VcGantt1.LeftTable
table.ColumnWidth(1) = 1500
```

Code-Beispiel C#

```
VcTable table = vcGantt1.LeftTable;
table.set_ColumnWidth(1,1500);
```

Name

Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie für die Tabelle einen Namen setzen oder erfragen. Diese Eigenschaft können Sie auch im Dialog **Tabelle bearbeiten** einstellen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Tabelle

NoOfColumns

Nur-Lese-Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie die Anzahl der Spalten der Tabelle erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Tabellenspalten

Position

Nur-Lese-Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie erfragen, ob die Tabelle rechts oder links vom Diagramm dargestellt wird.

	Datentyp	Beschreibung
Eigenschaftswert	VcTablePosition **Mögliche Werte:** .vcLeftTable 0 .vcRightTable 1	Position der Tabelle Tabelle links vom Diagramm Tabelle rechts vom Diagramm

Code-Beispiel VB.NET

```
Dim table As VcTable

table = VcGantt1.LeftTable
MsgBox(table.Position)
```

Code-Beispiel C#

```
VcTable table = vcGantt1.LeftTable;
MessageBox.Show(table.Position.ToString());
```

TableFormatCollection

Nur-Lese-Eigenschaft von VcTable

Mit dieser Eigenschaft haben Sie Zugriff auf das TableFormatCollection-Objekt, in dem alle zur Verfügung stehenden Tabellenformate zusammengefasst sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcTableFormatCollection	TableFormatCollection-Objekt

Code-Beispiel VB.NET

```
Dim table As VcTable
Dim formatCltn As VcTableFormatCollection

table = VcGantt1.LeftTable
formatCltn = table.TableFormatCollection
```

Code-Beispiel C#

```
VcTable table = vcGantt1.LeftTable;
VcTableFormatCollection formatCltn = table.TableFormatCollection;
```

UpdateBehaviorName

Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

Visible

Eigenschaft von VcTable

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Tabelle sichtbar ist oder nicht.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Tabelle sichtbar/unsichtbar

Code-Beispiel VB.NET

```
Dim table As VcTable

table = VcGantt1.LeftTable
table.Visible = True
```

Code-Beispiel C#

```
VcTable table = vcGantt1.LeftTable;
table.Visible = true;
```

Methoden

IdentifyFormatField

Methode von VcTable

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
Parameter:		
⇨ x	System.Int32	X-Koordinate der Position
⇨ y	System.Int32	Y-Koordinate der Position
⇦ format	VcTableFormat	Identifiziertes Format
⇦ formatFieldIndex	System.Int16	Format-Feldindex
Rückgabewert	System.Boolean	Ein Formatfeld befindet sich/befindet sich nicht an der angegebenen Position

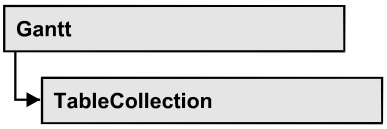
OptimizeColumnWidth

Methode von VcTable

Mit dieser Methode können Sie die optimierte Breite einer Spalte berechnen. Dabei entscheidet die Länge des längsten Textes in dieser Spalte über die Spaltenbreite. Die Angabe von ColumnNo = 0 bewirkt, dass alle Spalten optimiert werden.

	Datentyp	Beschreibung
Parameter:		
⇨ columnNo	System.Int16	Spaltennummer
Rückgabewert	Void	

7.70 VcTableCollection



Ein Objekt des Typs VcTableCollection beinhaltet alle bestehenden Tabellen. Über **For Each table InTableCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Tabellen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **TableByName** und **TableByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Tabellen kann über die Eigenschaft **Count** erfragt werden. Die aktuelle Tabelle kann durch die Eigenschaft **Active** gesetzt oder erfragt werden.

Eigenschaften

- Active
- Count

Methoden

- FirstTable
- GetEnumerator
- NextTable
- TableByIndex
- TableByName

Eigenschaften

Active

Eigenschaft von VcTableCollection

Mit dieser Eigenschaft kann die aktuell für die Darstellung verwendete Tabelle erfragt oder gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcTable	Aktuell verwendete Tabelle

Count

Nur-Lese-Eigenschaft von VcTableCollection

Mit dieser Eigenschaft kann die Anzahl der Tabellen in der Table-Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	Integer	Number of tables

Methoden

FirstTable

Methode von VcTableCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Tabelle des TableCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextTable** über die nachfolgenden Tabellen zu iterieren. Existiert keine Tabelle in der TableCollection, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcTable	First table

GetEnumerator

Methode von VcTableCollection

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 Tabellen-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	reference object

NextTable

Methode von VcTableCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Tabellen des TableCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstTable** den Initialwert erfasst haben. Sind alle Tabellen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcTable	Next table

TableByIndex

Methode von VcTableCollection

Mit dieser Methode können Sie auf eine einzelne Tabelle über ihren Index zugreifen. Existiert keine Tabelle an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index der Tabelle
Rückgabewert	VcTable	Ermitteltes Tabellenobjekt

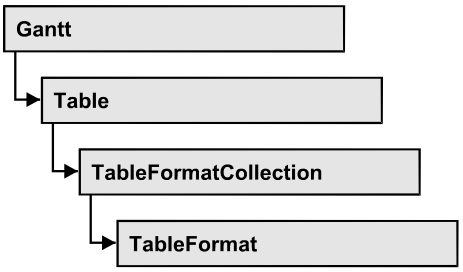
TableByName

Methode von VcTableCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Tabellenobjekt zugreifen. Existiert kein Tabellenobjekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
tableName	System.String	Name der Tabelle
Rückgabewert	VcTable	Table

7.71 VcTableFormat



Ein Objekt vom Typ VcTableFormat legt Inhalt und Erscheinungsbild einer Zeile der Tabelle fest. In einer Zeile der Tabelle stehen entweder die Daten eines Vorgangs oder eine Gruppenüberschrift. In einem Tabellenformat kann man für jedes Tabellenfeld (definiert durch die Angabe der Spalte) separat festlegen, welches Datenfeld es enthalten soll, sowie Schriftattribute (Schriftart und -farbe), Hintergrundfarbe, horizontale Ausrichtung und Ränder einstellen.

Verfügbare Tabellenformate:

- StandardList (für normale Vorgänge)
- ListFormat2 (alternatives Tabellenformat für normale Vorgänge, über Filter zuweisbar)
- ListFormat3 (alternatives Tabellenformat für normale Vorgänge, über Filter zuweisbar)
- Subtitle (für Gruppenüberschriften, wenn die Gruppe expandiert ist)
- Subtitle_n (für Gruppenüberschriften bei mehrstufiger Gruppierung, wenn die Gruppe expandiert ist)
- Collapsed (für Gruppenüberschriften, wenn die Gruppe kollabiert ist)
- Collapsed_n (für Gruppenüberschriften bei mehrstufiger Gruppierung, wenn die Gruppe kollabiert ist)
- Hierarchy (für Summenvorgänge bei hierarchischer Darstellung)
- HierarchyCollapsed (für kollabierte Summenvorgänge bei hierarchischer Darstellung)

Eigenschaften

- CollapseColumn
- FieldsSeparatedByLines
- FilterName
- FormatField

- FormatFieldCount
- IndentColumn
- IndentWidth
- Name
- SeparationLineColor
- ThreeDEffect

Methoden

- GetEnumerator

Eigenschaften

CollapseColumn

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie festlegen, ob und in welcher Spalte, die mehrere Zeilen enthält, +/- zum Ein- bzw. Ausblenden der weiteren Zeilen angezeigt werden soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzeige von +/- in Spalte eingeschaltet.

Code-Beispiel VB.NET

```
' Display of +/- in the fifth column
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("Hierarchy").
CollapseColumn = 5
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("HierarchyCol
lapsed").CollapseColumn = 5
```

Code-Beispiel C#

```
// Display of +/- in the fifth column
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("Hierarchy").
CollapseColumn = 5;
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("HierarchyCol
lapsed").CollapseColumn = 5;
```

FieldsSeparatedByLines

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie festlegen, ob die Tabellenfelder durch Linien getrennt werden (True) oder nicht (False).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Tabellenfelder werden durch Linien getrennt (True)/ nicht getrennt (False).

Code-Beispiel VB.NET

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
format.FieldsSeparatedByLines = True
```

Code-Beispiel C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
format.FieldsSeparatedByLines = true;
```

FilterName

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie den Filter setzen oder erfragen, der bestimmt, unter welcher Bedingung (d. h. für welche Vorgänge) dieses Tabellenformat in der Tabelle verwendet werden soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Filtername

Code-Beispiel VB.NET

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("ListFormat2")
format.FilterName = "Code1NotA"
```

Code-Beispiel C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("ListFormat2
format.FilterName = "Code1NotA";
```

FormatField

Nur-Lese-Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie ein VcTableFormatField-Objekt per Index holen. Der Index muss im Bereich von 0 bis FormatFieldCount-1 liegen.

Hinweis für Benutzer einer Version vor 3.0: Der Index zählt bei dieser Methode nicht wie in den bisherigen Feldeigenschaften von 1 bis FormatFieldCount!

Die Eigenschaft FormatField ist eine indizierte Eigenschaft, die in C# über die Methode get_FormatField (index) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
index	System.Int16	Index des Tabellenformatfeldes 0 ... FormatFieldCount-1
Eigenschaftswert	VcTableFormatField	Tabellenformatfeld

FormatFieldCount

Nur-Lese-Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie die Anzahl der Tabellenspalten dieses Tabellenformats erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Tabellenspalten

Code-Beispiel VB.NET

```
Dim format As VcTableFormat
Dim numberOfColumns As Integer

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
numberOfColumns = format.FormatFieldCount
```

Code-Beispiel C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
int numberOfColumns = format.FormatFieldCount;
```

IndentColumn

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie festlegen, welche Spalte einen Einzug bekommen soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Nummer der Spalte mit Einzug

Code-Beispiel VB.NET

```
' Second column is indented
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList
").IndentColumn = 2
```

Code-Beispiel C#

```
// Second column is indented
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList
").IndentColumn = 2;
```

IndentWidth

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie die Größe des Einzuges in mm festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Grösse des Einzugs

Code-Beispiel VB.NET

```
' Second column is indented by 100 mm
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList
").IndentColumn = 2
VcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList
").IndentWidth = 100
```

Code-Beispiel C#

```
// Second column is indented by 100 mm
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList
").IndentColumn = 2;
vcGantt1.TableCollection.Active.TableFormatCollection.FormatByName("StandardList
").IndentWidth = 100;
```

Name

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie den Namen des Tabellenformats erfragen oder setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Tabellenformatname

Code-Beispiel VB.NET

```
Dim format As VcTableFormat
Dim formatName As String

format = VcGantt1.LeftTable.TableFormatCollection.FirstFormat
formatName = format.Name
```

Code-Beispiel C#

```
VcTableFormat format = vcGantt1.LeftTable.TableFormatCollection.FirstFormat();
string formatName = format.Name;
```

SeparationLineColor

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft können Sie die Farbe der Trennlinien für die Tabellenfelder festlegen. Die Standardfarbe ist weiß.

	Datentyp	Beschreibung
Eigenschaftswert	Color RGB	Farbwert ({0...255},{0...255},{0...255}) **Standardwert:** RGB(0,0,0)

Code-Beispiel VB.NET

```
VcTableFormat.SeparationLineColor = RGB(255, 204, 204)
```

Code-Beispiel C#

```
VcTableFormat.SeparationLineColor = RGB(255, 204, 204);
```

ThreeDEffect

Eigenschaft von VcTableFormat

Mit dieser Eigenschaft wird festgelegt oder erfragt, ob das Tabellenformat einen 3D-Effekt erhält oder nicht.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	3D-Effekt eingeschaltet (True)/ausgeschaltet (False)

Code-Beispiel VB.NET

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
format.ThreeDEffect = True
```

Code-Beispiel C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
format.ThreeDEffect = true;
```

Methoden

GetEnumerator

Methode von VcTableFormat

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 Tabellenformatfelder iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

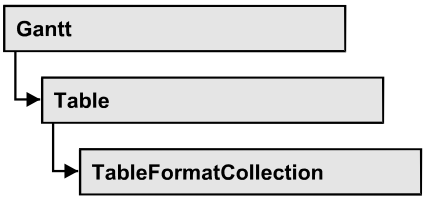
Code-Beispiel VB.NET

```
Dim format As VcTableFormat
Dim formatField As VcTableFormatField

For Each formatField In format
   Debug.Write(formatField.Index)
Next
```

Code-Beispiel C#

```
private void Form1_Load(object sender, System.EventArgs e)
   {
   VcTableFormat format;
   foreach (VcTableFormatField formatField in format)
      Console.Writ(formatField.Index);
   }
```

7.72 VcTableFormatCollection



In einem Objekt vom Typ VcTableFormatCollection sind alle verfügbaren Tabellenformate enthalten. Über **For Each tableFormat InTableFormatCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Formate zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **FormatByName** und **FormatBy-Index**. Die Anzahl der im Auflistungsobjekt vorhandenen Tabellenformate kann über die Eigenschaft **Count** erfragt werden.

Eigenschaften

- Count

Methoden

- FirstFormat
- FormatByIndex
- FormatByName
- GetEnumerator
- NextFormat

Eigenschaften

Count

Nur-Lese-Eigenschaft von VcTableFormatCollection

Mit dieser Eigenschaft können Sie die Anzahl der Tabellenformatobjekte in der TableFormat-Auflistung erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Tabellenformate

Code-Beispiel VB.NET

```
Dim formatCltn As VcTableFormatCollection
Dim numberOfFormats As Integer

formatCltn = VcGantt1.LeftTable.TableFormatCollection
numberOfFormats = formatCltn.Count
```

Code-Beispiel C#

```
VcTableFormatCollection formatCltn = vcGantt1.LeftTable.TableFormatCollection;
int numberOfFormats = formatCltn.Count;
```

Methoden

FirstFormat

Methode von VcTableFormatCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. das erste Tabellenformat des TableFormatCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextFormat** über die nachfolgenden Tabellenformate zu iterieren. Existiert kein Tabellenformat im TableFormatCollection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcTableFormat	Erstes Tabellenformat

Code-Beispiel VB.NET

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FirstFormat
```

Code-Beispiel C#

```
VcTableFormat format = vcGantt1.LeftTable.TableFormatCollection.FirstFormat();
```

FormatByIndex

Methode von VcTableFormatCollection

Mit dieser Methode können Sie auf ein einzelnes Tabellenformat über seinen Index zugreifen. Existiert kein Tabellenformat an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des Tabellenformats
Rückgabewert	VcTableFormat	Ermitteltes Tabellenformatobjekt

FormatByName

Methode von VcTableFormatCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf ein bestimmtes Tabellenformat zugreifen. Existiert kein Tabellenformat unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ formatName	System.String	Name des Tabellenformats
Rückgabewert	VcTableFormat	Tabellenformat

Code-Beispiel VB.NET

```
Dim format As VcTableFormat

format = VcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList")
```

Code-Beispiel C#

```
VcTableFormat format =
vcGantt1.LeftTable.TableFormatCollection.FormatByName("StandardList");
```

GetEnumerator

Methode von VcTableFormatCollection

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 Tabellenformate iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim format As VcTableFormat

For Each format In VcGantt1.LeftTable.TableFormatCollection
   Debug.Write(format.Name)
Next
```

Code-Beispiel C#

```
foreach (VcTableFormat format in vcGantt1.LeftTable.TableFormatCollection)
   Console.Write(format.Name);
```

NextFormat

Methode von VcTableFormatCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Tabellenformate des TableFormatCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstFormat** den Initialwert erfasst haben. Sind alle Tabellenformate durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcTableFormat	Folgendes Tabellenformat

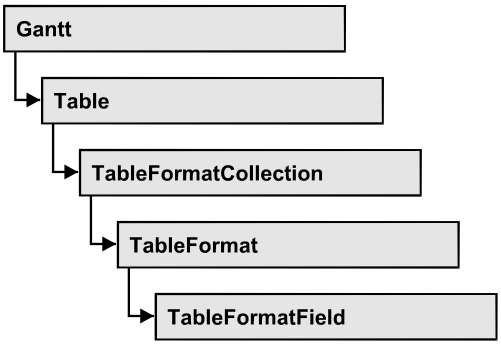
Code-Beispiel VB.NET

```
Dim formatCltn As VcTableFormatCollection
Dim format As VcTableFormat

formatCltn = VcGantt1.LeftTable.TableFormatCollection
format = formatCltn.FirstFormat
While Not format Is Nothing
  ListBox1.Items.Add(format.Name)
   format = formatCltn.NextFormat
End While
```

Code-Beispiel C#

```
VcTableFormatCollection formatCltn = vcGantt1.LeftTable.TableFormatCollection;
VcTableFormat format = formatCltn.FirstFormat();
while (format != null)
   {
   listBox1.Items.Add(format.Name);
   format = formatCltn.NextFormat();
   }
```

7.73 VcTableFormatField



Ein Objekt vom Typ **VcTableFormatField** stellt ein Tabellenformatfeld, also ein Feld eines **VcTableFormat**-Objekts dar. Ein Tabellenformatfeld besitzt im Gegensatz zu vielen anderen Objekten keinen Namen, sondern nur einen Index, unter dem es im Tabellenformat untergebracht ist. Eine Tabelle kann maximal 100 Formatfelder beinhalten.

Eigenschaften

- Alignment
- BottomMargin
- ConstantText
- FormatName
- GraphicsFileName
- GraphicsFileNameDataFieldIndex
- GraphicsFileNameMapName
- GraphicsHeight
- Index
- LeftMargin
- MaximumTextLineCount
- MinimumTextLineCount
- MultiState
- PatternBackgroundColorAsARGB
- PatternBackgroundColorDataFieldIndex
- PatternBackgroundColorMapName
- PatternColorAsARGB
- PatternColorDataFieldIndex
- PatternColorMapName
- PatternEx
- PatternExDataFieldIndex
- PatternExMapName

- RightMargin
- TextAndGraphicsCombined
- TextDataFieldIndex
- TextFont
- TextFontColor
- TextFontColorDataFieldIndex
- TextFontColorMapName
- TextFontDataFieldIndex
- TextFontMapName
- TopMargin
- Type

Eigenschaften

Alignment

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Ausrichtung des Inhalts im Tabellenformatfeld festlegen oder erfragen.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcFormatFieldAlignment **Mögliche Werte:** .vcFFABottom 28 .vcFFABottomLeft 27 .vcFFABottomRight 29 .vcFFACenter 25 .vcFFALeft 24 .vcFFARight 26 .vcFFATop 22 .vcFFATopLeft 21 .vcFFATopRight 23	Ausrichtung des Feldinhalts unten unten links unten rechts unten mittig links rechts oben oben links oben rechts

BottomMargin

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des unteren Randes des Tabellenformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Tabellenformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite des unteren Randes des Tabellenformatfeldes in mm 0...9

ConstantText

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie einen konstanten Text in dem Tabellenformatfeld ausgeben, falls der Typ des Tabellenformatfeldes auf **vcFFTText** und falls die Eigenschaft **TextDataFieldIndex** auf **-1** gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Konstanter Text

FormatName

Nur-Lese-Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Namen des Tabellenformats erfragen, zu dem dieses Tabellenformatfeld gehört.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Tabellenformats

GraphicsFileName

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** den Namen einer Grafikdatei setzen oder erfragen, deren Inhalt in dem Tabellenformatfeld ausgegeben wird. Der Name muss eine gültige Grafikdatei bezeichnen. 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 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
Eigenschaftswert	System.String	Name der Grafikdatei

GraphicsFileNameDataFieldIndex

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** den Datenfeldindex festlegen oder erfragen, der in der Eigenschaft **GraphicsFile-NameMapName** benötigt wird. Beim Wert **-1** wird in dem Tabellenformatfeld die Grafikdatei ausgeben, die in der Eigenschaft **GraphicsFileName** angegeben ist. Ist ein gültiger Datenfeldindex angegeben und keine Zuordnungstabelle, dann wird der Grafikdateiname direkt aus dem Inhalt des angegebenen Datenfelds entnommen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Datenfeldes

GraphicsFileNameMapName

Eigenschaft von VcTableFormatField

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 VcTableFormatField

Mit dieser Eigenschaft können Sie beim Typ **vcFFTGraphics** die Höhe der Grafik im Tabellenformatfeld festlegen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Höhe der Grafik in mm 0 ... 99

Index

Nur-Lese-Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Index des Tabellenformatfelds im zugehörigen Tabellenformat erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Index des Tabellenformatfeldes

LeftMargin

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des linken Randes des Tabellenformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Tabellenformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite des linken Randes des Tabellenformatfeldes in mm 0...9

MaximumTextLineCount

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die maximale Anzahl der Zeilen in dem Tabellenformatfeld setzen oder erfragen, falls das Tabellenformatfeld vom Typ **vcFFTText** ist. Bitte sehen Sie auch die Eigenschaft **MinimumText-LineCount**.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Maximale Zeilenzahl 0...9

MinimumTextLineCount

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die minimale Anzahl der Zeilen in dem Tabellenformatfeld setzen oder erfragen, falls der Typ des Tabellenformat-feldes auf **vcFFTText** 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	Minimale Zeilenzahl 0...9

MultiState

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie setzen oder erfragen, ob das Tabellenformatfeld ein Multi-State-Feld ist. MultiState-Felder werden verwendet, um z.B. beim Anklicken eines Tabellenfeldes eine zyklischen Abfolge von Zuständen wie auch der zugehörigen Datenfelder auszulösen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	MultiState-Feld (True) / kein MultiState-Feld (False)

PatternBackgroundColorAsARGB

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Hintergrundfarbe des Tabellenformatfeldes 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 Tabellenformats besitzen soll.

Wenn in der Eigenschaft **PatternPatternBackgroundColorMapName** eine Zuordnungstabelle angegeben ist, steuert diese die Hintergrundfarbe in Abhängigkeit von den Daten.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** -1

PatternBackgroundColorDataFieldIndex

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der Verbindung mit der Eigenschaft **PatternBackgroundColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

PatternBackgroundColorMapName

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn ein Name einer Farbzuordnungstabelle 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 **PatternBackgroundColor** ausgegeben.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

PatternColorAsARGB

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Musterfarbe desTabellenformatfeldes 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 Tabellenformats besitzen soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	Musterfarbe des Tabellenformatfeldes

PatternColorDataFieldIndex

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

PatternColorMapName

Eigenschaft von VcTableFormatField

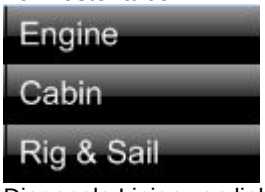
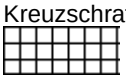
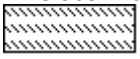
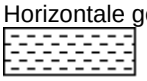
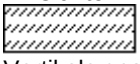
Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) setzen oder erfragen. Wird hier "" angegeben, dann wird keine Zuordnungstabelle verwendet. Nur wenn ein Name einer Farbzuordnungstabelle und ein Datenfeldindex in der Eigenschaft **PatternColorDataFieldIndex** angegeben sind, wird die Musterfarbe des Kalendergitters aus der Zuordnungstabelle ausgewählt. Trifft kein Datenfeldeintrag zu, wird die Musterfarbe aus der Eigenschaft **PatternColor** ausgegeben.

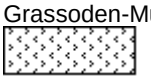
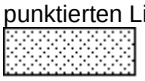
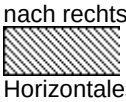
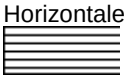
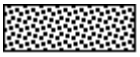
	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Farbzuordnungstabelle

PatternEx

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie für den Hintergrund des Tabellenformatfeldes ein Muster setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFillPattern	Mustertyp
	Mögliche Werte:	
	.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 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 

PatternExDataFieldIndex

Nur-Lese-Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der in Verbindung mit der Eigenschaft **PatternExMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

PatternExMapName

Nur-Lese-Eigenschaft von VcTableFormatField

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
Eigenschaftswert	System.String	Name der Musterzuordnungstabelle

RightMargin

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des rechten Randes des Tabellenformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Tabellenformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite des rechten Randes des Tabellenformatfeldes in mm 0...9

TextAndGraphicsCombined

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das Tabellenfeld ein Kombifeld ist. (Vgl. Dialog **Tabellenformat bearbeiten**).

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kombifeld (True)/ kein Kombifeld (False)

TextDataFieldIndex

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Index des Datenfelds, dessen Inhalt in dem Tabellenformatfeld dargestellt werden soll, erfragen oder setzen, sofern es sich um ein Feld des Datentyps **vcFFTText** 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 VcTableFormatField

Mit dieser Eigenschaft können Sie die Schriftart des Tabellenformatfeldes 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

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie die Schriftfarbe des Tabellenformatfeldes festlegen oder erfragen, falls der Typ des Feldes auf **vcFFTText** 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 Tabellenformatfelds

TextFontColorDataFieldIndex

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Datenfeldindex festlegen oder erfragen, der bei einer Schriftfarbe-Zuordnungstabelle in der Eigenschaft **TextFontColorMapName** benötigt wird. Wenn Sie hier **-1** angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

TextFontColorMapName

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Namen einer Farbzuordnungstabelle (Typ vcColorMap) für die Schriftfarbe setzen oder erfragen, falls der Typ des Formatfeldes auf **vcFFTText** gesetzt wurde. Wird für den Namen "" angegeben, dann wird keine Zuordnungstabelle verwendet. Wenn der Name einer Farbzuordnungstabelle und zusätzlich ein Datenfeldindex in der Eigenschaft **TextFontColorDataFieldIndex** angegeben ist, wird die Schriftfarbe aus der Zuordnungstabelle ausgewählt. Trifft kein Eintrag der Zuordnungstabelle zu, wird die Hintergrundfarbe verwendet, die über die Eigenschaft **TextFontColor** gesetzt wurde.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Schriftfarben-Zuordnungstabelle

TextFontDataFieldIndex

Eigenschaft von VcTableFormatField

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 angegeben, wird keine Zuordnungstabelle verwendet.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Datenfeldindex

TextFontMapName

Eigenschaft von VcTableFormatField

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 VcTableFormatField

Mit dieser Eigenschaft können Sie die Breite (in mm) des oberen Randes des Tabellenformatfeldes festlegen oder erfragen. Sie kann ebenfalls in dem Dialog **Tabellenformat bearbeiten** gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Breite des oberen Randes des Tabellenformatfeldes in mm 0...9

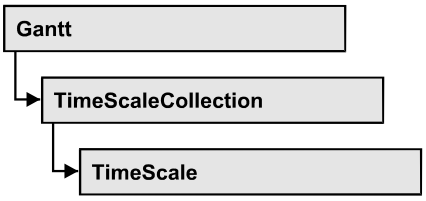
Type

Eigenschaft von VcTableFormatField

Mit dieser Eigenschaft können Sie den Typ des Tabellenformatfelds setzen/erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcFormatFieldType **Mögliche Werte:** .vcFFTGraphics 64 .vcFFTText 36	Typ des Tabellenformatfeldes Grafik Text

7.74 VcTimeScale



Das VcTimeScale-Objekt ist die Zeitskala oberhalb des Diagrammbereichs, in dem die Knoten dargestellt werden. Zum dargestellten Zeitbereich können Sie aus verschiedenen Zeitskalen die passende wählen. Die Farbe und diverse Schriftattribute können beliebig gesetzt werden. Beim Zeitskala-Objekt werden auch das (vertikale) Zeitraster sowie ggf. die Kennzeichnung der Wochenenden eingeschaltet.

Eigenschaften

- BackgroundColor
- CalendarGridsVisible
- DateGridsVisible
- Font
- FontColor
- Name
- Ribbon
- Section
- ThreeDEffect
- UpdateBehaviorName

Eigenschaften

BackgroundColor

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie die Hintergrundfarbe der Zeitskala erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timeScale.BackgroundColor = Color.Blue
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.BackgroundColor = Color.LightSteelBlue;
```

CalendarGridsVisible

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob arbeitsfreie Zeiten durch graue Schattierungen markiert werden sollen. Sie können diese Eigenschaft auch im Dialog **Zeitskala festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Arbeitsfreie Zeiten werden/werden nicht grau hinterlegt

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timeScale.CalendarGridsVisible = True
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.CalendarGridsVisible = true;
```

DateGridsVisible

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob vertikale Rasterlinien angezeigt werden. Sie können diese Eigenschaft auch im Dialog **Zeitskala festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Rasterlinien werden/werden nicht angezeigt

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timeScale.DateGridsVisible = True
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.DateGridsVisible = true;
```

Font

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie alle Schriftattribute der Zeitskala erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.DrawingFont	Schriftattribute der Zeitskala

Code-Beispiel VB.NET

```
Dim newFont As Font
newFont = VcGantt1.TimeScaleCollection.Active.Font
MsgBox(newFont.ToString())
```

Code-Beispiel C#

```
Font newFont = vcGantt1.TimeScaleCollection.Active.Font;
MessageBox.Show(newFont.ToString());
```

FontColor

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie die Schriftfarbe der Zeitskala erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255})

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timeScale.FontColor = Color.Blue
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.FontColor = Color.LightSteelBlue;
```

Name

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie den Namen einer Zeitskala setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name der Zeitskala

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
MsgBox("Active timescale: " + timeScale.Name)
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
MessageBox.Show("Active timescale: " + timeScale.Name);
```

Ribbon

Nur-Lese-Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie auf einen Zeitstreifen der Zeitskala zugreifen.

Die Eigenschaft Ribbon ist eine indizierte Eigenschaft, die in C# über die Methode get_Ribbon (ribbonIndex, sectionIndex) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ sectionIndex	System.Int16	Index des Zeitskalenabschnitts
⇨ ribbonIndex	System.Int16	Index des Zeitstreifens
Eigenschaftswert	VcRibbon	Zeitstreifen-Objekt

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim ribbon As VcRibbon

timescale = VcGantt1.TimeScaleCollection.Active
ribbon = timescale.Ribbon(0, 0)
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcRibbon ribbon = timeScale.get_Ribbon(0,0);
```

Section

Nur-Lese-Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie auf einen Skalenabschnitt der Zeitskala zugreifen.

Die Eigenschaft Section ist eine indizierte Eigenschaft, die in C# über die Methode get_Section (sectionIndex) angesprochen wird.

	Datentyp	Beschreibung
Parameter:		
⇨ sectionIndex	System.Int16	Index des Skalenabschnitts
Eigenschaftswert	VcSection	Skalenabschnittsobjekt

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale
Dim section As VcSection

timescale = VcGantt1.TimeScaleCollection.Active
section = timescale.Section(0)
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
VcSection section = timeScale.get_Section(0);
```

ThreeDEffect

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie erfragen oder festlegen, ob die Zeitskala mit 3D-Effekt dargestellt wird. Sie können diese Eigenschaft auch im Dialog **Zeitskala festlegen** setzen.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	3D-Effekt eingeschaltet (True)/ausgeschaltet (False)

Code-Beispiel VB.NET

```
Dim timescale As VcTimeScale

timescale = VcGantt1.TimeScaleCollection.Active
timeScale.ThreeDEffect = False
```

Code-Beispiel C#

```
VcTimeScale timeScale = vcGantt1.TimeScaleCollection.Active;
timeScale.ThreeDEffect = false;
```

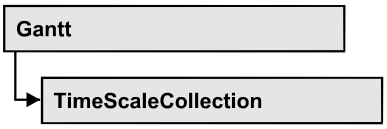
UpdateBehaviorName

Eigenschaft von VcTimeScale

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

7.75 VcTimeScaleCollection



Im VcTimeScaleCollection-Objekt sind automatisch alle verfügbaren Zeitskalen enthalten. Über **For Each timeScale InTimeScaleCollection** oder die Methoden **First...** und **Next...** können Sie in einer Schleife auf alle Zeitskalen zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **TimeScaleByName** und **TimeScaleByIndex**.Die Anzahl der im Auflistungsobjekt vorhandenen Zeitskalen kann über die Eigenschaft **Count** erfragt werden. Die aktuelle Zeitskala kann durch die Eigenschaft **Active** gesetzt oder erfragt werden.

Eigenschaften

- Active
- Count

Methoden

- FirstTimeScale
- GetEnumerator
- NextTimeScale
- TimeScaleByIndex
- TimeScaleByName

Eigenschaften

Active

Eigenschaft von VcTimeScaleCollection

Mit dieser Eigenschaft kann die aktuell für die Darstellung verwendete Zeitskala erfragt oder gesetzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	VcTimeScale	Aktuell dargestellte Zeitskala

Code-Beispiel VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.Active
```

Code-Beispiel C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.Active;
```

Count

Nur-Lese-Eigenschaft von VcTimeScaleCollection

Mit dieser Eigenschaft kann die Anzahl der Zeitskalen in der TimeScale-Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Zeitskalen

Code-Beispiel VB.NET

```
Dim numberOfTimeScales As Integer

numberOfTimeScales = VcGantt1.TimeScaleCollection.Count
```

Code-Beispiel C#

```
int numberOfTimeScales = vcGantt1.TimeScaleCollection.Count;
```

Methoden

FirstTimeScale

Methode von VcTimeScaleCollection

Mit dieser Methode können Sie auf den Initialwert, d.h. die erste Zeitskala des TimeScaleCollection-Objekts zugreifen, um anschließend in einer Schleife mit der Methode **NextTimeScale** über die nachfolgenden Zeitskalen zu iterieren. Existiert keine Zeitskala im Collection-Objekt, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcTimeScale	Erste Zeitskala

Code-Beispiel VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.FirstTimeScale
```

Code-Beispiel C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.FirstTimeScale();
```

GetEnumerator

Methode von VcTimeScaleCollection

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 Zeitskalen-Objekte iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

NextTimeScale

Methode von VcTimeScaleCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Zeitskalen des TimeScaleCollection-Objekts zugreifen, nachdem Sie mit der Methode **FirstTimeScale** den Initialwert erfasst haben. Sind alle Zeitskalen durchlaufen, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcTimeScale	Nachfolgende Zeitskala

Code-Beispiel VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection
Dim timeScale As VcTimeScale

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScale = timeScaleCltn.FirstTimeScale
While Not timeScale Is Nothing
   ListBox1.Items.Add(timeScale.Name)
   timeScale = timeScaleCltn.NextTimeScale
End While
```

Code-Beispiel C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
VcTimeScale timeScale = timeScaleCltn.FirstTimeScale();
while (timeScale != null)
   {
   listBox1.Items.Add(timeScale.Name);
   timeScale = timeScaleCltn.NextTimeScale();
   }
```

TimeScaleByIndex

Methode von VcTimeScaleCollection

Mit dieser Methode können Sie auf eine einzelne Zeitskala über ihren Index zugreifen. Existiert keine Zeitskala an dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	**Beschreibung**
Parameter:		
⇨ index	System.Int16	Index der Zeitskala
Rückgabewert	VcTimeScale	Ermitteltes Zeitskalenobjekt

TimeScaleByName

Methode von VcTimeScaleCollection

Mit dieser Methode können Sie unter Verwendung des Namens auf eine bestimmte Zeitskala zugreifen. Existiert keine Zeitskala unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	**Beschreibung**
Parameter:		
⇨ timeScaleName	System.String	Name der Zeitskala
Rückgabewert	VcTimeScale	Zeitskala

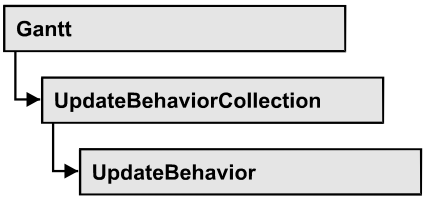
Code-Beispiel VB.NET

```
Dim timeScaleCltn As VcTimeScaleCollection

timeScaleCltn = VcGantt1.TimeScaleCollection
timeScaleCltn.Active = timeScaleCltn.TimeScaleByName("Days")
```

Code-Beispiel C#

```
VcTimeScaleCollection timeScaleCltn = vcGantt1.TimeScaleCollection;
timeScaleCltn.Active = timeScaleCltn.TimeScaleByName("Days");
```

7.76 VcUpdateBehavior



Ein Objekt vom Typ **VcUpdateBehavior** beinhaltet eine Sammlung von Eigenschaften und Methoden, die das Aktualisierungsverhalten der Objekte am Bildschirm steuern, denen es zugewiesen wurde.

Eigenschaften

- IsEditable
- Name
- Specification

Methoden

- PutInOrderAfter

Eigenschaften

IsEditable

Eigenschaft von VcUpdateBehavior

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob das spezifizierte Aktualisierungsverhalten zur Laufzeit editierbar sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Aktualisierungsverhalten editierbar (True) / nicht editierbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim updBeh As VcUpdateBehavior

updBeh = UpdateBehaviorCollection.UpdateBehaviorByName("Immediate")
updBeh.IsEditable = False
```

Code-Beispiel C#

```
VcUpdateBehavior updBeh =
UpdateBehavior.Collection.UpdateBehaviorByName("Immediate");
updBeh.IsEditable = false;
```

Name

Eigenschaft von VcUpdateBehavior

Mit dieser Eigenschaft können Sie den Namen eines Aktualisierungsverhaltens setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Name des Aktualisierungsverhaltens

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection
For Each updBeh In updBehCltn
   ComboBox1.Items.Add(updBeh.Name)
Next
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
foreach (VcUpdateBehavior updBeh in updBehCltn)
   comboBox1.Items.Add(updBeh.Name);
```

Specification

Nur-Lese-Eigenschaft von VcUpdateBehavior

Mit dieser Eigenschaft können Sie die Spezifikation dieses Aktualisierungsverhaltens 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 Aktualisierungsverhaltens mit der Methode **VcUpdateBehaviorCollection.AddBySpecification** benutzt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.String	Spezifikation des Aktualisierungsverhaltens

Code-Beispiel VB.NET

```
Dim updateBehaviorCltn As VcUpdateBehaviorCollection
Dim updateBehavior As VcUpdateBehavior

updateBehaviorCltn = VcGantt1.UpdateBehaviorCollection
updateBehavior = updateBehaviorCltn.FirstUpdateBehavior
MsgBox(updateBehavior.Specification)
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection boxCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updateBehavior = updateBehaviorCltn.FirstUpdateBehavior();
MessageBox.Show(updateBehavior.Specification);
```

Methoden

PutInOrderAfter

Methode von VcUpdateBehavior

Mit dieser Methode können Sie dieses Aktualisierungsverhalten in der Auflistung aller Aktualisierungsverhalten hinter das durch den Namen angegebene setzen. Wenn als Name "" angegeben wird, wird das Aktualisierungsverhalten an die erste Stelle gesetzt. Die Reihenfolge der Aktualisierungsverhalten in der Auflistung entscheidet darüber, in welcher Reihenfolge sie auf die zugewiesenen Objekte angewendet werden.

	Datentyp	Beschreibung
Parameter:		
refUpdateBehaviorName	System.String	Name des Aktualisierungsverhaltens, hinter das das aktuelle Aktualisierungsverhalten gesetzt werden soll

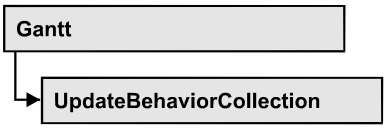
Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh1 As VcUpdateBehavior
Dim updBeh2 As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection()
updBeh1 = updBehCltn.Add("updBeh1")
updBeh2 = updBehCltn.Add("updBeh2")
updBeh1.PutInOrderAfter("updBeh2")
updBehCltn.Update()
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh1 = updBehCltn.Add("updBeh1");
VcUpdateBehavior updBeh2 = updBehCltn.Add("updBeh2");
updBeh1.PutInOrderAfter("updBeh2");
updBehCltn.Update();
```

7.77 VcUpdateBehaviorCollection



Im VcUpdateBehavior-Auflistungsobjekt sind alle verfügbaren Aktualisierungsverhalten zusammengefasst. Über **For Each UpdateBehavior In UpdateBehaviorCollection** oder die Methoden **First…** und **Next…** können Sie in einer Schleife auf alle Verhalten zugreifen. Sie haben Zugriff auf bestimmte Objekte über die Eigenschaften **UpdateBehaviorByName** und **UpdateBehaviorByIndex**. Die Anzahl der im Auflistungsobjekt vorhandenen Aktualisierungsverhalten kann über die Eigenschaft **Count** erfragt werden. Die Methoden **Add…**, **Copy…** und **Remove…** ermöglichen das Hinzufügen, Kopieren und Löschen von Aktualisierungsverhalten.

Eigenschaften

- Active
- Count

Methoden

- Add
- AddBySpecification
- Copy
- FirstUpdateBehavior
- GetEnumerator
- NextUpdateBehavior
- Remove
- UpdateBehaviorByIndex
- UpdateBehaviorByName

Eigenschaften

Active

Nur-Lese-Eigenschaft von VcUpdateBehaviorCollection

Mit dieser Eigenschaft können sie das aktuell verwendete Aktualisierungsverhalten setzen oder erfragen.

	Datentyp	Beschreibung
Eigenschaftswert	VcUpdateBehavior	Aktuell verwendetes Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

Set updBehCltn = VcGantt1.UpdateBehaviorCollection
Set updBeh = UpdateBehaviorCltn.Active
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh = updBehCltn.UpdateBehaviorByIndex(0);
updBehCltn.Remove(updBeh.Name);
```

Count

Nur-Lese-Eigenschaft von VcUpdateBehaviorCollection

Mit dieser Eigenschaft kann die Anzahl der Aktualisierungsverhalten in der Auflistung erfragt werden.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int32	Anzahl der Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim numberOfUpdateBehavior As Integer

numberOfUpdateBehavior = VcGantt1.UpdateBehaviorCollection.Count
```

Code-Beispiel C#

```
int numberOfUpdateBehviors = vcGantt1.UpdateBehaviorCollection.Count;
```

Methoden

Add

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie ein neues Aktualisierungsverhalten in der UpdateBehavior-Auflistung anlegen. Wenn der Name noch nicht verwendet wird, dann wird das neue Aktualisierungsverhalten zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ updateBehaviorName	System.String	Name des Aktualisierungsverhaltens
Rückgabewert	VcUpdateBehavior	Neues Aktualisierungsverhalten

Code-Beispiel VB.NET

```
newUpdateBehavior = VcGantt1.UpdateBehaviorCollection.Add("updBeh1")
```

Code-Beispiel C#

```
newUpdateBehavior = vcGantt1.UpdateBehaviorCollection.Add("updBeh1");
```

AddBySpecification

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie ein Aktualisierungsverhalten über eine Spezifikation erzeugen. Dies dient der Persistenz von Aktualisierungsverhalten. Die Spezifikation eines Aktualisierungsverhaltens kann erfragt und gespeichert werden. Bei einer neuen Sitzung kann das gleiche Aktualisierungsverhalten mit der wieder eingelesenen Spezifikation samt des gespeicherten Namens wieder erzeugt werden.

	Datentyp	Beschreibung
Parameter:		
⇨ updateBehaviorSpecification	System.String	Spezifikation des Aktualisierungsverhaltens
Rückgabewert	VcUpdateBehavior	Neues Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBehCltn.AddBySpecification(textSpecification)
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
updBehCltn.AddBySpecification(textSpecification);
```

Copy

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie ein Aktualisierungsverhalten kopieren. Wenn das Aktualisierungsverhalten mit dem angegebenen Namen existiert und der Name des neuen Aktualisierungsverhaltens noch nicht verwendet wird, wird

das neue Aktualisierungsverhalten zurückgegeben, sonst "Nothing" (Visual Basic) oder "0" (andere Sprachen als VB).

	Datentyp	Beschreibung
Parameter:		
⇨ updateBehaviorName	System.String	Name des zu kopierenden Aktualisierungsverhaltens
⇨ newUpdateBehaviorName	System.String	Name des neuen Aktualisierungsverhaltens
Rückgabewert	VcUpdateBehavior	Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBehCltn.Copy("UpdateBehviorOne", "NewUpdateBehvior")
updBehCltn.Update()
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
updBehCltn.Copy("UpdateBehaviorOne", "NewUpdateBehavior");
```

FirstUpdateBehavior

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie auf das erste Aktualisierungsverhalten der Auflistung zugreifen, um anschließend in einer Schleife mit der Methode **NextUdateBehavior** über die nachfolgenden Aktualisierungsverhalten zu iterieren. Existiert kein Aktualisierungsverhalten in der Auflistung, wird ein Leerobjekt übergeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcUpdateBehavior	Erstes Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBeh = updBehCltn.FirstUpdateBehavior
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh = updBehCltn.FirstUpdateBehavior();
```

GetEnumerator

Methode von VcUpdateBehaviorCollection

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 Aktualisierungsverhalten iterieren.

	Datentyp	Beschreibung
Rückgabewert	VcObject	Referenzobjekt

Code-Beispiel VB.NET

```
Dim updBeh As VcUpdateBehavior

For Each updBeh In VcGantt1.UpdateBehaviorCollection
   Debug.Print updBeh.Name
Next
```

Code-Beispiel C#

```
VcUpdateBehavior updBehCltn = vcGantt1.UpdateBehaviorCollection;
foreach (VcUpdateBehavior updBeh in updBehCltn)
   listBox1.Items.Add(updBeh.Name);
```

NextUpdateBehavior

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie in einer Schleife auf die nachfolgenden Aktualisierungsverhalten der Auflistung zugreifen, nachdem Sie mit der Methode **FirstUpdateBehavior** den Initialwert erfasst haben. Sind alle Aktualisierungsverhalten durchlaufen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Rückgabewert	VcUpdateBehavior	Folgendes Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBeh = updBehCltn.FirstUpdateBehavior

While Not updBeh Is Nothing
   ListBox1.Items.Add(updBeh.Name)
   updBeh = updBehCltn.NextUpdateBehavior
End While
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh = updBehCltn.FirstUpdateBehavior();

while (updBeh != null)
   {
   ListBox.Items.Add(updBeh.Name);
   updBeh = updBehCltn.NextUpdateBehavior();
   }
```

Remove

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie ein Aktualisierungsverhalten löschen. Wenn das Aktualisierungsverhalten noch in irgendeinem anderen Objekt benutzt wird, kann es nicht gelöscht werden. In diesem Fall wird False zurückgegeben, sonst True.

	Datentyp	Beschreibung
Parameter:		
⇨ updateBehaviorName	System.String	Name des Aktualisierungsverhaltens
Rückgabewert	System.Boolean	Aktualisierungsverhalten gelöscht (True)/nicht gelöscht (False)

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBeh = updBehCltn.UpdateBehaviorByIndex(0)
updBehCltn.Remove(updBeh.Name)
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh = updBehCltn.UpdateBehaviorByIndex(0);
updBehCltn.Remove(updBeh.Name);
```

UpdateBehaviorByIndex

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie auf eine einzelne Aktualisierungsverhalten über ihren Index zugreifen. Existiert kein Aktualisierungsverhalten unter dem angegebenen Index, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ index	System.Int16	Index des Aktualisierungsverhaltens
Rückgabewert	VcUpdateBehavior	Ermitteltes Aktualisierungsverhalten

Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim upBeh As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBeh = updBehCltn.UpdateBehaviorByIndex(0)
MsgBox(updBeh.Name)
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh = updBehCltn.UpdateBehaviorByIndex(0);
MessageBox.Show(updBeh.Name);
```

UpdateBehaviorByName

Methode von VcUpdateBehaviorCollection

Mit dieser Methode können Sie unter Verwendung des Namens des Aktualisierungsverhaltens auf ein bestimmtes Aktualisierungsverhalten zugreifen. Existiert kein UpdateBehavior-Objekt unter dem angegebenen Namen, wird ein Leerobjekt zurückgegeben (in Visual Basic: **Nothing**).

	Datentyp	Beschreibung
Parameter:		
⇨ updateBehaviorName	System.String	Name des Aktualisierungsverhaltens
Rückgabewert	VcUpdateBehavior	Aktualisierungsverhalten

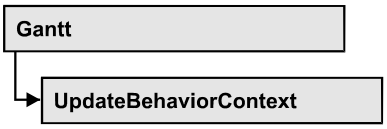
Code-Beispiel VB.NET

```
Dim updBehCltn As VcUpdateBehaviorCollection
Dim updBeh As VcUpdateBehavior

updBehCltn = VcGantt1.UpdateBehaviorCollection
updBeh = updBehCltn.UpdateBehaviorByName("UpdateBehaviorOne")
MsgBox(updBeh.Name)
```

Code-Beispiel C#

```
VcUpdateBehaviorCollection updBehCltn = vcGantt1.UpdateBehaviorCollection;
VcUpdateBehavior updBeh = updBehCltn.UpdateBehaviorByName("UpdateBehaviorOne");
MessageBox.Show(updBeh.Name);
```

7.78 VcUpdateBehaviorContext



Ein Objekt vom Typ **VcUpdateBehaviorContext** beschreibt den Kontext des Aktualisierungsverhaltens, d.h. das Verhalten anderer Objekte, die vom LiveUdate betroffen sind und das vom Benutzer einstellbar ist.

Eigenschaften

- DelayTime
- IsEditable
- Type
- UpdateMode

Eigenschaften

DelayTime

Eigenschaft von VcUpdateBehaviorContext

Mit dieser Eigenschaft können Sie die Verzögerungszeit setzen, nach der während des Verschiebvorgangs des Mauszeigers die Aktualisierung durch das live update erfolgen soll. Die Setzung dieser Eigenschaft ist nur sinnvoll, wenn die Eigenschaft **UpdateMode** auf **Verharren während des Verschiebens** gesetzt ist.

	Datentyp	Beschreibung
Eigenschaftswert	System.Int16	Anzahl der Millisekunden **Standardwert:** 500

Code-Beispiel VB.NET

```
Dim updBehCtx As VcUpdateBehaviorContext
Dim delTim As Integer

delTim = VcGantt1.updBehCtx.DelayTime
```

Code-Beispiel C#

```
int numOfMS = VcUpdateBehaviorContext.DelayTime;
```

IsEditable

Eigenschaft von VcUpdateBehaviorContext

Mit dieser Eigenschaft können Sie festlegen oder erfragen, ob der spezifizierte Kontext des Aktualisierungsverhaltens zur Laufzeit editierbar sein soll.

	Datentyp	Beschreibung
Eigenschaftswert	System.Boolean	Kontext des Aktualisierungsverhalten editierbar (True) / nicht editierbar (False) **Standardwert:** True

Code-Beispiel VB.NET

```
Dim updBehCtx As VcUpdateBehaviorContext

updBehCtx.Editable = False
```

Code-Beispiel C#

```
VcUpdateBehaviorContext updBehCtx.Editable = false;
```

Type

Nur-Lese-Eigenschaft von VcUpdateBehaviorContext

Mit dieser Eigenschaft können Sie einzelne Bereiche (Kontext-Typen) erfragen, die vom Live-Update betroffen sind und auf die die Eigenschaften **Editable**, **UpdateMode** und **DelayTime** anwendbar sind.

	Datentyp	Beschreibung
Eigenschaftswert	VcUpdateBehaviorContextType	Mögliche Aktualisierungsbereiche (Typen):
	Mögliche Werte:	
	.vcBoxesChangeAnchorNode 1403	Boxen wechseln den Ankerknoten
	.vcBoxesChangePosition 1402	Boxen ändern die Position
	.vcBoxesChangeSize 1401	Boxen ändern die Größe
	.vcCurvesChangeValue 1302	Kurven ändern Y-Wert
	.vcCurvesChangeXAndYValue 1303	Kurven ändern X- und Y-Wert
	.vcCurvesChangeXValue 1301	Kurven ändern X-Wert
	.vcDateLinesChangeDate 801	Terminstichlinien ändern das Datum
	.vcGroupLevelLayoutsAutoCollapseGroups 705	Im Gruppierungsebenen-Layout werden Gruppen automatisch kollabiert

.vcGroupLevelLayoutsAutoExpandTargetGroup 707	Im Gruppierungsebenen-Layout werden Zielgruppen automatisch expandiert.
.vcGroupLevelLayoutsChangeGroupsSortingOrder 701	Gruppierungsebenen-Layouts ändern die Sortierrichtung für Gruppen
.vcGroupLevelLayoutsNodesOptimization 702	Gruppierungsebenen-Layouts optimieren Knoten
.vcGroupLevelLayoutsOverlappingNodesSorting 704	Gruppierungsebenen-Layouts sortieren überlappende Knoten
.vcGroupLevelLayoutsRestoreAutoCollapsedGroups 706	Im Gruppierungsebenen-Layout werden automatisch kollabierte Gruppen automatisch wieder hergestellt.
.vcGroupLevelLayoutsRestoreAutoExpandedGroups 708	Im Gruppierungsebenen-Layout werden automatisch expandierte Zielgruppen wiederhergestellt
.vcGroupLevelLayoutsSummaryBarsCalculation 703	Gruppierungsebenen-Layouts berechnen die Summenknoten
.vcHierarchyLevelLayoutAutoCollapseGroups 302	Im Hierarchieebenen-Layout werden Gruppen automatisch kollabiert
.vcHierarchyLevelLayoutAutoExpandTargetGroups 304	Im Hierarchieebenen-Layout werden Zielgruppen automatisch expandiert
.vcHierarchyLevelLayoutRestoreAutoCollapsedGroups 303	Im Hierarchieebenen-Layout werden automatisch kollabierte Gruppen automatisch wieder hergestellt.
.vcHierarchyLevelLayoutRestoreAutoExpandedGroups 305	Im Hierarchieebenen-Layout werden werden automatisch expandierte Zielgruppen wiederhergestellt
.vcHierarchyLevelLayoutSummaryBarsCalculation 301	Das Hierarchieebenen-Layout berechnet die Summenknoten
.vcHistogramsLayerSourceCurvesCalculations 1101	Kurvenberechnung aus Layerdaten in Histogrammen
.vcLinksChangeSuccessorNode 402	Verbindungen wechseln den Nachfolgerknoten
.vcNodeLevelLayoutsChangeNodesSortingOrder 201	Knotenebenen-Layouts ändern die Sortierrichtung für Knoten
.vcNodesAutoScheduling 105	Automatische Zeiteinplanung für Knoten

.vcNodesChangeDatesDuration 101	Knoten ändern ihre Termine oder ihre Dauer
.vcNodesFiltering 102	Knoten werden gefiltert
.vcNodesGrouping 104	Knoten werden gruppiert
.vcNumericScalesChangeUnitWidth 1201	Numerische Skalen ändern die Breite der Einheit
.vcSaschesChangePosition 1501	Trennbalken ändern ihre Position
.vcTablesChangeColumnWidth 901	Tabellen ändern die Breite der Spalten
.vcTimeScalesChangeSectionStartDate 1002	Zeitskalen ändern das Anfangsdatum eines Zeitabschnitts
.vcTimeScalesChangeSectionStartDate 1002	Zeitskalen ändern das Anfangsdatum eines Zeitabschnitts
.vcTimeScalesChangeUnitWidth 1001	Zeitskalen ändern die Breite der Einheit

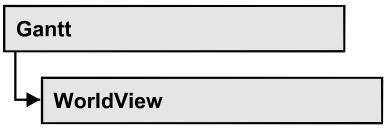
UpdateMode

Eigenschaft von VcUpdateBehaviorContext

Mit dieser Eigenschaft können Sie bei einem selbst generierten Aktualisierungsverhalten einstellen oder erfragen, bei welcher Aktion des Cursors die Aktualisierung durch das live update erfolgen soll. Ist diese Eigenschaft auf **Verharren während des Verschiebens** gesetzt, kann über die Eigenschaft **DelayTime** die gewünschte Verzögerungszeit eingestellt werden.

	Datentyp	**Beschreibung**
Eigenschaftswert	VcUpdateMode	Mögliche Aktionen des Cursors: **Standardwert:** vcOnMouseMove
	Mögliche Werte:	
	.vcOnMouseMove 1	Anzeige beim Bewegen des Mauszeigers
	.vcOnMouseUp 0	Anzeige wenn die linke Maustaste losgelassen wird
	.vcOnPauseWhileMouseMoving 2	Anzeige, wenn beim Bewegen des Mauszeigers angehalten wird

7.79 VcWorldView



Ein Objekt vom Typ **VcWorldView** bezeichnet das Komplettansicht-Fenster.

Eigenschaften

- Border
- BorderColor
- Height
- HeightActualValue
- Left
- LeftActualValue
- MarkingColor
- Mode
- 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
Eigenschaftswert	System.Boolean	Rahmen um die Komplettansicht (True)/kein Rahmen um die Komplettansicht (False) **Standardwert:** True

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Mode = VcWorldViewMode.vcNotFixed
VcGantt1.WorldView.Border = True
```

Code-Beispiel C#

```
vcGantt1.WorldView.Mode = VcWorldViewMode.vcNotFixed;
vcGantt1.WorldView.Border = true;
```

BorderColor

Eigenschaft von VcWorldView

Diese Eigenschaft setzt/liefert die des ggf. sichtbaren Rahmens.

	Datentyp	Beschreibung
Eigenschaftswert	System.Drawing.Color RGB ({0...255},{0...255},{0...255})	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** 0,0,0

Height

Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die vertikale Ausdehnung der Komplettansicht erfragt werden. In den Postionen **vcFixedAtTop**, **vcFixedAtBottom**, **vcNot-Fixed** 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
Eigenschaftswert	System.Int32	Höhe der Komplettansicht **Standardwert:** 100

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Height = 100
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche Höhe der Komplettansicht {0, ...} **Standardwert:** 100

Code-Beispiel VB.NET

```
VcGantt1.LegendView.Height = 300
```

Code-Beispiel C#

```
vcGantt1.LegendView.Height = 100;
```

Left

Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die linke Position der Komplettansicht erfragt werden. In den Postionen **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
Eigenschaftswert	System.Int32	Linke Position der Komplettansicht **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Left = 200
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche linke Position der Komplettansicht {0, ...} **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.LegendView.LeftActualValue = 150
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Drawing.Color	RGB-Farbwerte ({0...255},{0...255},{0...255}) **Standardwert:** RGB(0, 0, 255)

Code-Beispiel VB.NET

```
VcGantt1.WorldView.MarkingColor = Color.Red
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	VcWorldViewMode	Modus der Gesamtansicht **Standardwert:** vcPopupWindow
	Mögliche Werte:	
	.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.
	.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 **Schließen**-Schaltfläche in der Titelleiste ausgeschaltet werden.

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Mode = VcWorldViewMode.vcFixedAtBottom
```

Code-Beispiel C#

```
vcGantt1.WorldView.Mode = VcWorldViewMode.vcFixedAtBottom;
```

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 **Standardwert:** NoScrollBar
	Mögliche Werte:	
	.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

```
VcGantt1.WorldView.ScrollBarMode = vcAutomaticScrollbar
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Obere Position der Komplettansicht **Standardwert:** 0

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Top = 20
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche obere Position der Komplettansicht {0, ...}

Code-Beispiel VB.NET

```
VcGantt1.LegendView.TopActualValue = 40
```

Code-Beispiel C#

```
vcGantt1.LegendView.TopActualValue = 40;
```

UpdateBehaviorName

Nur-Lese-Eigenschaft von VcWorldView

Mit dieser Eigenschaft können Sie den Namen des Aktualisierungsverhaltens erfragen oder festlegen.

	Datentyp	Beschreibung
Eigenschaftswert	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
Eigenschaftswert	System.Boolean	Komplettansicht sichtbar (True)/unsichtbar (False) **Standardwert:** False

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Visible = True
```

Code-Beispiel C#

```
vcGantt1.WorldView.Visible = true;
```

Width

Eigenschaft von VcWorldView

Mit dieser Eigenschaft kann die horizontale Ausdehnung der Komplettansicht erfragt werden. In den Positionen **vcFixedAtLeft**, **vcFixedAtRight**, **vcNot-Fixed** 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
Eigenschaftswert	System.Int32	Horizontale Ausdehnung der Komplettansicht **Standardwert:** 100

Code-Beispiel VB.NET

```
VcGantt1.WorldView.Width = 200
```

Code-Beispiel C#

```
vcGantt1.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
Eigenschaftswert	System.Int32	Tatsächliche horizontale Ausdehnung der Komplettansicht {0, ...} **Standardwert:** 100

Code-Beispiel VB.NET

```
VcGantt1.LegendView.WidthActualValue = 600
```

Code-Beispiel C#

```
vcGantt1.LegendView.WidthActualValue = 600;
```

8 Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

Z