byron

Byron Informatik AG
Eisengasse 6
CH-4051 Basel
Tel. +41 (0)61 690 96 00
Fax +41 (0)61 690 96 09

byron@byron.ch
www.byron.ch

Byron/BIS – Technische Produktinformation
Benutzerdefinierte Formulare
Handbuch und Referenz

Inhaltsverzeichnis

1 Grundkonzept: Formulare in Byron/BIS

Die Modellierung von Formularen in Byron/BIS ist eine logische Erweiterung der Möglichkeit neue Objektklassen und Attribute mittels des Modellierungs-Werkzeugs in die Datenbank einzuführen. Wir können für die einzelnen Datenbankobjekte individuelle Formulare entwerfen und diese entweder mit der Öffnen-Funktion (Menüknopf, Popup-Menü, Doppelklick) oder mittels eines Menübefehls anzeigen.

1.1 Datenbankmodell

Zuerst ein paar Definitionen:

Datenbankobjekt	Ein einzelnes Objekt, das in der Datenbank gespeichert ist. Jedes Datenbankobjekt gehört genau einer Objektklasse an und kann beliebig viele Attribute und Assoziationen enthalten.
Objektklasse	Träger der Definition für eine Menge von Datenbankobjekten mit denselben Attributen und Assoziationen. Die Objektklasse definiert welche Attribute und Assoziationen für Datenbankobjekte erlaubt sind.
Attribut	Eigenschaft eines Datenbankobjektes. Ein Attribut besitzt einen Namen, einen Wert und bezieht sich auf einen Attributtyp, welcher die Art des Wertes (Text, Integer usw.) bestimmt. Zusätzlich kann ein Attribut eine Einheit und ein Format tragen.
Attributtyp	Definiert die Art des Wertes, z.B. Text, ganze Zahl, Gleitkommazahl, Aufzählung, Zeit, Intervall usw.
Assoziation	Beziehung zwischen zwei oder mehreren Datenbankobjekten. Eine Assoziation besitzt einen Namen und geht von einem Datenbankobjekt aus zu einem oder mehreren anderen Datenbankobjekten. Assoziationen kommen immer paarweise vor, d.h. zu jeder Assoziation gibt es einen Gegenpart, welcher in die umgekehrte Richtung zeigt.

1.2 Formulargrundtypen

Beim Erstellen eines Formulars muss ein Formular Formulargrundtyp gewählt werden. Der Formulargrundtyp ist verantwortlich für das Verhalten und die vordefinierten Komponenten des Formulars.

Der Formulargrundtyp lässt sich nachträglich nicht mehr verändern.

Im Moment gibt es nur einen Formulargrundtypen.

1.3 Ankopplung eines Formulars an die Datenbank

Alle bisherigen Formulargrundtypen arbeiten genau auf einem Datenbankobjekt. Beim Öffnen der Formulare wird das Datenbank-Objekt in die vordefinierte Komponente *BisDataSource* der Klasse TBisDataSource geladen.

Alle Benutzerschnittstellen-Komponenten mit Datenbankanbindung müssen durch die Eigenschaft *DataLink.DataSource* mit einer Komponente der Klasse TBisDataSource verknüpft sein – in der Regel die vordefinierte Komponente *BisDataSource*.

Die Operationen 'Laden der Daten aus der Datenbank' sowie 'Speichern der Daten in die Datenbank' werden von der *BisDataSource* an alle verknüpften Benutzerschnittstellen-Komponenten weitergegeben. Diese enthalten zusätzliche Angaben in der Eigenschaft *Datalink*, die festlegen, welches Attribut oder welche Assoziation dargestellt (bzw. geladen und abgespeichert) werden soll.

1.3.1 Die Felder der Eigenschaft DataLink

CDLAttribute	Name des darzustellenden Attributes.
CDLReference	Enthält optional den Namen einer Assoziation. Wenn das Feld gesetzt ist, wird nicht das Attribut vom Datenbankobjekt, das in BisDataSource steht verwendet sondern zuerst eine Navigationsoperation über die Assoziation durchgeführt und dann das Attribut des gefundenen Objektes gezeigt.
DataSource	Verbindung zur BisDataSource
InputRequired	Die Benutzerschnittstellen-Komponente darf beim Speichern nicht leer sein
OptimizedLoad	Wird diese Eigenschaft gesetzt, so wird, wenn möglich, das Laden von Elementen optimiert (zum Beispiel, wenn sich dieses auf einer zur Zeit unsichtbaren Registerseite eines PageControls befindet). Default: *true*.
UseAccessRights	Verwendet die in der Benutzerverwaltung definierten Zugriffsrechte auf Attribute und Assoziationen.
UseFormatFromDb	Benutzt zur Darstellung das Format aus der Datenbank. Muss bei Aufzählungstypen unbedingt gesetzt sein.

Die Bedeutung von *CDLAttribute* und vor allem *CDLReference* variiert je nach Benutzerschnittstellen-Komponente. Es wird deshalb empfohlen, die Beschreibung der jeweiligen Komponente zu konsultieren (Abschnitt: Beschreibung der einzelnen Komponenten).

2 Konfigurations-Explorer

Der Konfigurations-Explorer ist ein Tool zur Konfiguration der Byron/BIS-Datenbank. Er verhält sich wie die anderen Byron/BIS-Explorer. Sie können damit Datenbankobjekte der Objektklassen *Formular*, *Aktion, Menü, Ansicht, benannte Suchen, Beschriftung, Icon, Symbol, Ordner* und *Text* verwalten.

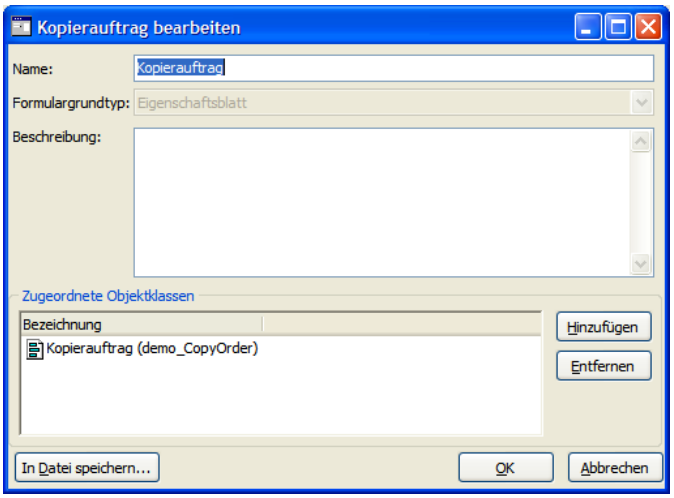
2.1 Aufgaben

Bezüglich *Formulare*, *Aktionen,* und *Menüs* können sie mit dem Konfigurations-Explorer folgende Aufgaben erledigen:

- Erstellen eines neuen Formulars und Anbinden an eine Objektklasse (d.h. der Menüpunkt Öffnen für diese Objektklasse öffnet das neue Formular)
- Erstellen eines neuen Formulars und eines neuen Menüpunktes zum Öffnen des Formulars
- Ändern/Löschen eines Formulars
- Erstellen/Ändern/Löschen eines Menüs
- Erstellen/Ändern/Löschen einer Aktion
- Anbinden eines externen Programms an einen Menüpunkt

Um ein Formular in den Formulareditor zu bekommen, wählen wir ein Formular an und benutzen die Menüposition 'Bearbeiten -> Formular bearbeiten' oder den Eintrag 'Formular bearbeiten' im Kontext-Menü.

2.2 Formular

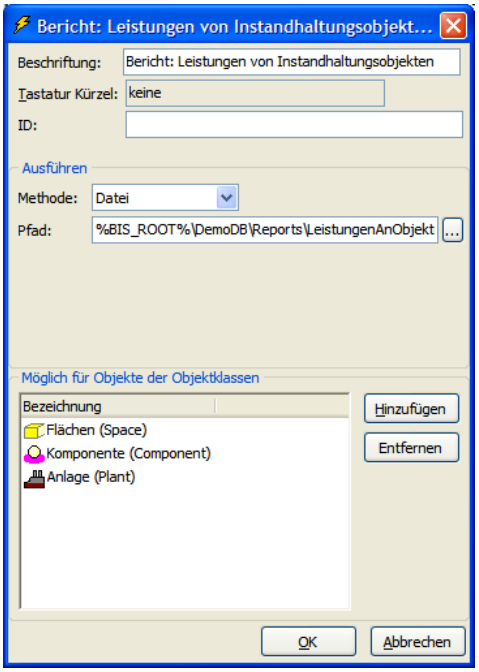


Name	Interne Bezeichnung des Formulars.
Formulargrundtyp	Im Moment gibt es nur einen Grundtypen: Eigenschaftsblatt.
Beschreibung	Eine kurze Beschreibung des Formulars (Funktion, Aufruf, …)
Zugeordnete Objektklassen	Für die Objektklassen in der Liste wird das Formular beim Benutzen des Menüpunktes 'Öffnen' geöffnet.

2.3 Aktion

Es können folgende Aktionen ausgeführt werden:

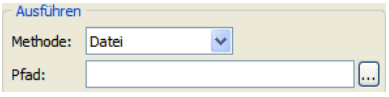
- Öffnen von Dateien
- Starten von externen Programmen
- Öffnen von Formularen
- Aufruf von globalen Funktionen in BIS



Beschriftung	Beschriftung, welche an der Oberfläche erscheint.
Tastaturkürzel	Funktionstaste oder Tastenkombination mit dem das Menü aktiviert werden kann. Bedienung: den Cursor in das Feld setzen und dann die entsprechende Taste oder Tastenkombination drücken.
ID	
Methode	Siehe Die Gruppe 'Ausführen'
Pfad	Siehe Die Gruppe 'Ausführen'
Möglich für Objekte	In dieser Liste wird definiert, für welche Objektklassen die Aktion aktiv sein soll. Wenn die Liste keinen Eintrag enthält, ist die Aktion für alle Objektklassen aktiv.

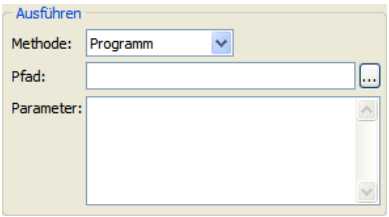
2.3.1 Die Gruppe 'Ausführen'

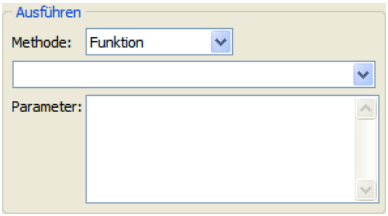
In der Gruppe *Ausführen* wird festgelegt, welche Aktion beim Anwählen des Menüs ausgeführt werden soll. Zur Auswahl stehen die Methoden *Datei*, *Programm*, *Formular* und *Funktion*. Je nach Wahl müssen andere Parameter eingegeben werden.



Pfad gibt den Ort einer Datei an, welche sich im Windows-Explorer mittels Doppelklick öffnen lässt. Der Knopf '...' hilft bei der Auswahl.

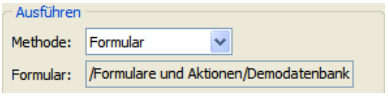
Beispiel:
Pfad: C:\temp\Formular-Referenz.pdf





Pfad gibt den Ort eines ausführbaren Programms (*.exe) an. Der Knopf '...' hilft bei der Auswahl. Im Feld *Parameter* können zusätzliche Angaben dem Programm mitgegeben werden.

Beispiel:
Pfad: C:\Programme\Microsoft Office\Office\WINWORD.EXE
Parameter: C:\temp\Formulare Referenz.doc



Formular gibt das Formular an, welches geöffnet werden soll. Das Formular muss dazu mit Drag & Drop aus dem Konfigurations-Explorer in das (gesperrte) Eingabefeld gezogen werden.

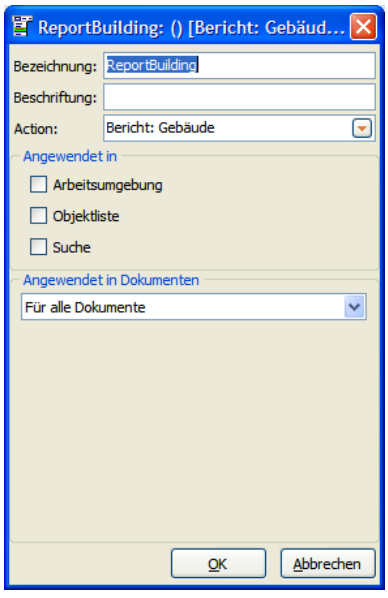
In der Combobox kann eine der folgenden Funktionen ausgewählt werden:

- Erzeugen eines Objekts
- Erzeugen von Aufträgen aus Instandhaltungsaufgaben

Das Verhalten der Funktion und deren Parameter wird in einem gelben Hinweis-Fenster über der Combobox kurz erklärt.

Die Funktionen sind mit Lizenzen verknüpft und deswegen evtl. nicht wählbar.

2.4 Menü



Bezeichnung	Interne Bezeichnung des Menüs
Beschriftung	Beschriftung, die an der Oberfläche erscheint. Spezielle Zeichen: Ein '-' (Minuszeichen) als Beschriftung wird als Trennlinie interpretiert. Ein '&' vor einem Buchstaben führt zum Unterstreichen desselben und zur Markierung als ShortCut. Beispiele: 'Bearbeiten' lässt sich also durch den Text '&Bearbeiten' erzeugen. 'Drag & Drop' wird durch den Text 'Drag && Drop' erzeugt.
Action	Action, welche durch das Menü aufgerufen wird.
Arbeitsumgebung	Der Menüpunkt soll in der Byron/BIS Arbeitsumgebung erscheinen.
Objektliste	Der Menüpunkt soll im Dialog *Objektliste* erscheinen.
Suche	Der Menüpunkt soll im Dialog *Suchen* erscheinen.

Angewendet in Dok.	Definiert in welchen Explorern der Menüpunkt erscheinen soll.

2.5 Ordner

Ordner dienen nur der Strukturierung der Formulare, Aktionen und Menüs im Explorer.

Ordner können andere Ordner enthalten.

2.6 Funktionen für Fortgeschrittene

2.6.1 Importieren und Exportieren von Formularen (Menüs)

Formulare und Menüs können wie alle BIS-Objekte mittels BIX von einer Datenbank zur anderen übertragen werden.

2.6.2 Formular als Text

Der Menüpunkt '*Extras -> Formular als Text...*' öffnet die Formulardefinitionen in einer textuellen Ansicht.

Hier kann der erfahrene Benutzer mit der notwendigen Vorsicht, Änderungen unabhängig vom unten beschriebenen Formular-Editor vornehmen.

2.6.3 Definition der Reihenfolge der Menüeinträge

Die Reihenfolge der Menüeinträge wird durch alphabetische Sortierung der Menü-Objekte nach ihrer Bezeichnung vorgegeben.

Die richtige Reihenfolge kann durch geschickte Wahl der Bezeichnung erzwungen werden (z.B. '22 Monatsbericht', '23 Trennlinie nach Monatsbericht', ...).

3 Der Formulareditor

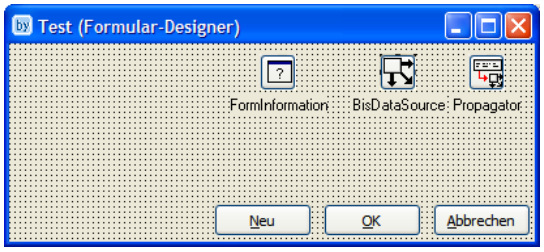
Um ein Formular in den Formulareditor zu bekommen, wählen wir im Konfigurations-Explorer ein Formular an und benutzen die Menüposition '*Bearbeiten -> Formular bearbeiten*' oder den Eintrag *Formular bearbeiten* im Popup-Menü.

Es erscheinen folgende Fenster:

Formular-Fenster	WYSIWYG-Darstellung des Formulars
Komponentenpalette	Enthält eine Sammlung aller Komponenten, welche in das Formular eingefügt werden können.
Objektinspektor	Hier werden die Eigenschaften der Komponenten im Formular angezeigt und können geändert werden.
Ausrichtungspalette	Hilfsfenster zum Anordnen der Komponenten.
Skript-Editor	Gibt die Möglichkeit bei Ereignissen im Formular (z.B. Wert in einem Feld geändert, Drücken eines Knopfes, …) Skripts auszuführen.

3.1 Das Formular-Fenster

Das Formular-Fenster beinhaltet das zu ändernde Formular in einer WYSIWYG-Darstellung. Gewisse Elemente sind bereits im leeren Formular vordefiniert.



3.1.1 Manipulation der Benutzerschnittstellen-Komponenten

Die Komponenten können mit der Maus gewählt und verschoben werden. Die meisten Komponenten erlauben auch eine Veränderung der Grösse (schwarze Rechtecke anwählen und ziehen).

Die Änderungen mit der Maus bewegen sich aber immer in einem Raster (dargestellt durch die Punkte). Eine gewählte Komponente kann aber auch mit Hilfe der Pfeiltasten bewegt (CTRL+Pfeiltasten) und vergrössert bzw. verkleinert werden (SHIFT+Pfeiltasten).

Mehrere Komponenten können durch zwei Verfahren gewählt werden:

- Mausklick bei gedrückter SHIFT-Taste.
- Selektion mit der Maus bei gedrückter CTRL-Taste (zieht eine Box auf).

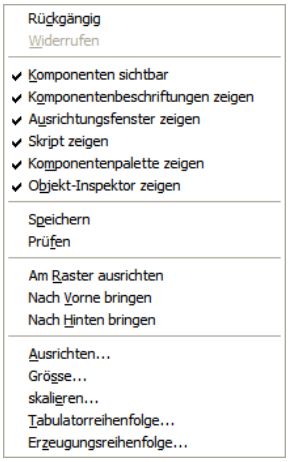
Die Eigenschaften einer aber auch mehrerer Komponenten gemeinsam können im Objektinspektor verändert werden.

Hinweis: Bei der Selektion mehrerer Komponenten wird immer die zuerst selektierte Komponente als Referenz verwendet (z.B. beim Ausrichten).

3.1.2 Das Kontextmenü

Das Kontextmenü wird beim Drücken der rechten Maustaste angezeigt. Es teilt sich in zwei Teile: Steuerung der Darstellung des Formulareditors (diese Einstellungen werden abgespeichert!) und zusätzliche Funktionen zur Manipulation der Komponenten.

Die Menüeinträge im einzelnen:



Komponenten sichtbar	Steuert die Sichtbarkeit der 'zur Laufzeit unsichtbaren' Komponenten wie FormInformation, BisDataSource, ...
Komp.beschriftungen ...	Steuert die Sichtbarkeit der Beschriftungen der 'zur Laufzeit unsichtbaren' Komponenten wie FormInformation, BisDataSource, ...
Ausrichtungsfenst. zeigen	Steuert die Sichtbarkeit des Ausrichtungsfensters.
Skript zeigen	Steuert die Sichtbarkeit des Skript Editors.
Komponentenpal. zeigen	Steuert die Sichtbarkeit der Komponentenpalette.
Objekt-Inspektor zeigen	Steuert die Sichtbarkeit des Objektinspektors.
Speichern	Speichert das aktuelle Formular.
Prüfen	Prüft das aktuelle Formular.
Am Raster ausrichten	Passt die selektierten Komponenten an das Raster an.
Nach Vorne bringen	Holt die selektierten Komponenten in den Vordergrund.
Nach Hinten bringen	Setzt die selektierten Komponenten in den Hintergrund.
Ausrichten…	Öffnet ein Dialogfenster zum Anordnen der selektierten Komponenten. Die angebotenen Funktionen sind dieselben wie auf dem Ausrichtungsfenster.
Grösse…	Öffnet ein Dialogfenster zum Anpassen der Breite bzw. Höhe der selektierten Komponenten (auf das Minimum, Maximum oder einen Zahlenwert). Das Setzen von Höhe und Breite auf einen Zahlenwert lässt sich allerdings auch im Objektinspektor bewerkstelligen.
skalieren...	Öffnet ein Dialogfenster zum Skalieren der selektierten Komponenten.
Tabulatorreihenfolge...	Öffnet ein Dialogfenster mit welchem die Reihenfolge verändert wird, mit welcher die Komponenten beim Drücken der Tabulatortaste durchlaufen werden. Die Reihenfolge ist durch die Eigenschaft *TabOrder* der Komponenten gegeben, welche sich aber mit dem Objektinspektor etwas mühsam bearbeiten lässt.
Erzeugungsreihenfolge...	nicht verwenden

3.1.3 Vordefinierte Komponenten

FormInformation	Definiert das globale Verhalten des Formulars.
BisDataSource	Enthält das Datenbankobjekt, für welches das Formular geöffnet wurde.
Propagator	Diese Komponente wird benötigt, wenn das Formular auf Änderungen der Selektion im Explorer (oder in der Grafik) reagieren soll (wird gesteuert durch die Eigenschaft *AllowBrowsing* von FormInformation*)*.

3.2 Die Komponentenpalette

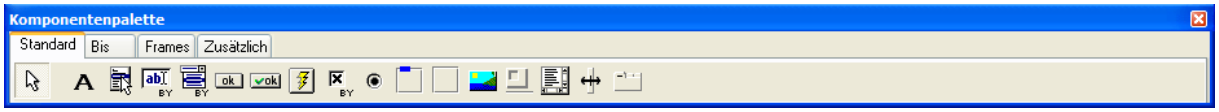
Das Palettenfenster enthält alle möglichen Komponenten, die in ein Formular eingefügt werden können.

Die Palette besitzt vier verschiedene Register:

Standard	Komponenten, die keine Beziehung zur Datenbank haben. Sie dienen vor allem der Strukturierung (Gruppierung der Felder etc.) des Formulars, sowie dem Auslösen von Aktionen (Menüs, Knöpfe).
Bis	Komponenten, die eine Beziehung zur Datenbank haben. Die meisten Komponenten dienen der Darstellung und Manipulation von Attributen der Datenbank-Objekten.
Frames	Modulspezifische Frames.
Zusätzlich	Weitere Komponenten, die keine Beziehung zur Datenbank haben.

Die Komponenten selbst sind auf der Palette nicht beschriftet; ihr Typ wird aber mit einem gelben Hinweisfenster angezeigt, sobald der Mauszeiger über die Symbole bewegt wird.

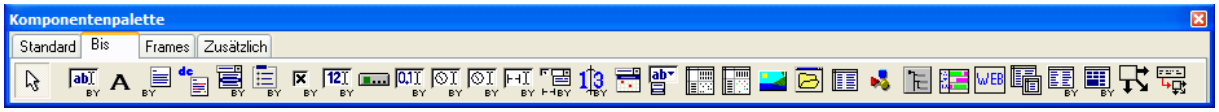
3.2.1 Die Palettenseite 'Standard'



Label	Einfaches Textfeld, dient der Beschriftung usw.
ByPopupMenu	Ein Kontextmenü (wird in der Regel über die rechte Maustaste aktiviert).
ByBaseEdit	Ein einfaches Eingabefeld (für Texte).
ByCombobox	Eine Combobox zur Eingabe von Aufzählungen oder zur Eingabe von Texten mit Vorgaben.
Button	Ein Knopf der eine Aktion (oder ein Skript) auslösen kann.
ByBitBtn	
SpeedButton	Ein Knopf, der mit einem Bitmap versehen werden kann.
ByCheckbox	Ein Eingabefeld für Werte wie 'ja/ nein', 'gesetzt/nicht gesetzt'.
RadioButton	
GroupBox	Rahmen mit Titel, dient der Zusammenfassung von Oberflächenelementen.
Panel	Rahmen ohne Titel, dient der Zusammenfassung von Oberflächenelementen. Der Rahmen selbst kann auch unsichtbar werden.
Image	Darstellung einer Grafik. Unterstützt werden die Formate JPG, BMP, ICO und Windows-Metafile (EMF, WMF).
Bevel	Einfaches grafisches Objekt (Box, Rahmen, Line)
ScrollBox	Ein Panel mit Scrollbalken.
Splitter	Ein Teiler, mit dem das Grössenverhältnis zwischen zwei Oberflächenelementen verändert werden kann.

ByPageControl	Das Oberflächenelement besteht aus mehreren Seiten, die mittels Registerkarten gewählt werden können.

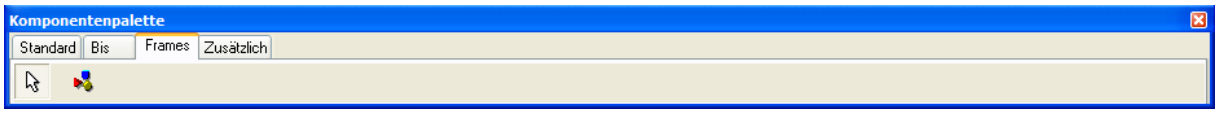
3.2.2 Die Palettenseite 'Bis'



FormInformation	Definiert das globale Verhalten des Formulars.
BisEditLinked	Einzeiliges Ein-/Ausgabefeld für Text-Attribute.
BisLabelLinked	Feld für Attributbezeichnungen.
BisMemoLinked	Mehrzeiliges Ein-/Ausgabefeld für Text-Attribute.
BisComboboxLinked	Ein-/Ausgabefeld für Attribute vom Typ *Aufzählung* für Datenbank-Objekten.
BisRadioGroupLinked	
BisCheckboxLinked	Ein-/Ausgabefeld für Attribute vom Typ *Boolean* ('Ja/Nein' bzw. 'gesetzt/nicht gesetzt').
BisIntEditLinked	Ein-/Ausgabefeld für Attribute vom Typ *ganze Zahl*.
BisColorButtonLinked	
BisFloatEditLinked	Ein-/Ausgabefeld für Attribute vom Typ *Gleitkommazahl*.
BisTimeEditLinked	Ein-/Ausgabefeld für Attribute vom Typ *Zeit*. Je nach Einstellung wird nur der Tag-Anteil oder nur der Zeit-Anteil oder beide Teile dargestellt.
BisDateEditLinked	Ein-/Ausgabefeld für Attribute vom Typ *Zeit*. Es wird nur der Tagesanteil bearbeitet. Das Feld enthält zusätzlich einen Knopf mit einem Popup-Kalender zur einfachen Eingabe von Kalendertagen.
BisIntervalEditLinked	Ein-/Ausgabefeld für Attribute vom Typ *Intervall*.
BisIntervalEditExLinked	Ein-/Ausgabefeld für Attribute vom Typ *Intervall*. Neuere Ausführung, die eine getrennte Eingabe von Wert (eine Zahl) und Einheit (Minute, Stunde, ...) erlaubt.
BisTerminEditLinked	
BisLookupCombobox	Combobox, für die Ein-/Ausgabe einer Beziehung. Verwendet *BisDataSet* zum Bestimmen der Objekt-Auswahl.
BisCrossTable	
BisObjectCrossTable	
BisImageLinked	Zeigt eine im Dateisystem existierende Bilddatei (*.bmp, *.jpg, ...).
BisDynOpenDialog	
BisDynSimpleGridII	Eine Liste mit Datenbank-Objekten mit ihren Attributen. In der Regel werden die Objekte einer Beziehung dargestellt (z.B. der Rauminhalt). Es lassen sich weitere Objekte zur Liste hinzufügen (mittels Drag & Drop) oder entfernen. Die Attribute der Objekte der Liste lassen sich ebenfalls verändern.
BisDynSimpleTree	Zeigt verknüpfte Objekte in einem Baum. Ausgehend von einem oder mehreren Startobjekte wird mit Hilfe der

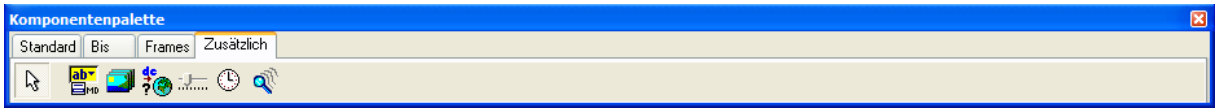
	Navigationsregeln eines *DataSets*. Eine Hierarchie aufgebaut und dargestellt.
BisPopupTree	
BisDynGanttLinked	
BisDynWebBrowser	
BisDataSet	Eine unsichtbare Komponente, die zu 'Beschaffung' von Objekten dient (z.B. für die Objektauswahl einer *BisLookupCombobox)*. Dazu enthält sie Navigationsregeln und Filter.
BisDataSource	Enthält ein Datenbankobjekt. Die Komponente wird beim Aufstarten des Formulars oder durch eine Komponente vom Typ *TBisObjectPropagator* mit einem Datenbankobjekt geladen. Die Operationen 'Laden der Daten aus der Datenbank' sowie 'Speichern der Daten in die Datenbank' werden von der *BisDataSource* an alle verknüpften Benutzerschnittstellen-Komponenten weitergegeben.
BisObjectPropagator	Eine Unsichtbare Komponente. Sie dient der Weitergabe von Selektionen an eine *BisDataSource*. Zu diesem Zweck wird Sie an eine *BisLookupCombobox*, ein *BisDynSimpleGridII* oder an einen *BisDynSimpleTree* angeschlossen.

3.2.3 Die Palettenseite 'Frames'



BisRemovalEdit

3.2.4 Die Palettenseite 'Zusätzlich'



ImageList	Unsichtbare Komponente, enthält eine Liste von Bildern. Im Moment nicht benötigt.
Animate	Zeigt eine Animation.
Timer	

3.3 Der Objektinspektor

Der Objektinspektor zeigt zum aktuell gewählten Oberflächenelement auf der Seite 'Eigenschaften' alle Eigenschaften an, welche geändert werden können.

Die Seite 'Ereignisse' enthält eine Liste der Ereignisse, die das gewählte Oberflächenelement auslösen kann, und mit denen Skript-Routinen angestossen werden können.

Der Objektinspektor kennt zwei Darstellungsmodi, welche mit der rechten Maustaste umgeschaltet werden können ('Zeige Kategorien'):

- Alphabetische Sortierung der Eigenschaften
- Gruppierung der Eigenschaften in Kategorien

3.4 Die Ausrichtungspalette

Das Ausrichtungsfenster ermöglicht ein schnelles Anordnen mehrer Komponenten. Die Bedeutung der einzelnen Symbole wird in einem gelben Hinweisfenster gezeigt, sobald der Mauszeiger über die Symbole fährt. Die Funktionen sind (von links nach rechts):

- Linke Kanten ausrichten
- Horizontale Mitten ausrichten
- Horizontal im Fenster zentrieren
- Abstand angleichen, horizontal
- Rechte Kanten ausrichten.
- Oberkanten ausrichten
- Vertikale Mitten ausrichten
- Vertikal im Fenster zentrieren
- Abstand angleichen, vertikal
- Unterkanten ausrichten.

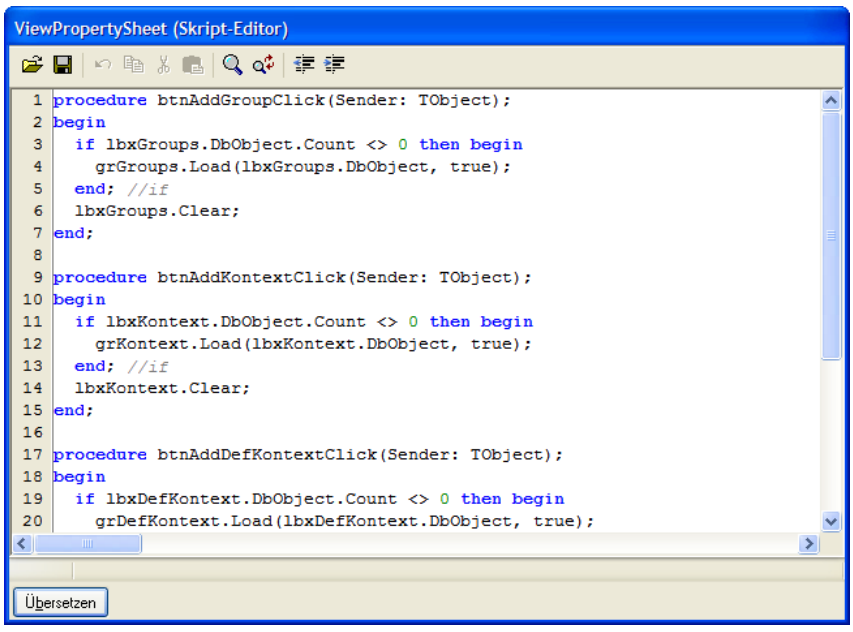
Hinweis: Bei der Selektion mehrerer Komponenten wird immer die zuerst selektierte Komponente als Referenz verwendet.

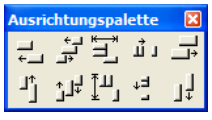
3.5 Der Skript Editor

Das Skriptfenster zeigt alle Ereignisprozeduren für alle Komponenten, die sich auf dem Formular befinden. Die in den Skripts verwendete Sprache ist (Object-)Pascal bzw. Delphi-Script.

Zusätzlich können Sie eigene Prozeduren schreiben und aufrufen (Bedingung – sie müssen oberhalb des Aufrufs definiert sein.)

Natürlich können auch andere Ereignisprozeduren aufgerufen werden – als Parameter für *Sender* wird am besten *nil* verwendet.





3.5.1 Skripts debuggen

Um die Skripts zu debuggen, muss der 'Dream Script Debugger' installiert werden. Eine Test-Version ist unter 'http://www.dreamcompany.com/dsd.html' erhältlich.

4 Beschreibung der einzelnen Komponenten

Im Folgenden werden die wichtigen Komponenten und Eigenschaften beschrieben.

Nicht beschriebene Komponenten und Eigenschaften sollten nicht oder nur nach Nachfrage mit der Byron Informatik AG verwendet werden.

4.1 Gemeinsame Eigenschaften

In der folgenden Auflistung werden die Eigenschaften beschrieben, welche bei vielen oder gar allen Komponenten verwendet werden können.

Align

Typ: Aufzählung

Die Eigenschaft *Align* bestimmt, wie das Steuerelement innerhalb seines Containers (übergeordneten Komponente) ausgerichtet wird.

Werte:

alNone	Das Steuerelement bleibt an der Stelle, an der es platziert wurde. Dies ist der Standardwert.
alTop	Das Steuerelement wird an den oberen Rand des übergeordneten Elements verschoben und nimmt dessen gesamte Breite ein. Die Höhe des Steuerelements ändert sich nicht.
alBottom	Das Steuerelement wird an den unteren Rand des übergeordneten Elements verschoben und nimmt dessen gesamte Breite ein. Die Höhe des Steuerelements ändert sich nicht.
alLeft	Das Steuerelement wird an den linken Rand des übergeordneten Elements verschoben und nimmt dessen gesamte Höhe ein. Die Breite des Steuerelements ändert sich nicht.
alRight	Das Steuerelement wird an den rechten Rand des übergeordneten Elements verschoben und nimmt dessen gesamte Höhe ein. Die Breite des Steuerelements ändert sich nicht.
alClient	Die Grösse des Steuerelements wird so verändert, dass es den Client-Bereich des übergeordneten Elements ausfüllt. Wenn ein Steuerelement bereits einen Teil des Client-Bereichs belegt, wird die Grösse des neuen Steuerelements so geändert, dass es den verbleibenden Client-Bereich ausfüllt.

Beschreibung:
Mit *Align* können Sie ein Steuerelement an der oberen, unteren, linken oder rechten Seite eines Formulars oder Panels ausrichten. Die Komponente bleibt auch dann an dieser Position, wenn sich die Grösse ihres Containers ändert. Die Grösse des ausgerichteten Steuerelements wird zusammen mit dem übergeordneten Objekt geändert, so dass es sich weiterhin über den oberen, unteren, linken oder rechten Rand des Containers erstreckt.

Wenn Sie beispielsweise ein Panel mit mehreren Steuerelementen als Werkzeugpalette verwenden wollen, setzen Sie *Align* für das Panel auf *alLeft*. Dieser Wert garantiert, dass die Werkzeugpalette auf der linken Seite des Formulars bleibt und immer der Client-Höhe des Formulars entspricht.

Align hat den Standardwert *alNone*. Dieser Wert bewirkt, dass das Steuerelement an der Position bleibt, an der es in das Formular oder Panel eingefügt wurde. Geben Sie einen anderen Wert ein, um das Steuerelement auf eine bestimmte Weise auszurichten.

Tipp:
Ist *Align* auf *alClient* gesetzt, füllt das Steuerelement den gesamten Client-Bereich aus, so dass das übergeordnete Formular nicht durch Anklicken ausgewählt werden kann. Verwenden Sie in diesem Fall den Objektinspektor, oder markieren Sie das Steuerelement im Formular und drücken die Taste ESC.

Sie können auch einer beliebigen Anzahl von untergeordneten Komponenten den gleichen *Align*-Wert zuweisen. Die Komponenten werden dann entlang des Rands der übergeordneten Komponente gestapelt. Durch Ziehen der Steuerelemente mit der Maus können Sie den Stapel auf die gewünschte Weise anordnen.

Hinweis:
Damit ein Steuerelement eine bestimmte Beziehung mit dem Rand seiner übergeordneten Komponente beibehält, ohne notwendigerweise direkt am Rand ausgerichtet zu werden, verwenden Sie stattdessen die Eigenschaft *Anchors*.

Alignment

Typ: Aufzählung

Mit *Alignment* legen Sie fest, wie der Text innerhalb des Begrenzungsrechtecks der Komponente ausgerichtet wird.
Der Effekt der Eigenschaft *Alignment* wird deutlicher, wenn die Eigenschaft *WordWrap* gesetzt ist und der Beschriftungstext mehrere Zeilen enthält.

Mögliche Werte (Aufzählung): taLeftJustify, taCenter, taRightJustify

Anchors

Typ: zusammengesetzt

Die Eigenschaft *Anchors* legt fest, wie das Steuerelement in seinem übergeordneten Objekt verankert wird.

Felder:

akTop	Das Steuerelement wird am oberen Rand des übergeordneten Objekts verankert.
akLeft	Das Steuerelement wird am linken Rand des übergeordneten Objekts verankert.
akRight	Das Steuerelement wird am rechten Rand des übergeordneten Objekts verankert.
akBottom	Das Steuerelement wird am unteren Rand des übergeordneten Objekts verankert.

Mit *Anchors* können Sie sicherstellen, dass ein Steuerelement selbst bei Grössenänderungen des Containers seine Position relativ zu dessen Rand beibehält. Wenn die Grösse des übergeordneten Objekts geändert wird, behält das Steuerelement seine Position relativ zu den Rändern bei, an denen es verankert ist.

Wenn ein Steuerelement an gegenüberliegenden Seiten seines übergeordneten Objekts verankert ist, wird es bei Grössenänderungen des Containers gestreckt. Hat beispielsweise die Eigenschaft *Anchors* eines Steuerelements den Wert *[akLeft, akRight]*, wird das Steuerelement gestreckt, wenn das übergeordnete Objekt verbreitert wird.

Anchors ist nur wirksam, wenn die Grösse des übergeordneten Objekts geändert wird. Wenn z.B. ein Steuerelement beim Entwurf an gegenüberliegenden Seiten eines Formulars verankert wird und das Formular maximiert erstellt wird, wird das Steuerelement nicht gestreckt, weil die Grösse des Formulars nach der Erzeugung des Steuerelements nicht verändert wurde.

Hinweis:
Wenn ein Steuerelement den Kontakt mit drei Rändern seines Containers behalten soll, verwenden Sie stattdessen die Eigenschaft *Align*. Im Gegensatz zu *Anchors* ermöglicht *Align*, dass Steuerelemente bei Grössenänderungen anderer gleichrangiger Komponenten sowie übergeordneter Objekte entsprechend angepasst werden.

AutoSize

Typ: Boolean

Mit *AutoSize* legen Sie fest, ob das Steuerelement seine Grösse automatisch anpasst. Ist *AutoSize* gesetzt, dann wird das Steuerelement automatisch angepasst, wenn sich sein Inhalt ändert.
Standardmässig ist *AutoSize* nicht gesetzt.

BorderStyle

Typ: Aufzählung

Die Eigenschaft *BorderStyle* legt fest, ob der Client-Bereich im Eingabefeld eine einzelne Linie als Rahmen verwendet.

Werte:

bsNone	Kein sichtbarer Rahmen
bsSingle	Einfacher Rahmen

BorderWidth

Typ: ganze Zahl

Die Eigenschaft *BorderWidth* gibt die Stärke des Steuerelementrahmens an.

Color

Typ: Farbauswahl (mit Dialog)

Mit *Color* können Sie die Hintergrundfarbe des Steuerelements ermitteln oder ändern.

Wenn die Eigenschaft *ParentColor* eines Steuerelements gesetzt ist, dann wirkt sich eine Änderung der Eigenschaft *Color* des übergeordneten Steuerelements automatisch auch auf die Eigenschaft *Color* des untergeordneten Steuerelements aus. Wenn Sie der Eigenschaft *Color* eines Steuerelements einen Wert zuweisen, wird *ParentColor* automatisch auf 'false' gesetzt.

Constraints

Typ: Zusammengesetzt

Die Eigenschaft *Constraints* gibt die Grössenbeschränkungen des Steuerelements an.

Felder:

	MaxHeight ganze Zahl, Maximale Höhe in Pixel
	MaxWidth ganze Zahl, Maximale Breite in Pixel
	MinHeight ganze Zahl, Minimale Höhe in Pixel
	MinWidth ganze Zahl, Minimale Breite in Pixel
DataLink	Typ: Zusammengesetzt
	Verbindung mit der Datenbank, zur Erklärung siehe weiter oben im Text.
DisabledColor	Typ: Farbauswahl (mit Dialog)
	Farbe, die das Feld annimmt wenn *ReadOnly* gesetzt ist
Enabled	Typ: Boolean
	Mit *Enabled* können Sie festlegen, ob das Steuerelement für den Benutzer verfügbar ist. Um es zu deaktivieren, setzen Sie die Eigenschaft auf 'false'. Deaktivierte Steuerelemente werden grau dargestellt und ignorieren Maus-, Tastatur- und Timer-Ereignisse
Font	Typ: Zusammengesetzt
	Wenn Sie die Attribute einer Schrift ändern wollen, weisen Sie den Eigenschaften *Charset*, *Color*, *Height*, *Name*, *Pitch*, *Size* oder *Style* die gewünschten Werte zu.
	Siehe auch *ParentFont*.
Format	Typ: Text
	Darstellungsformat, wird nur verwendet wenn in der Eigenschaft *DataLink* das Feld *UseFormatFromDb* nicht gesetzt ist.
Height	Typ: ganze Zahl
	Die Eigenschaft *Height* gibt die Höhe des Steuerelements in Pixel an.
HideSelection	Typ: Boolean
	Die Eigenschaft *HideSelection* legt fest, ob die visuelle Kennzeichnung des markierten Textes erhalten bleibt, wenn der Fokus an ein anderes Steuerelement übergeht.
Hint	Typ: Text
	Mit *Hint* können Sie einen Text zuweisen, der als Kurzhinweis oder an einer bestimmten Position (z.B. Statusleiste) als Hilfetext angezeigt wird.
InvalidColor	Typ: Farbauswahl (mit Dialog)
	Farbe, die das Feld annimmt wenn eine ungültige Eingabe erfolgt ist oder das Feld nicht korrekt an die Datenbank angeschlossen ist.
Left	Typ: ganze Zahl
	Die Eigenschaft *Left* gibt die horizontale Koordinate des linken Komponentenrandes relativ zu seinem übergeordneten Objekt in Pixel an.
NoSpaces	Typ: Boolean
	Die Eigenschaft *NoSpaces* legt fest ob Leerschläge als Eingabe erlaubt sind.

ParentColor	Typ: Boolean Setzen Sie *ParentColor* auf 'true', wenn ein Steuerelement die Farbe seines übergeordneten Steuerelements verwenden soll. Ist *ParentColor* nicht gesetzt, dann verwendet das Steuerelement seine eigene Eigenschaft *Color*. Setzen Sie die Eigenschaft *ParentColor* sämtlicher Komponenten, um sicherzustellen, dass alle Steuerelemente in einem Formular ein einheitliches Erscheinungsbild haben. Sie können dann beispielsweise durch Ändern der Hintergrundfarbe des Formulars allen Steuerelementen dieselbe Hintergrundfarbe zuweisen. Wenn Sie den Wert der Eigenschaft *Color* eines Steuerelements ändern, wird *ParentColor* automatisch auf 'false' gesetzt.
ParentFont	Typ: Boolean Setzen Sie *ParentFont*, wenn ein Steuerelement die Schrift seines übergeordneten Steuerelements verwenden soll. Hat *ParentFont* den Wert 'false', verwendet das Steuerelement seine eigene *Font*-Einstellung. Wenn alle Steuerelemente in einem Formular einheitlich aussehen sollen, setzen Sie die Eigenschaft *ParentFont* für alle Elemente. Ist beispielsweise *ParentFont* für alle Steuerelemente in einem Formular 'true', erhalten diese Steuerelemente die Schrift Courier 12 Punkt, wenn der Eigenschaft *Font* des Formulars diese Schrift zugewiesen wird. Wenn Sie den Wert der Eigenschaft *Font* eines Steuerelements ändern, wird *ParentFont* automatisch auf 'false' gesetzt. Ist die Eigenschaft *ParentFont* eines Formulars gesetzt, wird die Standardschriftart verwendet.
ParentShowHint	Typ: Boolean Mit *ParentShowHint* können Sie sicherstellen, dass bei allen Steuerelementen in einem Formular Kurzhinweise angezeigt werden oder nicht. Wenn *ParentShowHint* gesetzt ist, verwendet das Steuerelement die Eigenschaft *ShowHint* seines übergeordneten Objekts. Andernfalls verwendet das Steuerelement den Wert seiner eigenen Eigenschaft *ShowHint*. Wenn nur bestimmte Steuerelemente im Formular Kurzhinweise anzeigen sollen, setzen Sie die Eigenschaft *ShowHint* dieser Steuerelemente. In diesem Fall wird der Eigenschaft *ParentShowHint* automatisch 'false' zugewiesen.
PopupMenu	Typ: Komponente Gibt an welches Popupmenü auf der Komponente gezeigt werden soll, wenn der rechte Mausknopf gedrückt wird.
ReadOnly	Typ: Boolean Die Eigenschaft *ReadOnly* legt fest, ob der Benutzer den Text im Eingabefeld bearbeiten kann. **Achtung**: diese Eigenschaft wird bei vielen Komponenten beim Laden des Wertes aus der Datenbank zurückgesetzt. Änderungen durch das

	Formularskripting müssen also nach dem Laden (OnOperate) „wiederholt“ werden.
ShowHint	Typ: Boolean Der für ein Steuerelement angezeigte Kurzhinweis wird mit der Eigenschaft *Hint* angegeben. Mit *ShowHint* legen Sie fest, ob für das Steuerelement ein Kurzhinweis eingeblendet wird. Um diese Funktion für ein bestimmtes Steuerelement zu aktivieren, muss die Eigenschaft *ShowHint* des Anwendungsobjekts den Wert 'true' haben. Zusätzlich muss eine der folgenden Bedingungen erfüllt sein: Die Eigenschaft *ShowHint* des Steuerelements muss 'true' sein. Die Eigenschaft *ParentShowHint* des Steuerelements und die Eigenschaft *ShowHint* seines übergeordneten Objekts müssen 'true' sein.
Softvalidation	Typ: Boolean Wenn die Eigenschaft *Softvalidation* gesetzt ist, wird bei einer ungültigen Eingabe das Feld mit *InvalidColor* dargestellt, sonst wird die Eingabe zurückgewiesen.
TabOrder	Typ: ganze Zahl Die Eigenschaft *TabOrder* gibt die Position eines Steuerelements in der Tabulatorreihenfolge seines übergeordneten Fensters an.
TabStop	Typ: Boolean Die Eigenschaft *TabStop* bestimmt, ob der Benutzer mit der Tabulatortaste zu einem Steuerelement wechseln kann.
Top	Typ: ganze Zahl Die Eigenschaft *Top* gibt die Y-Koordinate der linken oberen Ecke eines Steuerelements relativ zu seinem Container in Pixel an.
Transparent	Typ: Boolean Setzen Sie *Transparent* auf 'true', wenn die Beschriftung keine anderen Steuerelemente verdecken soll. Auf diese Weise können Sie z.B. Text in eine Grafik integrieren.
ValidColor	Typ: Farbauswahl (mit Dialog) Farbe, die das Feld annimmt wenn das Feld einen gültigen Wert enthält..
Visible	Typ: Boolean Mit *Visible* können Sie die Sichtbarkeit des Steuerelements zur Laufzeit steuern. Wenn *Visible* gesetzt ist, wird das Steuerelement angezeigt. Hat die Eigenschaft den Wert 'false', ist es nicht zu sehen.
WantTabs	Typ: Boolean Die Eigenschaft *WantTabs* bestimmt, ob der Benutzer Tabulatorzeichen in den Text einfügen kann.
WantToDefault	Typ: Boolean

	Die Eigenschaft *WantToDefault* bestimmt, ob der Benutzer mit der Taste Fragezeichen den Wert auf den Default-Wert zurücksetzen kann.
Width	Typ: ganze Zahl Die Eigenschaft *Width* gibt die Breite des Steuerelements oder Formulars in Pixel an.

4.2 Palettenseite 'Standard'

4.2.1 Label

Label ist ein Steuerelement, mit dem Text in einem Formular angezeigt werden kann. Sie können diesen Text als Beschriftung für ein anderes Steuerelement verwenden und den Fokus an dieses Steuerelement übergeben, sobald der Benutzer ein im Text enthaltenes Tastenkürzel eingibt.

Spezifische Eigenschaften

Caption	Typ: Text Beschriftung die angezeigt wird
WordWrap	Die Eigenschaft legt fest, ob der Beschriftungstext bei Bedarf umbrochen wird.

4.2.2 ByPopUpMenu

4.2.3 ByBaseEdit

4.2.4 ByCombobox

4.2.5 Button

4.2.6 ByBitButton

4.2.7 SpeedButton

4.2.8 ByCheckbox

4.2.9 RadioButton

4.2.10 GroupBox

Die Komponente GroupBox repräsentiert ein Standard-Windows-Gruppenfeld, mit dem funktional zusammengehörige Steuerelemente eines Formulars gruppiert werden können. Sobald ein weiteres Steuerelement innerhalb einer Gruppe platziert wird, ist das Gruppenfeld das übergeordnete Objekt dieser Komponente.

Caption	Typ: Text Beschriftung die im Titel angezeigt wird

4.2.11 Panel

Verwenden Sie ein Panel, um eine leere Tafel in ein Formular einzufügen. Mit den zugehörigen Eigenschaften kann ein dreidimensional wirkender Rahmen um dieses Steuerelement gelegt werden. Ausserdem verfügen Tafeln über Methoden, die die Positionierung von untergeordneten Steuerelementen innerhalb der Tafel ermöglichen.

Spezifische Eigenschaften

Caption	Typ: Text Beschriftung die im Zentrum angezeigt wird
BevelInner	Typ: Aufzählung Die Eigenschaft *BevelInner* bestimmt den Stil der inneren Schrägkante einer Tafel. Werte: bvNone – Es wird kein abgeschrägter Rahmen angezeigt. bvLowered – Der abgeschrägte Rahmen wird vertieft dargestellt. bvRaised – Der abgeschrägte Rahmen wird erhaben dargestellt. bvSpace – Der abgeschrägte Rahmen hat eine abgeflachte Kante, wenn für sein Aussehen (TBevelKind) nicht bkTile festgelegt ist. Andernfalls wird er erhaben dargestellt.
BevelOuter	Typ: Aufzählung Die Eigenschaft *BevelOuter* bestimmt den Stil der äusseren Schrägkante einer Tafel. Werte wie bei *BevelInner.*
BevelWidth	Typ: ganze Zahl Die Eigenschaft *BevelWidth* legt die Gesamtbreite für die innere und äussere Schrägkante einer Tafel in Pixel fest.

4.2.12 Image

4.2.13 Bevel

4.2.14 Scrollbox

4.2.15 Splitter

Eine Splitter-Komponente ist eine Teilerleiste, die den Client-Bereich eines Formulars in einzelne Felder variabler Grösse gliedert.

Wenn in einem Formular eine Teilerleiste zwischen zwei oder mehr ausgerichteten Steuerelementen eingefügt wird, kann der Benutzer diese Steuerelemente zur Laufzeit in der Grösse verändern. Eines der Steuerelemente muss am Rand des Formulars ausgerichtet sein, während die anderen den verbleibenden Platz im Formular ausfüllen. Die Teilerleiste benötigt dieselbe Ausrichtung wie das Steuerelement, das an einem Rand des Formulars verankert ist. Sobald die Teilerleiste verschoben wird, passt sich die Grösse des verankerten Steuerelements automatisch an, wodurch der Client-Bereich des Formulars verändert wird. Die anderen Steuerelemente füllen dann immer den verbleibenden Rest des Client-Bereichs aus.

Eine Teilerleiste ist immer dann sinnvoll, wenn ein Formular aus mehreren Teilflächen mit variabler Grösse bestehen soll. Das letzte Steuerelement, das eine Teilfläche bildet (z.B. ein Memofeld) muss immer am Client ausgerichtet (*alClient*) sein, damit es den verbleibenden Platz automatisch ausfüllt.

Spezifische Eigenschaften

AutoSnap	Typ: Boolean

Die Eigenschaft *AutoSnap* legt fest, ob die Grösse von benachbarten Objekten auf Null gesetzt wird, wenn die Grösse dieser Objekte mit der Teilerleiste auf weniger als in *MinSize* angegeben gesetzt wird.

Beveled

Typ: Boolean

Die Eigenschaft *Beveled* bestimmt, ob die Teilerleiste entlang der verschiebbaren Kanten mit einer abgeschrägten Linie versehen wird.

MinSize

Typ: ganze Zahl

Die Eigenschaft *MinSize* bestimmt die minimale Grösse in Pixel des Bereichs auf jeder Seite der Teilerleiste.

ResizeStyle

Typ: Aufzählung

Die Eigenschaft *ResizeStyle* bestimmt die Auswirkung des Verschiebens der Teilerleiste.

Werte:

rsNone	Das Verschieben der Teilerleiste hat keine Auswirkung.
rsLine	Es wird eine Linie angezeigt, die die neue Position der Teilerleiste angibt, wenn der Benutzer die Maustaste an der aktuellen Position losliesse
rsUpdate	Die Grösse des Steuerelements, an dem die Teilerleiste ausgerichtet ist, wird entsprechend der aktuellen Mausposition geändert, und die Teilerleiste wird an diese Position verschoben.
rsPattern	Es wird ein Muster angezeigt, die die neue Position der Teilerleiste angibt, wenn der Benutzer die Maustaste an der aktuellen Position losliesse.

4.2.16 ByPageControl

4.3 Palettenseite 'BIS'

4.3.1 FormInformation

Definiert das globale Verhalten des Formulars.

Eigenschaften

ActiveControl

Die hier ausgewählte Benutzerschnittstellen-Komponente, erhält beim Öffnen des Formulars den Tastatur-Fokus (eingegebene Zeichen werden in dieser Komponente eingetragen).

Aber **Vorsicht** – der Fokus darf nicht an ein unsichtbares oder gesperrtes (Eigenschaft Enabled ist nicht gesetzt) Fenster vergeben werden.

AllowBrowsing

Wird diese Eigenschaft gesetzt, dann verändert sich der Inhalt des Formulars mit der Selektion (des Explorers). Die Knöpfe *Ok* und *Abbrechen* sind stattdessen mit *Speichern* und *Schliessen* beschriftet.

Hinweis:
Die Eigenschaft *FilterObjectTypes* der Komponente *Propagator* sollte ebenfalls gesetzt werden (siehe unten).

AutoLocalize	Wird diese Eigenschaft gesetzt, so wird das Formular zur Laufzeit in die aktuelle verwendete Sprache lokalisiert (übersetzt). Die Übersetzungen werden dem Textkatalog entnommen.
CanAddToTaskbar	Wird diese Eigenschaft gesetzt, so wird das Formular in der Taskleiste dargestellt.
DesignLanguage	Bestimmt die Sprache in welcher das Formular im Designer bearbeitet wird. Eine Lokalisierung zur Laufzeit erfolgt nur, wenn die eingestellte Designsprache ungleich der aktuell verwendeten Laufzeitsprache ist.
OpenCaption	Steuert die Beschriftung des Formulars. Die Beschriftung folgt derselben Konvention wie die Darstellungsformel der Objekte. Beispiel: 'Raumreinigung für $Object_Display_Kontext$'.
ShowModal	Wenn die Eigenschaft gesetzt ist, wird das Formular modal geöffnet, d.h. der Zugriff auf die anderen BIS-Fenster ist nicht möglich solange das Formular nicht wieder geschlossen wurde.
StoreInSection	Eindeutiger Name, in dem die Grösse und Position des Formulars gespeichert wird, damit sie zu einem späteren Zeitpunkt wieder an derselben Stelle und Position geöffnet werden kann.
StorePosSize	Wenn die Eigenschaft gesetzt ist, merkt sich das System, an welcher Stelle und mit welcher Grösse das Formular geöffnet war und benutzt diese Einstellung beim nächsten Öffnen des Formulars.
TextCatalogID	Bestimmt den Container und das Prefix für den Textkatalog, welcher für die Lokalisierung verwendet wird. Der Textkatalog für das Formular kann nur erzeugt werden, wenn die TextCatalogID definiert ist.

Methoden

```
Procedure CallCommand(const inTool: String;
                const callerTag: String;
                const commandIdent: String;
                const commandOption: String;
                const commandPrompt: String;
                const forObjects: TBisObjectProperty);
```

Damit wird ein Command abgesetzt. Die Antwort erfolgt durch die Callbacks OnCommand…

Ein Beispiel befindet sich in 6.1.20

inTool spezifiziert das Empfänger Tool:
,' (leer) Das Tool an dem das Formular angedockt ist.
,*' Alle (geöffneten) Tools
sonst: Die Tools mit dieser Tool ID

callerTag wird nicht interpretiert und bei der Antwort zurückgeliefert, damit kann die Anwendung verschiedene Aufrufe unterscheiden.

commandIdent Der Name des Befehls, zB ‚Pick', 'PickAtPos', 'PickBorder'.

Der Befehle Pick berücksichtigt die Optionen 012, womit bestimmte Fälle ausgeschlossen werden können. Die übrigen Befehle haben dieselbe Funktionalität ohne den Optonsausschluss.

commandPrompt Hinweis Text für den Benutzer

forObjects Objekte als Parameter, je nach Command.

Folgende Funktionen stehen nur bei konfigurierten Formularen zur Verfügung

```
Procedure FormInformation.PropagateData(
    const aName : String;
    const aCode : Integer;
    const aData : Variant;
    const objects: TBisObjectProperty); (ab v4.3.0)
```

löst ein propagate aus. aName muss dann mit dem Namen in der XML-Toolkonfiguration übereinstimmen.

```
Function FormInformation.GetToolValueParameter(
    const paramName : String): TVariantList; (ab v4.2.0)
```

Sucht nach einem Tool-Parameter vom Typ "Wert" (z.B. VAL_ToolId) und liefert das Ergebnis in einer Variantenliste zurück.

```
Procedure FormInformation.SetToolValueParameter(
    const valueName : String;
    const value     : Variant); (ab v4.10.5)
```

Setzt den Wert „valueName" im Speicher (global Storage) des Tools auf den <value>. Entspricht dem FilterNavigations-Befehl „SAVEAS GLOBAL <valueName> ‚=meinWert'".

```
Function FormInformation.GetToolObjectParameter(
    const paramName : String): TBisObjectProperty; (ab v4.2.0)
```

Sucht nach einem Tool-Parameter vom Typ "Objekt" (z.B. OBJ_Tool) und liefert das Ergebnis in einer BisObjectProperty zurück.

```
Procedure FormInformation.SetToolObjectParameter(
    const paramName : String
    const value     : TBisObjectProperty); (ab v4.5.10)
```

Setzt den Wert „valueName" im Speicher (global Storage) des Tools auf <value>. Entspricht dem FilterNavigations-Befehl „SAVEAS GLOBAL <valueName>".

```
Function FormInformation.ExecuteToolOperation(const operationID: String):
Boolean; (ab v4.2.0)
```

Führt eine Tool-Operation aus. Bei erfolgreichem Ausführen ist das Ergebnis „true".

(Beispiel „Goto" siehe „Fragen & Antworten")

```
Procedure FormInformation. WriteStatusBar (
    const thePanel : String
    const theText  : String); (ab v4.10.8)
```

Schreibt den Text <theText> in die Statuszeilenfläche <thePanel>. Der Entwickler muss darauf achten, dass

- genug Panels zur Verfügung stehen, die nicht von anderen Applikationselementen überschrieben werden (Element <StatusBar> der Toolkonfiguration)
- das richtige Panel verwendet wird (Element <StatusMapping> der Toolkonfiguration)

```
Function FormInformation.GetReportParameter(const paramName: String): Vari-
ant; (ab v4.4.0)
```

Sucht nach dem Berichts-Parameter „paramName" und liefert das Ergebnis als Variant zurück, falls der Parameter gefunden wurde.

Die Funktion liefert nur ein Ergebnis, wenn das Formular in einem Bericht verwendet wird.

Callbacks

```
property OnCommandResponse: TCommandResponseEvent
TCommandResponseEvent = procedure(const resultValues: TVariantList;
                        const resultObjects: TBisObjectProperty);

property OnCommandCancel: TCommandCancelEvent
property OnCommandChanged: TCommandChanged
```

`property OnReceiveToolPropagate;` (ab v4.2.0)
wird mit folgenden Parametern aufgerufen (beim load, select und custom)

```
  propagateType  : TeleLinkType;        // eltLoad, eltSelect, eltCustom
  theObjects     : TBisObjectProperty;  // Objektliste
  code           : Integer;             // 1)
  additionalData : TVariantList;        // 2)
```

1) 0 bei load und select, additional.code bei custom
2) ApplElement-ID (String) bei load und select = self.ID, bei custom = sender.ID, additional.Name (String), additional.Data (Variant) bei custom (bis jetzt immer nil, wegen fehlender XML-Konfiguration). Beispiel „GoTo“ siehe „Fragen & Antworten“

`property OnFilterNavigateParameter;` (ab v4.2.0)
wird mit folgenden Parametern aufgerufen

```
  paramName   : ShortString;         // Name des Parameters
  valueParam  : TVariantList;        // Rückgabewert für den Parameter
  objectParam : TBisObjectProperty;  // Rückgabeobjekt für den Parameter
```

Der Callback wird für jeden Parameter in einer Filternavigation aufgerufen, sobald die Filternavigation ausgeführt wird. Somit kann in einer Filternavigation auf einen Wert zugegriffen werden, welcher aus einem konfigurierten Formular stammt.

Beispiel:

```
  …
  if paramName = 'MYOBJ_SelectedOrder' then begin
    objectParam.Add(grOrders.SelectedObjects);
  end;
  …
```

(weiteres Beispiel „GoTo“ siehe auch „Fragen & Antworten“)

`property OnLoadWorkState;` (ab v4.11.8)
`property OnStoreWorkState;` (ab v4.11.8)
wird mit folgenden Parametern aufgerufen

```
  storage   : TStrings
```

Die Callbacks werden im Rahmen eines konfigurierten Tools (vgl. ToolKonfiguration) aufgerufen, wenn das Tool die Erweiterung des Application Work State (die aktive Anwendung und den Zustand der Applicationelemente) abspeichert bzw. lädt. Das Formular hat mit diesen Callbacks die Möglichkeit zusätzliche Daten in Form von Strings abzuspeichern.

N.B: die Verwendung der Application Work States setzt die Verwendung des Functioncontrol voraus – siehe Toolkonfiguration.

Das Aufnehmen eines Application Work States geschieht in zwei Schritten/Teilen:

- Basis (Startobjekte der Application Elements)
- Erweiterung (Formulardaten)

Das Function Control speichert die Basis **nach** dem Propagate einer Funktion und die Erweiterung **vor** dem nächsten Propagate einer Funktion.

Die Operation *opWorkStateRecord* (des Application Elements) speichert direkt hintereinander zuerst die Extension des letzten States und dann die Basis des neuen States.

Die Option storeWorkStateAfterPropagate des Application Elements speichert direkt hintereinander zuerst die Extension des letzten States und dann die Basis des neuen States.

Beispiel:

```
var
  lastFunc : string; // zuletzt ausgeführte Funktion

procedure FormInformationLoadWorkstate(storage: TStrings);
var
  v : string;
begin
  v := storage.Values['lastFunc']; // Funktion auslesen
  GridLoad(v);                     // und das Grid wieder entsprechend laden
end;

procedure FormInformationStoreWorkstate(storage: TStrings);
begin
  storage.Values['lastFunc'] := lastFunc; // letzte Funktion merken
end;
```

4.3.2 BisEditLinked

Einzeiliges Ein-/Ausgabefeld für Text-Attribute von Datenbank-Objekten. Nicht Text-Attribute werden nur dargestellt und können nicht bearbeitet werden.

4.3.3 BisLabelLinked

Label für die Anzeige von Attributen von Datenbank-Objekten, bzw. Bezeichnung und Einheit eines Attributes.

4.3.4 BisMemoLinked

Mehrzeiliges Ein-/Ausgabefeld für Text-Attribute von Datenbank-Objekten.

Spezifische Eigenschaften

ScrollBars	Typ: Aufzählung Die Eigenschaft *ScrollBars* bestimmt, ob das Memofeld mit Bildlaufleisten versehen wird. Wert: ssNone Keine Bildlaufleiste

ssHorizontal Horizontale Bildlaufleiste

ssVertical Vertikale Bildlaufleiste

ssBoth Horizontale und vertikale Bildlaufleiste

WantReturns Typ: Boolean

Die Eigenschaft *WantReturns* bestimmt, ob der Benutzer mit der Taste EINGABE einen Zeilenumbruch in den Text einfügen kann.

4.3.5 BisDreamMemoLinked

4.3.6 BisComboboxLinked

Ein-/Ausgabefeld für Attribute vom Typ Aufzählung für Datenbank-Objekten.

Wichtig: Das Feld *UseFormatFromDb* in der Eigenschaft *DataLink* muss gesetzt sein.

Spezifische Eigenschaften

EnumValue Typ: Integer (Variant)
Die Eigenschaft *EnumValue* enthält im Falle einer Aufzählung den Wert des Attributes in der Datenbank (*ItemIndex* enthält den Index in der Liste der Aufzählungswerte).

Style Typ: Aufzählung

Mit der Eigenschaft *Style* können Sie festlegen, wie das Kombinationsfeld angezeigt wird.

Erlaubte Werte:

csDropDown Erzeugt eine Dropdown-Liste mit einem Eingabefeld für manuell einzugebenden Text

csSimple Erzeugt ein Eingabefeld und eine Liste (ein Listenfeld), die unter dem Feld angezeigt wird. Die Länge der Liste wird durch die Eigenschaft Height des Kombinationsfeldes bestimmt.

csDropDownList Erzeugt eine Dropdown-Liste ohne Eingabefeld. Der Benutzer kann manuell keinen Text eingeben. Alle Einträge sind Strings, die dieselbe Höhe haben.

4.3.7 BisRadioGroupLinked

Ein-/Ausgabefeld für Attribute vom Typ Aufzählung.

Spezifische Eigenschaften

EnumValue Typ: Integer (Variant)

Die Eigenschaft *EnumValue* enthält im Falle einer Aufzählung den Wert des Attributes in der Datenbank (*ItemIndex* enthält den Index in der Liste der Aufzählungswerte).

4.3.8 BisCheckboxLinked

Ein-/Ausgabefeld für Attribute vom Typ Boolean (Ja/Nein) für Datenbank-Objekten.

Spezifische Eigenschaften

Caption	Typ: Text
	Beschriftung die hinter dem Kästchen angezeigt wird

4.3.9 BisIntEditLinked

Ein-/Ausgabefeld für Attribute vom Typ ganze Zahl für Datenbank-Objekten.

Spezifische Eigenschaften

Max

Min

4.3.10 BisColorButtonLinked

4.3.11 BisFloatEditLinked

Ein-/Ausgabefeld für Attribute vom Typ Gleitkomma-Zahl für Datenbank-Objekten.

Spezifische Eigenschaften

Digits	Typ: ganze Zahl
	Anzahl darzustellende Nachkommastellen
FormatType	Typ: Aufzählung
	Format für die Ausgabe.

Max

Min

4.3.12 BisTimeEditLinked

Ein-/Ausgabefeld für Attribute vom Typ Zeit für Datenbank-Objekten. Je nach Einstellung wird nur der Tag-Anteil oder nur der Zeit-Anteil oder beide.

Spezifische Eigenschaften

DateOnly	Typ: Boolean
	Es wird nur der Zeitanteil in der Datenbank gesetzt.
TimeOnly	Typ: Boolean
	Es wird nur der Datumanteil in der Datenbank gesetzt.

4.3.13 BisDateEditBtnLinked

Ein-/Ausgabefeld für Attribute vom Typ Zeit für Datenbank-Objekten. Das Feld enthält zusätzlich einen Knopf mit einem Popup-Kalender zur einfachen Eingabe von Kalendertagen.

Spezifische Eigenschaften

DateOnly	Typ: Boolean
	Es wird nur der Zeitanteil in der Datenbank gesetzt.
ShowButton	Typ: Boolean
	Der Knopf für den Popup-Kalender wird gezeigt.
TimeOnly	Typ: Boolean

Es wird nur der Datumanteil in der Datenbank gesetzt.

4.3.14 BisIntervalEditLinked

4.3.15 BisIntervalEditExLinked

Ein-/Ausgabefeld für Attribute vom Typ Intervall für Datenbank-Objekten.

Neuere bessere Ausführung mit zwei Teilen. Im ersten Teil kann der Wert eingegeben werden, im zweiten Teil die Auflösung ausgewählt werden.

Spezifische Eigenschaften

SeparatorSize	Typ: ganze Zahl
	Breite des Teilers zwischen den beiden Eingabefeldern.
ValueFormat	Typ: Text
	Format für den Value-Teil
ValueMax	Typ: Gleitpunktzahl
	Maximaler Wert
ValueMin	Typ: Gleitpunktzahl
	Minimaler Wert
ValueSize	Typ: ganze Zahl
	Breite des Value-Teils in Pixels

4.3.16 BisTerminEditLinked

4.3.17 BisTerminControl

4.3.18 BisLookupCombobox

Lookup-Comboboxen (*BisLookupCombobox*) verwenden wir dann, wenn wir die Beziehung eines Datenbankobjekt zu einem anderen Datenbankobjekt im Formular manipulieren können wollen.

Im Gegensatz zur Komponente *BisDynSimpleGridII*, bei der wir mit Drag & Drop mehrere Datenbankobjekte der Beziehung zuweisen können, arbeitet die *BisLookupCombobox* ähnlich wie eine Combobox, in der ich das Datenbankobjekt wählen kann, das in die Beziehung eingefügt werden soll.

Eine *BisLookupCombobox* erfordert immer eine zugeordnete *BisDataSet*-Komponente, die alle möglichen Datenbankobjekte bestimmt, die an der Beziehung teilnehmen können.

Spezifische Eigenschaften

AllowDrop	Typ: Boolean
	Wenn gesetzt, dann können Objekte auch mit Drag & Drop der Komponente zugewiesen werden.
AutoEnter	Typ: Boolean
	Wenn die Eigenschaft *AutoEnter* bestimmt, ob wenn die Eingabe eindeutig ein Datenbankobjekt identifiziert, die Eingabe komplettiert wird.
AutoLookupDelay	Typ: ganze Zahl (Millisekunden)

	Wartezeit bis zur automatischen Komplettierung von Zeilen.
AutoSelect	Typ: Boolean Die Eigenschaft *AutoSelect* bestimmt, ob der gesamte Text im Eingabefeld automatisch markiert wird, wenn das Steuerelement den Fokus erhält.
DataSet	Typ: Auswahl Beziehung zu einer Komponente vom Typ *BisDataSet*, welche alle möglichen Datenbankobjekte für die Auswahl liefert.
MinCharsToLookup	Typ: ganze Zahl Minimale Anzahl Zeichen, bis es möglich ist, das gelbe Dreieck zu klicken um die Liste aller möglichen Datenbankobjekte zu bekommen.

Options

lcoAllowDblClickOpen

lcoAllowDrag

lcoAllowOpen

lcoAllowProperties

lcoAllowSearchDocument

lcoNoItemsSort

Durch Setzen dieser Option wird die Ergebnismenge des zugrundeliegenden DataSets nicht weiter sortiert. Die Sortierung sollte in diesem Fall durch das DataSet vorgenommen werden.

lcoShowImage

lcoNoCompareAttribute

Durch Setzen dieser Option wird der implizite Vergleich mit dem CDLAttribute im DataLink, bzw. mit dem SearchViewCdl beim Ausführen des Lookups unterbunden. Dh. das DataSet (FilterNavigation) an der LookupCombobox bestimmt die Menge der gefundenen Objekte.
Auf diese Weise kann zum Beispiel eine Person über Personalnummer, Vorname oder Nachname oder klassisch gesucht werden (man beachte die $$-Notation!):

```
START INSTANCES Person
[
  VALUE bisB_PersonalNumber ~= $$
    OR
  VALUE bisB_FirstName ~= $$
    OR
  VALUE bisB_LastName ~= $$
    OR
  VALUE Object_Display_Kontext ~= $$ ]
```

Oder, man kann einfach ein häufig vorkommendes Prefix, automatisch davorstellen. Hier wäre das das Prefix "HB-" für eine Schweizer Flugzeug-Immatrikulation:

```
  START INSTANCES bGMT_Aircraft
  SAVEAS vPattern $$ [
    VALUE bGMT_Immatriculation ~= '="HB-" + vPattern'  OR
    VALUE bGMT_Immatriculation ~= DATAOF vPattern      OR
```

```
  VALUE bGMT_AircraftType ~= DATAOF vPattern          OR
  VALUE Object_Display_Kontext ~= DATAOF vPattern ]
```

Ab Version 4.10.1 gibt es eine zweite Variante in der Filternavigation des DataSets die Eingabe der LookupBox zu verwenden: den Parameter **VAL_Input**.

```
START INSTANCES bGMT_Aircraft
[
  VALUE bGMT_Immatriculation ~= '="HB-" + VAL_Input'  OR
  VALUE bGMT_Immatriculation ~= DATAOF VAL_Input      OR
  VALUE bGMT_AircraftType ~= DATAOF VAL_Input         OR
  VALUE Object_Display_Kontext ~= DATAOF VAL_Input ]
```

lcoLookupOnExit

Wird eine BisLookupCombobox "verlassen" ohne dass zuvor ein Lookup (durch Eingabe von Return, bzw. Klicken auf den Pfeil) ausgeführt worden ist, wird die LookupCombobox ungültig, dh. rot. Durch Setzen der Option lcoLookupOnExit kann eine BisLookupCombobox z. Bsp. mit <Tab> verlassen werden. Beim Exit wird nun ein "vereinfachtes" Lookup ausgeführt und bei genau einem Treffer dieser übernommen.

4.3.19 BisCrossTable

Stellt die Beziehung zweier Objektmengen über eine Assoziation in einer Kreuztabelle dar. Die Objektmengen sowie die Assoziation werden in TBisCrossTableDataLink definiert.

Spezifische Eigenschaften

Columns	Typ: Liste von Spalten Beschreibt die unteren linken Spalte in der Kreuztabelle. (Attribut und Assoziation von BisDataSetRow)
CrossTableOptions	Typ: Menge Beschreibt die Optionen der Kreuztabelle
CrossTableRows	Typ: Liste von Zeilen Beschreibt die oberen rechten Zeilen in der Kreuztabelle. (Attribut und Assoziation von BisDataSetCol)
DataLink	Typ: TBisCrossTableDataLink …
FontVertical	Typ: TFont Beschreibt die Schriftformatierung der CrossTableRows.
SortedCols, SortedRows	Typ: Integer Beschreibt nach welcher Spalten-, Zeilen-Nummer sortiert werden soll.

4.3.20 TBisCrossTableDataLink

Gibt zwei Objektmengen sowie eine Assoziation an.

Spezifische Eigenschaften

AssociationColumnToRow	Typ: String Gibt die darzustellende Assoziation an.

DataSetColumn	Typ: BisDataSet Gibt die Objektmenge der Spalten an.
DataSetRow	Typ: BisDataSet Gibt die Objektmenge der Zeilen an.
EvalRowToColumn	Typ: Boolean Wenn gesetzt, dann wird nicht die Assoziation *AssociationColumn-ToRow* ausgewertet, sondern die Partnerassoziation (RowToColumn). Muss verwendet werden, wenn einseitige Assoziationen bearbeitet werden (>= v5.0.3)

4.3.21 BisObjectCrosstable

Stellt die Beziehung dreier Objektmengen über zwei Assoziation in einer Kreuztabelle dar, sowie die Werte aller drei Objektmengen. Die Objektmengen sowie die Assoziationen werden in TBisObjectCrossTableDataLink definiert.

Spezifische Eigenschaften

Columns	Typ: Liste von Spalten Beschreibt die unteren linken Spalte in der Kreuztabelle. (Attribut und Assoziation von BisDataSetRow)
CrossTableOptions	Typ: Menge Beschreibt die Optionen der Kreuztabelle
CrossTableRows	Typ: Liste von Zeilen Beschreibt die oberen rechten Zeilen in der Kreuztabelle. (Attribut und Assoziation von BisDataSetCol)
DataLink	Typ: TBisObjectCrossTableDataLink …
FontVertical	Typ: TFont Beschreibt die Schriftformatierung der CrossTableRows.
SortedCols, SortedRows	Typ: Integer Beschreibt nach welcher Spalten-, Zeilen-Nummer sortiert werden soll.

4.3.22 TBisObjectCrossTableDataLink

Gibt die Objektmengen sowie eine Assoziation an.

Spezifische Eigenschaften

Association	Typ: String Gibt die Assoziation zwischen der Referenzierte-Objektemenge mit der Zellen-Objektmenge an. Welches die Referenzierte-Objektemenge ist wird bei RowIsPatent definiert.
AssociationColumn	Typ: String (ab v.4.10.8) Gibt die Assoziation zwischen der Objektemenge der Spalten mit der Zellen-Objektmenge an.
AssociationRow	Typ: String (ab v.4.10.8) Gibt die Assoziation zwischen der Objektemenge der Zeilen mit der Zellen-Objektmenge an.

CDLAttribute	Typ: String Gibt die darzustellende Attribut an. Wenn das Attribut nicht gesetzt wird, wird mit Häkchen angezeigt wenn Zellen-Objekte vorhanden sind.
DataSetColumn	Typ: BisDataSet Gibt die Objektmenge der Spalten an.
DataSetRow	Typ: BisDataSet Gibt die Objektmenge der Zeilen an.
DataSetParent	Typ: BisDataSet (ab v.4.10.8) Gibt den Container der Zellen-Objekte an.
ObjectType	Type: String Gibt die Klasse der Zellen-Objekte an. Wird benötigt wenn neue Objekte erzeugt werden müssen.
RowIsParent	Type: Boolean Gibt an welche Objektmenge die Referenzierte-Objektemenge ist. Ist "RowIsParent" nicht gesetzt so ist "DataSetColumn" die Referenzierte-Objektemenge sonst ist es "DataSetRow".

Anmerkung:

Wenn die Zellen-Objekte ein Zeilen- oder Spaltenobjekt als Container (Parent) haben sollen, dann müssen folgende Eigenschaften verwendet werden:

- Association
- RowIsParent

Wenn die Zellen-Objekte ein Zeilen- oder Spaltenobjekt nicht als Container (Parent) haben sollen, dann müssen folgende Eigenschaften verwendet werden:

- AssociationColumn
- AssociationRow
- DataSetParent

4.3.23 BisImageLinked

4.3.24 BisDynOpenDialog

4.3.25 BisDynSimpleGridII

Eine Liste mit Datenbank-Objekten mit ihren Attributen als Spalten. Dient der Darstellung und Manipulation von Beziehungen und der Veränderung von Attributen Datenbankobjekte in der Liste.

Spezifische Eigenschaften

Columns	Typ: Liste von Text Definition der Spalten, zur genauen Beschreibung siehe weiter unten
Propagator	Typ: Auswahl Bei jedem Wechsel der Selektion wird die angewählte *BisObjectPropagator* benachrichtigt und kann damit eine abhängige *BisDataSource* setzen.
MaxAutoFilterItems	Typ: Integer (ab v4.10.3)

Definiert die maximale Anzahl der AutoFilter-Werte in der AutoFilter-Combobox.
Kann mit dem globalen Parameter P_MaxAutoFilterItems für alle Grids (auch in Toolkonfiguration) eingestellt werden.
Default = 1000

GridHierarchyBuilder .SubContentsNav Typ: String (ab v.4.10.8)

Definiert eine FilterNavigation zur Anzeige von Untergeordneten Objekten im Grid. Funktioniert analog zum Tree.

.SubContentsExistsNav Typ: String (ab v.5.4.9)

Definiert eine FilterNavigation welche angibt, ob es Untergeordnete Objekte gibt (die FilterNavigation gibt in diesem Fall mehr als 0 Objekte zurück). Ist die Eigenschaft leer, wird SubContentsNav verwendet. Funktioniert analog zum Tree.

SimpleGridOptions Typ: Menge – die einzelnen Optionen sind wie folgt

sgoAllowCopy	Kopieren erlaubt
sgoAllowDelete	Löschen von Objekten erlaubt (Löschen erscheint im Kontextmenü)
sgoAllowDrag	Ziehen mit der Maus erlaubt
sgoAllowDrop	Ablegen von Objekten erlaubt (sofern in der Assoziation erlaubt)
sgoAllowDuplicateRows >= v10.2.2	Mehrere Zeilen zu einem Objekt können geladen werden.
sgoAllowEmptyRows	Das Einfügen leerer Zeilen ist erlaubt. Für leere Zeilen werden beim Speichern Objekte erzeugt (Objektklasse ist konfiguriert unter…)
sgoAllowOpen	Das Öffnen von Objekten ist erlaubt (erscheint im Kontextmenü). Öffnen funktioniert auch über Doppelklick
sgoAllowOpenDocuments	Über die Assoziation `Doc_Documents` verknüpfte Dokumente werden im Kontextmenü im Untermenü *Dokumente* angezeigt und können geöffnet werden.
sgoAllowPaste	Einfügen erlaubt
sgoAllowProperties	Das Anzeigen des Eigenschaftsblatts ist erlaubt (erscheint im Kontextmenü)
sgoAllowRemove	Entfernen von Objekten aus der Assoziation / dem Grid erlaubt (Entferen erscheint im Kontextmenü
sgoAllowSearchDocument	obsolet
sgoBroaodcastSelection	Die gewählten Objekte werden im gesamten Byron/BIS verbreitet (broadcastet)
sgoCopyObjects	Ein hinzugefügtes Zeilenobjekt wird beim Speichern unter das Objekt des Grids kopiert.
sgoSelectByBroadcast	Objekte im Grid werden über Broadcast selektiert

sgoSelectFirstRowAfterLoad	Die erste Zeile wird nach dem Laden selektiert, der Propagator aufgerufen.
sgoStrictReadOnly >= v4.10.2	Ist die DataSource ReadOnly, dann werden im Normalfall die Operationen gesperrt, mit denen die assoziation verändert werden kann (Drop, Remove, Delete, Paste, EmptRows). Die Eigenschaften der Objekte im Grid können aber noch bearbeitet werden (das Grid ist nicht ReasOnly/grau). Ist diese Option gesetzt, dann wird das Grid ReadOnly, sofern die DataSource ReadOnly ist.

TrackScroll

Typ: Boolean (ab v4.11.8), Vorgabe: gesetzt

Steuert das Verhalten des Grids, wenn die Bildlaufleiste mit der Maus gepackt und verschoben wird.
Gesetzt: der Inhalt des Grids wird während des Verschiebens neu gezeichnet.
Nicht gesetzt: der Inhalt wird erst beim Loslassen des neu gezeichnet.
Wichtiger Nebeneffekt: ist TrackScroll nicht gesetzt, dann wird der Inhalt der Zellen des Grids erst bei Bedarf (Anzeige, Sortieren, Filter, …) nachgeladen. Dies sollte zu einer Verbesserung der Lade-Performance führen. Der Effekt wird durch die Verwendung von mehrzeiligen Spalten zunichte gemacht.

Callbacks:

OnLookupActionButtonCanShow(Sender: TObject; rowId, colId: String)

Typ: Boolean

Gibt zurück, ob in einer Lookupcombobox im Grid, neben dem Lupensymbol auch der ActionButton angezeigt werden soll. Mit Hilfe von `rowId` und `colId` kann die Anzeige auf Objekt- und Spaltenbasis ein- bzw. ausgeschaltet werden.

OnLookupActionButtonClicked(Sender: TObject; rowId, colId: String)

Wird in OnLookupActionButtonCanShow `true` zurückgeliefert, kann hier auf den Klick des LookupActionButtons reagiert werden. Typischerweise wird hier ein Formular mittels ShowModalEx (mit Parameterliste) aufgerufen.

OnCellClick(Sender: TObject; colId, rowId: String; obj: TBisObjectProperty; var clickHandled: Boolean)

(ab v4.10.5) Wird aufgerufen, wenn mit der Maus auf eine Zelle geklickt wird. Wechselt die Zelle in den Editiermodus, wird der Callback nicht aufgerufen.
Wenn DisplayAsHyperlink gesetzt ist und der Hyperlink-Aufruf verhindert werden soll, muss `clickHandled` auf `true` gesetzt werden. Der Standardwert von `clickHandled` ist `false`.
Mit Hilfe der Parameter `colId` und `rowId` kann auf die Zelle zugegriffen werden. Der Parameter `obj` enthält das Datenobjekt der Zeile.

OnCellIsEditable(Sender: TObject; colId, rowId: String; obj: TBisObjectProperty; var isEditable: Boolean)

(ab v4.10.5) Wird aufgerufen, wenn die Zelle editierbar ist und bevor die Zelle in den Editiermodus wechseln will. Der Callback wird ohne Transaktion aufgerufen. Wenn verhindert werden soll, dass die Zelle editierbar ist, muss `isEditable` auf `false` gesetzt werden. Der Standardwert von `isEditable` ist `true`.
Mit Hilfe der Parameter `colId` und `rowId` kann auf die Zelle zugegriffen werden. Der Parameter `obj` enthält das Datenobjekt der Zeile.

OnRowAddedEx(Sender: TObject; rowId: String; obj: TBisObjectProperty)

(ab v5.0.1) Wird aufgerufen, wenn dem Grid eine Zeile (Row) hinzugefügt wird..
Mit Hilfe dem Parameter `rowId` kann auf die Zeile, bzw. eine Zelle dieser zugegriffen werden. Der Parameter `obj` enthält das Datenobjekt der Zeile.
Anwendung: Anreichern der Zeile um html-Elemente wie Löschen-„Button", o.ä.

4.3.26 BisDynSimpleGridColumn

Beschreibt eine Spalte im Gitter.

Spezifische Eigenschaften

DisableImage — Typ: Boolean

Unterdrückt ein allfälliges Icon.

HasAutoFilter — Typ: Boolean

Schaltet die AutoFilter-Funktion für diese Spalte ein.

LookupDataSet — Typ: TBisDataSet

Zuordnung eines DataSets für die LoopupCombobox, welche für die Auswahl des referenzierten Objekts in der Spalte verwendet wird. Es muss eine Assoziation in der Eigenschaft Role angegeben werden und die Eigenschaft ReadOnly muss deaktiviert sein.

LookupNoCompareAttribute — Typ: Boolean

Wenn gesetzt, wird bei der Suche über das LookupDataSet das Attribut (View) der Spalte nicht verwendet. Erst ab Byron/BIS v4.11.8.

Options — Typ: Menge

Beschreibt die Optionen einer Spalte im Gitter.

Werte:

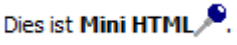
gpoNoAutoAdjustSize
Diese Option verhindert die automatische Berechnung der Spaltenbreite.

gpoNoUserAdjustSize
Diese Option verhindert das Verändern der Breite durch den Benutzer.

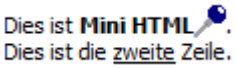
	gpoNoZeroSize Diese Option verhindert, dass die Spaltenbreite auf Null gesetzt werden kann. *gpoNotLessThanInitialSize* Diese Option verhindert, dass die Spaltenbreite kleiner als die Anfangsbreite sein kann.
Path	Typ: String Gibt das darzustellende Attribut bzw. die Assoziation an. Assoziation und Attribut werden über das Prozent-Zeichen "%" getrennt. Es sind mehr als eine Assoziation erlaubt. Hinweis: Löscht beim Setzen die Eigenschaften Role und View!
Role	Typ: String Gibt die darzustellende Assoziation an. Falls View gesetzt ist, wird das Attribut dargestellt. Hinweis: Löscht beim Setzen die Eigenschaft Path!
View	Typ: String Gibt das darzustellende Attribut an. Falls Role gesetzt ist, wird zuerst zur Assoziation navigiert und anschliessend das Attribut verwendet. Hinweis: Löscht beim Setzen die Eigenschaft Path!
UseAccessRights	Typ: Boolean Verwendet die in der Benutzerverwaltung definierten Zugriffsrechte auf Attribute und Assoziationen
DisplayAsHyperlink	Typ: Boolean (ab v4.10.6) Gibt an, ob der Inhalt der Zelle als Hyperlink dargestellt werden soll. Dies gilt nur für die Datentypen Text, Aufzählung, ganze Zahl, Gleitkommazahl und Zeit. Zusätzlich wird der Hand-Mauszeiger angezeigt, sobald sich die Maus innerhalb einer Hyperlink-Zelle befindet. Nach dem Klicken auf die Zelle wird der Hyperlink aufgerufen. Wenn auf den Mausklick reagiert werden soll, kann das Event `OnCellClick` im Formularscripting verwendet werden. Beispiel 1: www.byron.ch Beispiel 2: mailto:support@byron.ch Beispiel 3: http://www.byron.ch/downloads/Dokumente/ByronBIS_FormularReferenz.pdf Beispiel 4: L:\Dokumentation\Technische Dokumentation\ByronBIS_FormularReferenz.docx
RenderHTML	Typ: Boolean (ab v4.10.6) Gibt an, ob der Inhalt der Zelle als HTML dargestellt werden soll. Dies gilt nur für den Datentyp Text. Sind DisplayAsHyperlink und RenderHTML gleichzeitig gesetzt, wird zwar der Hand-Mauszeiger angezeigt, aber der Text als HTML dargestellt. Mini-HTML-Referenz: http://www.tmssoftware.com/site/minihtml.asp Byron/BIS-Erweiterung: Für das Image-Element `<IMG src="specifier:name">` wird ein neuer `specifier` „bis“ unterstützt, welcher das kleine Icon (small image) aus `name` verwendet. Beispiel für Mini-HTML:

```
Dies ist <B>Mini HTML</B><IMG src="bis:bisKL_Key">.<BR>Dies
ist die <U>zweite</U> Zeile.
```

Anzeige im Gird ohne MultiLine:



Anzeige im Grid mit MultiLine:



Wenn auf den Mausklick reagiert werden soll, kann das Event `OnCell-Click` im Formularscripting verwendet werden.

4.3.27 BisDynSimpleTree

4.3.28 BisPopupTree

4.3.29 BisDynGanttLinked

4.3.30 BisDynWebBrowser

In diesem WEB Browser können WEB Dokumente angezeigt werden. Die Komponente basiert auf I-WebBrowser2 und ist hier MSDN-webbrowser2 im Detail beschrieben.

Mode	steuert was im Browser dargestellt werden soll: wbmAutomatic: Es wird versucht den richtigen Modus zu ermitteln. wbmFixUrl: Es wird die Seite welche man im 'URL' angegeben hat angezeigt, d.h. es erfolgt kein Datenbank Zugriff. wbmDBUrl: Der Wert des DataLinks wird als URL eingesetzt. Vgl. Bemerkung bei URL wbmDBcontents: Der Wert des DataLinks wird als Dokument Inhalt eingesetzt. vgl. MIMEType
MIMEType	Default = 'xhtml' Diese Einstellung wird bei der Methode OpenDoc verwendet, welche bei Mode wbmDBcontents verwendet wird. Mit diesem Default und Mode=wbmDBcontents muss also das Attribut ein vollständiges html Dokument sein d.h. typischerweise vom Typ 'mehrezeiliger Text'.
DataLink	CDLAttribute: Bestimmt das Attribut, welches verwendet wird (Mode = wbmDB...). Navigation: Bestimmt das Objekt mittels FilterNavigation, ausgehend vom Objekt der *BisDataSource*.
URL	Enthält den URL der angezeigten Seite. In doppelten Anführungszeichen können globale Parameter und Umgebungsvariablen verwendet werden. Bsp: www. "Firma".ch Wenn es nun einen globalen Parameter mit Name Firma = byron gibt, dann kommt das gut.
FixContents	Kann auf den Inhalt des Dokumentes zugegriffen werden, bzw. zur Designzeit ein Dokument definiert werden. Wie bei wbmDBcontents wird dann dieses Dokument dargestellt (vgl. MIMEType).

	Beispiel: URL = '' Mode= wbmAutomatic; DataLink = leer; MIMEType = 'xhtml'; FixContents=vollständiger Text eines HTML Dokumentes.
BISOnNavigateError	Tritt auf, wenn eine Webseite einen Fehler zurückgibt (z.B. 404 wenn die Seite nicht gefunden wurde). Der Fehlercode kann im Parameter StatusCode abgefragt werden. **ACHTUNG: BIS**OnNavigateError ist zu verwenden nicht OnNavigateError!
BISOnBeforeNavigate	Tritt auf, bevor die Webseite zu einem Link weiternavigiert (z.B. mit einem <a href="#myLink"). Mit dem VAR-Prameter „Cancel" kann die Weiterleitung abgebrochen werden. Mit diesem Ereignis ist es möglich, Links auf einer Webseite für die Navigation zu anderen Applikationselementen umzuleiten. **ACHTUNG: BIS**OnBeforeNavigate ist zu verwenden nicht OnBeforeNavigate2!
OnDownloadBegin	
OnDownloadComplete	Treten auf, wenn eine Seite anfängt zu laden bzw. das Laden beendet wird.
BISOnDocumentComplete	Tritt auf, wenn das HTML-Dokument fertig geladen ist. Ab diesem Zeitpunkt können z.B. JavaScripts ausgeführt werden. **ACHTUNG: BIS**OnDocumentComplete ist zu verwenden nicht OnDocumentComplete!
OnProgressChange	Eine Internet Explorer Fortschrittsinformation. Kann im Zusammenspiel mit einer Progressbar verwendet werden.

4.3.31 BisDataSet

Unsichtbare Komponente, welche Navigationsregeln und Filter enthält. Wird von *BisLookupComboboxLinked* verwendet

Spezifische Eigenschaften

DataSource	Typ: Auswahl einer *BisDataSource* Falls die Eigenschaft *DataSource* gesetzt ist, liefert die assoziierte *DataSource* das Ausgangsobjekt für die Navigation
Filters	Typ: mehrzeiliger Text (StringList) Die Eigenschaft *Filters* enthält eine Liste von Namen von Objektklassen (pro Zeile eine). Wenn die Liste nicht leer ist, werden nur Datenbankobjekte deren Oberklasse in der Liste ist zurückgegeben.
NavRules	Typ: mehrzeiliger Text (StringList) Ein Liste von Navigationsregeln (pro Zeile eine). Format: <von Objektklasse>, <nach Objektklasse >, <Assoziation> Wenn: <von Objektklasse> und <Assoziation> leer sind, wird das Wurzelobjekt von <nach Objektklasse> gesucht.
UseIndex	Typ: Boolean Keine Wirkung

FilterNavigation

Navigation zu den Objekten. Formalismus gem. separater Dokumentation[1]. Parameter können im Event OnSetVariables zugewiesen und in der FilterNavigation mit dem definierten Variablenamen verwendet werden.

Beispiel:
(OnSetVariables-Event)

```
procedure dSetDeliveryOrder(Sender: TObject);
begin
dSetDeliveryOrder.SetVariable('ORDERTYPE', cbxType.ItemIn-
dex);
end;
```

(FilterNavigation im DataSet)

```
START INSTANCES bisKL_DeliveryOrder
VALUE bisKL_DeliveryOrderType = DATAOF ORDERTYPE
```

Verwendung mit der LookupBox

DataSets werden hauptsächlich für die Objektsuche mit der LookupBox verwendet. In diesem Fall steht für die Objektsuche der Parameter `$$` und ab Version 4.10.1 der Parameter `VAL_Input` zur Verfügung. Beide Parameter enthalten den in der LookupBox eingegebenen Text (inklusive eines ‚*').

N.B. Wenn Wertevergleiche in der Filternavigation des DataSets durchgeführt werden, dann muss die LookupBox daran gehindert werden, das Ergebnis noch einmal zu filtern. Verwenden Sie hierzu die Option **IcoNoCompareAttribute** der LookupBox.

Beispiele einer Filternavigation:

```
[
  START VALUE bisB_LastName ~= $$
    OR
  START INSTANCES Person VALUE Object_Display_Kontext ~= $$
]
```

-- ab Version 4.10.1

```
[
  START VALUE bisB_LastName ~= DATAOF VAL_Input
    OR
  START INSTANCES Person VALUE Object_Display_Kontext ~= DATAOF VAL_Input
]
```

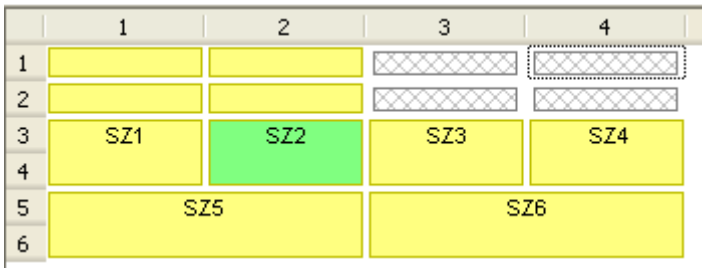
```
START VALUE bisB_LastName ~= "'*' + VAL_Input"
```

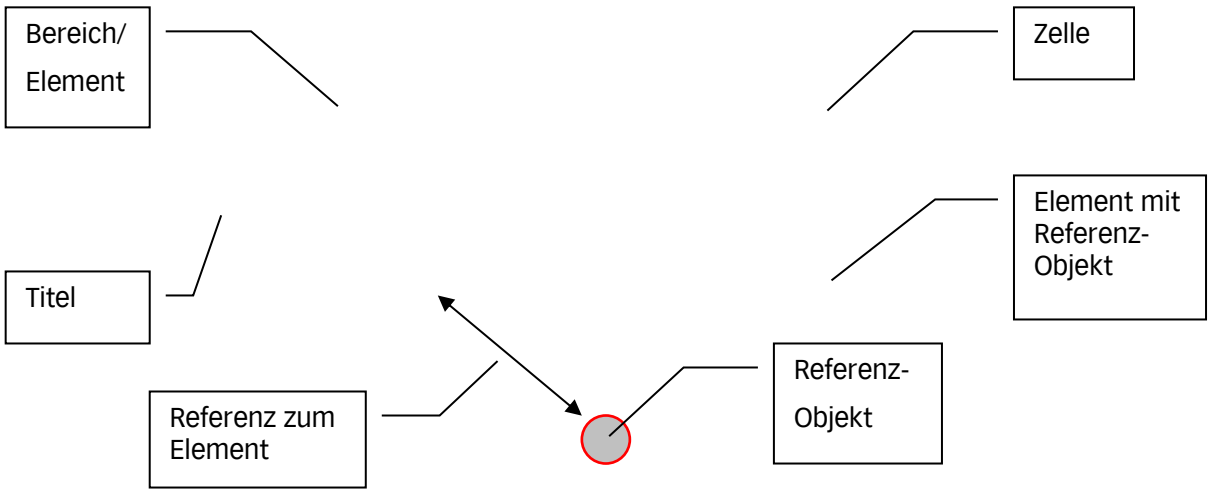
4.3.32 BisDynamicPropertyList

4.3.33 BisCellGrid

In diesem Grid, das aus Zellen besteht, können graphische Bereiche (Elemente) angezeigt werden. Diese Bereiche können sich über mehrere Zellen (vertikal sowie horizontal) erstrecken. Für jeden Bereich wird ein Element-Objekt erzeugt. Diesem Element kann ein Referenz-Objekt per mit Drag & Drop zugeordnet werden.

[1] www.byron.ch/downloads/dokumente/FilterNavigationDescription.pdf





DataLink	Typ: TBisDataLink CDLAttribute: Wird nicht verwendet CDLReferenz: Wird nicht verwendet DataSource: Verbindung zur *BisDataSource*
ColCountAttr	Typ: Text Attributname um den Wert für ColCount auszulesen
RowCountAttr	Typ: Text Attributname um den Wert für RowCount auszulesen
ColCount	Typ: ganze Zahl Anzahl der Zellen horizontal (Spalten)
RowCount	Typ: ganze Zahl Anzahl der Zellen vertikal (Zeilen)
CellBrush	Typ: TBrush Farbe und Style der Zell-Brush
CellPen	Typ: TPen Farbe, Mode, Style und Breite des Zell-Pen

RegionBrush	Typ: TBrush Farbe und Style der Bereichs-Brush
RegionPen	Typ: TPen Farbe, Mode, Style und Breite des Bereichs-Pen
ElementClassName	Typ: Text Element-Klassenname zum Erzeugen von Elementen (als Object_To_Children)
ElementAttrDisplay	Typ: Text Attributname zur Anzeige in einem Bereich
ElementAttrTop	Typ: Text Attributname für den Wert Top von Element
ElementAttrLeft	Typ: Text Attributname für den Wert Left von Element
ElementAttrWidth	Typ: Text Attributname für den Wert Width von Element
ElementAttrHeight	Typ: Text Attributname für den Wert Height von Element
DropAssoziation	Typ: Text Referenz-Beziehung zum Erstellen von Element-Referenz-Zuordnungen
AllowOverrideDrop	Typ: TByTriState triFalse: Kein Überschreiben von Element-Referenz-Zuordnung erlaubt triTrue: Überschreiben von Element-Referenz-Zuordnung erlaubt (ohne Nachfrage) triUndef: Überschreiben von Element-Referenz-Zuordnung erlaubt (mit Nachfrage)
OverrideDropMessage	Typ: Text Message-String für die Abfrage beim Überschreiben einer Element-Referenz-Zuordnung. Wird nur verwendet, wenn AllowOverrideDrop = triUndef
DropCondition	Hier kann eine Filternavigation angegeben werden. Als Input wird das gezogene Objekt übergeben. Es muss mindestens ein Objekt resultieren, damit die Drop Operation akzeptiert wird.
ElementEditTop	Typ: TByBaseEdit Verbindung zu einem TByBaseEdit für den Wert Top eines Elements
ElementEditLeft	Typ: TByBaseEdit Verbindung zu einem TByBaseEdit für den Wert Left eines Elements
ElementEditWidth	Typ: TByBaseEdit Verbindung zu einem TByBaseEdit für den Wert Width eines Elements
ElementEditHeight	Typ: TByBaseEdit Verbindung zu einem TByBaseEdit für den Wert Height eines Elements
ColWidth	Typ: ganze Zahl Zellenbreite (Spaltenbreite)
RegionSelected	Typ: Boolean (ReadOnly) Liefert TRUE, wenn ein Bereich selektiert ist, sonst FALSE.

HasAssoziation	Typ: Boolean (ReadOnly) Liefert TRUE, wenn eine Element-Referenz-Zuordnung existiert, sonst FALSE.
CellGridOptions	Typ: Aufzählung CellGrid-Optionen (Beschreibungen siehe weiter unten bei SetCellGridOption)

Methoden für das Scripting

```
procedure SetCellGridOption(const cgOption: TCellGridOption; const Value: Boolean);
  // cgOption ist eine der folgenden CellGrid-Optionen:
  //   cgoShowTitle                Titel anzeigen
  //   cgoDisplayAssoziation       FALSE = ElementAttrDisplay ist ein Attribut von
  //                                       Element
  //                               TRUE  = ElementAttrDisplay ist ein Attribut von
  //                                       Referenz
  //   cgoAllowOpenElement         Element bei Doppelklick öffnen, falls keine
  //                               Referenz vorhanden
  //   cgoAllowOpenAssoziation     Referenz bei Doppelklick öffnen
  //   cgoBroadcastElement         Broadcast bei SelectionChanged für Element
  //   cgoBroadcastAssoziation     Broadcast bei SelectionChanged für Referenz
  //   cgoAllowProperties          PopUp Menu ‚Eigenschaften'
  //   cgoAllowDeleteAssocElement  PopUp Menu ‚Verknüpfung löschen'
  //   cgoAllowDeleteElement       PopUp Menu ‚Element löschen'
  //   cgoSelectByBroadcast        externe Selektionen setzen die Selektion
  //                               vom CellGrid (ab v4.10.0)

  // Mit Value wird die Option auf an (TRUE) bzw. aus (FALSE) gesetzt
procedure CreateRegions;
  // Erzeugt automatisch Bereiche für die leeren Zellen
procedure CreateRegion;
  // Erzeugt einen neuen Bereich mit den Werten Top, Left, Width und Height aus den
  // Element-(TByBase)Edit's
procedure DeleteRegion;
  // Löscht den aktuellen Bereich
procedure DeleteAssoziation;
  // Löscht die aktuelle Element-Referenz-Zuordnung
```

4.3.34 BisDynChart (ab v3.11.0)

Obsolete, bitte die Komponente BisDynChartII verwenden!

Ist eine Diagramm Komponente auf der Basis von TChart. http://www.steema.net/teechart-docs/

```
TBisSerieTypeS =(bstLineSeries,
                 bstBarSeries,
                 bstHorizBarSeries,
                 bstAreaSeries,
                 bstPointSeries,
                 bstPieSeries,
                 bstFastLineSeries,
                 bst_byChart); // wird von Chart (BisChartType) übernommen

TBisSerieType =  bstLineSeries .. bstFastLineSeries;

TBisSerieFunctions = (sfuNone,
                      sfuSUM );

TBisChartOptions = set of (copMapToOneYear);
// mit copMapToOneYear werden alle X-Werte (müssen vom Typ Datum sein)
// auf dasselbe Jahr abgebildet. Mit dieser Option wird im Title der
// Serie der Text <YEAR> durch die Jahreszahl ersetzt.

TBisChart = class(TChart)
  property BisChartType: TBisSerieType
  property DataLink: TBisDataLinkChart
  property BisSerie: TBisSerieCollection -> TBisSerie
  property MarginPercentHor : Integer
  property MarginPercentVer : Integer
  property BisOptions : TBisChartOptions;
end

TBisSerie = class(TCollectionItem)
  property BisChartType: TBisSerieTypeS  //
  property DrawBetweenPoints : Boolean
  property Title       : String
  property MarkVisible : Boolean
  property PointerVisible : Boolean
  property Color  : TColor
  property ColorAutomatic  : Boolean
  property PointerStyle : TSeriesPointerStyle
  property ShowInLegend : Boolean
  property VertAxis     : TVertAxis
  property FunctionOperation :  TBisSerieFunctions
  property AttributeX : String
  property AttributeY : String
  property Navigation : TStrings
end
```

Prinzip:

Durch die `DataLink.Navigation` werden die Objekte bestimmt, welche je eine Serie (Datenreihe) ergeben. Wenn die Property `DataLink.Navigation` leer ist, wird das Objekt aus der DataSource verwendet. Für jedes Objekt aus `DataLink.Navigation` werden alle definierten Serien erstellt. Die Serien werden mit der Property `BisSerie` definiert.

BisSerie:

Mit `Navigation` werden aus dem Serien-Objekt die Objekte für die Datenpunkte ermittelt. In `AttributeX` und `AttributeY` werden die Attribute für die Werte (Koordinaten auf der X- und auf der Y-Achse) spezifiziert. Die AttributeX –Y können auch als Expression formuliert werden, dies wird durch ein ‚=' als erstes Zeichen formuliert. Z.B. "= - ATTRIBUTE(‚x_offset') + 3.5 / 150".
Dokumentation der Expressions

www.byron.ch/downloads/dokumente/ExpressionsForVariablesAndValues.pdf

Wenn FunctionOperation auf sfuSUM gesetzt wird, dann werden die Y-Werte jeweils kumuliert. In diesem Fall können AttributeX, -Y, Navigation weggelassen werden, in diesem Fall wird die ‚Vorgänger-Serie' verwendet.

Wenn die X-Werte Zeitwerte sind, dann setzen Sie im Chart:
`BottomAxis.ExactDateTime` auf true
und
`BottomAxis.Increment` auf 7 bei Woche, 30 für Monat etc.

Bemerkungen:

- Bei folgenden Properties können Ausdrücke in $-Konvention eingegeben werden (werden ausgewertet):
  `Chart.Title.Text`
  `Serie.Title`
  bei letzterem wird auch <YEAR> bei Option copMapToOneYear ausgewertet
- Eine Serie kann auch einer Achse zugeordnet werden. So können 2 Serien definiert werden, welche unterschiedliche Einheiten besitzen, die eine der linken, die andere der rechten Achse zugeordnet.

Weitere Methoden

```
Procedure PrintTo(OutputFormat: Integer; const FileName: String);
// OutputFormat:
// 10 = Chart im Querformat drucken (Parameter FileName wird ignoriert)
// 11 = Chart im Hochformat drucken (Parameter FileName wird ignoriert)
// 20 = Chart als Bitmap in die Zwischenablage kopieren
//      (Parameter FileName wird ignoriert)
// 21 = Chart als Metafile in die Zwischenablage kopieren
//      (Parameter FileName wird ignoriert)
// 22 = Chart als Enhanced-Metafile in die Zwischenablage kopieren
//      (Parameter FileName wird ignoriert)
// 30 = Chart in eine Bitmap-Datei (FileName) speichern
// 31 = Chart in eine Metafile-Datei (FileName) speichern
// 32 = Chart in eine Enhanced-Metafile-Datei (FileName) speichern
```

Hinweis: Mit der PrintTo-Funktion (verfügbar ab v3.12.11) können Sie die Chart-Grafik ausdrucken, in die Zwischenablage kopieren oder als Datei speichern.

```
Procedure SetNavigationOfSerie(serieIndex: Integer; const Navigation: String);
// i Serie Index  value
```

Hinweis: Um die Navigation des Charts im Scripting zu ändern, empfiehlt sich nur einzeilige Navigationen zu formulieren. Diese kann dann durch folgende Zuweisung geändert werden:

```
Chart.DataLink.Navigation[0] := 'VIA Object_To_Children CLASS HIP_Querprofil';
Procedure SetNavigationOfSerie(serieIndex: Integer; const Navigation: String);
```

4.3.35 BisDynChartII (ab v4.5.4)

Ist eine Diagramm Komponente auf der Basis von TChart (TeeChart Pro VCL 2012) und löst die Komponente BisDynChart ab.
http://www.steema.com/teechart/vcl

```
  TBisSerieTypeS = (bstLineSeries,
                    bstBarSeries,
                    bstHorizBarSeries,
                    bstAreaSeries,
                    bstPointSeries,
                    bstPieSeries,
                    bstFastLineSeries,
                    bst_byChart); //  wird von Chart (BisChartType) übernommen

  TBisSerieType = bstLineSeries .. bstFastLineSeries;

  TBisFuncionTypes = (bftAdd,
                      bftSubtract,
                      bftHigh,
                      bftLow,
                      bftCount,
                      bftMultiply,
                      bftDivide,
                      bftAverage,
                      bftMedian,
                      bftMode,
                      bftCrossPoints,
                      bftCumulative,
                      bftCurveFitting,
                      bftExponentialTrend,
                      bftExponentialAverage,
                      bftPerformance,
                      bftRootMeanSquare,
                      bftSmoothing,
                      bftStandardDeviation,
                      bftTrend,
                      bftDownSampling,
                      bftCorrelation,
                      bftVariance,
                      bftPerimeter,
                      bftAverageDirectionalChange,
                      bftMovingAverage,
                      bftExponentialMovingAverage,
                      bftRelativeStrengthIndex,
                      bftMomentum,
                      bftMomentumDivision,
                      bftMovingAverageConvergenceDivergence,
                      bftStochastic,
```

```
                    bftBollinger,
                    bftCompress,
                    bftAccumulationDistributionLine,
                    bftOnBalanceVolume,
                    bftCommodityChannelIndex,
                    bftPercentageVolumeOscillator,
                    bftStopAndReverse,
                    bftSubset,
                    bftHistogram);

TSeriesMarksStyle = (smsValue,             { 1234 }
                     smsPercent,           { 12 % }
                     smsLabel,             { Cars }
                     smsLabelPercent,      { Cars 12 % }
                     smsLabelValue,        { Cars 1234 }
                     smsLegend,            { (Legend.Style) }
                     smsPercentTotal,      { 12 % of 1234 }
                     smsLabelPercentTotal, { Cars 12 % of 1234 }
                     smsXValue,            { 1..2..3.. or 21/6/1996 }
                     smsXY,                { 123 456 }
                     smsSeriesTitle,       { Series1 }
                     smsPointIndex,        { 1..2..3... }
                     smsPercentRelative);  { 100%..90%..120%... }

TSeriesPointerStyle = (psRectangle,
                       psCircle,
                       psTriangle,
                       psDownTriangle,
                       psCross,
                       psDiagCross,
                       psStar,
                       psDiamond,
                       psSmallDot,
                       psNothing,
                       psLeftTriangle,
                       psRightTriangle,
                       psHexagon,
                       psVisual);

TVertAxis = (aLeftAxis, aRightAxis, aBothVertAxis, aCustomVertAxis);

TChartListOrder = (loNone, loAscending, loDescending);

TMultiBar = (mbNone, mbSide, mbStacked, mbStacked100, mbSideAll,
             mbSelfStack);

TBarStyle = (bsRectangle, bsPyramid, bsInvPyramid,
             bsCilinder, bsEllipse, bsArrow, bsRectGradient, bsCone,
             bsBevel, bsSlantCube, bsDiamond, bsInvArrow, bsInvCone,
             bsCustom, bsRoundRectangle);

TBisGroupedGridOperation = (gopAttribute,
                            gopGroupValue,
                            gopCount,
                            gopAttributeSum,
                            gopAttributeMin,
```

```
                                gopAttributeMax,
                                gopAttributeAvg);

  TFunctionPeriodAlign = (paFirst, paCenter, paLast);

  TFunctionPeriodStyle = (psNumPoints, psRange);

  TBisChartII = class(TChart)
    property BisChartType     : TBisSerieType
    property DataLink         : TBisDataLinkChart
    property BisSerie         : TBisSerieCollection -> TBisSerie
    property BisFunctionSerie : TBisFunctionSerieCollection -> TBisFunction-
Serie
    property MarginPercentHor : Integer
    property MarginPercentVer : Integer
  end

  TBisSerieAbstract = class(TCollectionItem)
    property BisChartType      : TBisSerieTypeS // Default = bst_byChart
    property DrawBetweenPoints : Boolean
    property Title             : String
      // Zugriff auf den Gruppenwert mit <GROUPVALUE>
    property MarkVisible       : Boolean
    property MarkStyle         : TSeriesMarksStyle
    property PointerVisible    : Boolean
    property Color             : TColor
    property ColorAutomatic    : Boolean
    property PointerStyle      : TSeriesPointerStyle
    property ShowInLegend      : Boolean
    property VertAxis          : TVertAxis
    property SortXOrder        : TChartListOrder
    property Visible           : Boolean        // Default = true
    property BarOptions        : TBarSeriesOptions
  end

  TBisSerie = class(TBisSerieAbstract)
    property AxisTitle  : TAxisTitleEvaluation
    property Name       : String
    property Navigation : TStrings
    property Criteria   : TOwnedCollection -> TBisGroupedGridCriterion
    property XValue     : TBisChartValue
    property YValue     : TBisChartValue
  end

Neu ab RQRQ

  TBisFunctionSerie = class(TBisSerieAbstract)
    property FunctionType    : TBisFuncionTypes     // Default = bftAverage
    property Period          : Double
    property PeriodAlign     : TFunctionPeriodAlign // Default = paCenter
    property PeriodStyle     : TFunctionPeriodStyle // Default = psNumPoints
    property ForBisSeries    : TForBisSerieCollection
      // verwendete BisSeries für die Funktion
    property ForEachBisSerie : Boolean              // Default = true
  end
```

```
TBarSeriesOptions = class(TPersistent)
  property MultiBar        : TMultiBar // Default = mbNone
  property BarStyle        : TBarStyle // Default = bsRectangle
  property BarWidthPercent : Integer   // Default = 70
  property OffsetPercent   : Integer   // Default = 0
  property DepthPercent    : Integer   // Default = 100
end

TAxisTitleEvaluation = class(TPersistent)
  property Title      : String
  property Navigation : String
end;

TBisGroupedGridCriterion = class(TBisCustomGroupedGridCriterion)
  property FilterNavigation : TStringList
  property Attribute        : String
  property Format           : String
end;

TBisChartValue = class(TPersistent)
  property Caption          : String
  property Attribute        : String
  property Format           : String
  property Operation        : TBisGroupedGridOperation
  property FilterNavigation : TStringList
end;
```

Prinzip:

Durch die `DataLink.Navigation` werden die Objekte bestimmt, welche je eine Serie (Datenreihe) ergeben. Wenn die Property `DataLink.Navigation` leer ist, wird das Objekt aus der DataSource verwendet. Für jedes Objekt aus `DataLink.Navigation` werden alle definierten Serien erstellt. Die Serien werden mit der Property `BisSerie` definiert.

BisSerie:

Mit `Navigation` werden aus dem Serien-Objekt die Objekte für die Datenpunkte ermittelt. In `XValue.Attribute` und `YValue.Attribute` werden die Attribute für die Werte (Koordinaten auf der X- und auf der Y-Achse) spezifiziert. Die Attribute für X und Y können auch als Expression formuliert werden. Dies wird durch ein ‚=' als erstes Zeichen formuliert. Z.B. "= - ATTRIBUTE('x_offset') + 3.5 / 150".
Die Dokumentation der Expressions finden Sie hier:

www.byron.ch/downloads/dokumente/ExpressionsForVariablesAndValues.pdf

BisFunctionSerie:

Mit der Property `BisFunctionSerie` können Funktionen für Serien definiert werden. Der Funktionstyp wird über `FunctionType` festgelegt. Mit `ForBisSeries` werden die Serien festgelegt, für welche die Funktion ihre Daten holt und berechnet.

Vor-Gruppierung:

Mit der Property `DataLink.Criteria` können Objekte aus `DataLink.Navigation` gruppiert werden. Für jedes Gruppierungskriterium muss ein `TBisGroupedGridCriterion` erstellt werden. Das Attribut zur Gruppierung wird mit der Property `Attribute` festgelegt. Falls über eine Assoziation gruppiert werden soll, kann die Property `FilterNavigation` verwendet werden. Die Formatierung

des Gruppenwerts kann mit `Format` festgelegt werden. So kann z.B. für ein Datumsattribut nach Jahr gruppiert werden, wenn das Format `yyyy` enthält.

Gruppierung der Serie:

Mit der Property `Criteria` können auch Objekte aus der Serie gruppiert werden. Die Konfiguration entspricht dem der Vor-Gruppierung (siehe oben).

Bemerkungen:

- Bei folgenden Properties können Ausdrücke in $-Konvention eingegeben werden (werden ausgewertet):
  `Chart.Title.Text`
  `Serie.Title`
- Eine Serie kann auch einer Achse zugeordnet werden. So können 2 Serien definiert werden, welche unterschiedliche Einheiten besitzen, die eine der linken, die andere der rechten Achse zugeordnet.

Weitere Methoden

```
Procedure PrintTo(OutputFormat: Integer; const FileName: String);
// OutputFormat:
// 10 = Chart im Querformat drucken (Parameter FileName wird ignoriert)
// 11 = Chart im Hochformat drucken (Parameter FileName wird ignoriert)
// 20 = Chart als Bitmap in die Zwischenablage kopieren
//      (Parameter FileName wird ignoriert)
// 21 = Chart als Metafile in die Zwischenablage kopieren
//      (Parameter FileName wird ignoriert)
// 22 = Chart als Enhanced-Metafile in die Zwischenablage kopieren
//      (Parameter FileName wird ignoriert)
// 30 = Chart in eine Bitmap-Datei (FileName) speichern
// 31 = Chart in eine Metafile-Datei (FileName) speichern
// 32 = Chart in eine Enhanced-Metafile-Datei (FileName) speichern
```

Hinweis: Mit der PrintTo-Funktion können Sie die Chart-Grafik ausdrucken, in die Zwischenablage kopieren oder als Datei speichern.

```
Procedure SetNavigationOfSerie(serieIndex: Integer; const Navigation: String);
```

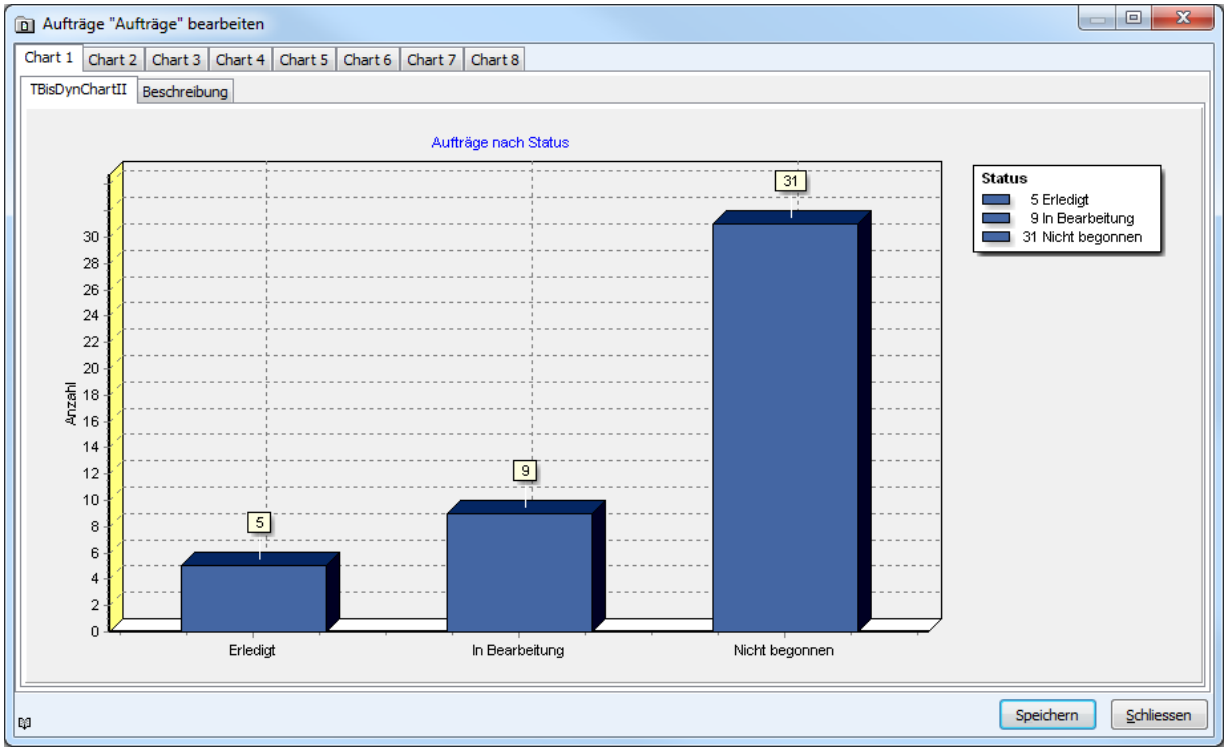
Hinweis: Um die Navigation des Charts im Scripting zu ändern, empfiehlt sich nur einzeilige Navigationen zu formulieren. Diese kann dann durch folgende Zuweisung geändert werden:

```
Chart.DataLink.Navigation[0] := 'VIA Object_To_Children CLASS HIP_Querprofil';
Procedure SetNavigationOfSerie(serieIndex: Integer; const Navigation: String);

Function GetSerie(serieIndex: Integer) : TBisSerie;
```

Hinweis: Mit der GetSerie-Funktion können Sie auf einer Serie zugreifen.

Beispiel 1 „Aufträge nach Status“ (Balken):

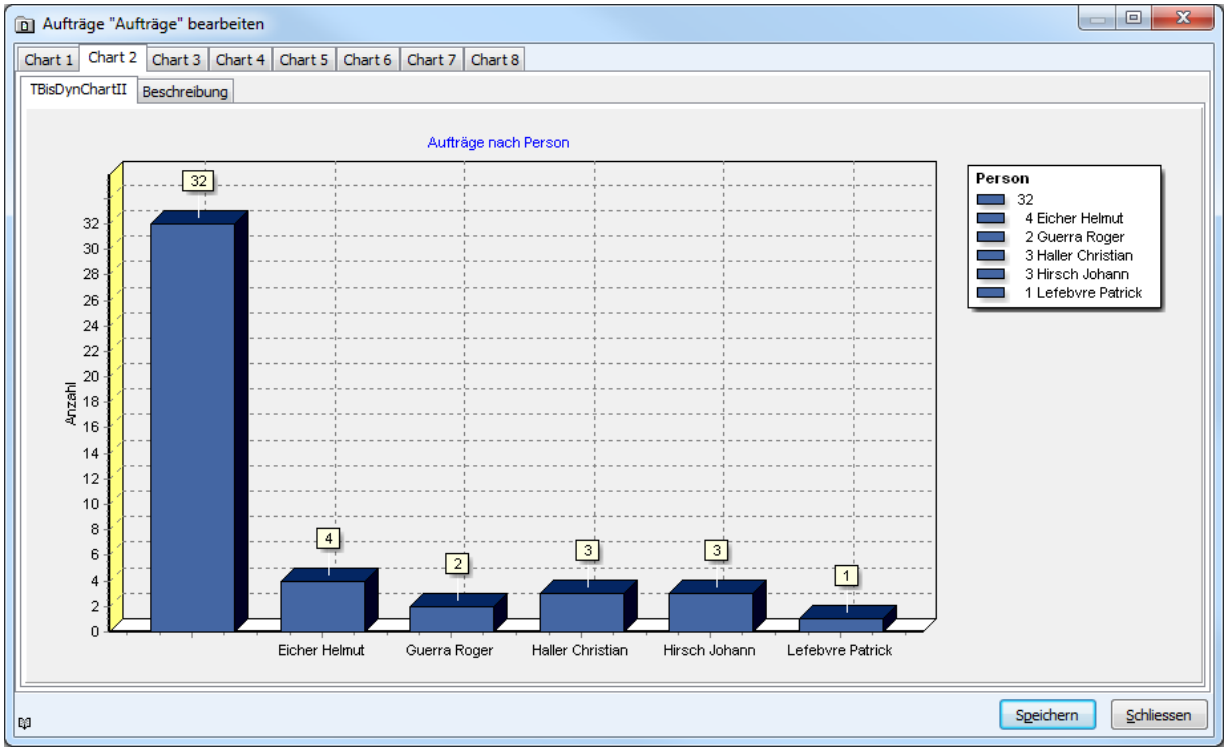


Quellcode 1 (Formular als Text):

```
object BisDynChartII1: TBisDynChartII
  Left = 0
  Top = 0
  Width = 805
  Height = 682
  Legend.Title.Text.Strings = (
    'Status')
  Title.Text.Strings = (
    'Auftr'#228'ge nach Status')
  Align = alClient
  TabOrder = 0
  BisChartType = bstLineSeries
  DataLink.DataSource = BisDataSource
  DataLink.CDLAttribute = '.'
  DataLink.Criteria = <>
  BisSerie = <
    item
      BisChartType = bstBarSeries
      DrawBetweenPoints = False
      Title = 'Auftr'#228'ge'
      MarkVisible = True
      MarkStyle = smsValue
      PointerVisible = False
      Color = clBlack
      ColorAutomatic = True
      PointerStyle = psRectangle
      ShowInLegend = True
      VertAxis = aLeftAxis
      SortXOrder = loNone
      AxisTitle.Title = 'Anzahl'
      Name = 'BisSerie1'
      Navigation.Strings = (
```

```
        'LOOP (VIA Object_To_Children) CLASS bisP_MaintOrder')
      Criteria = <
        item
          Attribute = 'bisP_OrderState'
        end>
      XValue.Attribute = 'bisP_OrderState'
      XValue.Operation = gopGroupValue
      YValue.Attribute = 'bisP_OrderState'
      YValue.Operation = gopCount
    end>
  BisFunctionSerie = <>
  MarginPercentHor = 0
  MarginPercentVer = 0
  ColorPaletteIndex = 13
end
```

Beispiel 2 „Aufträge nach Person" (Balken):

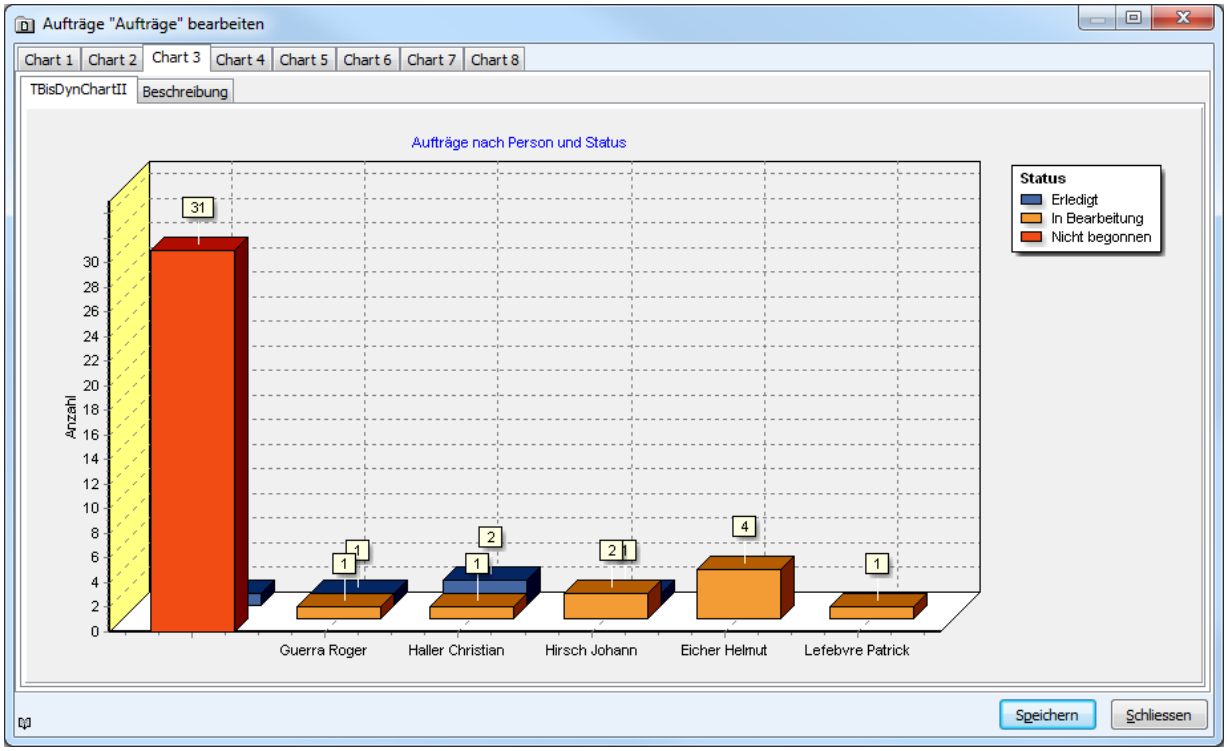


Quellcode 2 (Formular als Text):

```
object BisDynChartII2: TBisDynChartII
  Left = 0
  Top = 0
  Width = 805
  Height = 682
  Legend.Title.Text.Strings = (
    'Person')
  Title.Text.Strings = (
    'Auftr'#228'ge nach Person')
  Align = alClient
  TabOrder = 0
  BisChartType = bstLineSeries
  DataLink.DataSource = BisDataSource
  DataLink.CDLAttribute = '.'
```

```
  DataLink.Criteria = <>
  BisSerie = <
    item
      BisChartType = bstBarSeries
      DrawBetweenPoints = False
      Title = 'Auftr'#228'ge'
      MarkVisible = True
      MarkStyle = smsValue
      PointerVisible = False
      Color = clBlack
      ColorAutomatic = True
      PointerStyle = psRectangle
      ShowInLegend = True
      VertAxis = aLeftAxis
      SortXOrder = loNone
      AxisTitle.Title = 'Anzahl'
      Name = 'BisSerie1'
      Navigation.Strings = (
        'LOOP (VIA Object_To_Children) CLASS bisP_MaintOrder')
      Criteria = <
        item
          FilterNavigation.Strings = (
            'VIA bisP_MaintOrderToMaintPerson')
          Attribute = 'Object_Display_Kontext'
        end>
      XValue.Attribute = 'Object_Display_Kontext'
      XValue.Operation = gopGroupValue
      YValue.Attribute = 'Object_Display_Kontext'
      YValue.Operation = gopCount
    end>
  BisFunctionSerie = <>
  MarginPercentHor = 0
  MarginPercentVer = 0
  ColorPaletteIndex = 13
end
```

Beispiel 3 „Aufträge nach Person und Status“ (Balken):

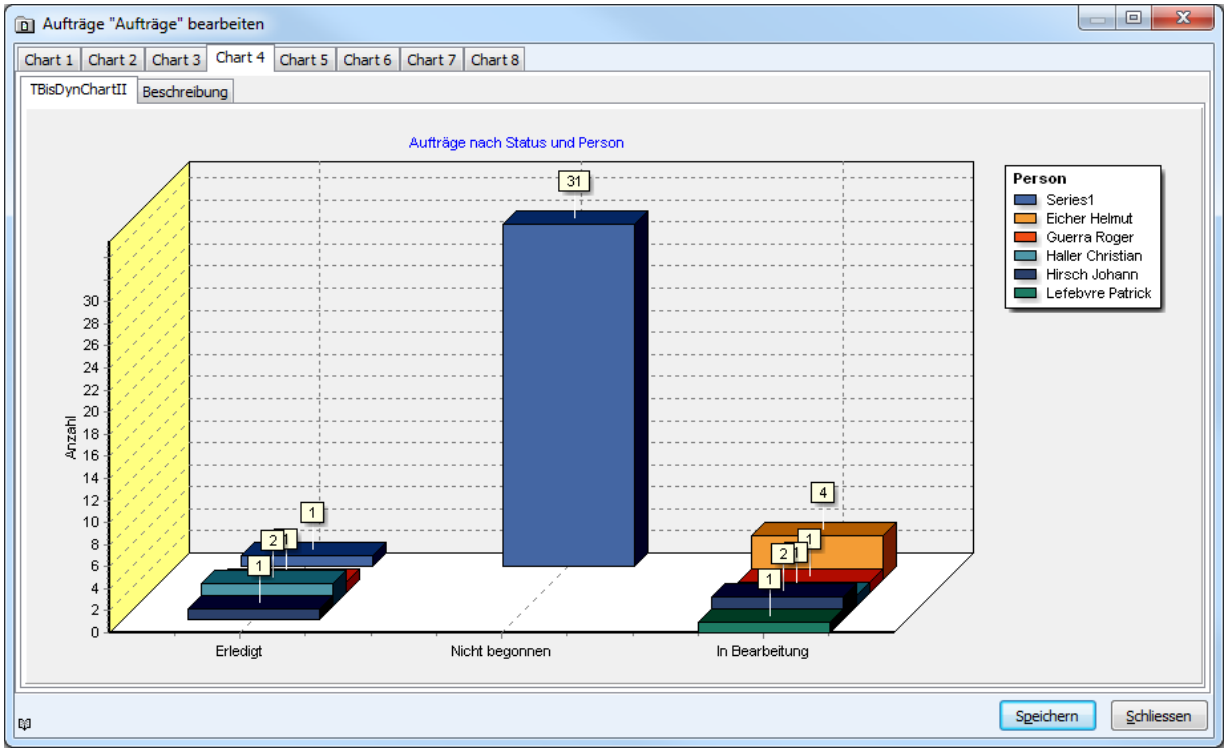


Quellcode 3 (Formular als Text):

```
object BisDynChartII3: TBisDynChartII
  Left = 0
  Top = 0
  Width = 805
  Height = 682
  Legend.Title.Text.Strings = (
    'Status')
  Title.Text.Strings = (
    'Auftr'#228'ge nach Person und Status')
  Align = alClient
  TabOrder = 0
  BisChartType = bstLineSeries
  DataLink.DataSource = BisDataSource
  DataLink.Navigation.Strings = (
    'LOOP (VIA Object_To_Children) CLASS bisP_MaintOrder')
  DataLink.CDLAttribute = '.'
  DataLink.Criteria = <
    item
      Attribute = 'bisP_OrderState'
    end>
  BisSerie = <
    item
      BisChartType = bstBarSeries
      DrawBetweenPoints = False
      Title = '<GROUPVALUE>'
      MarkVisible = True
      MarkStyle = smsValue
      PointerVisible = False
      Color = clBlack
      ColorAutomatic = True
      PointerStyle = psRectangle
      ShowInLegend = True
```

```
        VertAxis = aLeftAxis
        SortXOrder = loNone
        BarOptions.BarWidthPercent = 50
        AxisTitle.Title = 'Anzahl'
        Name = 'BisSerie1'
        Criteria = <
          item
            FilterNavigation.Strings = (
              'VIA bisP_MaintOrderToMaintPerson')
            Attribute = 'Object_Display_Kontext'
          end>
        XValue.Attribute = 'Object_Display_Kontext'
        XValue.Operation = gopGroupValue
        YValue.Attribute = 'Object_Display_Kontext'
        YValue.Operation = gopCount
      end>
    BisFunctionSerie = <>
    MarginPercentHor = 0
    MarginPercentVer = 0
    ColorPaletteIndex = 13
  end
```

Beispiel 4 „Aufträge nach Status und Person" (Balken):

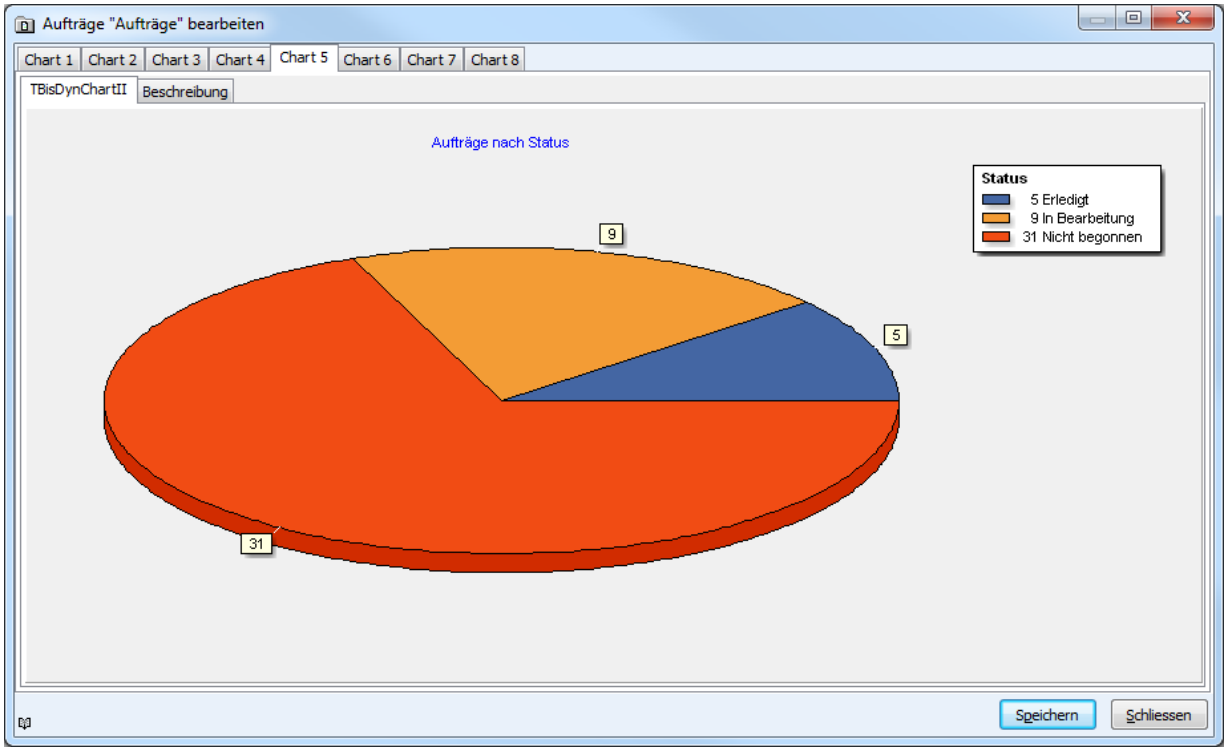


Quellcode 4 (Formular als Text):

```
object BisDynChartII4: TBisDynChartII
  Left = 0
  Top = 0
  Width = 805
  Height = 682
  Legend.Title.Text.Strings = (
    'Person')
  Title.Text.Strings = (
```

```
    'Auftr'#228'ge nach Status und Person')
  Align = alClient
  TabOrder = 0
  BisChartType = bstLineSeries
  DataLink.DataSource = BisDataSource
  DataLink.Navigation.Strings = (
    'LOOP (VIA Object_To_Children) CLASS bisP_MaintOrder')
  DataLink.CDLAttribute = '.'
  DataLink.Criteria = <
    item
      FilterNavigation.Strings = (
        'VIA bisP_MaintOrderToMaintPerson')
      Attribute = 'Object_Display_Kontext'
    end>
  BisSerie = <
    item
      BisChartType = bstBarSeries
      DrawBetweenPoints = False
      Title = '<GROUPVALUE>'
      MarkVisible = True
      MarkStyle = smsValue
      PointerVisible = False
      Color = clBlack
      ColorAutomatic = True
      PointerStyle = psRectangle
      ShowInLegend = True
      VertAxis = aLeftAxis
      SortXOrder = loNone
      BarOptions.BarWidthPercent = 50
      AxisTitle.Title = 'Anzahl'
      Name = 'BisSerie1'
      Criteria = <
        item
          Attribute = 'bisP_OrderState'
        end>
      XValue.Attribute = 'Object_Display_Kontext'
      XValue.Operation = gopGroupValue
      YValue.Attribute = 'Object_Display_Kontext'
      YValue.Operation = gopCount
    end>
  BisFunctionSerie = <>
  MarginPercentHor = 0
  MarginPercentVer = 0
  ColorPaletteIndex = 13
end
```

Beispiel 5 „Aufträge nach Status“ (Pie):

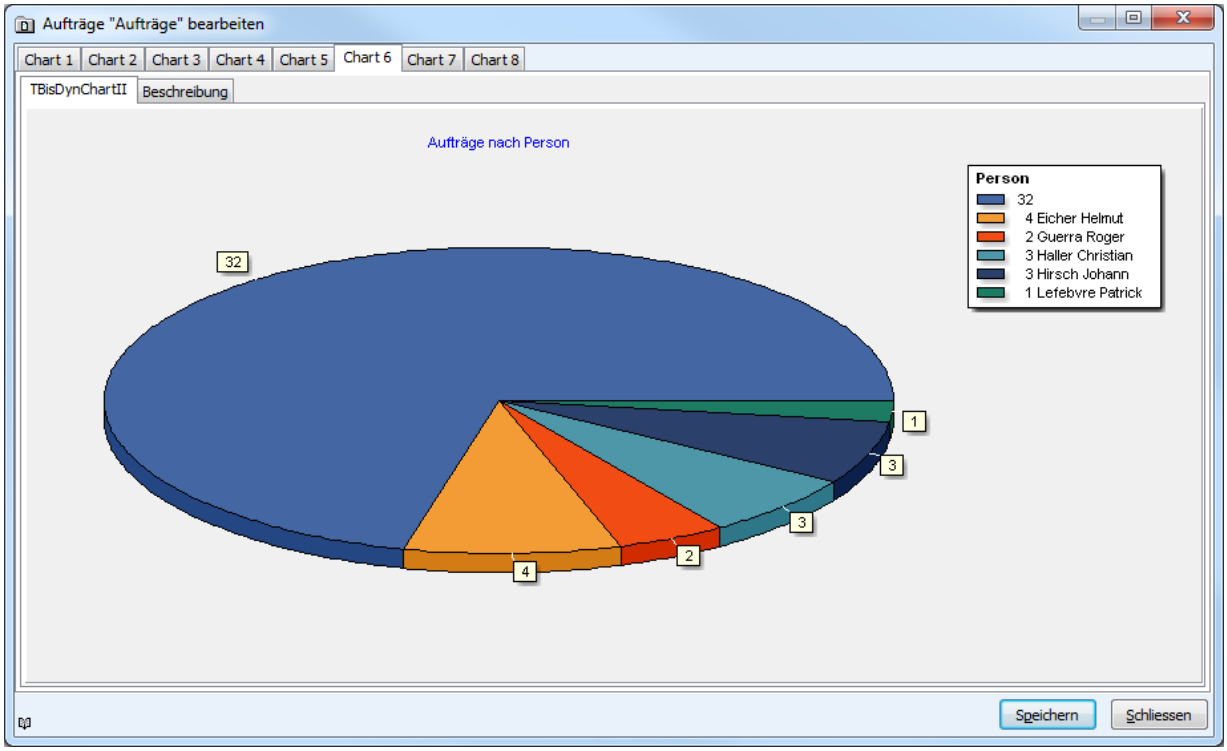


Quellcode 5 (Formular als Text):

```
object BisDynChartII5: TBisDynChartII
  Left = 0
  Top = 0
  Width = 805
  Height = 682
  Legend.Title.Text.Strings = (
    'Status')
  Title.Text.Strings = (
    'Auftr'#228'ge nach Status')
  Align = alClient
  TabOrder = 0
  BisChartType = bstLineSeries
  DataLink.DataSource = BisDataSource
  DataLink.CDLAttribute = '.'
  DataLink.Criteria = <>
  BisSerie = <
    item
      BisChartType = bstPieSeries
      DrawBetweenPoints = False
      Title = '<GROUPVALUE>'
      MarkVisible = True
      MarkStyle = smsValue
      PointerVisible = False
      Color = clBlack
      ColorAutomatic = True
      PointerStyle = psRectangle
      ShowInLegend = True
      VertAxis = aLeftAxis
      SortXOrder = loNone
      AxisTitle.Title = 'Anzahl'
      Name = 'BisSerie1'
      Navigation.Strings = (
```

```
        'LOOP (VIA Object_To_Children) CLASS bisP_MaintOrder')
      Criteria = <
        item
          Attribute = 'bisP_OrderState'
        end>
      XValue.Attribute = 'Object_Display_Kontext'
      XValue.Operation = gopGroupValue
      YValue.Attribute = 'Object_Display_Kontext'
      YValue.Operation = gopCount
    end>
  BisFunctionSerie = <>
  MarginPercentHor = 0
  MarginPercentVer = 0
  ColorPaletteIndex = 13
end
```

Beispiel 6 „Aufträge nach Person" (Pie):



Quellcode 6 (Formular als Text):

```
object BisDynChartII6: TBisDynChartII
  Left = 0
  Top = 0
  Width = 805
  Height = 682
  Legend.Title.Text.Strings = (
    'Person')
  Title.Text.Strings = (
    'Auftr'#228'ge nach Person')
  Align = alClient
  TabOrder = 0
  BisChartType = bstLineSeries
  DataLink.DataSource = BisDataSource
  DataLink.CDLAttribute = '.'
```

```
  DataLink.Criteria = <>
  BisSerie = <
    item
      BisChartType = bstPieSeries
      DrawBetweenPoints = False
      Title = '<GROUPVALUE>'
      MarkVisible = True
      MarkStyle = smsValue
      PointerVisible = False
      Color = clBlack
      ColorAutomatic = True
      PointerStyle = psRectangle
      ShowInLegend = True
      VertAxis = aLeftAxis
      SortXOrder = loNone
      AxisTitle.Title = 'Anzahl'
      Name = 'BisSerie1'
      Navigation.Strings = (
        'LOOP (VIA Object_To_Children) CLASS bisP_MaintOrder')
      Criteria = <
        item
          FilterNavigation.Strings = (
            'VIA bisP_MaintOrderToMaintPerson')
          Attribute = 'Object_Display_Kontext'
        end>
      XValue.Attribute = 'Object_Display_Kontext'
      XValue.Operation = gopGroupValue
      YValue.Attribute = 'Object_Display_Kontext'
      YValue.Operation = gopCount
    end>
  BisFunctionSerie = <>
  MarginPercentHor = 0
  MarginPercentVer = 0
  ColorPaletteIndex = 13
end
```

4.3.36 BisPageControl (ab v3.12.5)

TBisPageControl dient dazu, Dialogfelder aufzubauen, die aus mehreren, sich überlappenden Seiten (Registerkarten) bestehen. Die Registerkarten (TBisTabSheet) können Datenbank-gesteuert ein- und ausgeblendet werden.

Eigenschaften von TBisPageControl:

DataLink	Bestimmt die DataSource der Datenbank-Anbindung und mittels der Navigation das Datenbank-Objekt, welches für die Filter der Register-karten verwendet wird.
OptimizedLoad	Wird diese Eigenschaft gesetzt, so werden Elemente auf den TabSheets erst geladen, wenn das TabSheet vom Benutzer angewählt wird. Das verzögerte Laden kann für einzelne Elemente deaktiviert werden, indem an deren DataLink die Eigenschaft OptimizedLoad nicht gesetzt wird.

Eigenschaften von TBisTabSheet:

FilterDef	Typ: String Definition eines Filters mittels FilterNavigation

OnUpdateVisibility	Typ: TNotifyEvent (Ereignis) Ereignis zur Bestimmung der Sichtbarkeit mittels Skript.
ObjectClassFilter	Typ: TStrings (erst ab v4.10.2)
ObjectClassFilterIncl	Typ: Boolean (erst ab v4.10.2)

Die Sichtbarkeit der Registerkarten wird folgendermassen berechnet:

- Ist ein Ereignis zu OnUpdateVisibility gesetzt, wird die Berechnung vollständig an das Ereignis delegiert.
- Ansonsten wird die FilterNavigation in FilterDef mit dem Objekt des DataLinks ausgeführt. Die Registerkarte ist sichtbar, wenn die Ausführung der FilterDef mindestens ein Objekt liefert.
- Ab v4.10.2: Ist keine FilterDef angegeben, wird der Klassenfiler in ObjectClassFilter (und ObjectClassFilterIncl) zur Prüfung der Sichtbarkeit angewendet.
- Wichtiger Hinweis: Beim Erstellen eines neues Objekts wird das Default-Objekt der Objektklasse des zu erzeugenden Objekts als Startmenge für die FilterDef verwendet. Existiert kein solches, erfolgt der Aufruf der FilterDef mit der leeren Startmenge.

TabVisible	Typ: Boolean Sichtbarkeit des Registers (Ersetzt Eigenschaft «Visible»)

4.3.37 TBisGroupedGrid (ab v4.3.0)

TBisGroupedGrid ist eine Grid-Komponente, mit welcher Objekte gruppiert dargestellt werden können. Dabei entspricht jede Zeile im Grid einer Gruppierung. Ab v4.10.5 werden nicht mehr die Gruppenobjekte, sondern die (Eingangs-)Objekt angezeigt. Ist kein aktives Gruppierungskriterium vorhanden, werden alle (Eingangs-)Objekt angezeigt. Sind aktive Gruppierungskriterien vorhanden, werden die (Eingangs-)Objekt anhand der Gruppierungskriterien gruppiert. Für jede Gruppierung (nicht jedes Gruppenkriterium) wird das erste (Eingangs-)Objekt verwendet und dargestellt.

DataLink.DataSource	Bestimmt eine DataSource oder ein DataSet, welche die (Eingangs-)Objekte für die Gruppierung liefern. Um mehrere Objekte zu gruppieren, wird meisten ein DataSet verwendet.
Criteria	Typ: TBisGroupedGridCriterion Bestimmt die Gruppierungskriterien für die Gruppierung der Objekte.
Columns	Typ: TBisGroupedGridColumn Bestimmt die Spalten, welche angezeigt werden sollen.
GroupedGridOptions	Typ: Menge Beschreibt die Optionen für das Gruppen-Grid.

ggoAllowDrag	Erlaubt das Ziehen (Drag) von Gruppenobjekten.
ggoAllowCopy	Erlaubt das Kopieren in die Zwischenablage durch CTRL+C oder Popupmenü von Gruppenobjekten.
ggoAllowOpen	Erlaubt das Öffnen durch Doppelklick oder Popupmenü von Gruppenobjekten.

ggoAllowProperties	Erlaubt das Öffnen des Eigenschaftsblattes durch Popupmenü von Gruppenobjekten.
ggoBroaodcastSelection	Leitet die Selektion der Gruppenobjekte an andere Komponenten weiter.
ggoSelectByBroadcast	Empfängt von anderen Komponenten eine Selektion und selektiert die gefundenen Gruppenobjekte.
ggoSelectFirstRowAfterLoad	Selektiert die erste Zeile (erstes Gruppenobjekt) nach dem Laden.

4.3.38 TBisGroupedGridCriterion

Beschreibt die Gruppierungskriterien im Gruppen-Grid.

Aktive	Typ: Boolean (ab v4.10.5) Bestimmt, ob das Gruppierungskriterium verwendet (Aktiv=true) werden soll oder nicht. Default = true
Attribute	Typ: Text (Optional) Bestimmt das Attribut, nach welchem gruppiert werden soll. Als Standard wird die Objekt-URL verwendet. Expressions sind auch erlaubt und müssen mit dem Gleichheitszeichen ("=") beginnen z.B.: "=SUBSTR(ATTRIBUTE('Name'), 1, 1)" gruppiert nach dem 1. Zeichen aus der Bezeichnung (Name)
FilterNavigation	Typ: String (Optional) Bestimmt mittels FilterNavigation das Objekt für die Gruppierung. Die Objekte aus DataLink.DataSource werden als Eingangsobjekte für die Filternavigation verwendet.
Format	Typ: Text (Optional) Bestimmt das Format für das Attribut, wenn Attribut keine Expression enthält.

4.3.39 TBisGroupedGridColumn

Beschreibt eine Spalte im Gruppen-Grid.

Spezifische Eigenschaften

Attribute	Typ: Text (Optional) Gibt das darzustellende Attribut an. Ist für die Attribut-Operationen (gopAttributeXXX) notwendig.
Caption	Typ: Text (Optional) Beschriftung die angezeigt wird. Als Standard wird die Beschriftung des Attributs aus dem Modell übernommen.
FilterNavigation	Typ: String (Optional) Bestimmt mittels FilterNavigation das Objekt für die Darstellung von referenzierten Attributen. Das erste Objekt aus einer Gruppierung (Zeile) wird als Eingangsobjekt für die Filternavigation verwendet.
Format	Typ: Text (Optional) Legt die Formatierung für den angezeigten Wertes fest.

Auf die Werte der Gruppierungskriterien kann mittels %0:s zugegriffen werden. Der erste (Gruppen-)Wert beginnt mit dem Index 0.
Z.B. "%0:s - %1:s" stellt den ersten und zweiten Gruppenwert getrennt durch " - " dar.

Operation

Typ: Aufzählung
Die Eigenschaft *Operation* bestimmt den Typ der Spalte.

Werte:

gopAttribute	Der Wert des angegebenen Attributs wird angezeigt.
gopAttributeAvg	Der durchschnittliche Wert des angegebenen Attributs wird angezeigt.
gopAttributeMax	Der maximale Wert des angegebenen Attributs wird angezeigt.
gopAttributeMin	Der minimale Wert des angegebenen Attributs wird angezeigt.
gopAttributeSum	Die Summe der Werte des angegebenen Attributs wird angezeigt.
gopCount	Die Anzahl der Objekte in der Gruppierung wird angezeigt.
gopGroupValue	Der Gruppierungswert wird angezeigt.

PreOperationNav

Typ: String (Optional) (ab v4.10.6)
Wendet die FilterNavigation für jede Zeile auf die gruppierten Objekte an, bevor die Operation durchgeführt wird.
Die FilterNavigation wird nur bei den Operationen gopAttributeMin, gopAttributeMax, gopAttributeAvg, gopAttributeSum und gopCount ausgeführt.

SummaryOperation

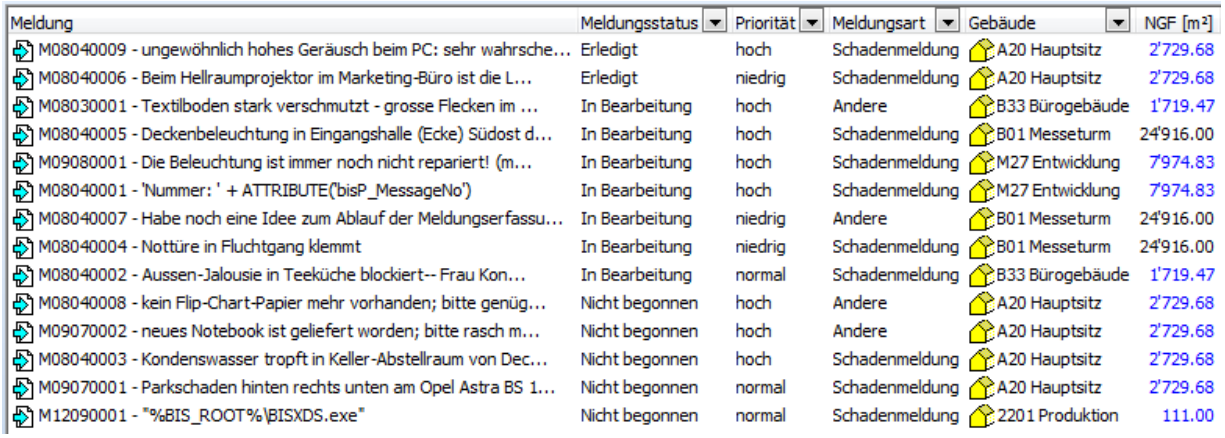
Typ: Aufzählung (ab v4.10.5)
Die Eigenschaft *SummaryOperation* bestimmt den Datentyp für die Zelle in der Summenzeile. Wenn SummaryText eine Expression enthält, ist der Datentyp immer „Text". Die Summenzeile wird nur angezeigt, wenn mindestens eine sichtbare Spalte einen anderen Wert als gsopNone enthält.

Werte:

gsopNone	Keine Anzeige in der Summenzeile.
gsopText	Datentyp: Text Der Text aus *SummaryText* wird angezeigt.
gsopSum	Datentyp: Gleitkommazahl Die Summe der Werte aus allen Zeilen in dieser Spalte wird angezeigt.
gsopMin	Datentyp: Gleitkommazahl Der minimale Wert aus allen Zeilen dieser Spalte wird angezeigt.
gsopMax	Datentyp: Gleitkommazahl Der maximale Wert aus allen Zeilen dieser Spalte wird angezeigt.

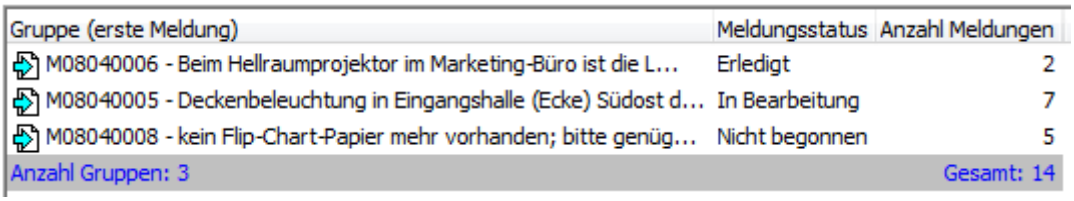
	gsopAvg	Datentyp: Gleitkommazahl Der durchschnittliche Wert aus allen Zeilen dieser Spalte wird angezeigt
	gsopCount	Datentyp: ganze Zahl Die Anzahl der Objekte in der Gruppierung aus allen Zeilen dieser Spalte wird angezeigt.
	gsopRowCount	Datentyp: ganze Zahl Die Anzahl der Zeilen (ohne sie Summenzeile) wird angezeigt.
SummaryText	Typ: String (Optional) (ab v4.10.5) Bestimmt den Text für die Spalte in der Summenzeile. Muss angegeben werden, wenn SummaryOperation=gsopText. Kann angegeben werden, wenn SummaryOperation<>gsopText. Expressions sind auch erlaubt und müssen mit dem Gleichheitszeichen ("=") beginnen z.B.: "='Anzahl Zeilen: ' + FORMAT(RowCount, '%d')" → "Anzahl Zeilen: 12". In den Expressions kann auf folgende Variablen zugegriffen werden: RowCount: Typ: Ganze Zahl Anzahl der Zeilen im Grid (ohne Summenzeile). SummaryValue: Typ: abhängig von der SummaryOperation Wert für die Summenzeile aus eigenen Spalte. SummaryValue_<col_id>: Typ: abhängig von der SummaryOperation Wert für die Summenzeile aus Spalte <col_id>. z.B.: "='Gesamtsumme: ' + FORMAT(SummaryValue_Col_4, '%.2f')“ → "'Gesamtsumme: 123.45".	
SummaryBackgroundColorIndex	Typ: Integer (ab v4.10.5) Bestimmt die Hintergrundfarbe für die Zelle in der Summenzeile für diese Spalte. Werte: -1 = Grid.Color 1 = Grid.BackgroundColor1 2 = Grid.BackgroundColor2 3 = Grid.BackgroundColor3 4 = Grid.BackgroundColor4 5 = Grid.BackgroundColor5 Default = -1	
SummaryFontIndex	Typ: Integer (ab v4.10.5) Bestimmt die Schrift für die Zelle in der Summenzeile für diese Spalte. Werte: 0 = Grid.Font 1 = Grid.Font1 2 = Grid.Font2 3 = Grid.Font3 4 = Grid.Font4 5 = Grid.Font5 Default = 0	

Normales Grid mit Meldungen ohne Gruppierung:



Meldung	Meldungsstatus	Priorität	Meldungsart	Gebäude	NGF [m²]
M08040009 - ungewöhnlich hohes Geräusch beim PC: sehr wahrsche...	Erledigt	hoch	Schadenmeldung	A20 Hauptsitz	2'729.68
M08040006 - Beim Hellraumprojektor im Marketing-Büro ist die L...	Erledigt	niedrig	Schadenmeldung	A20 Hauptsitz	2'729.68
M08030001 - Textilboden stark verschmutzt - grosse Flecken im ...	In Bearbeitung	hoch	Andere	B33 Bürogebäude	1'719.47
M08040005 - Deckenbeleuchtung in Eingangshalle (Ecke) Südost d...	In Bearbeitung	hoch	Schadenmeldung	B01 Messeturm	24'916.00
M09080001 - Die Beleuchtung ist immer noch nicht repariert! (m...	In Bearbeitung	hoch	Schadenmeldung	M27 Entwicklung	7'974.83
M08040001 - 'Nummer: ' + ATTRIBUTE('bisP_MessageNo')	In Bearbeitung	hoch	Schadenmeldung	M27 Entwicklung	7'974.83
M08040007 - Habe noch eine Idee zum Ablauf der Meldungserfassu...	In Bearbeitung	niedrig	Andere	B01 Messeturm	24'916.00
M08040004 - Nottüre in Fluchtgang klemmt	In Bearbeitung	niedrig	Schadenmeldung	B01 Messeturm	24'916.00
M08040002 - Aussen-Jalousie in Teeküche blockiert-- Frau Kon...	In Bearbeitung	normal	Schadenmeldung	B33 Bürogebäude	1'719.47
M08040008 - kein Flip-Chart-Papier mehr vorhanden; bitte genüg...	Nicht begonnen	hoch	Andere	A20 Hauptsitz	2'729.68
M09070002 - neues Notebook ist geliefert worden; bitte rasch m...	Nicht begonnen	hoch	Andere	A20 Hauptsitz	2'729.68
M08040003 - Kondenswasser tropft in Keller-Abstellraum von Dec...	Nicht begonnen	hoch	Schadenmeldung	A20 Hauptsitz	2'729.68
M09070001 - Parkschaden hinten rechts unten am Opel Astra BS 1...	Nicht begonnen	normal	Schadenmeldung	A20 Hauptsitz	2'729.68
M12090001 - "%BIS_ROOT%\BISXDS.exe"	Nicht begonnen	normal	Schadenmeldung	2201 Produktion	111.00

Beispiel 1 „Meldungen gruppiert nach Meldungsstatus“:



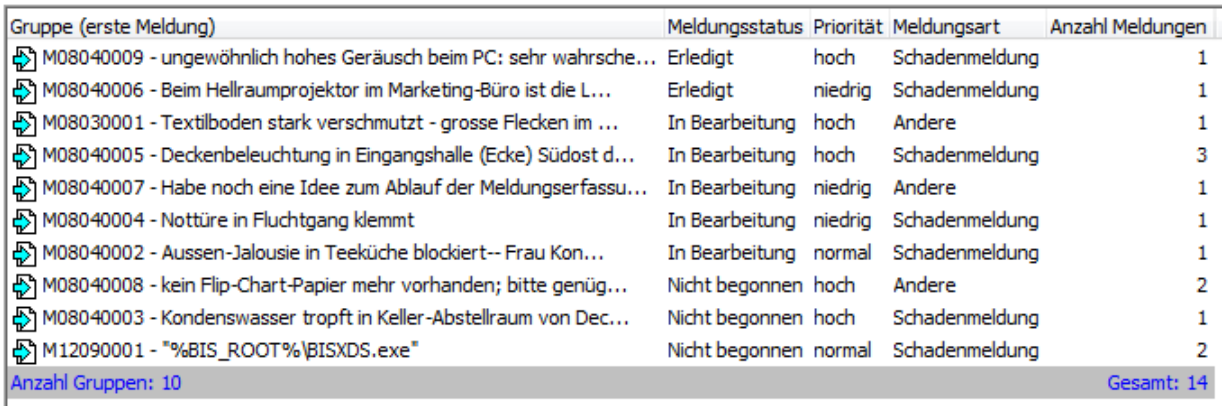
Gruppe (erste Meldung)	Meldungsstatus	Anzahl Meldungen
M08040006 - Beim Hellraumprojektor im Marketing-Büro ist die L...	Erledigt	2
M08040005 - Deckenbeleuchtung in Eingangshalle (Ecke) Südost d...	In Bearbeitung	7
M08040008 - kein Flip-Chart-Papier mehr vorhanden; bitte genüg...	Nicht begonnen	5
Anzahl Gruppen: 3		Gesamt: 14

Quellcode 1 (Formular als Text):

```
      object BisGroupedGrid1: TBisGroupedGrid
        Left = 0
        Top = 0
        Width = 660
        Height = 631
        Criteria = <
          item
            FilterNavigation.Strings = (
              ' ')
            Attribute = 'bisP_MessageState'
          end>
        GroupedGridOptions = [ggoAllowDrag, ggoAllowOpen]
        DataLink.DataSource = BisDataSet1
        Columns = <
          item
            Attribute = 'Object_Display_Kontext'
            Operation = gopAttribute
            Id = 'Col_15'
            FontIndex = 0
            BackgroundColorIndex = -1
            SummaryOperation = gsopRowCount
            SummaryText = '='#39'Anzahl Gruppen: '#39' + FORMAT(SummaryValue,
'#39'%d'#39')'
            SummaryFontIndex = 1
            SummaryBackgroundColorIndex = 1
            Caption = 'Gruppe (erste Meldung)'
          end
          item
            Attribute = 'bisP_MessageState'
            Operation = gopAttribute
            Id = 'Col_1'
            FontIndex = 0
            BackgroundColorIndex = -1
            SummaryOperation = gsopNone
```

```
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Meldungsstatus'
    end
    item
      Operation = gopCount
      Id = 'Col_2'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopCount
      SummaryText = '='#39'Gesamt: '#39' + FORMAT(SummaryValue, '#39'%d'#39')'
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Anzahl Meldungen'
    end>
  SortedCols = '2;3;4'
  Font1.Charset = DEFAULT_CHARSET
  Font1.Color = clWindowText
  Font1.Height = -11
  Font1.Name = 'Tahoma'
  Font1.Style = [fsBold]
  Font2.Charset = DEFAULT_CHARSET
  Font2.Color = clWindowText
  Font2.Height = -11
  Font2.Name = 'Tahoma'
  Font2.Style = [fsItalic]
  Font3.Charset = DEFAULT_CHARSET
  Font3.Color = clWindowText
  Font3.Height = -11
  Font3.Name = 'Tahoma'
  Font3.Style = [fsUnderline]
  Font4.Charset = DEFAULT_CHARSET
  Font4.Color = clWindowText
  Font4.Height = -11
  Font4.Name = 'Tahoma'
  Font4.Style = [fsStrikeOut]
  Font5.Charset = DEFAULT_CHARSET
  Font5.Color = clWindowText
  Font5.Height = -11
  Font5.Name = 'Tahoma'
  Font5.Style = [fsBold, fsItalic, fsUnderline]
  BackgroundColor1 = clSilver
  BackgroundColor2 = clWindow
  BackgroundColor3 = clWindow
  BackgroundColor4 = clWindow
  BackgroundColor5 = clWindow
  Align = alClient
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
  Font.Name = 'Tahoma'
  Font.Style = []
  ParentFont = False
  ParentShowHint = False
  ShowHint = True
  TabOrder = 0
end
```

Beispiel 2 „Meldungen gruppiert nach Meldungsstatus, Priorität und Meldungsart“:



Gruppe (erste Meldung)	Meldungsstatus	Priorität	Meldungsart	Anzahl Meldungen
M08040009 - ungewöhnlich hohes Geräusch beim PC: sehr wahrsche...	Erledigt	hoch	Schadenmeldung	1
M08040006 - Beim Hellraumprojektor im Marketing-Büro ist die L...	Erledigt	niedrig	Schadenmeldung	1
M08030001 - Textilboden stark verschmutzt - grosse Flecken im ...	In Bearbeitung	hoch	Andere	1
M08040005 - Deckenbeleuchtung in Eingangshalle (Ecke) Südost d...	In Bearbeitung	hoch	Schadenmeldung	3
M08040007 - Habe noch eine Idee zum Ablauf der Meldungserfassu...	In Bearbeitung	niedrig	Andere	1
M08040004 - Nottüre in Fluchtgang klemmt	In Bearbeitung	niedrig	Schadenmeldung	1
M08040002 - Aussen-Jalousie in Teeküche blockiert-- Frau Kon...	In Bearbeitung	normal	Schadenmeldung	1
M08040008 - kein Flip-Chart-Papier mehr vorhanden; bitte genüg...	Nicht begonnen	hoch	Andere	2
M08040003 - Kondenswasser tropft in Keller-Abstellraum von Dec...	Nicht begonnen	hoch	Schadenmeldung	1
M12090001 - "%BIS_ROOT%\BISXDS.exe"	Nicht begonnen	normal	Schadenmeldung	2
Anzahl Gruppen: 10				Gesamt: 14

Quellcode 2 (Formular als Text):

```
          object BisGroupedGrid2: TBisGroupedGrid
        Left = 0
        Top = 0
        Width = 660
        Height = 631
        Criteria = <
          item
            FilterNavigation.Strings = (
              ' ')
            Attribute = 'bisP_MessageState'
          end
          item
            Attribute = 'bisP_MessagePriorty'
          end
          item
            Attribute = 'bisP_MessageKind'
          end>
        GroupedGridOptions = [ggoAllowDrag, ggoAllowOpen]
        DataLink.DataSource = BisDataSet1
        Columns = <
          item
            Attribute = 'Object_Display_Kontext'
            Operation = gopAttribute
            Id = 'Col_15'
            FontIndex = 0
            BackgroundColorIndex = -1
            SummaryOperation = gsopRowCount
            SummaryText = '='#39'Anzahl Gruppen: '#39' + FORMAT(SummaryValue,
'#39'%d'#39')'
            SummaryFontIndex = 1
            SummaryBackgroundColorIndex = 1
            Caption = 'Gruppe (erste Meldung)'
          end
          item
            Attribute = 'bisP_MessageState'
            Operation = gopAttribute
            Id = 'Col_1'
            FontIndex = 0
            BackgroundColorIndex = -1
            SummaryOperation = gsopNone
            SummaryFontIndex = 1
            SummaryBackgroundColorIndex = 1
            Caption = 'Meldungsstatus'
          end
          item
```

```
      Attribute = 'bisP_MessagePriorty'
      Operation = gopAttribute
      Id = 'Col_8'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopNone
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Priorit'#228't'
    end
    item
      Attribute = 'bisP_MessageKind'
      Operation = gopAttribute
      Id = 'Col_13'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopNone
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Meldungsart'
    end
    item
      Operation = gopCount
      Id = 'Col_2'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopCount
      SummaryText = '='#39'Gesamt: '#39' + FORMAT(SummaryValue, '#39'%d'#39')'
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Anzahl Meldungen'
    end>
  SortedCols = '2;3;4'
  Font1.Charset = DEFAULT_CHARSET
  Font1.Color = clWindowText
  Font1.Height = -11
  Font1.Name = 'Tahoma'
  Font1.Style = [fsBold]
  Font2.Charset = DEFAULT_CHARSET
  Font2.Color = clWindowText
  Font2.Height = -11
  Font2.Name = 'Tahoma'
  Font2.Style = [fsItalic]
  Font3.Charset = DEFAULT_CHARSET
  Font3.Color = clWindowText
  Font3.Height = -11
  Font3.Name = 'Tahoma'
  Font3.Style = [fsUnderline]
  Font4.Charset = DEFAULT_CHARSET
  Font4.Color = clWindowText
  Font4.Height = -11
  Font4.Name = 'Tahoma'
  Font4.Style = [fsStrikeOut]
  Font5.Charset = DEFAULT_CHARSET
  Font5.Color = clWindowText
  Font5.Height = -11
  Font5.Name = 'Tahoma'
  Font5.Style = [fsBold, fsItalic, fsUnderline]
  BackgroundColor1 = clSilver
  BackgroundColor2 = clWindow
  BackgroundColor3 = clWindow
  BackgroundColor4 = clWindow
```

```
    BackgroundColor5 = clWindow
    Align = alClient
    Font.Charset = DEFAULT_CHARSET
    Font.Color = clWindowText
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = []
    ParentFont = False
    ParentShowHint = False
    ShowHint = True
    TabOrder = 0
  end
```

Beispiel 3 „Meldungen gruppiert nach Gebäude (ohne PreOperationNav)“:

Gruppe (erste Meldung)	Gebäude	Summe NGF [m²]	Anzahl Meldungen
M12090001 - "%BIS_ROOT%\BISXDS.exe"	2201 Produktion	111.00	1
M09070002 - neues Notebook ist geliefert worden; bitte rasch m...	A20 Hauptsitz	16'378.07	6
M08040004 - Nottüre in Fluchtgang klemmt	B01 Messeturm	74'748.00	3
M08040002 - Aussen-Jalousie in Teeküche blockiert-- Frau Kon...	B33 Bürogebäude	3'438.94	2
M09080001 - Die Beleuchtung ist immer noch nicht repariert! (m...	M27 Entwicklung	15'949.66	2
Anzahl Gruppen: 5	Gesamt:	110'625.67	14

Quellcode 3 (Formular als Text):

```
  object BisGroupedGrid3: TBisGroupedGrid
    Left = 0
    Top = 0
    Width = 660
    Height = 631
    Criteria = <
      item
        FilterNavigation.Strings = (
          'VIA bisP_MessageToSpace')
      end>
    GroupedGridOptions = [ggoAllowDrag, ggoAllowOpen]
    DataLink.DataSource = BisDataSet1
    Columns = <
      item
        Attribute = 'Object_Display_Kontext'
        Operation = gopAttribute
        Id = 'Col_15'
        FontIndex = 0
        BackgroundColorIndex = -1
        SummaryOperation = gsopRowCount
        SummaryText = '='#39'Anzahl Gruppen: '#39' + FORMAT(SummaryValue,
'#39'%d'#39')'
        SummaryFontIndex = 1
        SummaryBackgroundColorIndex = 1
        Caption = 'Gruppe (erste Meldung)'
      end
      item
        Attribute = 'Object_Display_kontext'
        FilterNavigation.Strings = (
          'VIA bisP_MessageToSpace ')
        Operation = gopAttribute
        Id = 'Col_5'
        FontIndex = 0
        BackgroundColorIndex = -1
        SummaryOperation = gsopText
```

```
      SummaryText = 'Gesamt:'
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Geb'#228'ude'
    end
    item
      Attribute = 'bisB_NetGroundArea'
      FilterNavigation.Strings = (
        'VIA bisP_MessageToSpace')
      Operation = gopAttributeSum
      Id = 'Col_3'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopSum
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Summe NGF [m'#178']'
    end
    item
      Operation = gopCount
      Id = 'Col_2'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopCount
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Anzahl Meldungen'
    end>
  SortedCols = '2;3'
  Font1.Charset = DEFAULT_CHARSET
  Font1.Color = clWindowText
  Font1.Height = -11
  Font1.Name = 'Tahoma'
  Font1.Style = [fsBold]
  Font2.Charset = DEFAULT_CHARSET
  Font2.Color = clWindowText
  Font2.Height = -11
  Font2.Name = 'Tahoma'
  Font2.Style = [fsItalic]
  Font3.Charset = DEFAULT_CHARSET
  Font3.Color = clWindowText
  Font3.Height = -11
  Font3.Name = 'Tahoma'
  Font3.Style = [fsUnderline]
  Font4.Charset = DEFAULT_CHARSET
  Font4.Color = clWindowText
  Font4.Height = -11
  Font4.Name = 'Tahoma'
  Font4.Style = [fsStrikeOut]
  Font5.Charset = DEFAULT_CHARSET
  Font5.Color = clWindowText
  Font5.Height = -11
  Font5.Name = 'Tahoma'
  Font5.Style = [fsBold, fsItalic, fsUnderline]
  BackgroundColor1 = clSilver
  BackgroundColor2 = clWindow
  BackgroundColor3 = clWindow
  BackgroundColor4 = clWindow
  BackgroundColor5 = clWindow
  Align = alClient
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
```

```
    Font.Height = -11
    Font.Name = 'Tahoma'
    Font.Style = []
    ParentFont = False
    ParentShowHint = False
    ShowHint = True
    TabOrder = 0
  end
```

Beispiel 4 „Meldungen gruppiert nach Gebäude (mit PreOperationNav)“:

Gruppe (erste Meldung)	Gebäude	NGF [m²]	Anzahl Meldungen
M12090001 - "%BIS_ROOT%\BISXDS.exe"	2201 Produktion	111.00	1
M09070002 - neues Notebook ist geliefert worden; bitte rasch m...	A20 Hauptsitz	2'729.68	6
M08040004 - Nottüre in Fluchtgang klemmt	B01 Messeturm	24'916.00	3
M08040002 - Aussen-Jalousie in Teeküche blockiert-- Frau Kon...	B33 Bürogebäude	1'719.47	2
M09080001 - Die Beleuchtung ist immer noch nicht repariert! (m...	M27 Entwicklung	7'974.83	2
Anzahl Gruppen: 5	Gesamt:	37'450.98	14

Quellcode 4 (Formular als Text):

```
object BisGroupedGrid4: TBisGroupedGrid
  Left = 0
  Top = 0
  Width = 660
  Height = 631
  Criteria = <
    item
      FilterNavigation.Strings = (
        'VIA bisP_MessageToSpace')
    end>
  GroupedGridOptions = [ggoAllowDrag, ggoAllowOpen]
  DataLink.DataSource = BisDataSet1
  Columns = <
    item
      Attribute = 'Object_Display_Kontext'
      Operation = gopAttribute
      Id = 'Col_15'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopRowCount
      SummaryText = '='#39'Anzahl Gruppen: '#39' + FORMAT(SummaryValue,
'#39'%d'#39')'
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Gruppe (erste Meldung)'
    end
    item
      Attribute = 'Object_Display_kontext'
      FilterNavigation.Strings = (
        'VIA bisP_MessageToSpace ')
      Operation = gopAttribute
      Id = 'Col_5'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopText
      SummaryText = 'Gesamt:'
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
```

```
      Caption = 'Geb'#228'ude'
    end
    item
      Attribute = 'bisB_NetGroundArea'
      FilterNavigation.Strings = (
        '')
      Operation = gopAttributeSum
      PreOperationNav.Strings = (
        'VIA bisP_MessageToSpace')
      Id = 'Col_3'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopSum
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'NGF [m'#178']'
    end
    item
      Operation = gopCount
      Id = 'Col_2'
      FontIndex = 0
      BackgroundColorIndex = -1
      SummaryOperation = gsopCount
      SummaryFontIndex = 1
      SummaryBackgroundColorIndex = 1
      Caption = 'Anzahl Meldungen'
    end>
  SortedCols = '2;3'
  Font1.Charset = DEFAULT_CHARSET
  Font1.Color = clWindowText
  Font1.Height = -11
  Font1.Name = 'Tahoma'
  Font1.Style = [fsBold]
  Font2.Charset = DEFAULT_CHARSET
  Font2.Color = clWindowText
  Font2.Height = -11
  Font2.Name = 'Tahoma'
  Font2.Style = [fsItalic]
  Font3.Charset = DEFAULT_CHARSET
  Font3.Color = clWindowText
  Font3.Height = -11
  Font3.Name = 'Tahoma'
  Font3.Style = [fsUnderline]
  Font4.Charset = DEFAULT_CHARSET
  Font4.Color = clWindowText
  Font4.Height = -11
  Font4.Name = 'Tahoma'
  Font4.Style = [fsStrikeOut]
  Font5.Charset = DEFAULT_CHARSET
  Font5.Color = clWindowText
  Font5.Height = -11
  Font5.Name = 'Tahoma'
  Font5.Style = [fsBold, fsItalic, fsUnderline]
  BackgroundColor1 = clSilver
  BackgroundColor2 = clWindow
  BackgroundColor3 = clWindow
  BackgroundColor4 = clWindow
  BackgroundColor5 = clWindow
  Align = alClient
  Font.Charset = DEFAULT_CHARSET
  Font.Color = clWindowText
  Font.Height = -11
```

```
  Font.Name = 'Tahoma'
  Font.Style = []
  ParentFont = False
  ParentShowHint = False
  ShowHint = True
  TabOrder = 0
end
```

4.3.40 BisDataSource

Platzhalter für ein Datenbankobjekt. Der Platzhalter wird beim Aufstarten des Formulars oder durch eine Komponente vom Typ *BisObjectPropagator* mit einem Datenbankobjekt verbunden und teilt das allen abhängigen Komponenten (Ein-/Ausgabefelder und Listen) mit.

Eigenschaften

CatchNotify	Wenn gesetzt, dann reagiert das Formular auf Änderungen am Objekt (z.B. in einer Liste) und führt diese nach.
ShowInputRequiredMessage	Zeigt beim Speichern die Standardmeldung "*Fehlende Eingabe in einem Pflichtfeld*", wenn mind. ein Pflichtfeld leer ist.
LockObject	Wenn gesetzt, dann wird das Objekt der DataSource als gesperrt markiert und kann von anderen Benutzern nur noch schreibgeschützt geöffnet werden. *Wichtig*: Es wird vorausgesetzt, dass der Objektklasse des Objekts das Attribut "bisB_ObjectLock" zugewiesen worden ist.
LinkedDataSource	Über `LinkedDataSource` kann eine Kette von Datasources aufgebaut werden. Alle Datasources der Kette führen alle Operationen der ersten DataSource der Kette auch aus. Das Objekt wird jeweils aus der ersten DataSource übernommen. Anwendung: Z. Bsp. Festlegung des ReadOnly-Zustandes für unterschiedliche Attribute/Assoziationen/Komponenten über die Eigenschaft `WriteAllowedNav`.

Ereignisse

OnOperate	Das Ereignis `OnOperate` wird ausgeführt *nachdem* die DataSource (und alle gelinkten DataSources) eine Operation ausgeführt hat. Vorsicht: Gewisse Operationen implizieren eine laufende Transaktion!
OnStoreFinished	Das Ereignis `OnStoreFinished` wird nach dem Speichern und `OnOperate(bdoStoreData)` ausgeführt. Es ist keine Transaktion am Laufen!

4.3.41 BisObjectPropagator

Unsichtbare Komponente. Die der Weitergabe von Selektionsereignissen an Komponenten vom Typ *BisDataSource*.

Spezifische Eigenschaften

CatchNotify	Typ: Boolean Wenn gesetzt, dann wird eine Selektionsänderung (TBisSelectNotification) weitergegeben.
FilterIncl	Typ: Boolean

	Wenn die Eigenschaft *FilterIncl* gesetzt ist, werden nur Datenbankobjekte deren Objektklassen in *FilterObjectTypes* aufgeführt ist, weiter propagiert, sonst werden genau diese Datenbankobjekte von der Propagierung ausgeschlossen.
FilterObjectTypes	Typ: Liste von Text Liste von Namen von Objektklassen, die als Filter dienen sollen. Siehe auch *FilterIncl*.
ModifiedAction	Typ: Aufzählung Die Eigenschaft *ModifiedAction* bestimmt, was einem Wechsel des Objektes geschehen soll, wenn am Objekte bereits Änderungen stattgefunden haben. Werte: macAskStore es wird nachgefragt, ob die Änderungen in die Datenbank gespeichert werden sollen macNone die Änderungen werden nicht in die Datenbank gespeichert macStore die Änderungen werden ohne Nachfrage in die Datenbank gespeichert
PropagateNoObject	Typ: Boolean Wenn die Eigenschaft *PropagateNoObject* gesetzt ist, wird wenn in der Ursprungsliste die Selektion leer ist ein Null-Objekt propagiert
PropagateStore	Typ: Boolean Wenn die Eigenschaft *PropagateStore* gesetzt ist, wird wenn in der Ursprungsliste ein Speicher ausgelöst wird, das Speichern weiter propagiert.
Target	Typ: Auswahl von *BisDataSource*-Komponenten / BisObject..Propagator Die Target-*BisDataSource* ändert ihr Datenbankobjekt entsprechend dem propagierten Objekt. Alle abhängigen Komponenten werden nachgeführt.

4.3.42 BisObjectToolPropagator

Unsichtbare Komponente. Die der Weitergabe von Selektionsereignissen an Komponenten vom Typ *BisDataSource*, oder andere propagator (Verkettung)

Spezifische Eigenschaften

FilterNavigation	Bedingung ob der propagator ausgelöst werden soll (Anzahl resultierender Objekte > 0). Die Objekte werden nicht durch diese Navigation bestimmt.

4.4 Palettenseite 'Frames'

4.4.1 BisRemovalEdit

4.5 Palettenseite 'Zusätzlich'

4.5.1 BisDynSchemaLookupBox

4.5.2 ImageList

4.5.3 HTMLParser

4.5.4 TrackBar

4.5.5 Timer

4.5.6 Animate

4.5.7 BisDynSimpleExpressionParser

Unsichtbare Komponente. Ermöglicht Syntax-Highlighting und Syntax-Checking von einfachen Expressions. Bsp.: (var1 > 0) and (var2 < 3). Diese Komponente wird mit einer BisDreamMemoLinked mit Hilfe der Property „SyntaxParser" verknüpft.

OnCheckIdentifier	Callback wird aufgerufen wenn beim Checken der Syntax ein Identifier gefunden wird. Der Rückgabewert (Boolean) bestimmt, ob der Identifier gültig ist.
OnGetColorData	TokenStr enthält z.B. den Identifier. An der jeweiligen BisDreamMemoLinked wird die Property Textstyles definiert. Die Property „ColorData" in OnGetColorData enthält für JEDES Zeichen den Textstyle-Index als Char. Soll z.B. der Textstyle mit dem Index 7 für den Identifier „procedure" verwendet werden, so sieht „ColorData" folgendermassen aus: ColorData := #7#7#7#7#7#7#7#7#7 //9 mal weil 9 Zeichen

4.5.8 BisDynWebServiceCalls

Unsichtbare Komponente. WebService-Aufrufe werden an dieser Komponente implementiert und aus dem Skripting heraus verwendet. Die Proxy Einstellungen sind ab Version 5.0.3 implementiert.

ServiceURL	Typ: String URL des aufzurufenden Webservices.
ServiceResponse	Typ: TStringList Enthält die Antwort eines Webservice-Aufrufs in relevante Blöcke aufgeteilt. Die Länge der String Liste und Aufteilung ist für jede Funktion unterschiedlich (siehe TBisDynamicWebServiceCalls).
CallFailed	Typ: Boolean True wenn der Aufruf ein Fehler zurückmeldet.
LastError	Typ: String Enthält die Fehlermeldung wenn CallFailed = true.

Mode	Proxy Handling. - wam_ProxyWieIE: Einstellungen vom IE werden übernommen - wam_KeinProxy: kein Proxy verwenden. - wam_AngegebeneProxy: die hier eingegebenen Proxy Einstellungen werden verwendet. - wam_ProxIEsonstAngegeben: wenn es Einstellungen am IE gibt werden diese verwendet, sonst die hier angegebenen.	
ProxyServer	zB. '127.0.0.1'	vgl. Mode = wam_AngegebeneProxy
ProxyPort	zB. 8888	vgl. Mode = wam_AngegebeneProxy

5 Scripting

Die in den Skripts verwendete Sprache ist (Object-)Pascal bzw. Delphi-Script. Auf Eigenschaften und Methoden von Objekten wird über eine punktierte Schreibweise zugegriffen: <Objektname>.<Eigenschaft> bzw. <Objektname>.<Prozedur>.

Ein Beispiel:

```
BisEditLinked1.Text := 'Hallo';
BisComboboxLinked2.AcceptPendingValues;
```

Beim Scripting kann auf (fast) alle Eigenschaften der Komponenten zugegriffen werden, welche im Objektinspektor sichtbar sind. Zusätzliche Methoden und Eigenschaften, die im Scripting verwendet werden können, sind anschliessend zusammengestellt.

5.1 Zusätzliche Units einbinden (ab v4.11.1)

Um Funktionen und Prozeduren auslagern und aus verschiedenen Formularen verwenden zu können, müssen in Byron/BIS **Formularskripting Units** erstellt werden. Dies sind Objekte der Klasse **bisB_FormScriptingUnit**.

5.1.1 Einbinden einer zusätzlichen Unit in ein Formular

```
uses
  MyUnit;

procedure Button1Click(Sender: TObject);
begin
  MySpecialProcedure(Sender);
end;
```

5.1.2 Beispiel Unit "MyUnit" (Doc_ID = "MyUnit")

```
procedure MySpecialProcedure(btn: TButton);
begin
  btn.Caption := "My very special caption";
end;
```

5.1.3 Callbacks aus Units verwenden (ab v4.11.2)

Soll aus einer Unit eine eigene Prozedur (Callback) verwendet werden, so kann der Callback über die Komponente `TScriptNotifier` (Registerseite „Zusätzlich“) realisiert werden. Die (optionale) Parameterübergabe kann über die `property Params` (TVariantList) oder die `property Tag` (Integer) erfolgen.

Vorgehen:

- `TScriptNotifier` auf Formular platzieren (ScriptNotifier1)
- Callback-Ereignis im ObjectInspector anschliessen (`ScriptNotifier1Notify`)
- Bsp. im aufrufenden Formular:

```
procedure Button1Click(Sender: TObject);
begin
  ScriptNotifier1.Params := NewVariantList;
  ScriptNotifier1.Params[0] := 'Ein Parameter';
  AnyProcedure(ScriptNotifier1); // aus anderer Unit
end;
```

```
procedure ScriptNotifier1Notify(Sender: TObject);
begin
  ShowMessage(ScriptNotifier1.Params[1]);
end;
```

- Bsp. in der Unit

```
procedure AnyProcedure(notifier: TScriptNotifier);
begin
  notifier.Params[1] := DoAnyThing(notifier.Params[0]);

  notifier.Notify; // Callback auslösen
end;
```

5.1.4 Grenzen / Workarounds

- Globale TVariantList in Units können durch andere Units / Formular nicht verändert werden. Beim Assign (z.B. `varLst[0] := "mein Wert"`) erscheint die Fehlermeldung "Ungültiges Argument".
  - Anstelle von TVariantLists sollten Arrays verwendet werden. Damit können die Uses umgehen. Ansonsten kann die VariantList einer lokalen Variable zugewiesen und diese verwendet werden. Beispiel: `myVarLst := varLst; myVarLst[0] := "mein Wert";`

5.2 Zusätzliche Eigenschaften und Methoden der Komponenten der Palettenseite 'BIS'

5.2.1 TFormInformation

```
procedure FreeMemory;
function.ReplaceParameter(theExpr : string) : string;
// ersetzt Parameter in % (sowohl Datenbank Parameter wie auch Umgebungsvariablen
property CurrentUser: TBisObjectProperty read;
property GlobalParameter[name: String]: String read; write;
// write: ab v4.11.4
//        schreibt nur Db-Parameter (gelesen werden User- und Db-Parameter)
//        darf nur in einer Updatetransaktion gesetzt werden
//        wird ein leerer Text zugewiesen, dann wird der Parameter gelöscht
property ExternalSelection: TBisObjectProperty read, write;
```

5.2.2 TBisEditLinked

```
procedure Clear;
property Modified: Boolean read, write;
function Validate(messages: Boolean) : Boolean;
property Empty: Boolean read;
procedure ShowValid(value: Boolean);
```

5.2.3 TBisMemoLinked

```
procedure Clear;
property Modified: Boolean read, write;
procedure SetCaret(toPos: Integer);
procedure SetTabStops(theTabs: array of Integer; inPixel: Boolean = true);
property Text: String read, write;
```

5.2.4 TBisAdvMemoLinked

```
procedure InsertText(text: String);
  //Fügt Text beim Cursor ein. Falls Selektion vorhanden wird diese überschrieben
```

5.2.5 TBisDreamMemoLinked

```
property SelText: String read, write;
  //Text kann beim Cursor eingefügt werden
```

5.2.6 TBisComboboxLinked

```
procedure AcceptPendingValue;
procedure ShowValid(value: Boolean);
property ItemIndex: Integer read, write;
  // '-1' wenn nicht gesetzt
property Modified: Boolean read, write;
property EnumValue: Variant read, write;
  // Enthält eigentlich einen Integerwert (null wenn nicht gesetzt);
  // setzen von null funktioniert nicht. Stattdessen ItemIndex := -1; verwenden
```

5.2.7 TBisRadioGroupLinked

```
property ItemIndex: Integer read, write;
  // '-1' wenn nicht gesetzt
property Modified: Boolean read, write;
property EnumValue: Variant read, write;
  // Enthält eigentlich einen Integerwert (null wenn nicht gesetzt);
```

5.2.8 TBisCheckBoxLinked

```
property Modified: Boolean read, write;
```

5.2.9 TBisIntEditLinked

```
procedure Clear;
property Modified: Boolean read, write;
function Validate(messages: Boolean) : Boolean;
property Empty: Boolean read;
procedure ShowValid(value: Boolean);
property Valid: Boolean read;
```

5.2.10 TBisFloatEditLinked

```
procedure Clear;
property Modified: Boolean read, write;
function Validate(messages: Boolean) : Boolean; //overload;
property Empty: Boolean read;
procedure ShowValid(value: Boolean);
property Valid: Boolean read;
```

5.2.11 TBisTimeEditLinked

```
procedure Clear;
property Modified: Boolean read, write;
function Validate(messages: Boolean) : Boolean;
property Empty: Boolean read;
procedure ShowValid(value: Boolean);
property Valid: Boolean read;
```

5.2.12 TBisDateEditBtnLinked

```
procedure Clear;
property Modified: Boolean read, write;
function Validate(messages: Boolean) : Boolean;
property Empty: Boolean read;
procedure ShowValid(value: Boolean);
property Valid: Boolean read;
```

5.2.13 TBisIntervalEditLinked

```
procedure Clear;
property Modified: Boolean read, write;
function Validate(messages: Boolean) : Boolean;
property Empty: Boolean read;
procedure ShowValid(value: Boolean);
property Valid: Boolean read;
```

5.2.14 TBisIntervalEditExLinked

```
procedure Clear;
property Empty: Boolean read;
property Modified: Boolean read, write;
property Valid: Boolean read;
```

5.2.15 TBisTerminEditLinked

```
procedure Clear;
procedure SetDefault(setModified: Boolean);
  // Setzt einen geeigneten Vorgabewert (je nach Style).
  // Wenn setModified = true, dann ist 'Modified' anschliessend gesetzt
property Empty: Boolean read;
property Modified: Boolean read, write;
property Value: TBisTermin read, write;
```

5.2.16 TBisTerminControl

```
// Dient sowohl zum Bearbeiten von Terminen, als auch zum Navigieren
// innerhalb von Serieterminen (nächster, vorheriger, Gruppe, …)
// nur die spezifischen Eigenschaften werden erklärt
property Empty: Boolean read;
property Modified: Boolean read, write;
property Style: TTerminControlStyle read, write default tcsDateTime;
  // bestimmt, wie der Termin dargestellt werden soll
  // tcsDateTime      - Ein Datumsfeld, zwei Zeitfelder
  // tcsTwoDatesTime  - Zwei Datumsfelder, zwei Zeitfelder
  // tcsDate          - Ein Datumsfeld
  // tcsTwoDates      - Zwei Datumsfelder
  // tcsYear          - eine Jahres-Combobox
  // tcsTwoYears      - Zwei Jahres-Comboboxen
property DateStart: TDateTime read, write;
  // enthält das Startdatum
property DateStop: TDateTime read, write;
  // enthält das Enddatum
property SeparatorSize: Integer read, write, default 4;
  // Abstand zwischen den einzelnen Feldern des Controls
property TimeComboSize: Integer read, write, default 55;
  // Breite der Zeitfelder
property Options: TTerminControlOptions read, write;
  // Optionen: tcoEnableNavigation – erlaube die Navigation
property Mode: TTerminControlMode read FMode;
  // Beschreibt den Modus, in welchem sich das Control befindet:
  // tcmShowSimple - wir editieren einen einfachen Termin
  // tcmShowSimpleOfCompound - wir editieren einen einfachen Termin, der Teil
  //                           eines Compounds ist
  // tcmShowCompound - wir editieren einen zusammengesetzten Termin
  // tcmShowSimpleCompound - wir editieren einen zusammengesetzten Termin mit nur
  //                         einem einfachen Termin als Inhalt
property TimeComboResolution: TTimeResolution read, write, default tres15Min;
  // Abstand der Einträge in den Zeitausfahlfeldern.
```

```
property TimeComboStartTime: String read, write, default "6:00";
  // Erster Eintrag in Zeitausfahlfeldern.
property TimeComboEndTime: String read, write, default "18:00";
  // Letzter Eintrag in Zeitausfahlfeldern
property DataLink: TBisDataLinkBase read;
property SimpleTerminType: String read, write;
  // Objekttyp der Elemente eines Serietermins. Wird ein Serietermin bearbeitet
  // dann erzeugt das Control Serieterminelemente dieses Typs
property Propagator: TBisObjectPropagator read, write;
  // Zum Navigieren benötigter Propagator. Beim Navigieren innerhalb eines
  // Serietermins wird primär die Selektion (Brwosing des Formulars) verwendet.
  // Hat dies keinen Effekt, dann wird versucht, über den Propagator das
  // Objekt zu ändern.
  // Wird Normalerweise auf den Propagator des Formulars gesetzt.
```

5.2.17 TBisDataSet

```
// Das Startobjekt kann über die property 'DataSource' manipuliert werden
procedure SetVariable(const aName: String; const aValue: Variant);
property DeepObjects[attribute, pattern: String]: TBisObjectProperty
property Objects[attribute, pattern: String]: TBisObjectProperty
  // Auf attribute und pattern kann verzichtet werden ('')
```

5.2.18 TBisLookupCombobox

```
procedure Clear;
property DbObject: TBisObjectProperty read, write;
property LookupObjects: TBisObjectProperty read, write;
  // Zum Zuweisen der auswählbaren Objekte.
  // Funktioniert nur ohne zugewiesenes BisDataSet
property Modified: Boolean read, write;
property Empty: Boolean read;
```

5.2.19 TBisObjectCombobox

```
property DbObject: TBisObjectProperty read, write;
property Items read, write;
property ItemIndex read, write;
property StartObject: TBisAbstractSource read, write;
property ObjectNav: TStrings read, write;
```

Ab V5.5.0 kann die Komponente aus ohne `CDLReference` genutzt werden. Dazu muss noch `UseAccessRights` auf false gesetzt werden. Bei jedem load wird die `ObjectNav` ausgeführt, mit `StartObject` oder wenn nicht vorhanden `DataLink.DataSource` als Eingangsobjekt. Mit `DataLink.InputRequired` wird bestimmt, ob noch zusätzlich ein ungesetzt Eintrag erstellt wird.

5.2.20 TBisDynSimpleGridII

```
procedure Clear(markAsRemoved: Boolean);
procedure Load(theObjects: TBisObjectProperty; markAsNew: Boolean);
procedure Remove(theObjects: TBisObjectProperty; markAsRemoved: Boolean);
procedure Delete(theObjects: TBisObjectProperty; markAsDeleted: Boolean);
procedure BeginUpdate; //ab Version 4.10.3
  //Beim Load wird nicht Sort, AdjustWidths und SelectFirstRow gemacht.
  //Erst sobald EndUpdate aufgerufen wird. Achtung: Immer EndUpdate aufrufen!
procedure EndUpdate; //ab Version 4.10.3
  //Ruft Sort, AdjustWidths, SelectFirstRow und ggf. FilterRows auf nachdem
  //mehrere Objekte einzeln in das Grid geladen wurden.
procedure AdjustWidths;
procedure AdjustHeights; //ab Version 5.4.7
procedure SelectAll;
```

```
procedure DeselectAll;
  // Setzt die Selektion des Grids zurück
procedure SetBackgroundColorIndex(rowId: String; colId: String; index: Integer);
  // ab Version 4.9.1
  //
  // Setzt den Hintergrundfarbindex (BackgroundColorIndex) für
  // Zellen, Spalten, Zeilen oder für das ganze Gird:
  //
  // rowId colId  index für
  // ----------------------
  // <id>  <id>   Zelle
  // <id>  leer   Zeile
  // leer  <id>   Spalte
  // leer  leer   Grid
  //
  // Der index (BackgroundColorIndex) muss zwischen 1 und 5 liegen.
  // Dazu können bis zu fünf Hintergrundfarben am Grid definiert werden:
  // BackgroundColor1 : TColor
  // BackgroundColor2 : TColor
  // BackgroundColor3 : TColor
  // BackgroundColor4 : TColor
  // BackgroundColor5 : TColor
procedure SetFokus(rowId     : String;  colId: String;
                   directEdit: Boolean; selectRow: Boolean)
  // ab Version 4.9.9 - Achtung: Fokus mit "k" geschrieben!
  //
  // Setzt den Fokus auf eine bestimmte Grid-Zelle.
  // directEdit: Falls nicht sowieso goAlwaysEdit gesetzt ist, wird Zelle direkt
                 in den Editier-Modus versetzt.
  // selectRow : Die fokussierte Zeile erhält auch die Selektion
procedure SetFontIndex(const theRowId: String; theFontIndex: Integer);
  // Setzt die Font (0…5) der entsprechenden Zeile
procedure ScrollInView(const theColId, theRowId: String);
  // Scrollt die entsprechende Grid-Zelle in den sichtbaren Bereich
function CreateRow : String;
  // Erzeugt eine neue Zeile im Grid und liefert die Id der Zeile
function RowOfObject(const theObject: TBisObjectProperty) : String;
  // Liefert die Id der Zeile des Objekts
function IndexOfRow(const theId : string) : Integer;
  // Liefert den Index einer Row zur ID
  // ab 4.8.0
function IndexOfCol(const theId : string) : Integer;
  // Liefert den Index einer Col zur ID
  // ab 4.8.0
function ValueOf (row, col : Integer) : Variant;
  // Liefert den Inhalt der Zelle (row, col)
  // ab 4.8.0 eine performante Alternative zur property Cells
  // Bsp. c := IndexOfCol('Col_1');
  // for i := 0 to grid.RowCount -1 do begin
  //   v := grid.ValueOf(i, c);
  //   …
  // end; //for
procedure AddCol(
              theId: AnsiString;
              theType: TTempColDataType;
              theCaption: AnsiString;
              posBefore: AnsiString;
              readOnly: boolean;
              autoFilter: boolean);
TTempColDataType = (tdtBoolean, tdtInteger, tdtEnum, tdtFloat, tdtText, tdtDate,
                    tdtTimeInterval, tdtColor)
  // fügt eine temporäre Spalte hinzu, erst ab Byron/BIS v4.9.1
```

```
procedure DeleteCol(theId: AnsiString);
  // löscht die angegebene Spalte, erst ab Byron/BIS v4.9.1

procedure SetCol(
              theId: AnsiString;
              theCaption: AnsiString;
              posBefore: AnsiString;
              readOnly: boolean;
              autoFilter: boolean);
  // ändert die Eigenschaften einer Spalte, erst ab Byron/BIS v4.9.6
procedure SetRowId(
              theId: AnsiString;
              theNewId: AnsiString);
  // ändert die Id der Row <theId> zu <theNewId>. Ab Byron/BIS v4.10.2.
  // Wird z.B. dann verwendet, wenn mit Load mehrere Objekte in das Grid
  // geladen werden, die dann im ersten Moment alle dieselbe Id besitzen.
procedure SelectRow(
              theIndexOfRow: Integer;
              deselectAllFirst: Boolean);
  // Selektiert die Row bei Index <theIndexOfRow>. Ab Byron/BIS v4.10.2.
  // <deselectAllFirst> gibt an, ob zuerst alle Rows deselektiert werden sollen.


function SetAutoFilter(
           theColId: AnsiString;
           theType: TByAutoFilterType;
           theOperation1: TByAutoFilterOperation;
           theOperation2: TByAutoFilterOperation;
           theValue1: AnsiString;
           theValue2: AnsiString;
           theAndFilter: Boolean;
           theFilterFormat: AnsiString) : Boolean;
TByAutoFilterType = (aftAll, aftUserDefined, aftEmptyValues, aftNotEmptyValues,
                     aftValue)
TByAutoFilterOperation = (afoNone, afoEqual, afoNotEqual, afoIsGreaterThan,
                          afoIsGreaterOrEqual, afoIsSmallerThan,
                          afoIsSmallerOrEqual, afoBeginWith, afoNotBeginWith,
                          afoEndWith, afoNotEndWith, afoContains, afoNotContains)
  // Setzt den aktuellen AutoFilter für die Spalte <theColId>.
  // <theColId> definiert die Spalten-ID.
  // <theType> definiert den AutoFilter-Typ.
  //   aftAll           = alles anzeigen (kein AutoFilter gesetzt)
  //   aftUserDefined   = benutzerdefinierter AutoFilter
  //   aftEmptyValues   = nur leere Werte anzeigen
  //   aftNotEmptyValues= nur nichtleere Werte anzeigen
  //   aftValue         = AutoFilter für einen bestimmen Wert
  // <theOperation1> definiert die AutoFilter-Operation für den Wert <theValue1>.
  //   afoNone                = keine Operation
  //   afoEqual               = entspricht dem Wert
  //   afoNotEqual            = entspricht nicht dem Wert
  //   afoIsGreaterThan       = ist grosser als der Wert
  //   afoIsGreaterOrEqual    = ist grosser oder gleich als der Wert
  //   afoIsSmallerThan       = ist kleiner als der Wert
  //   afoIsSmallerOrEqual    = ist kleiner oder gleich als der Wert
  //   afoBeginWith           = beginnt mit dem Wert
  //   afoNotBeginWith        = beginnt nicht mit dem Wert
  //   afoEndWith             = endet mit dem Wert
  //   afoNotEndWith          = endet nicht mit dem Wert
  //   afoContains            = enthält den Wert
  //   afoNotContains         = enthält nicht den Wert
  // <theValue1> definiert den AutoFilter-Wert.
  //   (wird nur bei aftUserDefined und aftValue berücksichtigt)
```

```
  // <theOperation2> definiert die zweite AutoFilter-Operation
  //   für den Wert <theValue2>.
  //   (wird nur bei aftUserDefined berücksichtigt)
  // <theValue2> definiert den zweiten AutoFilter-Wert.
  //   (wird nur bei aftUserDefined berücksichtigt)
  // <theAndFilter> definiert die AutoFilter-Bedingung.
  //   TRUE = Filter1 und Filter2, FALSE = Filter1 oder Filter2
  //   (wird nur bei aftUserDefined berücksichtigt)
  // <theFilterFormat> definiert das AutoFilter-Format
  //   zur Anzeige der AutoFilter-Werte.
  // <theValueN> kann reguläre Ausdrücke (RegExpr) enthalten, wenn
  // <theType> = aftUserDefined und <theOperationN> = afoEqual oder afoNotEqual.
  // Der Rückgabewert ist false, wenn die Spalte nicht existiert.
  // Ab Byron/BIS v4.10.3.
function GetAutoFilter(
           theColId: AnsiString;
           var theType: TByAutoFilterType;
           var theOperation1: TByAutoFilterOperation;
           var theOperation2: TByAutoFilterOperation;
           var theValue1: AnsiString;
           var theValue2: AnsiString;
           var theAndFilter: Boolean;
           var theFilterFormat: AnsiString) : Boolean;
  // Liefert den aktuellen AutoFilter für die Spalte <theColId>.
  // Der Rückgabewert ist false, wenn die Spalte nicht existiert.
  // Ab Byron/BIS v4.10.3.
procedure FilterRows;
  // Führt den AutoFilter für das Grid aus.
  // Wenn bei aktivem AutoFilter Werte in gefilterten Zeilen per Scripting
  // geändert werden, muss FilterRows aufgerufen werden, damit der AutoFilter
  // für die Zeilen erneut ausgeführt wird.
  // Ab Byron/BIS v4.10.3.
procedure ResetAutoFilter;
  // Entfernt den AutoFilter aller Spalten und aktualisiert das Grid.
  // Ab Byron/BIS v4.10.3.
procedure SetRowsReadOnly(const theObjects: TBisObjectProperty;
                          const rowReadOnly: Boolean);
  // Setzt ReadOnly für die Zeilen der Objekte (ab v4.10.5)
procedure Export(const fileName: String;
                 const separator: String;
                 const inclHeader: Boolean;
                 const onlySelected : Boolean);
  // Exportiert alle Zeilen des Grids in die angegebene Datei.
  // Wenn separator leer ist, wird TAB als Trennzeichen verwendet.
  // Wenn separator nicht leer ist, wird das erste Zeichen verwendet.
  // Wenn inclHeader true ist, werden die Spaltenüberschriften auch exportiert.
  // Wenn onlySelected true ist, werden die nur die selektierten Zeilen
  // exportiert.
  // Ab Byron/BIS v4.10.6.
procedure Print(formCaption: String;
                preview: Boolean;
                layoutDef: String;
                toolObj: TBisObjectProperty;
                containerObj: TBisObjectProperty);
  // Druckt das Grid mit allen Zeilen aus.
  // formCation bestimmt den Titel der Druckvorschau, wenn preview = true ist.
  // Wenn preview = true ist, wird die Druckvorschau (Seitenansicht) angezeigt.
  // layoutDef enthält die Definition des Layouts.
  // (siehe auch <PageDecoration> in der ToolKonfiguration-Dokumentation)
  // toolObj bestimmt das Tool-Objekt, welches für die Auswertung
  // der Tool-Parameter in der Layoutdefinition verwendet wird.
  // containerObj bestimmt das Container-Objekt, welches für die Auswertung
```

```
    // der Container-Parameter in der Layoutdefinition verwendet wird.
    // Ab Byron/BIS v4.10.5.

  property SelectedObjects: TBisObjectProperty read, write;
  property AllObjects: TBisObjectProperty read;
  property RowCount: Integer read;
    // Liefert die Anzahl der Zeilen
  property ColCount: Integer read;
    // Liefert die Anzahl der Spalten
    // Ab Byron/BIS v4.10.6.
  property SelectedCount: Integer read;
  property Rows[index: Integer]: String read;
    // Liefert Id der Zeile, index beginnt mit 0
  property Cols[index: Integer]: String read;
    // Liefert Id der Spalte, index beginnt mit 0
    // Ab Byron/BIS v4.10.6.
  property Cells[row, col: String]: Variant read, write;
    // Liefert oder setzt den Inhalt der Zelle
  property ObjectOfRow[idx: Integer]: TBisObjectProperty read;
  property SelectedRow[index: Integer]: String read;
    // Liefert Id der selektierten Zeile
  property ColVisible[id: String]: Boolean read, write
  property RowVisible[id: String]: Boolean read, write
    // Sichtbarkeit von Spalten und Zeilen
  property RowReadOnly[id: String]: Boolean read, write
    // ReadOnly von Zeilen (ab v4.10.5)
  property ColReadOnly[id: String]: Boolean read, write
    // ReadOnly von Spalten (ab v4.10.6)
  property SortedCols: String read, write
    // Ein String bestehend aus Spaltennummer(n) zur Sortierung
    // Wird diese Eigenschaft gesetzt, dann wird das Grid sortiert.
    // Format: [-]<Spalten-Nr.>[;[-]<Spalten-Nr.>]
    // Anmerkungen:
    // - Trennzeichen für mehrere Spalten ist das Semikolon (;)
    // - Eine positive Spalten-Nr. sortiert die Spalte aufsteigend
    // - Eine negative Spalten-Nr. sortiert die Spalte absteigend
    // Beispiele 1:
    //   keine Sortierung: 0
    //   1. Spalte aufsteigend:  1
    //   2. Spalte absteigend:   -2
    //   1. Spalte aufsteigend, 2. Spalte absteigend:      1;-2
    //   2. Spalte absteigend, 1. Spalte aufsteigend:      -2;1

  // folgende Eigenschaften sind obsolet:
  property SortCol: Integer read, write

  property OnStatusTextUpdate: TStatusTextUpdateEvent; (ab v4.11.7)
    // Wird aufgerufen, wenn eine dem Grid zugehörige Statuszeile aktualisiert
    // werden muss.
```

5.2.21 TBisGroupedGrid

```
  property ShowSummaryRow: Boolean read;
    // true, wenn mindestens eine aktive Spalte einen anderen Wert
    // als gsopNone in SummaryOperation enthält
  property HasCriteria: Boolean read;
    // true, wenn mindestens ein aktives (Active=true) Gruppenkriterium existiert
  property CriteriaItems[index: Integer]: TBisGroupedGridCriterion read;
    // Liefert das entsprechende Gruppenkriterium (index beginnt mit 0)
  procedure Load(theObjects: TBisObjectProperty; markAsNew: Boolean);
    // Lädt das Grid,  merkt sich die (Eingangs-)Objekte und
```

```
  // führt die Grupperung durch
  // Die folgenden internen Load-Parameter werden wie folgt gesetzt:
  //    useAccessRight  : Boolean            = false
  //    withState       : TRowState          = rwLoaded (wenn markAsNew=false)
  //                                         = rwNew    (wenn markAsNew=true)
  //    checkDuplicates : Boolean            = true
  //    filter          : TDbObjectClassList = nil
  //    incl            : Boolean            = true
procedure Reload;
  // Löscht alle Zeilen und führt die Grupperung anhand der
  // gemerkten (Eingangs-)Objekte neu durch
```

5.2.22 TBisSerie (Property aus TBisChartII)

```
property CriteriaItems[index: Integer]: TBisGroupedGridCriterion read;
  // Liefert das entsprechende Gruppenkriterium (index beginnt mit 0)
  // erst ab v5.5.5
```

5.2.23 BisCrosstable (ab v4.9.6)

```
property Rows[index: Integer]: String read;
  // Liefert Id der Zeile, index beginnt mit 0 (ab v4.10.6)
property Cols[index: Integer]: String read;
  // Liefert Id der Spalte, index beginnt mit 0 (ab v4.10.6)
property Cells[row, col: integer]: Variant read, write;
  // Liefert oder setzt den Inhalt der Zelle (ab v4.10.6)
property RowReadOnly[row: Integer]: Boolean read, write
  // ReadOnly von Zeilen (ab v4.10.6)
property ColReadOnly[col: Integer]: Boolean read, write
  // ReadOnly von Spalten (ab v4.10.6)
procedure SetRowsReadOnly(const theObjects: TBisObjectProperty;
                          const rowReadOnly: Boolean);
  // Setzt ReadOnly für die Zeilen der Objekte (ab v4.10.6)
procedure SetColsReadOnly(const theObjects: TBisObjectProperty;
                          const colReadOnly: Boolean);
  // Setzt ReadOnly für die Spalten der Objekte (ab v4.10.6)
procedure SetBackgroundColorIndex(row: integer; col: integer; index: Integer);
  // ab Version 4.10.6
  //
  // Setzt den Hintergrundfarbindex (BackgroundColorIndex) für
  // Zellen, Spalten, Zeilen oder für die ganze CrossTable:
  //
  // row          col             Bedeutung
  // ---------------------------------------
  // <row-num>    <col-num>       Zelle
  // <row-num>    -1              Zeile
  // -1           <col-num>       Spalte
  // -1           -1              Grid
  //
  // Dazu können bis zu fünf Hintergrundfarben an der CrossTable definiert werden:
  // BackgroundColor1 : TColor
  // BackgroundColor2 : TColor
  // BackgroundColor3 : TColor
  // BackgroundColor4 : TColor
  // BackgroundColor5 : TColor
procedure Export(const fileName: String;
                 const separator: String;
                 const inclHeader: Boolean;
                 const onlySelected : Boolean);
  // Exportiert alle Zeilen der CrossTable in die angegebene Datei.
```

```
    // Wenn separator leer ist, wird TAB als Trennzeichen verwendet.
    // Wenn separator nicht leer ist, wird das erste Zeichen verwendet.
    // Wenn inclHeader true ist, werden die Spaltenüberschriften auch exportiert.
    // Wenn onlySelected true ist, werden die nur die selektierten Zeilen
    // exportiert.
    // Ab Byron/BIS v4.10.8.

  property SelectedObjects: TBisObjectProperty read, write;
  property SelectedCount: Integer read;
```

5.2.24 BisObjectCrosstable (ab v4.7.8)

```
  property Rows[index: Integer]: String read;
    // Liefert Id der Zeile, index beginnt mit 0 (ab v4.10.6)
  property Cols[index: Integer]: String read;
    // Liefert Id der Spalte, index beginnt mit 0 (ab v4.10.6)
  property Cells[row, col: integer]: Variant read, write;
    // Liefert oder setzt den Inhalt der Zelle (ab v4.10.6)
  property RowReadOnly[row: Integer]: Boolean read, write
    // ReadOnly von Zeilen (ab v4.10.6)
  property ColReadOnly[col: Integer]: Boolean read, write
    // ReadOnly von Spalten (ab v4.10.6)
  procedure SetRowsReadOnly(const theObjects: TBisObjectProperty;
                            const rowReadOnly: Boolean);
    // Setzt ReadOnly für die Zeilen der Objekte (ab v4.10.6)
  procedure SetColsReadOnly(const theObjects: TBisObjectProperty;
                            const colReadOnly: Boolean);
    // Setzt ReadOnly für die Spalten der Objekte (ab v4.10.6)
  procedure SetBackgroundColorIndex(row: integer; col: integer; index: Integer);
    // ab Version 4.10.6
    //
    // Setzt den Hintergrundfarbindex (BackgroundColorIndex) für
    // Zellen, Spalten, Zeilen oder für die ganze CrossTable:
    //
    // row          col             Bedeutung
    // ---------------------------------------
    // <row-num>    <col-num>       Zelle
    // <row-num>    -1              Zeile
    // -1           <col-num>       Spalte
    // -1           -1              Grid
    //
    // Dazu können bis zu fünf Hintergrundfarben an der CrossTable definiert werden:
    // BackgroundColor1 : TColor
    // BackgroundColor2 : TColor
    // BackgroundColor3 : TColor
    // BackgroundColor4 : TColor
    // BackgroundColor5 : TColor
  procedure Export(const fileName: String;
                   const separator: String;
                   const inclHeader: Boolean;
                   const onlySelected : Boolean);
    // Exportiert alle Zeilen der CrossTable in die angegebene Datei.
    // Wenn separator leer ist, wird TAB als Trennzeichen verwendet.
    // Wenn separator nicht leer ist, wird das erste Zeichen verwendet.
    // Wenn inclHeader true ist, werden die Spaltenüberschriften auch exportiert.
    // Wenn onlySelected true ist, werden die nur die selektierten Zeilen
    // exportiert.
    // Ab Byron/BIS v4.10.8.

  property SelectedObjects: TBisObjectProperty read, write;
  property SelectedCount: Integer read;
```

5.2.25 TBisDynSimpleTree

```
procedure Clear;
procedure ExpandLevels(levels: integer); (ab Byron/BIS v4.8.0)
property RootObjects: TBisObjectProperty read, write;
property SelectedObjects: TBisObjectProperty read, write;
property CheckedObjects: TBisObjectProperty read, write; (ab Byron/BIS v4.9.6)
  // wenn property CheckBoxes entsprechend gesetzt ist
```

5.2.26 TBisDataSource

```
procedure Operate(theOp: TBisDataOperation);
property DbObject: TBisObjectProperty read, write;
property Container: TBisObjectProperty read, write;
property ObjectType: String read, write;
  // Die Objektklasse wird beim Erzeugen von neuen Objekten benötigt
property Modified: Boolean read;
property OnInputRequired: TNotifyEvent read, write;
  // Wird aufgerufen, wenn beim Speichern mind. ein Pflichtfeld leer ist
property OnIsValid : TOnIsValid read, write;
  // Wird vor dem Speichern aufgerufen und gibt an, ob alle Eingabe-
  // felder/Komponenten einen gültigen Inhalt haben.
  // Falls nicht, wird furch BisDataSource KEINE Meldung angezeigt. Bsp.:
  // function BisDataSourceIsValid(sender: TObject): Boolean;
  // var
  //   u : TBisObjectProperty;
  // begin
  //   u := FormInformation.CurrentUser;
  //   result := u.Expression('REGEXP("' + edit.Text + '", "[0-9]{4}")') = true;
  //                                  // = true ist leider nötig – siehe Expression
  //   if not result then ShowMessage(Eingabe ungültig');
  // end;
property OnModified: TOnModified read, write;
  // Callback, mit welchem TBisDataSource.Modifed verändert werden kann.
  // der var-Parameter isModified des Callbacks wird durch die BisDataSource
  // anhand der Eingabefelder/Komponenten vorbelegt und kann auf den gewünschten
  // Wert geändert werden. Bsp.:
  // procedure BisDataSourceModified(sender: TObject; var isModified: Boolean);
  // begin
  //   isModified := isModified or edKennzeichen.Modified;
  // end;
property OnOperate: TOnOperate read, write;
property OnStoreFinished: TOnStoreFinished read, write;
```

5.2.27 TBisObjectPropagator

```
procedure Propagate(theObjects: TBisObjectProperty);
```

5.2.28 TBisGanttLinked

```
procedure Print;
  // Ruft den Drucken-Dialog des Gantts auf
```

5.2.29 TBisCellGrid

```
procedure SetCellGridOption(const cgOption: TCellGridOption;
                            const Value: Boolean);
procedure CreateRegion;
procedure CreateRegions;
```

```
procedure DeleteRegion;
procedure DeleteAssoziation;
procedure Clear;
procedure Load(theObject: TDbBaseObject);
procedure Store(theObject: TDbBaseObject);
property Modified: Boolean read, write;
```

5.2.30 TBisDynReportViewer

```
procedure Execute(
            const repObj : TBisObjectProperty;
            const sel    : TBisObjectProperty);
procedure Print;
  // ruft den Druckdialog auf und druckt den Bericht
procedure PrintEx(
            const printerName: String;
            const showDialog: Boolean);
  // Druckt einen Bericht direkt auf einen Drucker, ausser wenn
  // showDialog = true ist. Dann wird der Druckendialog angezeigt.
  // Um auf den Standarddrucker zu drucken, printerName leer lassen.
procedure SetParam(
            const id: string;
            const value: variant);

property OnSetParams: TNotifyEvent;
  // Dieses Event wird aufgerufen, wenn der Report zugeordnete Parameter hat.
  // Die Parameter können dann über SetParam gesetzt werden.
```

5.2.31 TBisDynSimpleExpressionParser

```
function CheckSyntax(const strData: String;
                     var   Error  : String) : Boolean;
  // Parst den String in strData und liefert True im Erfolgsfall zurück. Falls
  // False zurückgeliefert wird enthält Error eine Fehlermeldung
```

5.2.32 TBisDynWebBrowser

```
procedure ExecScript(sExpression, sLanguage: string); //nur bis und mit IE 10!
  // Führt ein Script im Browser aus (wie das HTML <script>-Tag).
  // Bsp.: web.ExecScript('alert("hallo welt");', 'JavaScript');
  // ACHTUNG: Erst nach/in BISOnDocumentComplete ausführen!

procedure Run(cmnd: Integer);
  // cmnd ist ein Befehls Code, welcher den entsprechenden Befehl auslöst.
  // Die Codes -1 bis -8 entsprechen den angegebenen Methoden des IwebBrowser2
  // Interface: MSDN-webbrowser2
  // -8: GoSearch
  // -7: GoHome
  // -6: GoForward
  // -5: GoBack
  // -4: Quit
  // -2: Refresh
  // -1: Stop
  // 0 macht verständlicherweise nichts.
  // Positive Werte werden an ExecWB von WebBrowser2 weitergegeben.
  // (vgl. OLECMDID)
```

5.2.33 TBisDynWebServiceCalls

```
function REGIS_TempID(username, password: string) : String;
  // Beschreibung:
  // Generiert bei REG-IS eine Session-ID, welche beim Abruf von Facilities
  // als Login-Kennung verwendet werden kann. Die generierte SessionID wird als
  // Parameter TempID an die URL angehängt. Bsp.:
  // Bsp.: http://www.my.reg-is.de/forms/facilities.aspx?ext=1&FacID=773&TempID=htbpq3410h
  // ServiceResponse-Property:
  // ServiceResponse[0]: mit CallFailed-Property abgedeckt
  // ServiceResponse[1]: Session-ID
  // ServiceResponse[2]: String "Ablaufdatum"
  // ServiceResponse[3]: Ablaufdatum
  // ServiceResponse[4]: Ablaufzeit
```

5.2.34 TBisDynChromiumBrowser (ab v5.5.11)

Eigenschaften

```
property Url: string;
  // legt die angezeigte Url fest, kann auch mit Dateipfaden genutzt werden
property Mode: TBisDynChromiumBrowserMode;
  // legt fest, wie der Browser zum Inhalt kommt
  //   wcmAutomatic – ermittelt den Mode aufgrund der Angaben im Datalink
  //   wcmFixUrl – die Url wird NICHT aus der Datenbank geladen
  //   wcmDBUrl – die Url wird aus dem Attribut im Datalink geladen
  //   wcmDBcontents – das Html wird aus dem Attribut im Datalink geladen
  // Vorgabe: wcmAutomatic
property ExePath: string;
  // legt fest, welches externe Programm gestartet wird
  // Vorgabe = '%BIS_ROOT%SimpleCefSharpHost\SimpleCefSharpHost.exe'
  // ab v5.7.4
property DevTools: boolean;
  // legt fest ob die Chrome-Entwicklertools geöffnet werden können
  // Vorgabe: false
property ShowUrl: boolean;
  // legt fest ob die Adresszeile angezeiugt wird
  // Vorgabe: false
property LocalDownloadForbidden: boolean;
  // legt fest, ob der Browser lokale Downloads verbieten soll
  // Vorgabe: false
property EvaluateVariables: boolean;
  // legt fest, ob Variablen in der Url ausgewertet werden
  // Vorgabe: false

property Html: string;
  // nur mittels Scripting zugreifbar
  // Wird diese Property gesetzt, wird das übergebene Html angezeigt
```

Events

```
property OnLoadComplete: TnotifyEvent
  // Wird aufgerufen, sobald Chromium die Seite geladen hat
property OnCustomCommandReceived: TNotifyEvent
  // Wird aufgerufen, sobald die Komponente einen CustomCommand vom
  // externen Programm erhalten hat.
  // die Daten können jetzt und nur innerhalb von OnCustomCommandReceived mit
  // GetCustomCommandOperation und GetCustomCommandObjects bezogen warden
  // ab v5.7.4
```

Funktionen

```
procedure SendCustomCommand (
            const operation : string;
            objects : TDbBaseObjectList);
  // Schickt einen CustomCommand mit den übergebenen Daten an das
  // externe Programm
  // ab v5.7.4
function GetCustomCommandOperation() : string;
  // Liefert die Operation des empfangenen CustomCommand
  // Kann nur innerhalb von OnCustomCommandReceived verwendet werden
  // ab v5.7.4
function GetCustomCommandObjects() : TDbBaseObjectList;
  // Liefert die Objekte des empfangenen CustomCommand
  // Kann nur innerhalb von OnCustomCommandReceived verwendet warden
  // ab v5.7.4
```

5.2.35 TBisDynOpenDialog

```
function Execute : Boolean;
  // Öffnet einen "Datei-Öffnen"-Dialog und gibt true zurück, wenn eine
  // oder mehrere Dateien ausgewählt wurden oder false, wenn der Benutzer
  // auf "Abbrechen" geklickt hat.

property Files: TStrings;
  // Wenn ofAllowMultiSelect in den Optionen erlaubt wurde, kann hier
  // auf alle ausgewählten Dateien zugegriffen werden. Wird nur eine
  // Datei ausgewählt, kann auch die Property FileName verwendet werden.
```

5.3 Hilfsobjektklassen

5.3.1 TStrings

```
  procedure Clear;
  procedure Add(theLine: String);
  procedure SaveToFile(theFilename: String); // ab v5.8.2
  procedure LoadFromFile(theFilename: String); // ab v5.8.2
  procedure Add(theLine: String);
..function  IndexOf(theString: String): Integer;
    //Index von theString in Liste. Falls nicht gefunden ist das Resultat < 0
  [index: Integer]: String read, write
  property Values[index: string]: String read, write (ab v4.11.7)
```

5.3.2 TBisTermin

```
  // Mit dieser Klasse können Termin-Attribute behandelt werden
  function IsIn(theTime: TDateTime) : Boolean;
  function HasCollision(theTermin: TBisTermin) : Boolean;
  // Iteration durch Termin
  function IterateStart(lo, hi: TDateTime) : Boolean;
  function IterateNext : Boolean;
  property CurrentStart: TDateTime;
  property CurrentStop: TDateTime;
  // Verwendung des Iterators:
  // if termin.IterateStart(myLoDate, myHiDate) then begin
  //   repeat
  //     DoAnyThing(termin.CurrentStart);
  //     DoAnyThing(termin.CurrentStop);
  //   until not termin.IterateNext;
  // end; //if
```

```
// Anfang und Ende des Termins
property Start: TDateTime;
property Stop: TDateTime;
```

5.3.3 TVariantList

```
// Mit dieser Klasse können Werte für verschiedene Datentypen ausgetauscht
// werden. Es Kann auch eine neue Liste erstellt werden mit NewVariantList
procedure Assign(theSource : TVariantList); // ab V4.9.12
//  kopiert alle Eintäge aus Source
property Count: Integer;    // read write  Anzahl Werte
property AsVariant: Variant // readonly    Die ganze Liste als Variant
property Items[index: Integer]: Variant // Liefert die Werte
```

Um einen 2 dimmensionalen Array zu bilden, können die beiden Indices auf einen abgebildet werden:

```
function IndexOf(i, j : integer) : integer;
var
  k : integer;
  res : integer;
begin
  if j = 0 then begin
    res := i * (i+1) DIV 2;
  end else begin
    res := (j+2)*(j+1) DIV 2 - 1;
    for k := 0 to i-1 do begin
      res := res + j + k + 1;
    end;
  end;
  result := res;
end;
```

(0, 0) =	0	(1, 0) =	1	(2, 0) =	3	(3, 0) =	6	(4, 0) =	10	(5, 0) =	15	(6, 0) =	21	(7, 0) =	28
(0, 1) =	2	(1, 1) =	4	(2, 1) =	7	(3, 1) =	11	(4, 1) =	16	(5, 1) =	22	(6, 1) =	29	(7, 1) =	37
(0, 2) =	5	(1, 2) =	8	(2, 2) =	12	(3, 2) =	17	(4, 2) =	23	(5, 2) =	30	(6, 2) =	38	(7, 2) =	47
(0, 3) =	9	(1, 3) =	13	(2, 3) =	18	(3, 3) =	24	(4, 3) =	31	(5, 3) =	39	(6, 3) =	48	(7, 3) =	58
(0, 4) =	14	(1, 4) =	19	(2, 4) =	25	(3, 4) =	32	(4, 4) =	40	(5, 4) =	49	(6, 4) =	59	(7, 4) =	70
(0, 5) =	20	(1, 5) =	26	(2, 5) =	33	(3, 5) =	41	(4, 5) =	50	(5, 5) =	60	(6, 5) =	71	(7, 5) =	83
(0, 6) =	27	(1, 6) =	34	(2, 6) =	42	(3, 6) =	51	(4, 6) =	61	(5, 6) =	72	(6, 6) =	84	(7, 6) =	97
(0, 7) =	35	(1, 7) =	43	(2, 7) =	52	(3, 7) =	62	(4, 7) =	73	(5, 7) =	85	(6, 7) =	98	(7, 7) =	112
(0, 8) =	44	(1, 8) =	53	(2, 8) =	63	(3, 8) =	74	(4, 8) =	86	(5, 8) =	99	(6, 8) =	113	(7, 8) =	128

5.3.4 TBisObjectProperty

```
// Mit dieser Klasse können Datenbank-Objekte zwischen den Benutzerschnitt-
// stellen-Komponenten ausgetauscht werden.
procedure Clear
procedure Add(theObjects: TbisObjectProperty)
procedure Remove(theObjects: TBisObjectProperty);
```

```
procedure Intersect(theObjects: TBisObjectProperty);
function Equals(theObjects: TBisObjectProperty): Boolean;
function Clone : TBisObjectProperty;
procedure Filter (const theType: String; incl: Boolean);
function FilterNavigate (const theCommands: String) : TBisObjectProperty;
  // führt die in theCommands übergebene Filternavigation aus.
  // Als Startmenge für die Filternavigation wird Self übergeben
function FilterNavigateEx (const theCommands: String) : TBisObjectProperty;
  // // erst ab Byron/BIS v4.10.8
  // führt die in theCommands übergebene Filternavigation aus.
  // Als Startmenge für die Filternavigation wird Self übergeben
  // Kann ausserhalb einer Transaktion aufgerufen werden,
  // da eine Lese- oder Schreibtransaktion automatisch eröffnet wird.
procedure ObjectFilter(const thePath: String;
                       const reachObject: TBisObjectProperty;
                       incl: Boolean);
 // Dient zum Testen, ob die Objekte der Menge mit den Objekten in reachObject
 // verknüpft sind (incl = true -> sie müssen verknüpft sein, damit sie nicht
 // entfernt werden).
 // Beispiel:
 //  theComponents.ObjectFilter('Room_Of_Component', theRooms, true);
 //  entfernt alle Komponenten aus theComponents, die nicht in den Räumen in
 //  theRooms liegen.
 //  thePath darf mehrere Assoziationen enthalten (<asso1>%<asso2>)
procedure ApplyObjectRights(const readNeeded  : Boolean;
                            const changeNeeded: Boolean;
                            const createNeeded: Boolean;
                            const deleteNeeded: Boolean);
  // Testet die Objektrechte für die enthaltenen Objekte. Die Parameter
  // readNeeded .. deleteNeeded beschreiben die Objektrechte, die der
  // angemeldete Benutzer auf die Objekte haben muss. Besitzt er diese Rechte
  // nicht, dann wird das Objekt aus der Liste gelöscht.
procedure BroadcastAsSelection;
  // Verschickt die enthaltenen Objekte als Selektion (Zeilen in Explorern bzw.
  // Objekte in der Grafik werden selektiert)
function GetAttribute(const attribut: String) : Variant;
function GetAttributeAsText(const attribut, format: String) : String;
function GetTermin(const attribut: String) : TBisTermin;
procedure DbDelete;
  // Löscht die Objekte in der Datenbank
procedure ModifyAttribute(const attribut: String; value: Variant);
procedure DeleteAttribute(const attribut: String);
procedure ResetAttribute(const attribut: String);
procedure CopyAttribute(const fromObject: TBisObjectProperty;
                        const attribute: String);
procedure CopyAllAttributes(const fromObject: TBisObjectProperty);
procedure UnbindAll(const assoc: String);
procedure Unbind(const assoc: String; const theObjects: TBisObjectProperty);
procedure Bind(const assoc: String; const theObjects: TBisObjectProperty);
procedure Rebind(const assoc: String; const theObjects: TBisObjectProperty);
procedure CopyAllReferences(const fromObject: TBisObjectProperty);
function PropertyIsCalculated(const prop: String) : Boolean;
procedure Sort(const attribut1: String; ascending1: Boolean;
               const attribut2: String; ascending2: Boolean;
               const attribut3: String; ascending3: Boolean);
procedure CopyToClipboard; // erst ab Byron/BIS v4.10.2
function  CallMethod (const methodName   : String; // = "classname.methodname"
                      const objectParams : TBisObjectProperty; // darf nil sein
                      const valueParam1  : Variant;
                      const valueParam2  : Variant;
                      const valueParam3  : Variant;
                      const valueParam4  : Variant;
```

```
                        const valueParam5  : Variant) : Variant;
  // vgl. unten
function Evaluate(const formula: String): String;
  // Wertet die in formula angegebene Formel aus und gibt das Ergebnis zurück
  // z.B. res := obj.Evaluate('Name: $Name$');
function Expression(const expr: String): Variant;
  // Wertet die in expr angegebene Expression aus und gibt das Ergebnis zurück
  // z.B. res := obj.Expression('"Name: " + ATTRIBUTE("Name")');
  // Achtung: bei Expressions, welche einen Boolean zurückgeben sollte
  //          sicherheitshalber = true angehängt werden. Bsp:
  //          u.Expression('REGEXP("'123456'", "[0-9]{4}")') = true;
property Count: Integer read;
property Items[index: Integer]: TBisObjectProperty read;
  // Liefert die Objekte einzeln
property ObjectsOf[path: String]: TBisObjectProperty read;
  // Navigiert und liefert die resultierenden Objekte.
  // path darf mehrere Assoziationen enthalten ('<asso1>%<asso2>)'
```

Bemerkungen zur Funktion `CallMethod`:

- Die aufgerufene Methode muss folgendes Profil haben:

```
method (objectParams : array of object) return
       Text/Integer/Time/Float/Boolean is;
```

oder

```
method (objectParams : array of object;
        valueParam1 : Text/Integer/Time/Float) return
        Text/Integer/Time/Float/Boolean is;
```

oder

```
method (objectParams : array of object;
        valueParam1 : Text/Integer/Time/Float/Boolean;
        valueParam2 : Text/Integer/Time/Float/Boolean) return
        Text/Integer/Time/Float/Boolean;
```

oder

```
...
```

- `valueParam1..5` mit dem Wert null werden ignoriert und werden nicht an die Methode weitergegeben.
- Die aufgerufene Methode hat ein "self" - es entspricht dem ersten Element, welches sich in der "self"-TBisObjectProperty befindet.
- Eine Lesetransaktion wird in jedem Fall geöffnet.
- Objekte können durch CallMethod nur in Form eines Strings zurückgegeben werden (Image).

5.3.5 TBisDataOperation

```
// Das Ereignis OnOperate hat als ersten Parameter eine Variable dieses Typs
bdoInitFormat (Integerwert: 0)
  // Initialisierung der Komponenten (z.B. Test, ob das gezeigte Attribut in
  // der Datenbank existiert etc.)
  // Achtung: bei bdoInitFormat ist das Objekt der DataSource, die Objektklasse
  //          der DataSource und FormInformation.ExternalSelection noch nicht
  //          geladen!
  // Achtung: bdoInitFormat wird pro Formularinstanz nur einmal ausgelöst.
bdoClear (Integerwert: 1)
  // Felder, Gitter Listen werden gelöscht bzw. geleert.
bdoSetDefault (Integerwert: 2)
  // Der Vorgabewert des Attributs (definiert im Modellexplorer) wird geladen.
```

```
    // Diese Operation wird zu Zeit nur aufgerufen, wenn ein neues Objekt erfasst
    // wird.
  bdoPrepareCreate (Integerwert: 3)
    // Soll ein neues Objekt mit einer 'DataSource' erzeugt werden, dann muss vor
    // bdoStoreData ein Operate(bdoPrepareCreate) ausgeführt werden. Bei
    // bdoPrepareCreate ist die Objektklasse der DataSource bereits geladen.
  bdoLoadData (Integerwert: 4)
    // Werte werden aus der Datenbank geladen.
  bdoLoadUnModified (Integerwert: 5)
    // Werte werden aus der Datenbank geladen, sofern der angezeigte Wert
    // nicht modifiziert wurde.
  bdoStoreData (Integerwert: 6)
    // Werte werden in die Datenbank gespeichert (Beziehungen erzeugt oder ent-
    // fernt, Objekte erzeugt oder gelöscht).
  bdoClearModified (Integerwert: 7)
    // Die Markierung ob der Inhalt einer Komponente modifiziert wurde (d.h. der
    // Wert wurde geändert) wird zurückgesetzt (nicht modifiziert).
  bdoInvalidDef (Integerwert: 8)
    // nicht verwenden
```

Die folgende Dokumentation der Aufrufreihenfolgen gilt ab Version v4.11.2

Reihenfolge der Operationen beim Erzeugen eines Objekts in einem **eingebetteten** Formular (durch Toolkonfiguration – ApplicationElment `<Form>` und `opCreate` **mit** `intoElement`):

1. `bdoPrepareCreate`
2. `bdoInitFormat`
3. `bdoSetDefault`

Reihenfolge der Operationen beim Anzeigen eines Objekts in einem **eingebetteten** Formular (durch Toolkonfiguration – ApplicationElment `<Form>` und `<Propagate>`):

1. `bdoInitFormat`
2. `bdoLoadData`

Reihenfolge der Operationen beim Erzeugen eines Objekts in einem **freien** Formular (durch Toolkonfiguration – `opCreate` **ohne** `intoElement` oder durch Skripting `CreateNewObjectModal`):

1. `bdoInitFormat`
2. `bdoPrepareCreate`
3. `bdoSetDefault`

Reihenfolge der Operationen beim Anzeigen eines Objekts in einem **freien** Formular (durch „Öffnen (F9)“ in der Benutzerschnittstelle oder durch Skripting `OpenObject` oder `ShowModalEx`):

1. `bdoInitFormat`
2. `bdoLoadData`

Reihenfolge der Operationen beim Speichern eines Formulars:

1. `bdoStoreData`

5.3.6 TBisDataLink

```
// Typ der Eigenschaft 'Datalink' der meisten Komponenten
procedure Operate(theOp: TBisDataOperation);
```

5.4 Zusätzliche Eigenschaften und Methoden sonstiger Komponenten

5.4.1 TControl

```
// Oberklasse aller sichtbaren Komponenten (Eingabefelder, Knöpfe, ...)
```

```
procedure Click;
  // Simuliert einen Mausklick
```

5.4.2 TByPageControl

```
property PageCount: Integer;
property Pages[ix: Integer]: TTabSheet;
```

5.4.3 TDynamicForm

```
property ModalResult: TModalResult; // ab V4.10.8
```

5.4.4 TCompareOperator

```
= (coEqual,
   coNotEqual,
   coMatch
   coNotMatch,
   coLess,
   coLessEqual,
   coGreater,
   coGreaterEqual);
coNone = coMatch;
```

5.5 Globale Prozeduren und Funktionen

```
procedure ShowMessage(const msg: String);
  // Zeigt msg in einem Dialogfenster
procedure LogMessage(evtType: TByEventLogType; const msg: String);
TByEventLogType = (eltFatal,  eltError, eltWarning, eltInformation, eltDebug);
  // Schreibt msg in das Log-File. Das Log-File wird nur beschrieben, wenn
  //der aktuelle Debug-Level (Umgebungsvariable BIS_DEBUG_LEVEL) grösser oder
  // gleich evtType ist.
  // Ab Byron/BIS v4.8.0
procedure Log (evtType: TByEventLogType;
               verbose: boolean;
               const msg: String;
               const theObjects: TBisObjectProperty);
  // Analog LogMessage(). Zusätzlich werden bei Angabe von theObjects die
  // übergebenen Objekte in das Log geschrieben. Bei verbose=true ausführlich,
  // dh. Mit URL und Object_Display_Typed, sonst nur die Anzahl der Objekte.
  // Ab Byron/BIS v4.9.3
function ShowMessageEx(
          const msg: string;
          dlgType: TMsgDlgType;
          buttons: integer;
          activeButton: integer): integer;
  // Zeigt ein Dialogfenster an. Typ und Buttons sind wählbar; analog MessageDlg
  // dlgType = (mtWarning, mtError, mtInformation, mtConfirmation , mtCustom);
  // Liefert als Ergebnis den gedrückten Button (mrOK, mrCancel, …)
  // Bsp.
  // if ShowMessageEx(,Fortfahren?', mtConfirmation, mbYes+mbNo, mbNo) = mrYes…
function ShowConfirmation(const msg: String; const showOk: Boolean;
                          const showYes: Boolean; const showNo: Boolean;
                          const showCancel: Boolean) : Integer;
  // Zeigt ein Bestätigungsdialogfenster an. Liefert als Ergebnis mrYes, mrNo,
  // mrOk oder mrCancel
procedure OpenContextHelp;
  // Öffnet die kontextsensitive Hilfe (F1)
  // Ab Byron/BIS v5.2.1
```

```
procedure CopyToClipboard(const s: string);
  // Kopiert den Text in die Zwischenablage
  // Ab Byron/BIS v5.2.1
function Now : TDateTime;
  // Gibt das aktuelle Datum und die aktuelle Zeit
function Null : Variant;
  // Liefert eine Variante mit dem Wert Null.
Function NewObjectProperty: TBisObjectProperty;
  // Liefert eine leere BisObjectProperty
Function NewVariantList: TVariantList;
  // Liefert eine leere VariantList, ab Byron/BIS v4.7.8
function IsNull(v: Variant) : Boolean;
  // Die Funktion gibt an, ob die Variante den Wert Null hat.
function VarIsNull(v: Variant) : Boolean;
  // Die Funktion prüft, ob die Variante den Wert Null hat oder leer ist.
function VarIsStr(v: Variant) : Boolean;
  // Die Funktion prüft, ob die der Type der Variante ein String ist (ab v4.10.5)
function VarIsFloat (v: Variant) : Boolean;
  // Die Funktion prüft, ob die der Type der Variante ein Float ist (ab v4.10.5)
function VarIsInteger (v: Variant) : Boolean;
  // Die Funktion prüft, ob die der Type der Variante ein Integer ist (ab v4.10.5)
function VarIsBoolean (v: Variant) : Boolean;
  // Die Funktion prüft, ob die der Type der Variante ein Boolean ist (ab v4.10.5)
function FileExists(fName: String) : Boolean;
// Tip: zuerst FormInformation.ReplaceParameter anwenden!
function DirectoryExists(dName: String): Boolean;
  // Tip: zuerst FormInformation.ReplaceParameter anwenden!
  // FileExists und DirectoryExists geben true zurück, wenn die im
  // Parameter angegebene Datei, bzw. das Verzeichnis, vorhanden ist.
  // Allfällige Umgebungsvariablen im Pfad werden ausgewertet.
// Tip: zuerst FormInformation.ReplaceParameter anwenden!
function FileCopy(FromFName, ToFName: String;
                  ReplaceExisting: Boolean) : Boolean
  // FileCopy gibt true zurück wenn das File erfolgreich kopiert wurde,
  // ansonsten gibt es false zurück. FileCopy kann mit Umgebungsvariablen
  // umgehen.
// Tip: zuerst FormInformation.ReplaceParameter anwenden!
function ~~ExtendedPathOf(const path: String) : String;~~
  // Wertet rekursiv alle Umgebungsvariablen im angegebenen Pfad aus.
// obsolete -> ersetzt durch FormInformation.ReplaceParameter

function ExtractFilePath(const fName: String) : String;
function ExtractFileName(const fName: String) : String;
function ExtractFileExt(const fName: String) : String;
  // Extrahiert die entsprechenden Teile aus einem Pfad.
function CheckFilterNav(const statements: String; var err: String) : Boolean;
function FindIndexed(const anAttribute: String; const aPattern: String;
                     match, ignoreCase: Boolean) : TBisObjectProperty;
  // Sucht Objekte über indiziertes Attribut
function FindIndexedEx(const anAttribute: String; value1: Variant;
                       cmp1: TCompareOperator; value2: Variant;
                       cmp2: TCompareOperator) : TBisObjectProperty;
  // Sucht Objekte über indiziertes Attribut (erweiterte Version)
procedure CreateNewObject(ofType: String; container: TBisObjectProperty;
                          selection: TBisObjectProperty);
  // Erzeugt ein neues Objekt der Klasse ofType d.h. das Eigenschaftsblatt wird
  // gezeigt. Die übergebene Selektion ist in der Eigenschaft ExternalSelection
  // der Komponente FormInformation zugreifbar.
function CreateNewObjectModal(ofType: String; container: TBisObjectProperty;
                              selection: TBisObjectProperty): TBisObjectProperty;
  // Erzeugt ein neues Objekt der Klasse ofType d.h. das Eigenschaftsblatt wird
  // modal gezeigt, d.h. alle anderen Fenster der Applikation sind gesperrt.
```

```
  // Die übergebene Selektion ist in der Eigenschaft ExternalSelection
  // der Komponente FormInformation zugreifbar.
function CreateNewObjectWithForm(ofType: String; withForm: string;
                            container: TBisObjectProperty;
                            selection: TBisObjectProperty;
                            doModal: boolean): TBisObjectProperty;
  // Erzeugt ein neues Objekt der Klasse ofType mittels dem angegebenen
  // Formular. doModal bestimmt, ob das Formular modal angezeigt und
  // das erzeugte Objekt als Funktionsergebnis zurückgeliefert wird.
function CreateNewObjectFromTemplate(
                            templates: TBisObjectProperty;
                            const containerNav   : string;
                            const canChildNav    : string;
                            const innerAssocs    : string;
                            const classAttribute : string;
                            const assocToTemplate: string): TBisObjectProperty;


  // ACHTUNG: Dieser Aufruf löst ein OnCopy-Event aus!

  // Kopiert die in <templates> angegebenen Objekte in den von der <containerNav>
  // angegebenen Container. Die Navigation in <containerNav> wird für jedes der
  // in <templates> angegebenen Objekte ausgeführt. Die Funktion kopiert auch
  // alle Child-Objekte mit, unter der Bedingung, dass die Navigation in
  // <canChildNav> ein Objekt zurückliefert. <canChildNav> erhält als
  // Eingangsobjekt das zu kopierende Child-Objekt.

  // Wird <containerNav> leer gelassen entspricht dies der Navigation "VIA
  // Object_To_Parent".

  // <innerAssocs> ist ein Text, welcher eine Liste von ";" (Semikolon)
  // getrennten Assoziationen beinhaltet. Diese Assoziationen werden unabhängig
  // vom Copy-Flag an der Objektklasse mitkopiert. Ist ein Objekt über eine in
  // <innerAssocs> angegebene Assoziation mit einem in diesem Aufruf kopierten
  // Objekt verknüpft, so wird das originale Objekt durch das neu erstellte
  // ersetzt, ansonsten bleibt das originale Objekt über die Assoziation
  // verknüpft(!!!). Um eine innere Assoziation so zu konfigurieren, dass sie
  // entweder an ein kopiertes Objekt oder gar nicht verknüpft wird, kann dem
  // Assoziationsnamen in der ";" getrennten liste ein "-" (Minus) als Vorzeichen
  // übergeben werden (z.B. "Room_Of_Component;-Room_To_Mst").

  // <classAttribute> enthält den Namen des Attributs, welches den Namen der zu
  // erstellenden Klasse beinhaltet. So kann ein Vorlagenobjekt, welches von
  // einer Vorlagen-Objektklasse abgeleitet ist kopiert werden und gleichzeitig
  // ein Klassenwechsel stattfinden. Ist der Parameter <classAttribute> leer,
  // wird automatisch das Attribut Object_TypeId verwendet. Falls der Wert des in
  // <classAttribute> angegebenen Attributs leer ist, erzeugt CreateFromTemplate
  // für das aktuelle Objekt keine Kopie.

  // Falls die Assoziation <assocToTemplate> am neuen Objekt erlaubt ist, wird
  // die Vorlage, aus der das neue Objekt hervorgeht, an diese Assoziation
  // gebunden. Dieser Parameter darf auch leer sein.

  // Der Rückgabewert, enthält die Kopien der in <templates> angegebenen Objekte.

Function ShowModalEx(
                  theForm: TBisObjectProperty;
                  theParams: TVariantList): Integer;
  // öffnet ein Formular und zeigt es modal an, ab Byron/BIS v4.7.8.
  // vgl. auch FAQ - Formular im Skripting öffnen
procedure OpenObject(theObject: TBisObjectProperty; readOnly: Boolean);
  // Öffnet das Eigenschaftsblatt eines Objekts
```

```
procedure OpenObjectEx(theObject: TBisObjectProperty;
                       theSelection: TBisObjectProperty;
                       withForm: TBisObjectProperty;
                       readOnly: Boolean);
  // Öffnet das Objekt mit dem registrierten, bzw. angegebenen Formular.
  // Die Objekte in theSelection werden an der FormInformation.ExternalSelection
  // abgelegt, erst ab Byron/BIS v4.9.1
procedure OpenObjectIn(theObject: TBisObjectProperty;
                       inTool   : String;
                       inView   : String;
                       style    : Integer);
  // Öffnet ein Tool und zeigt darin theObject
  //   inTool: default='ApGrafikEditor'
  //           bestimmt das Tool, welches geöffnet wird. Es gibt dabei 2
  //           Methoden:
  //           1. Es wird die Klasse des Tools spezifiziert, z.B. "ApRoomExpl"
  //              oder "ApGrafikEditor". Das Tool wird dann nach einer fixen
  //              Strategie gesucht.
  //              ( [Room_Of_Component] { Expl_Root_Of Object_To_Parent } )
  //           2. Filter-Navigation, ausgehend von theObject. z.B. "VIA
  //              Room_Of_Component VIA Object_To_Parent VIA Expl_Root_Of".
  //   inView: kein default
  //           spezifiziert die Ansicht welche im Tool eingestellt werden soll.
  //           Wenn keine Ansicht angegeben ist, dann wird diese nicht geändert.
  //           Im Layout-Editor (ApGrafikEditor) kann zusätzlich noch die Färbung
  //           spezifiziert werden, abgetrennt durch ein ";".Beispiele:
  //           "Schliesszylinder;DIN 277" (Schliesszylinder ist eine Grafik
  //           Ansicht und DIN 277 eine Färbung.
  //   Style:  kein default
  //           Hat zur Zeit nur im Layout-Editor eine Bedeutung. Die einzelnen
  //           Ziffern haben folgende Bedeutung ZZMLFG, dh. 231021 wird als ZZ=23
  //           M=1 L=0 F=2 und G=1 verstanden.
  //           G ewählt (Selektion)
  //             1: theObject wird selektiert
  //           F enster
  //             1: normales (Wiederherstellen)
  //             2: Maximiert
  //           L egende (Farblegende)
  //             1: eingeblendet
  //             2: ausgeblendet
  //           M odus (Raumflächenmodus)
  //             1: Solid (Farbe)
  //             2: Umrandung
  //           ZZ oom
  //             0: Es wird zu den theObject hingezoomed, gleich wie im ,Drop
  //                Zeigen'
  //             1: Der Zoom bleibt undverändert
  //             2: Alle Objekte umfassend (Drawing Extent)
  //             3: Blatt
  //             4-99: fester Wert für Breite = Höhe in Meter gemäss der Formel
  //                   (ZZ-2)*0.5*(ZZ-3) das ergibt u.a. 4=1m  5=3m  10=28m
  //                   20=153m  50=1128m  99=4656m
  //           Der Wert Style=0 entspricht also dem Verhalten beim Befehlt
  //           "Dokument Suchen".

procedure ShellExecute(fileName: String; parameters: String;
                       withObjects: TBisObjectProperty);
  // Führt ein ShellExecute aus
procedure ShellExecuteEx(filename      : String;
                         parameters    : String;
                         operation     : String;
                         show          : Boolean;
```

```
                        wait          : Boolean;
                        waitTimeoutMS : Integer;
                        killProcessAfterTimeout :
                        withObjects   : TBisObjectProperty); (ab v4.9.8)
  // Führt ein ShellExecute mit folgenden Parameter aus:
  // filename:      bestimmt den Dateiname (inkl. Dateipfad) zum Ausführen
  // parameters:    enthält die Parameter für den Ausruf von filename
  // operation:     bestimmt die ShellExecute-Operation z.B. "print" oder "open"
  //                Default = leer (normales Ausführen)
  // show:          legt fest, ob beim Ausführen von ShellExecute die verknüpfte
  //                Anwendung angezeigt werden soll (true) oder nicht (false)
  // wait:          legt fest, ob auf das Ausführen von ShellExecute gewartet
  //                werden soll (true) oder nicht (false)
  // waitTimeoutMS: falls wait = true ist, kann hier ein TimeOut in ms für das
  //                Warten angegeben werden
  // killProcessAfterTimeout: bestimmt, ob der Prozess bei wait = true und
  //                          waitTimeoutMS > 0 automatisch beendet werden soll
  // withObjects:   enthält die zu übergebenden BIS-Objekte (über BisObjects.dat)
procedure WinExec(cmd: String; params: String; withObjects: TBisObjectProperty;
                  wait: Boolean);
  // Führt ein WinExec aus
  // cmd kann Umgebungsvariablen enthalten
  // wait = auf Prozessende warten
procedure SetWaitCursor;
  // Setzt den Hourglass-Cursor
procedure RestoreCursor;
  // Setzt den Cursor auf den Default-Cursor zurück
procedure FocusControl(const aControl: TWinControl);
  // Setzt den Fokus auf das angegebene Control
function FocusedControl: TControl;
  // Liefert das Control mit dem Fokus oder Null, wenn kein Foku auf dem Form
  // vorhanden ist.
procedure ExecuteAction(const id: String; const selection: TBisObjectProperty;
                        const params: String);
  // Führt die Action mit der angegebenen id aus, d.h. es wird das Menü ausge-
  // führt, welches im Attribut bisB_ActionId den Wert des Parameters id hat
procedure ResetSerialNumbers(const viewName: String);
  // Setzt das Nummernband der angegebenen View zurück
  // ab v5.8.1 nicht mehr verfügbar
function GetSerialNumbersMinimum(const viewName: String) : Integer;
  // Liefert das Minimum des Nummernband der angegebenen View
  // ab v5.8.1 nicht mehr verfügbar
procedure SetSerialNumbersMinimum(const viewName: String; aValue: Integer);
  // Setzt das Minimum des Nummernband der angegebenen View auf aValue
  // ab v5.8.1 nicht mehr verfügbar
function GetCurrentTranslationOf(const textId: String) : String;
  // Sucht nach dem Text-Objekt <textId> und gibt den Text in
  // bisB_TextCurrentLanguage zurück
procedure RefreshTool('');
  // RefreshTool mit einem Leer-String aufgerufen aktualisiert Byron/BIS (wie
  // Shift + F5).
function Power(const base: Double; const exponent: Double) : Double;
  // Rechnet die Pozenz (base ^ exponent)
procedure InsertObjectMenuItems(
            const menuItem: TMenuItem;
            const objects: TBisObjectProperty);
  // Fügt für jedes Objekt in objects ein Menuitem in das Menü menuItem
  // ein. Die "alten" Items werden zuvor entfernt. Durch Klicken auf ein
  // Item wird das zugehörige Objekt geöffnet.
  // Anwendung: Bsp. Popup-Menü mit zugeordneten Dokumenten.
  // Erst ab Byron/BIS v4.10.2
```

```
function HasObjectsInClipboard: Boolean;
  // Erst ab Byron/BIS v4.10.2
function HasLicense(feature: String) : Boolean;
// Prüft auf die Existenz einer Lizenz, erst ab Byron/BIS v5.00
function ObjectsFromClipboard: TBisObjectProperty;
  // Erst ab Byron/BIS v4.10.2
procedure GlobalNotify(const theObject: TBisObjectProperty;
                       const opCode: Integer;
                       const theText: Text);
  // Ruft ein GlobalNotify auf (siehe auch Byron/BIS-Hilfe der Methodensprache)
  // theObject muss 1 Objekt enthalten
  // Ab Byron/BIS v4.10.6
function MapiSendMail(
           recipTo, recipCC, recipBCC: String;
           subject, body: String;
           attachments: TVariantList): integer;
  // Versendet eine E-Mail über den installierten MAPI-Client
  // (Microsoft Outlook o.ä.).
  // Es werden nur SMTP-Empfänger unterstützt, mehrere E-Mail-Adressen
  // werden durch „;“ getrennt.
  // attachments ist eine Liste von Dateipfaden (String)
  // Rückgabewert ist 0 bei Erfolg, sonst siehe MAPISendMailW function (MSDN)
  // Ab Byron/BIS v4.11.2
```

5.5.1 Behandlung von Schema und Schema-Rechten

```
function IsSubClassOf(const aSubClass, aClass: String) : Boolean;
function InstanceCount(const aClass: string; inclDerived: Boolean): Integer
  // liefert die Anzahl der Objekte einer Objektklasse, erst ab Byron/BIS v4.11.7
function GetSchemaRight(const theObject: TBisObjectProperty;
                        const aProperty: String) : TDbAccessType;
  // liefert das Schema-Recht für die angegebene Property am übergebenen Objekt:
  //   actNo_Access, actRead_Only oder actRead_Write
  // In theObject muss genau ein Objekt übergeben werden!
```

5.5.2 Behandlung von Aufzählungen von Textattributen

```
procedure GetViewEnumTexts(const viewName: String; aValue: TStrings);
procedure SetViewEnumTexts(const viewName: String; aValue: TStrings);
```

Achtung: Nach Aufruf von SetViewEnumTexts muss „Alles Aktualisieren“ aufgerufen werden, da die Änderung sonst nicht richtig propagiert wird. Dies lässt sich innerhalb eines Tools am besten mit

```
FormInformation.ExecuteToolOperation('ActualizeAll');
```

erreichen, sofern im Tool eine entsprechende Operation definiert ist. Sie auch hier.

5.5.3 Behandlung von Aufzählungen

```
function EnumGetItemCount(const viewName: String) : Integer;
  // Liefert die Anzahl der Enum-Items für das Attribut
procedure EnumGetItem(const viewName: String; idx: Integer;
                      var value: Integer; var text: String);
  // Liefert den idx-ten Eintrag der Aufzählung
procedure EnumSetItem(const viewName: String; aValue: Integer;
                      const aText: String);
  // Ändert einen Eintrag. Fügt den Eintrag falls nötig hinzu.
procedure EnumDeleteItem(const viewName: String; aValue: Integer);
  // Löscht den Eintrag
function DateTimePlusInterval(a: TDateTime;
                              const b: TBisIntervalEditExLinked) : TDateTime;
  // addiert ein Interval (aus einem TBisIntervalEditExLinked) zu einem TDateTime
```

5.5.4 Positionieren von Objekten

```
procedure SetPosition(const objects: TBisObjectProperty;
                      const anAssociation: String; const aPositionAttr: String);
  // Psitioniert die Objekte in Räumen.
  // Die Räume werden über die angegebene Assoziation gesucht und die Position in
  // das angegebene Attribut geschrieben. Wird keine Assoziation, bzw. kein
  // Attribut übergeben, wird 'Room_Of_Component' und 'Position' verwendet.
```

5.5.5 String Operationen

```
function StrToInt(s: String) : Integer;
  // StrToInt konvertiert den in s angegebenen String in einen Integer-Wert.
function StrToFloat(s: String) : Double;
  // StrToFloat konvertiert den in s angegebenen String in einen Double-Wert.
function StrToDateTime(s: String) : TDateTime;
  // StrToDateTime konvertiert den in s angegebenen String in einen
  // Zeit/Datum-Wert.
function TrimTrim(OrigText: String;
                  TrimChars: String;
                  OpMode: Integer) : String;
//  Liefert den entsprechend den Parameter veränderten OrigText
//    Beispiel: TrimTrim(' +41 61 6909600 ', ' ', 0) = '+41 61 6909600 '
//              TrimTrim(' +41 61 6909600 ', ' ', 1) = ' +41 61 6909600'
//              TrimTrim(' +41 61 6909600 ', ' ', 2) = '+41 61 6909600'
//              TrimTrim(' +41 61 6909600 ', ' ', 3) = '+41616909600'
//              TrimTrim(' +41 61 6909600 ', '0..9', 13) = '41616909600'
function Pos(const substr, str: String) : Integer;
function PosPos(const sub: String; const instr: String; nte: Integer) : Integer;
function Length(const str: String) : Integer;
function StringLength(const s: String) : Integer;
function Copy(const str: String; pos, len: Integer) : String;
function StringCopy(const str: String; pos, len: Integer) : String;
function StringReplace(const s, oldPattern, newPattern:
                       string; flags: integer): string; // vgl. Delphi, ab v4.11.7
function Format(const Format: String; const Args: array of const) : String;
function FormatDateTime(const Format: String; dateTime: TDateTime) : String;
  // vgl. Delphi
```

5.5.6 Funktionen auf Terminen (Serietermine, Serieelemente)

```
procedure FilterOverlappingDates(const theDates: TBisObjectProperty;
                                 const fromDate: TDateTime;
                                 const toDate  : TDateTime;
                                 const incl    : Boolean);
// incl = true  -> Entfernt alle Serieelemente aus theDates, die sich mit dem
//                 Zeitabschnitt <fromDate> bis <ToDate> nicht überschneiden
// incl = false -> Entfernt alle Serieelemente aus theDates, die sich mit dem
//                 Zeitabschnitt <fromDate> bis <ToDate> überschneiden
procedure FilterResources(const theResources: TBisObjectProperty;
                          const fromDate    : TDateTime;
                          const toDate      : TDateTime;
                          const findFree    : Boolean);
// Freie Ressourcen gesucht (incl)
// findFree = true  -> Entfernt alle Objekte aus <theResources>, die
//                     zwischen <fromDate> und <ToDate> belegt sind.
// findFree = false -> Entfernt alle Objekte aus <theResources>, die
//                     zwischen <fromDate> und <ToDate> nicht belegt sind.
procedure FilterResourcesForDate(const theResources: TBisObjectProperty;
                                 const termin      : TBisObjectProperty;
                                 const findFree    : Boolean);
// Freie Ressourcen gesucht (incl)
```

```
  // findFree = true  -> Entfernt alle Objekte aus <theResources>, die
  //                     an den durch <termin> definierten Daten belegt sind.
  // findFree = false -> Entfernt alle Objekte aus <theResources>, die
  //                     an den durch <termin> definierten Daten nicht belegt sind.
  // <termin> kann sowohl ein Serietermin als auch ein Serieelement sein.
  // ruft intern FilterResources auf.
  function FindConflictingDates(
           const termin : TBisObjectProperty) : TBisObjectProperty;
  // liefert eine Liste von Terminen (Serieelementen), welche dieselben Resourcen
  // wie termin zu denselben Zeiten belegen
  function EditTermin(const termin     : TBisObjectProperty;
                      const elementType: String;
                      const modal      : Boolean) : Boolean;
  // Öffnet ein Fenster, mit welchem ein Serietermin (d.h. dessen Elemente)
  // bearbeitet werden kann. Werden neue Serieelemente erzeugt, so wird als
  // Objektklasse <elementType> verwendet
  // Kann auch Serieelemente und Einzeltermine darstellen
```

5.5.7 Auslesen der History (Trace) eines Objektes (ab v4.11.2)

```
TTraceEntryList = class
  public
    property  Items[ix: Integer]: TTraceEntry; default;
end;

  TTraceOperation = (topLogin, topLogout, topCreate, topDestroy,
                     topModifyAttr, topModifyMethod, topDeleteAttr,
                     topLinkAttr, topBind, topUnbind, topReparent,
                     topModifySpace, topModifyArea, topCreateType,
                     topDestroyType, topReparentType, topChangeDynamicData,
                     topVerifyDB, topCustom, topTraceFile, topArchive);

TTraceEntry = class
  published
    property Date: TDateTime;
    property Who: string;
    property WhoUrl: string;
    property Op: TTraceOperation;
    property Prop: string read;
    property Value: string read;
end;

function TBisObjectProperty.GetTraceEntries: TTraceEntryList;
```

Hinweis: Die in Dateien archivierten Trace-Einträge werden nicht ausgelesen!

Anwendungsbeispiel:

Über folgendes Ereignis eines Buttons kann in einem Formular das Trace (nur die Spalten *Wann* und *Wer*) des dargestellten Objektes in ein Grid geladen werden:

```
procedure btnHistoryClick(Sender: TObject);
var
  trc: TTraceEntryList;
  i: integer;
  rId: string;
begin
```

```
  SetWaitCursor;

  grHistory.Clear(false);
  trc := BisDataSource.DbObject.GetTraceEntries;
  for i := 0 to trc.Count - 1 do begin
    rId := grHistory.CreateRow;
    grHistory.Cells[rId, 'Col_0'] := trc[i].Date;
    grHistory.Cells[rId, 'Col_1'] := trc[i].Who;
  end;
  grHistory.AdjustWidths;

  RestoreCursor;
end;
```

5.5.8 Diverse Funktionen

```
  function Assigned(p: Pointer) : Boolean;
  function Abs(e: Extended) : Extended;
  function Ord(v: Variant) : Integer;
  function Pred(i: Integer) : Integer;
  function Succ(i: Integer) : Integer;
```

5.5.9 Propagate aus der Tool-Konfiguration

```
procedure FormInformationReceiveToolPropagate(propagateType: TEleLinkType;
                                              theObjects: TBisObjectProperty;
                                              code: Integer;
                                              additionalData: TVariantList);
    // Empfängt Informationen aus der Toolkonfiguration,
    // welche mittels propatage übergeben wurden
    // Siehe Bsp: Wie kann ich von einem Formular zu ...
```

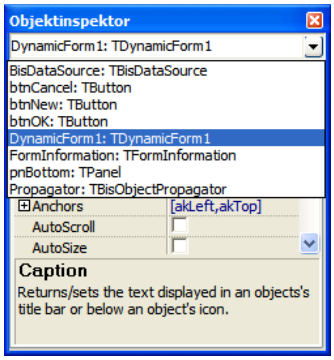
5.5.10 Propagate aus der Tool-Konfiguration

```
procedure FormInformationFilterNavigateParameter(const paramName: ShortString;
                                                 valueParam: TVariantList;
                                                 objectParam: TBisObjectProperty);
    // ermöglicht innerhalb einer Filternavigation eines Elements aus der
    // Toolkonfiguration einen Parameter (paramName) abzufragen,
    // welcher vom Formular ausgewertet wird.
    // Der Rückgabewert kann sowohl ein Value (valueParam) oder ein
    // Objekt (objectParam) sein
    // Siehe Bsp: Wie kann ich von einem Formular zu ...
```

6 Fragen & Antworten – Wie mache ich ...?

6.1.1 Wie setze ich Eigenschaften des Formulars selbst?

Die Eigenschaften des Formulars können verändert werden, indem in der Combobox des Objektinspektors der Eintrag 'DynamicForm1', bzw. 'DynamicForm*x*', angewählt wird.



Wie entferne ich die Knöpfe in der oberen rechten Ecke des Formularrahmens?

Verwenden Sie die Eigenschaft *BorderIcons* des Formulars.

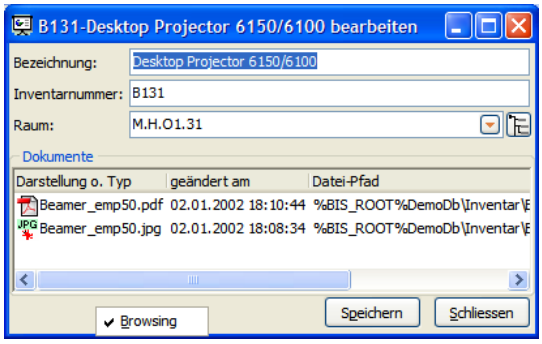
6.1.2 Ich möchte, dass das Formular immer im Vordergrund steht

Setzen Sie die Eigenschaft *FormStyle* des Formulars auf *fsStayOnTop*.

6.1.3 Mein Formular soll immer das momentan selektierte Objekt zeigen

Um Ihr Formular in ein so genanntes *Browse*-Formular umzuwandeln, setzen Sie die Eigenschaft *AllowBrowsing* der Komponente *Forminformation.*

Übrigens – das Browsing einer benutzerdefinierten Formular lässt sich mit Hilfe des Kontextmenüs auf der Arbeitsfläche ganz unten ausschalten. (vgl. folgendes Bild)



6.1.4 Mein Formular lässt sich nicht an einen Explorer andocken

Browse-Formulare können in jedem Fall angedockt werden. (vgl. Abschnitt 'Mein Formular soll immer das momentan selektierte Objekt zeigen')

Falls Ihr Formular nicht das jeweils selektierte Objekt zeigt (kein Browse-Formular), dann können Sie das Docking erlauben, indem sie folgende Eigenschaften des Formulars setzen:

- DragKind auf dkDock
- DragMode auf dmAutomatic

6.1.5 Wie setze ich den Tastaturfokus (aktiviertes Feld) beim Öffnen des Formulars?

Den initialen Tastaturfokus (Angebe, welches Feld als erstes die Tastatureingaben empfängt) können Sie mit der Eigenschaft *ActiveControl* der Komponente *Forminformation* festlegen.

Der Tastaturfokus kann zur Laufzeit mit der Eigenschaft *ActiveControl* des Formulars manipuliert werden. Beispiel:

```
Self.ActiveControl := BisEditLinked1;
```

6.1.6 Wie kann ich die Grösse des Formulars einfrieren?

Wenn Sie die Änderung der Formulargrösse einschränken möchten, dann Bearbeiten Sie die Eigenschaft *Constraints* des Formulars. In den Feldern *MaxHeight*, *MaxWidth*, *MinHeight* und *MinWidth* können Sie die minimalen und maximalen Abmessungen des Formulars angeben. Die aktuellen Werte entnehmen Sie am besten den Eigenschaften *Height* und *Width*.

Wenn Sie Änderungen der Formulargrösse generell verbieten möchten setzen Sie die Eigenschaft *BorderStyle* des Formulars auf *bsSingle*.

6.1.7 Wie kann ich die Position und Grösse des Formulars speichern?

Setzen Sie die Eigenschaft *StorePosSize* der Komponente *FormInformation* und Geben Sie in die Eigenschaft *StoreInSection* einen Namen für den Speicherort an. Die Position und Grösse wird in der Windows Registrierung gespeichert.

6.1.8 Wie kann ich Eigenschaften des Formulars (z.B. Caption) im Skript ändern?

Die Eigenschaften der Komponenten auf dem Formular werden über Ihren Namen zugegriffen (z.B. `BisEditLinked1.Text := '<noch leer>';`). Für das Formular muss der Name *Self* verwendet werden. Ein Beispiel:

```
Self.Caption := 'Mein Titel';
```
Setzt den Text im Rahmen des Formulars.

6.1.9 Warum werden Komponenten zur Eingabe von Zahlen rot dargestellt?

Geben Sie in der Eigenschaft *Text* oder *Value* der Komponente einen gültigen Zahlenwert ein, der innerhalb von Minimum und Maximum liegt.

6.1.10 Was ist der Unterschied zwischen den Eigenschaften ReadOnly und Enabled?

Die Verwendung von ReadOnly wird empfohlen.

Die Eigenschaft *Enabled*

Alle sichtbaren Komponenten besitzen diese Eigenschaft. Ist sie nicht gesetzt, dann kann die Komponente nicht angewählt werden (den Tastaturfokus erhalten). Versucht ein Programm (bzw. Skript) den Tastaturfokus auf eine Komponente zu setzen, deren Eigenschaft *Enabled* nicht gesetzt ist, kommt es zu einem Fehler.

Enabled wird an enthaltene Komponenten weitergegeben. Ist ein Panel nicht *Enabled*, so gilt dies auch für alle im Panel enthaltene Komponenten.

Die Eigenschaft ReadOnly

Einige Eingabefelder besitzen diese Eigenschaft. Sie verhindert Eingaben auf diesem Feld, nicht aber ein Anwählen (Tastaturfokus setzen) und ein Kopieren mittels CTRL-C.
Achtung: die Eigenschaft ReadOnly wird bei vielen Komponenten beim Laden des Wertes aus der

Datenbank zurückgesetzt. Änderungen durch das Formularskripting müssen also nach dem Laden (OnOperate) „wiederholt" werden.

6.1.11 Wie kann ich Komponenten ausblenden?

Mit der Eigenschaft *Visible.* Die Auswirkungen zeigen sich aber erst zur Laufzeit.

6.1.12 Wie kann ich mehrere Registerseiten in einem Formular einrichten?

Um Registerkarten in einem Dialog zu verwenden wählen Sie zuerst eine Komponente vom Typ *ByPageControl* aus dem Register *Standard* der Palette aus.

Wählen Sie jetzt den Eintrag *Neu Seite...* im Kontextmenü des Formulars

Durch die Selektion einer Seite, werden dessen Eigenschaften im Objektinspektor dargestellt und können verändert werden.

Jetzt können auf die einzelnen Seiten die benötigten Komponenten eingefügt werden.

6.1.13 Wie kann ich auf meinem Formular neue Objekte erzeugen ohne weitere Formulare aufzurufen?

Mit Hilfe einer DataSource können Sie direkt neue Objekte erzeugen wenn die Eigenschaft *AllowCreate* der DataSource gesetzt ist und der DataSource eine Objektklasse zugewiesen wurde.

Ablauf:

- Setzen Sie `AllowCreate` an der DataSource
- Weisen Sie der DataSource mittels Scripting eine Objektklasse zu:
  ```
  srcNewObject.ObjectType := 'Room';
  ```
  Dieser Schritt muss nur (einmal) durchgeführt werden, wenn die DataSource kein Objekt enthält.
- Bereiten Sie das Erzeugen vor (im Skript).
  Altes Objekt entfernen und Vorbereiten der DataSource (zwingend):
  ```
  srcNewObject.Operate(bdoPrepareCreate);
  ```
  Laden der Vorgabewerte in die Felder (fakultativ):
  ```
  srcNewObject.Operate(bdoSetDefault);
  ```
  Setzen des Containers des neuen Objekts (fakultativ):
  ```
  srcNewObject.Container := BisDataSource.DbObject;
  ```
- Lassen Sie den Benutzer Änderungen durchführen.
- Beim nächsten 'Store' (ausgelöst durch das Formular oder durch das Scripting oder ...) wird ein neues Objekt erzeugt.

6.1.14 Wie kann ich ein Formular in eine andere Sprache übersetzen?

Formulare können zur Laufzeit automatisch in die aktuell eingestellte Systemsprache lokalisiert (übersetzt) werden.

Ablauf:

- Setzen Sie in der FormInformation des Formulars die Eigenschaften "*AutoLocalize*", "*DesignLanguage*" und "*TextCatalogID*" auf die entsprechenden Werte.
- Erzeugen Sie den Textkatalog mit der gewünschten "*TextCatalogID*" mit Hilfe der Funktion "*Textkatalog erzeugen*" aus dem Kontextmenü der FormInformation. Alle im Formular verwendeten Texte werden dadurch im Textkatalog abgelegt.

- Übersetzen Sie die Texte der so erstellten Textkatalog-Objekte (Explorer "*Konfigurationen*"). Nicht benötigte Textobjekte dürfen auch gelöscht werden.
- Sie können die Übersetzung im Formular-Designer überprüfen, indem Sie die Funktion "Textkatalog laden…" im Kontextmenü der Forminformation verwenden.

Hinweise zur Verwendung von Textkatalogen (TextCatalogID) und Texten (Text-ID):

- **Wichtig!** Nach dem Erzeugen des Textkatalogs dürfen sowohl die Namen der Formularkomponenten als auch die TextCatalogID **nicht mehr geändert** werden, da sonst die Verknüpfung der Komponenten zu den Textkatalog-Objekten verloren geht. Wird dies trotzdem gemacht, muss der Textkatalog neu erzeugt und nicht mehr benötigte/doppelte Textkatalog-Objekte von Hand bereinigt werden.
- Es wird empfohlen, für jedes Formular einen eigenen Textkatalog (d.h. eine eigene TextCatalogID) zu verwenden. Zwar entstehen dadurch viele gleichartige Texte, dafür ist eine Vermischung von "Formular1.label1" mit "Formular2.label1" ausgeschlossen.
- Zur Unterstützung der Übersetzung von Texten, die in anderen Textkatalogen bereits übersetzt wurden, enthält Byron/BIS die Funktion "Automatisches Übersetzen" (das BMO kann bei Bedarf vom Support angefordert werden).
  Beim automatischen Übersetzen werden für ein Textkatalog-Objekt mit fehlenden Übersetzungen (Englisch, Italienisch, …) alle Textkatalog-Objekte durchsucht. Wird ein Textkatalog-Objekt mit demselben deutschen Text gefunden, dann werden dessen Übersetzungen übernommen.
  Mehrzeilige Texte (Texte mit Zeilenumbrüchen) können nicht automatisch übersetzt werden. Zeilenumbrüche können mit der Funktion "Mehrzeilige Texte umwandeln" durch "\r\n" ersetzt werden. Dieselbe Funktion ersetzt auch "\r\n" durch Zeilenumbrüche.
- Texte in den Formularen werden identifiziert durch die TextCatalogID.KomponentenName. Soll ein Textkatalog (d.h. eine TextCatalogID) von mehreren Formularen gemeinsam verwendet werden (entgegen der Empfehlung oben), dann muss darauf geachtet werden die Namen der Komponenten so zu vergeben, dass eine eindeutige Zuordnung von Komponentenname zu Textinhalt (= Textkatalog-Objekt) möglich ist (z.B. label1 -> "Bezeichnung" auf allen Formularen).

6.1.15 Wie kann ich das Laden von Elementen auf unsichtbaren Registerseiten eines BisPageControls optimieren, bzw. beschleunigen?

Ablauf:

- Verwenden Sie als PageControl ein PageControl der Klasse TBisPageControl.
- Setzen Sie am DataLink des Elements die Eigenschaft "OptimizedLoad".
- Setzen Sie am BisPageControl die Eigenschaft "OptimizedLoad".

6.1.16 Wie kann das Image für einen Eintrag in einer TBisLookupCombobox angezeigt werden?

- Die Option „IcoShowImage“ muss in der Eigenschaft „Options“ gesetzt sein
- Falls die LookupCombobox ohne Assoziation verwendet wird, muss im DataLink „UseAccessRights“ deaktiviert sein!

6.1.17 Wie kann ich mit Scripting aus einem Formular ein anderes Formular öffnen und von diesem ein Ergebnis erhalten?

Verwenden Sie ShowModalEx! Mit ShowModalEx kann ein Formular mit einer Parameterliste aufgerufen werden. Das aufgerufene Formular kann Werte in diese Parameterliste abspeichern und diese so an das aufrufende Formular zurückgeben.

Aufrufendes Formular und aufgerufendes Formular müssen aufeinander abgestimmt werden – siehe folgendes Beispiel.

Im aufrufenden Formular:

Das aufgerufene Formular muss ermittelt und die Parameterliste muss zusammengestellt werden.

```
procedure Button1Click(Sender: TObject);
var
  res: integer;
  f: TBisObjectProperty;
  p: TVariantList;
begin
  // Parameter erzeugen und belegen
  p := NewVariantList; // neu ab v4.7.8
  p[0] := 'Hallo';
  p[1] := 'Herr';
  p[2] := 'Müller';

  // Das aufzurufende Formular suchen
  f := FormInformation.CurrentUser.FilterNavigate(
         'START INSTANCES bisB_FormObject VALUE Name = "ShowModal"');

  res := ShowModalEx(f, p); // neu ab v4.7.8

  if res = mrOK then begin
    // OK gedrückt, Ergebnis des Formulars in p[3] anzeigen
    ShowMessage(p[3]);
  end; //if
end;
```

Im aufgerufenen Formular:

Die Parameterliste muss gespeichert und die BisDataSource muss initialisiert werden (bis und mit v4.11.1). Dies geschieht im Ereignis/Event OnSetParameters der FormInformation.

```
var
  globParams: TVariantList;


// Event „OnSetParameters“ an der FormInformation
procedure FormInformationSetParameters(values: TVariantList);
begin
  // wir merken uns das “Parameter-Objekt”,
  // damit wir es bei OK belegen können.
  globParams := values;

  // BisDataSource initialisieren (bis und mit v4.11.1 notwendig,
  // wenn die BisDataSource verwendet wird)
  BisDataSource.Operate(bdoInitFormat);

  // Eingehende Parameter verwenden
  edParam1.Text := globParams[0];
  edParam2.Text := globParams[1];
  edParam3.Text := globParams[2];
```

```
end;

procedure btnOKClick(Sender: TObject);
begin
  globParams[3] := edResult.Text; // Bei Klick auf OK die Ergebnisse zurückschreiben
end;
```

6.1.18 Wie kann ich von einem Formular zu einem anderen Formular in einer anderen Applikation springen (GoTo)?

Voraussetzung: Funktioniert nur für konfigurierte Formulare innerhalb eines Tools.

Ablauf:

- Toolkonfiguration: Definition einer Propagate-Operation (opPropagate)

```
<Operation ID="MyGotoOrder" opCode="opPropagate"
           toApplication="Details"
           toElement="MyOrderForm"
           fromParameter="MyOBJ_Order"
           propagateType="custom"
           propagateName="LoadOrder"
           propagateCode="10">
</Operation>
```

- Quell-Formular: Verwenden der Funktion „FormInformation.ExecuteToolOperation()“ zum Aufrufen der Propagate-Operation

```
procedure btGotoOrderClick(Sender: TObject);
begin
  FormInformation.ExecuteToolOperation('MyGotoOrder');
end;
```

- Quell-Formular: Verwenden des Callbacks „FormInformation.FilterNavigateParameter()“ zum Abfüllen des Propagate-Parameters

```
procedure FormInformationFilterNavigateParameter(const paramName: ShortString;
            valueParam: TVariantList; objectParam: TBisObjectProperty);
begin
  if paramName = 'MyOBJ_Order' then begin
    objectParam.Add(grOrders.SelectedObjects);
  end;
end;
```

- Ziel-Formular: Verwenden des Callbacks „FormInformation.ReceiveToolPropagate()“ zum Empfangen der Propagate-Operation

```
procedure FormInformationReceiveToolPropagate(propagateType: TEleLinkType;
            theObjects: TBisObjectProperty; code: Integer;
            additionalData: TVariantList);
begin
  if (propagateType = eltCustom) and (Code = 10) then begin
    // Die Variable "theObjects" enthält den Wert von "MyOBJ_Order", also
    // den selektierten Auftrag aus dem Quellformular (grOrders.SelectedObjects)
```

```
    end;
  end;
```

6.1.19 Wie kann ich eine Property vom Typ "set of ..." im Skripting setzen?

- Dazu muss der Index des zu setzenden Konstantenwerts bekannt sein oder die Konstante freigegeben sein. Beispiel:

```
memoStyler.AllStyles[0].Font.Style := (1 shl fsBold) + (1 shl fsUnderline);

// Setzt die Schrift auf fett und unterstrichen.
// Font.Style ist wie folgt definiert:
//
// TFontStyle = (fsBold {0}, fsItalic {1}, fsUnderline{2}, fsStrikeOut {3})
// TFontStyles = set of TFontStyle
```

6.1.20 Wie kann ich eine Position und Objekte in der Grafik abfragen:

Mit CallCommand.

Auslösen des Befehls:

```
var
  opt : string;
begin
  opt := '';
  if not rbxRaum.checked then begin
    if not rb0Raum.checked then opt := opt + '0';
    if not rb1Raum.checked then opt := opt + '1';
    if not rb2Raum.checked then opt := opt + '2';
  end;
  FormInformation.CallCommand(
     '',       // inTool
     0,        // callerTag
     'Pick',   // commandIdent
      opt,     // commandOption
     'Bitte Durchgang platzieren', //commandPrompt
      nil);
end;
```

Der Befehl liefert sicher eine Position. Durch die Option kann gesteuert werden, wie viele Räume sich an dieser Position befinden sollen. Diese werden dann als Objekte zurückgeliefert. Die Option wirkt ausschliessend, d.h. eine leere Option erlaubt alle Möglichkeiten. ‚02' bedeutet es darf nur ein Raum sein (0 und 2 ausgeschlossen). Der Befehl wird an die Grafik weitergeleitet, ohne das ‚propagates' definiert sein müssen.

Empfangen der Antwort in Forminformation CommandResponse:

resultValues enthält die Position (x, y) und die Drehung.
resultObjects enthält die Räume.

```
procedure FormInformationCommandResponse(const callerTag: Integer;
  const resultValues: TVariantList;
  const resultObjects: TBisObjectProperty);
var
  i : integer;
begin
  for i := 0 to resultValues.Count - 1 do begin
    FormInformation.ValueList.Items[i] :=  resultValues.Items[i];
```

```
  end; //for
  if callerTag = 0 then begin
    if (resultObjects.Count = 2) and (rb2Raum.checked or rbxRaum.checked) then begin
        DataSourceFlaeche1.DbObject := resultObjects.Items[0];
        DataSourceFlaeche1.Operate(bdoLoadData);
        DataSourceFlaeche2.DbObject := resultObjects.Items[1];
        DataSourceFlaeche2.Operate(bdoLoadData);
        DataSourceUebergang.ObjectType := 'CS_RaumUebergang';
        DataSourceUebergang.Operate(bdoPrepareCreate);
        DataSourceUebergang.Operate(bdoSetDefault);
        DataSourceUebergang.Operate(bdoStoreData);
    end else if (resultObjects.Count = 1)
                and (rb1Raum.checked or rbxRaum.checked) then begin
        DataSourceFlaeche1.DbObject := resultObjects.Items[0];
        DataSourceFlaeche1.Operate(bdoLoadData);
        DataSourceFlaeche2.DbObject := nil;
        DataSourceFlaeche2.Operate(bdoLoadData);
        DataSourceUebergang.ObjectType := 'CS_RaumUebergang';
        DataSourceUebergang.Operate(bdoPrepareCreate);
        DataSourceUebergang.Operate(bdoSetDefault);
        DataSourceUebergang.Operate(bdoStoreData);
    end else if (resultObjects.Count = 0)
                and (rb0Raum.checked or rbxRaum.checked) then begin
        DataSourceFlaeche1.DbObject := nil;
        DataSourceFlaeche2.DbObject := nil;
        DataSourceUebergang.ObjectType := 'CS_RaumUebergang';
        DataSourceUebergang.Operate(bdoPrepareCreate);
        DataSourceUebergang.Operate(bdoSetDefault);
        DataSourceUebergang.Operate(bdoStoreData);
    end else begin
      case resultObjects.Count of
        0:ShowMessage('Hier ist kein Raumübergang möglich (Kein Raum gewählt)');
        1:ShowMessage('Hier ist kein Raumübergang möglich (nur ein Raum gewählt)');
  else ShowMessage('Hier ist kein Raumübergang möglich (mehr als 2 Räume gewählt)');
      end; //case
    end; //if
  end;
end;
```

Die Daten werden in DataSource Operate in das neue Objekt abgespeichert:

```
procedure DataSourceUebergangOperate(theOp: TBisDataOperation; obj: TDbBaseObject);
var
  pos : Variant;
  rot : Double;
begin
  if theOp = bdoStoreData then begin
    DataSourceUebergang.DbObject.Bind('Room_Of_Component', obcFloor.DbObject);
    DataSourceUebergang.DbObject.Bind('x_UebergangNachInnen',
                                      DataSourceFlaeche1.DbObject);
    DataSourceUebergang.DbObject.Bind('x_UebergangNachAussen',
                                      DataSourceFlaeche2.DbObject);
    DataSourceUebergang.DbObject.Bind('x_UebergangZuZylinder', lock);
```

```
    rot :=  FormInformation.ValueList[2];
    if ABS(rot) > 0.00001 then begin
      DataSourceUebergang.DbObject.ModifyAttribute('Z_Drehwinkel', rot);
    end; //if
    FormInformation.ValueList.Items[2] := 0.0;  // Z-Koordinate = 0
    pos := FormInformation.ValueList.AsVariant;
    DataSourceUebergang.DbObject.ModifyAttribute('Position', pos); //globCoord);
  end; //if
end;
```

Um zwei Räume zu selektieren, muss das Cursor-Quadrat beide Räume berühren. Die Grösse des Quadrates kann abgefragt und verändert werden. Dies geschieht durch Custom-Propagates.

Wenn ein Wert > 0 propagiert wird, dann wird der Pickbereich auf diesen Wert gesetzt. Bsp.:

```
   FormInformation.PropagateData('PickBoxSize', 0, 8, nil);
```

Damit wird die Quadratgösse auf 16 Pixel (= 2 * 8) gesetzt, vorausgesetzt das wird an die Grafik propagiert:

```
   <Propagate operation="custom" name="PickBoxSize" target="Graphic"/>
```

Die Grösse kann auch abgefragt werden:

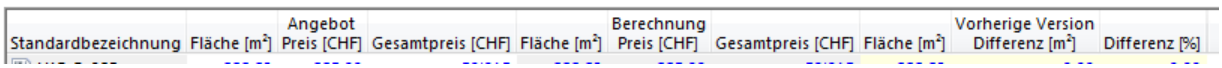
```
  FormInformation.PropagateData('PickBoxSize', 0, -1, nil);
```

Damit erhält man die Antwort in FormInformation ReceiveToolPropagate:

```
procedure FormInformationReceiveToolPropagate(propagateType: TEleLinkType;
                theObjects: TBisObjectProperty;
                code: Integer; additionalData: TVariantList);
begin
  if propagateType = eltCustom then begin
    if (additionalData.count > 2) and
       (additionalData.Items[1] = 'PickBoxSize') then begin
      TrackBar.enabled  := true;
      TrackBar.Position := additionalData.Items[2]; // hier steht der Wert
    end;
  end;
end;
```

6.1.21 Mehrzeilige Header im BisSimpleGridII

Beispiel:



Um dies zu erreichen benötigen Sie folgendes:

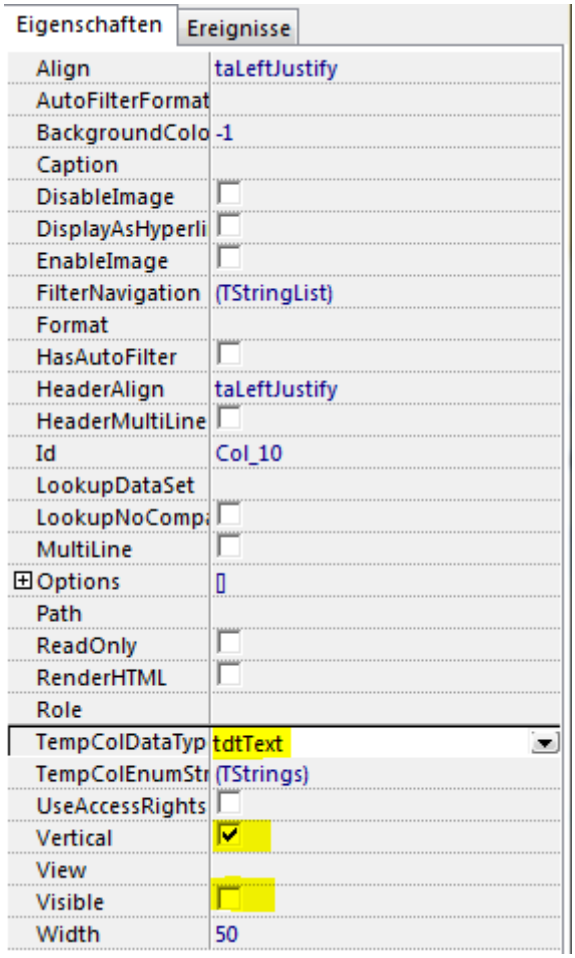
1. Die richtige Konfiguration der Spalten mit mehrzeiligen Header (aus *Formular als Text*):

```
item
  LookupNoCompareAttribute = False
  View = 'CIR_PreisAngeboten'
  Id = 'Col_3'
  UseAccessRights = False
```

```
    HeaderAlign = taCenter
    HeaderMultiLine = True
    Align = taRightJustify
    Caption = 'Angebot'#13#10'Preis [CHF]'
    DisableImage = False
    EnableImage = False
    BackgroundColorIndex = -1
  End
```

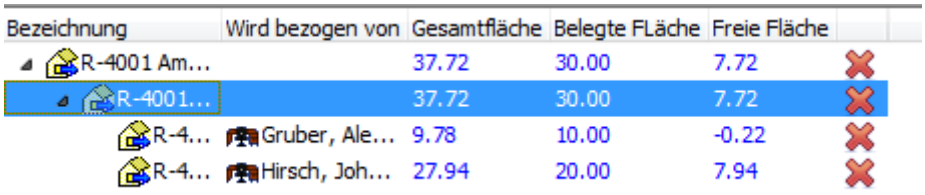
Hinweis: Die Caption lässt sich nur in der Textansicht des Formulars (oder im Skripting) mit Zeilenumbrüchen eingeben.

2. Eine zusätzliche unsichtbare vertikale Spalte

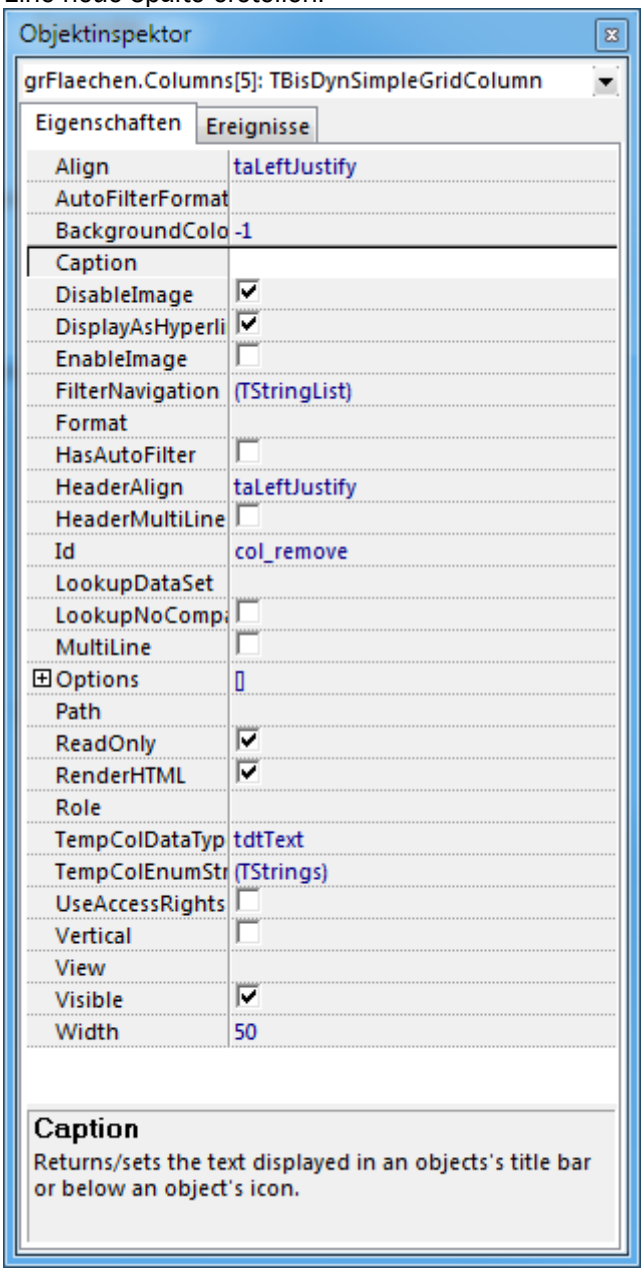


6.1.22 Wie kann ich in SimpleGridII Zeilen um active HTML-Elemente ergänzen?

Beispiel, um eine Zeile mit einem Löschen-Kreuz zu ergänzen:



1. Eine neue Spalte erstellen:



2. Am Grid das Ereignis OnRowAddedEx (erst ab v5.0.1!) anschliessen:

```
procedure grFlaechenRowAddedEx(Sender: TObject; const row: Short-
String; const obj: TBisObjectProperty);
begin
    Sender.Cells[row, 'col_remove'] := '<IMG src="bis:opDelete">';
end;
```

3. Am Grid das Ereignis OnCellClick anschliessen:

```
procedure grFlaechenCellClick(Sender: TObject; const col, row: Short-
String; const obj: TBisObjectProperty; var clickHandled: Boolean);
var
 conf: integer;
```

```
 objs: TBisObjectProperty;
begin
   if col = 'col_remove') then begin
     conf := ShowConfirmation(obj.Evaluate('Wollen Sie wirklich … ent-
fernen?'), false, true, true, false);
     if conf = mrYes then begin
       // do Something;
     end;
     clickHandled := true;
   end;
end;
```

6.1.23 Hinweise im Umgang mit High-dpi-Systemen

Bei High-Dpi-Systeme wie den Surface-Geräten von Microsoft skaliert Windows die Grösse von Text, Apps etc. Davon betroffen sind auch die Formulare von Byron/BIS. Grundsätzlich kann Byron/BIS damit umgehen. Beim Bau der Formulare sind jedoch die nachfolgenden Punkte zu beachten.

Align statt **Anchors**

Das Platzieren von Elementen mittels Anchors führt gerade bei komplexeren Formularen mit vielen Elementen zu Fehlern bei der Anzeige. Stattessen sollte zum Platzieren auf Align gesetzt werden.

Durch Strukturieren des Formulars mit GroupBoxen und Panels, welche mittels Align platziert sind (nicht «alNone»), lassen sich Darstellungsfehler weitestgehen vermeinden.

Keine absoluten Werte für Grösse / Position verwenden

Auf die Verwendung von absoluten Werten für Grösse (.Heigt oder .Width) oder Position (.Left oder .Top) sollte verzichtet werden, da diese bei Hight-DPI-Sysemen nicht skaliert werden und die Anzeige deshalb i.d.R. falsch ist.

Ein- / Ausblenden von Elementen

Sollen Elemente (Labels, Editfelder etc.) dynamisch ein /- ausgeblendet werden, folgende Hinweise:

- Nach Möglichkeit sollten ganze Panels / GroupBoxen ein-/ausgeblendet werden. Jeweils so konfiguriert, dass der hierbei freiwerdende Platz durch ein mittels alClient konfiguriertes Element eingenommen wird.
- Objger Punkt kann durch geschicktes «Verschachteln» von Panels / GroupBoxen auch bei komplizierereren Formularen umgesetzt werden.
- Muss die Grösse (z.B. eines Panesl) auf Grund des ändernden Platzbedarfes per Scripting geändert werden, so sollte die Modifikation von .Heigt oder .Width mittels Addition / Subtraktion anderer .Heigt oder .Width – Elemente passieren und nicht mittels absoluter zahlen (z.B. mittels Subtraktion der .Heigt des ausgeblendenten Panels).

Test des Formulars

Ein Test des Formulars auf einem High-DPI-System ist sehr empfehlenswert, soll dieses später definitv auf solch einem Gerät zum Einsatz kommen.