GDI-Business-Line 3.x Bildschirmdesigner

Factur- und CRM-Bereich

Stand: 17.04.2018
Version 3.8.1.2201

Änderungshistorie

Datum	Änderung	Autor
12.09.2002	Änderung Startdialog	GL
15.11.2002	komplette Überarbeitung Stand 2.0.1.15	GL
09.01.2003	nur ein Startwert bei nbSearch	GL
26.03.2003	Ergänzung Startdialog, kleinere Korrekturen, Ergänzung Tipps	GL
15.04.2003	LoadFromForm ebsDialog	GL
22.10.2003	CheckText-Erweiterung	GL
08.06.2004	ClientDBText, ClientMask	GL
09.06.2004	NavBar, Feldname=[max,Inkrement]	GL
08.07.2004	Grid, FixedColor, FixedCounter	GL
03.01.2005	2.0.4.2 GDI-BASIC-Speed-Button	GL
27.01.2005	2.0.4.5 cbsBoolean im Grid Mehrfachanzeige	GL
02.09.2005	Feinkorrektur ShowAll beim Grid	GL
11.01.2006	Hinweis: Desgin FABelegEK_1	GL
26.04.2006	Tabelle Griddefs, Maskenschutz, Animate, Splitter	GL
25.01.2007	visible bei Navbar	GL
23.04.2007	Inhaltsverz., Modal/nicht modal, kleine Korrekturen	GL
02.05.2007	TreeView Hideselection, Radio Default	GL
03.07.2007	Ausblenden Tabsheet aus TreeView	GL
24.07.2007	cbsBoolean: Farbe statt CheckBox	GL
14.01.2008	Zugriff auf Maske über @Form	GL
05.05.2008	CheckText im Grid	GL
30.05.2008	Gridsuche --> dgEditing true	GL
10.10.2012	TsButton, cbsImageList, Sessionname	GL
24.05.2013	Suchen im Grid, keine HK bei Navbar.Default	GL
09.10.2013	Multiselect F4-Artikelauswahl	GL
05.05.2014	Rechtesteuerung	GL
13.05.2014	div. Aktualisierung	GL
16.05.2014	MasterSlave Gridsuche	GL
17.10.2014	Komplettüberarbeitung **à** TsTable mit Beispielen, Maskenaufrufe im Facturbereich, Click-Ereignisse bei Buttons Neu aufgenommen: Maskendesign im CRM-Bereich	GL
06.11.2014	Click-Ereignisse bei NavBar-Buttons, Doppelclick-Ereignis beim Grid, "alte" TTable und TQuery eliminiert, TpsDBMemoText ergänzt, Syntax bei Aufruf von Belegen (Startdialog, ebsDialog) , Ordner "MandPfad\Bilder" bei DBImage	GL
15.04.2015	ClientDBText: ClientDatabaseName entfernt, SessionName ergänzt, Abschnitt "Kommunikation mit Masken" überarbeitet, diverse kleinere Korrekturen, Historie ergänzt	GL
28.04.2015	Neu aufgenommen: GDI-Basic zur Gridansteuerung, $GDI-BASIC im Grid, weitere Überarbeitungen und Korrekturen	GL
05.05.2015	ValueChecked bei TpsDBCheck	GL
30.10.2015	ClientDBText mit mehreren Parametern/Variablen	GL
20.04.2016	DBImage wandelt andere Grafiktypen ggfs. in JPG um	GL
09.05.2016	Toggle-Ereignis bei CheckBox, DBRadio	GL
12.05.2016	$GDI-Basic TableBrowser: Datenzugriff über %sTab_Grid… wie bei Gridbutton	GL
10.01.2017	TpsNavBar: neue Option "nbPlus"	GL
07.03.2017	Schreibfehler IsBandView - IsBandsView beseitigt	GL
24.05.2017	ItemIndex "-1" bei DBRadio, Change-Ereignis bei DBRadio	GL
11.04.2018	GDIAnzeigeSQL bei psEdit, PsPopUp etc.	GL
17.04.2018	TpsForm ergänzt plus diverse (kleinere) Ergänzungen	GL

Der Bildschirmdesigner des Factur-Bereiches (Maskendesigner)

- Aufbau und Bedienung
- Die zur Verfügung stehenden Objekte und ihre wichtigsten Eigenschaften

Mit dem **Bildschirm-Designer des Factur-Bereiches** der GDI-Business-Line (Bline) steht ein leistungsfähiges Werkzeug für Änderung/Gestaltung bestehender Masken oder für die Neuerstellung von eigenen Masken bereit. Im folgenden soll auf die wichtigsten Grundzüge dieses Bildschirm-Designers eingegangen werden. **Hinweis: Der Designer für Masken des CRM-Bereiches ist im zweiten Teil dieser Dokumentation ab Seite 73 beschrieben.**

Start des Bildschirm-Designers

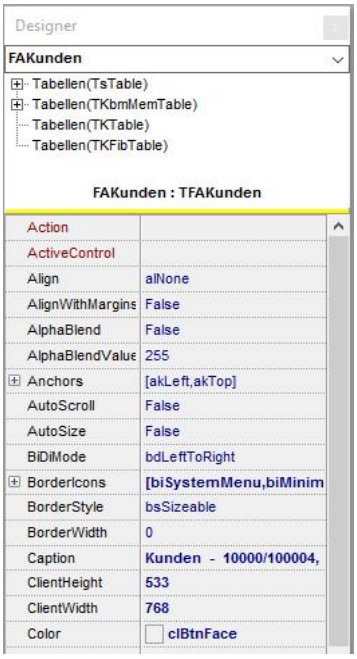
Der Bildschirm-Designer wird über den Eintrag *Bildschirm designen* im Systemmenü einer Maske aktiviert. Die Maske wird hierbei in den Design-Modus gestellt und zwei zusätzliche Fenster werden eingeblendet: Das Object- und das Property-Pad. Wichtig: Die zu designende Maske darf sich weder im Vollbild-Modus befinden, noch darf der MDI-Manager zur Fensterverwaltung aktiv sein.

- Unmittelbar nach Aufruf des Designers befindet man sich auf der untersten Ebene der Maske, der sog. "Form" und die Eigenschaften der Form werden im Property-Pad angezeigt. Hier kann man beispielsweise GDIBasic hinterlegen.
- Seit Version 3.6 sind Property- oder Object-Pad immer im Vordergrund. Sofern diese in älteren Versionen durch die zu designende Maske verdeckt werden, können diese mit <F11> in den Vordergrund geholt werden

Aufbau des Bildschirm-Designers

- oben: Object-Pad (Karteikarten "Standard", "Datenbank" und "Tabellen" mit den dazugehörenden Objekten)

- links: Property-Pad. Das Property-Pad unterteilt sich in drei Bereiche (von oben nach unten):



  - Combobox zur direkten Anwahl eines in der Maske enthaltenen Objektes. Dies ist sehr hilfreich in solchen Fällen, in denen ein Objekt nicht mit der Maus markiert werden kann, wenn es durch andere Objekte verdeckt ist.
  - Baumstruktur über die Tabellenobjekte der Maske: Um Datenbankfelder in einer Maske ansprechen zu können, müssen die dazugehörigen Datenbanktabellen in Form von sog. Table-Objekten in der Maske hinterlegt sein. Die vorhandenen Tabellenobjekte werden in der Baumstruktur angezeigt, neue können hier eingefügt werden.
  - Property-Tabelle: Die Eigenschaften (Properties) eines Bildschirm-Objektes werden über die zweispaltige Property-Tabelle eingestellt. Spaltenüberschrift: links Objektname, rechts Objekttyp; Spalteninhalt: links Property, rechts Wert/Ausprägung der Property. In der Property-Tabelle wurden einzelne Objekteigenschaften mit einem "+" versehen. Mit einem Maus-Klick können hier weitere Einträge eingeblendet werden.

Beenden des Bildschirmdesigners

Durch Schließen des Object-Pads oder des Property-Pads wird der Designermodus beendet. Falls der Hauptbildschirm (also das zu designende Fenster) aktiv ist, genügt ein Drücken der <ESC>-Taste. Ebenso kann der Designer über den Kontextmenü-Eintrag "Editor beenden" (rechte Maustaste) geschlossen werden.

Laden und Speichern von Bildschirmmasken

Eine veränderte oder neu gestaltete Bildschirmmaske steht nur dann dauerhaft zur Verfügung, wenn sie abgespeichert wurde. Über das Systemmenü der Maske kann über den Menüeintrag *Bildschirm speichern* ein Dialogfenster aufgerufen werden. Nach Eingabe einer Bezeichnung wird die Maske als Datei entweder im Mandantenpfad, Unterverzeichnis "Masken" oder im Programmpfad, Unterverzeichnis "Masken" abgelegt (in GDILine 2.x jeweils Unterverzeichnis "Factur\Masken"). Beim nächsten Aufruf der Bildschirmmaske wird automatisch die gespeicherte Maske angezogen.

- Masken können als msk-Dateien oder txt-Dateien gespeichert und auch wieder geladen werden. Msk-Dateien haben den Vorteil, dass sie etwas schneller geladen werden können, die Ablage einer Maske als txt-Datei bietet den Vorteil, dass eine "Nachbearbeitung" mit einem Texteditor (z.B. Notepad, Wordpad) möglich ist.
- Beim Speichern wird per default die Ablage der Maske auf Mandantenebene vorgeschlagen.
- Beim Laden von Masken sucht das Programm zunächst im Mandantenpfad, Unterverzeichnis "Masken" nach der betreffenden Datei. Erst danach wird im Programmpfad, Unterverzeichnis "Masken" gesucht.

Berechtigungen

In der Menü/Rechteverwaltung **à** Karteikarte "Rechte > System" findet man drei Einträge, um bedienergruppenabhängig Berechtigungen bzgl. des Bildschirmdesignens vornehmen zu können:

- **Maskendesigner**: Hierüber kann der Aufruf des Bildschirm-Designers gesperrt werden. Die zugehörigen Einträge im Systemmenü werden ausgeblendet.
- **Maskenaufruf**: Hierüber kann der Aufruf angepasster/eigener Bildschirm-Masken gesperrt werden, wenn in einer Bedienergruppe keinerlei angepasste Masken gewünscht sind. Sofern man (temporär) mit den GDI-Standardmasken arbeiten möchte, kann man in der 3.x unter dem Menüpunkt "Einstellungen" (bzw. in der 2.x unter Basisdaten | Firmendaten > Einstellungen) das Flag "Standardmaske verwenden" setzen. Dies gilt dann nur für den aktuellen Bediener.
- **Maskenumschaltung**: Im Systemmenü einer Maske wird angezeigt, ob die GDI-Standardmaske oder eine selbstdefinierte Maske geladen ist (unterster Eintrag des Systemmenüs). Wird eine selbstdefinierte Maske angezeigt, so kann durch Mausklick auf diesen Menüeintrag zur Standardmaske umgeschaltet werden. Umgekehrt gilt: Ist die Standardmaske geladen, so kann durch Mausklick auf den in diesem Falle angezeigten Eintrag *Standardmaske* ein Dialog zum Laden externer Masken (Msk- oder Txt-Dateien) geöffnet werden. Nach vorgenommener Umschaltung wird der Maskenwechsel nach Schließen und Öffnen der Maske durchgeführt. Diese Methode stellt eine alternative Vorgehensweise zur eigentlichen Maskendatei-Zuweisung im Menüdesigner dar (komplett eigene Masken können allerdings nur über den Menüdesigner am entsprechenden Menüpunkt zugewiesen werden). In der Menü/Rechteverwaltung kann bedienergruppenabhängig diese Umschaltung zwischen Standardmaske und externer Maske (Msk- oder Txt-Datei) gesperrt werden. Der entsprechende Eintrag im Systemmenü wird ausgeblendet.

Einfügen von Objekten in die Bildschirmmaske
Das gewünschte Objekt wird im Object-Pad per Mausklick ausgewählt. Anschließend kann es durch weiteren Mausklick an der gewünschten Stelle in der Maske eingefügt werden.

Bearbeiten von Objekten der Bildschirmmaske
Zur Bearbeitung muss das gewünschte Objekt zunächst per Mausklick markiert werden (ein markiertes Objekt wird durch schwarze Quadrate am Umfang dargestellt). Alternativ kann ein Objekt über die Combobox des Property-Pads ausgewählt werden. Bei Datenbank-Tabellen wird die Tabelle in der Baumstruktur des Property-Pads markiert. Anschließend stellt man die Eigenschaften in der Property-Tabelle des Property-Pads ein. Vor der Änderung von Eigenschaften sollte man sich immer anhand der Spaltenüberschrift der Property-Tabelle des Property-Pads vergewissern, dass man das gewünschte Objekt ausgewählt hat.

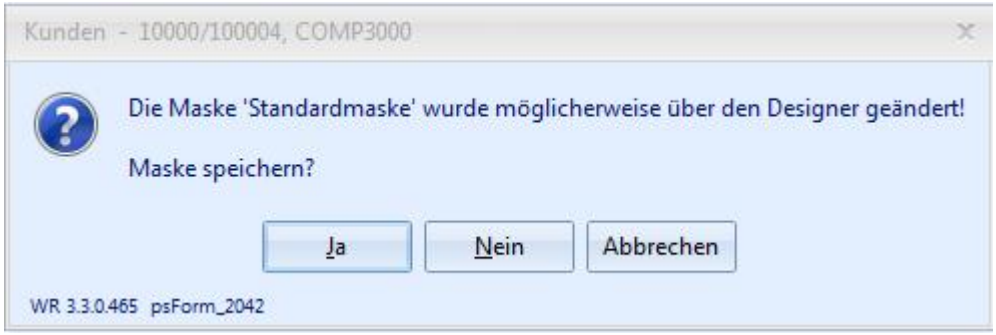
§ Markieren von mehreren Objekten
Mit <Shift>-Taste plus Maustaste können mehrere Objekte gleichzeitig markiert werden. Anschließend können diese gemeinsam verschoben oder ausgerichtet werden (**à** rechte Maustaste, Kontextmenü)

§ Tabulatorreihenfolge
Die Bearbeitung der Tabulatorreihenfolge von Objekten kann nach Markieren des Bildschirmelementes, auf welchem sich die Objekte befinden durch Drücken der rechten Maustaste (Kontextmenü) aktiviert werden.

§ Copy & Paste
Der Designer unterstützt Copy & Paste über das Kontextmenü. Ein Cut & Paste ist (leider) nicht möglich.

§ Modale und nicht modale Masken
Über das Menü eingebundene (eigene) Masken werden meist nicht modal aufgerufen. Dies bedeutet, dass bei geöffneter Maske andere Menüpunkte angewählt werden können.
Werden Masken aus anderen Masken heraus aufgerufen, z.B. über einen Editor mit Buttonstyle ebsDialog, so erfolgt ein modaler Aufruf. Dies bedeutet, dass andere Menüpunkte/Masken erst bearbeitet werden können, wenn zuvor die aufgerufene Maske geschlossen wurde.
Bei Verwendung der GDI-BASIC-Funktion Startdialog kann bei Aufruf einer Maske über Parameter entschieden werden, ob der Aufruf modal oder nicht modal (bzw. ab 3.x auch "freistehend") erfolgen soll (siehe auch *Bedeutung des Anzeigemodus* im nächsten Kapitel).

§ Nachfrage "Maske speichern?"
In der 3er Version erfolgt ab 3.3.2.x eine Sicherheitsnachfrage beim Schliessen einer Maske, sofern man zuvor den Designer geöffnet hatte. Diese Meldung erfolgt unabhängig davon, ob tatsächlich an der Maske eine Änderung vorgenommen wurde oder nicht.



Maskenaufrufe in der Bline 3.x (Factur-Bereich, MSK bzw. TXT):

Im Maskenaufruf ergaben sich gegenüber der GDILine 2.x Änderungen (und neue Möglichkeiten) aufgrund der Tatsache, dass das neue Menüsystem keine explizite Aliasliste führt. Es gibt also keine direkte Verknüpfung zwischen einem Alias und einer Maske, welche systemweit gültig ist, sondern bei einem Menüpunkt kann jeweils eine Maske angegeben werden. In der Bline 3.x ist somit z.B. ein Aufruf des Kundenstammes über zwei Menüpunkte mit unterschiedlichen gestalteten Masken möglich. Dies war in der GDILine 2.x nicht gegeben.
Bei Aufruf von Masken von anderen Stellen wird das Menüsystem in der Reihenfolge Menü, Toolbar und Systemmenü nach einem passenden Menüpunkt durchsucht, um hieraus verknüpfte Masken und Berechtigungen zu ermitteln. Beispiel: Aufruf des Kundenstammes aus Belegmaske **à** Es wird ermittelt, ob mit "TFAKunden" im Menü eine Maske verknüpft ist.

Aufrufe über das Menü:

Maskenaufrufe über das Menüsystem erfolgen ab der zweiten designbaren Hierarchieebene (z.B. Menü > Adressen > Kunden). Ein direkter Aufruf einer Maske über einen auf der ersten Ebene liegenden Menüpunkt ist nicht möglich (z.B. Menü > Kunden).

- Aufruf von "GDI-Masken": Im Menüsystem wird ein Menüpunkt über die Eigenschaft "Programmaufruf" zunächst mit dem zu startenden Programm-Modul verknüpft (z.B. TFAKunden oder TFABelegVK). Sofern dabei auf eine Maskendatei (MSK oder TXT) zugegriffen werden soll, wird diese als "Parameter 1" eingetragen. Über einen Auswahldialog wird die Zuordnung erleichtert. Parameter 2 und 3 sind für Belegaufrufe vorgesehen (Parameter 2 enthält das Belegart-Kürzel z.B. "AN" und Parameter 3 den Bezeichner der Belegart z.B. "Angebot").
- Eigene per Maskendesigner erstellte Masken werden über den Programmaufruf "ExterneMaske" plus Angabe der Maskendatei (TXT oder MSK) als "Parameter 1" eingebunden.

Aufrufe per GDI-BASIC-Funktion Startdialog:

Zusammen mit dem "Programmaufruf" werden die im Menüsystem optional angebbaren Parameter 1 - 4 bei der Startdialog-Funktion in deren **erstem** Parameter "Dialog" übergeben, und zwar durch Pipe ("|") getrennt. Obige Produktionsplanungs-Maske würde man also in GDI-Basic über `Startdialog("ExterneMaske|ProdPlanung.txt");` aufrufen. Ein Belegaufruf könnte wie folgt aussehen, im Unterschied zum Aufruf über das Menü kann dabei auch ein Startwert gesetzt werden: `Startdialog("TFABelegVK|*|AU|Auftrag",true,"BelegNr=101200008");`

Hinweis: Im CRM-Bereich erfolgt über das Menü lediglich die Zuweisung des entsprechenden Programm-Modules einschl. seiner Rechteeinstellung zu einem Menüpunkt. Die Verwaltung der MSK2-Masken des CRM-Bereiches erfolgt

nicht (benutzergruppen-spezifisch) über das Menü, sondern (bedienerspezifisch) über den Dateinamen.

Bedeutung des Anzeigemodus

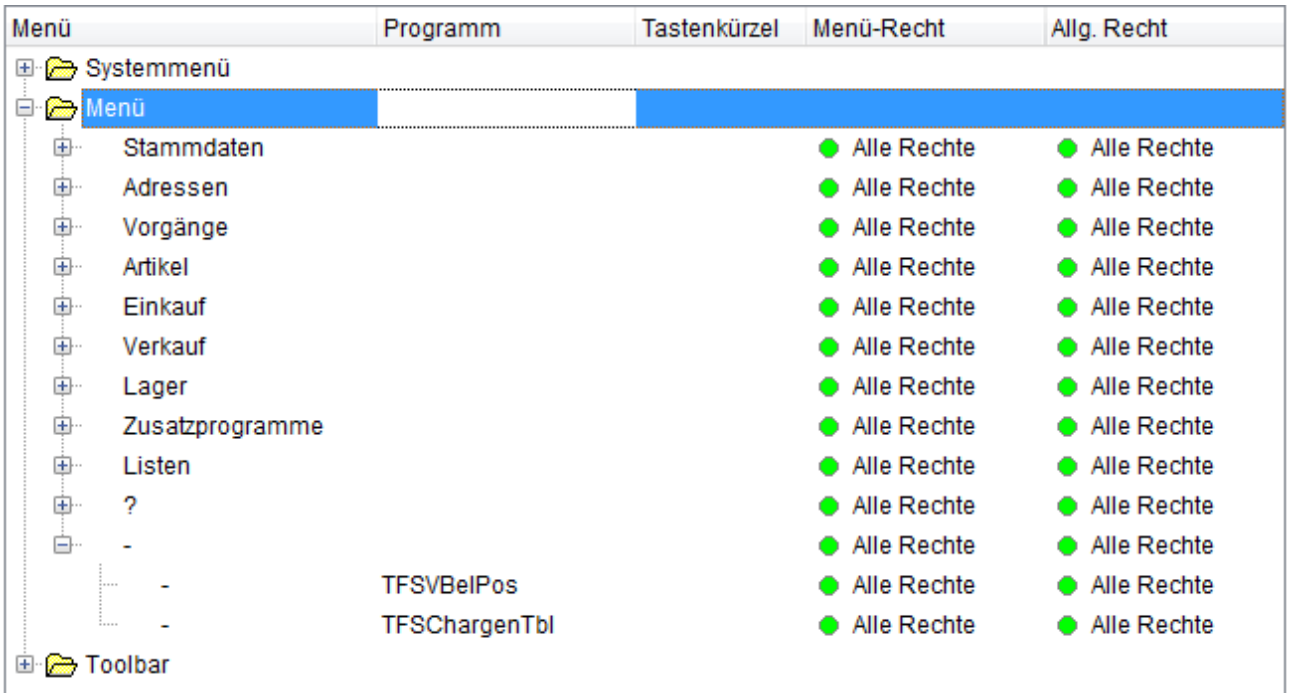
Der Anzeigemodus entspricht dem **zweiten** Parameter der GDI-Basic-Funktion "Startdialog".
Er entscheidet darüber, ob nach Aufruf der jeweiligen Maske andere Masken geöffnet werden können, ob sich eine Maske aus dem Hauptfenster der Bline hinausschieben lässt oder ob beispielsweise eine weitere Instanz derselben Maske geöffnet werden kann oder nicht. Es sind fünf Modi möglich, die Ziffern entsprechen dem Parameterwert in GDI-Basic:

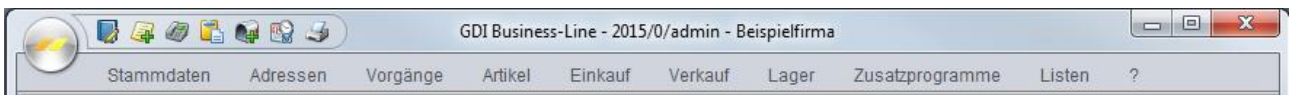
0 = nicht modal (Standard)
1 = modal (= default-Wert bei GDI-Basic-Funktion "Startdialog")
2 = nicht modal und freistehend. Die Maske lässt sich dann aus dem GDI-Hauptfenster herausschieben. Z.B. wenn man zwei Bildschirme hat.
3 = nicht modal und einzeln
4 = nicht modal, freistehend und einzeln

Verbergen von Menüpunkten

Aufgrund der fehlenden Aliasliste zur direkten Verknüpfung zwischen einem Alias und einer Maske müssen Menüpunkte auch für Masken generiert werden, die an sich nicht über das Menü aufgerufen werden - sofern man diese designen möchte. Beispiele hierfür sind Masken wie die Positionsinfo in der Belegerfassung, der Assistent zur Auswahl von Chargen, die Belegübernahme etc.

Über nachfolgenden Trick kann man die Verknüpfung anlegen, aber den Menüpunkt "ausblenden". Wichtig ist, dass man beide Ebenen "füllt", aber jeweils als Bezeichnung ein "-" (Minuszeichen) angibt!





Grundlage für die Anzeige/Bearbeitung von Daten in Masken: Das Tabellenobjekt TsTable

In den meisten Fällen sollen in Masken Daten aus Datentabellen der Datenbank angezeigt bzw. auch bearbeitet werden können. In diesem Falle muss, um den Bezug zur Datenbank herzustellen die entsprechende Datentabelle in Form einer TsTable in die Maske eingebunden werden. Die TsTable dient dann als DataSource für die in der Maske befindlichen Anzeigeelemente (z.B. Editfelder, Grids). Die TsTable findet man im Designer nicht wie die "sichtbaren Objekte" im Object-Pad, sondern im Property-Pad.
Bei selbstdesignten Masken sollte zu jedem Tabellenobjekt eine Navigationsleiste (NavBar) in die Maske implementiert werden. Diese ist z.B. für das Öffnen der TsTable bzw. Ausführen der bei ihr hinterlegten SQL verantwortlich (näheres siehe Objekttyp TpsNavBar).

Hinweise:
In der GDIline 2.x und in älteren Versionen der Bline gab es auch die BDE-behafteten Datenzugriffe über die Tabellenobjekttypen TpsTable und TQuery **à** siehe ältere Dokumentationen.
Neben der TsTable sind in der Bline 3.x noch weitere gdi-seitig verwendete Tabellen-Objekttypen in der Baumstruktur des Objektpads zu finden (z.B. TKbmMemTable). Diese sind i.d.R. für eigene Designs nicht relevant.

§ Einbinden einer TsTable in die Bildschirmmaske

Markieren des Astes *Tabellen (TsTable)* in der Baumstruktur des Property-Pads. Anschließend Tastenkombination <Strg> + <Einfg> drücken **à** ein Unterast mit der Bezeichnung STAB_1 wird eingefügt. Anschließend diesen Unterast markieren und die notwendigen Einstellungen vornehmen (siehe nächster Abschnitt).
Durch das Einfügen der TsTable wird zugleich auch eine sog. DataSource mit Namen DS_1 angelegt. Es wird empfohlen diese über die Auswahlbox des Objektinspektors auszuwählen und deren Namen in Anlehnung an den Namen der TsTable passend abzuändern. Wurde die STAB_1 z.B. nach TA_Kunden umbenannt, bietet es sich an die DataSource DS_1 nach DS_Kunden umzubenennen. An die DataSource werden später Objekte der Maske angeschlossen, welche letztlich die Datenanzeige/-Erfassung ermöglichen (Editoren, Grids etc.).

§ Die Eigenschaften einer TsTable

Die TsTable besitzt eine Vielzahl von Einstellungsmöglichkeiten. In nachfolgender Übersicht sind die wichtigsten Eigenschaften der TsTable (Tabellenobjekt vom Typ TsTable) in alphabetischer Reihenfolge (wie im Objektinspektor des Maskendesigners) zusammengefasst. Ein "X" in der letzten Spalte signalisiert, dass es sich um eine der "wichtigen" Eigenschaften handelt, die beim Maskendesign angepackt/kontrolliert werden müssen. Konkrete Beispiele über TsTable-Einstellungen finden Sie in einem gesonderten Abschnitt.

TsTable-Eigenschaft	Beschreibung	
Active	Zeigt an, ob die Datenmenge geöffnet (SQL ist ausgeführt) ist oder nicht. Wird bei angeschlossener Navigationsleiste (NavBar) über diese gesteuert oder kann auch per GDI-Basic gesetzt und somit die Datenmenge geöffnet/geschlossen werden (`@Objektname.Active = true;` bzw. `@Objektname.Active = false;`)	
AllowedUpdateKinds	Bestimmt, welche Aktionen an Datensätzen möglich sind. Bei rein lesenden Zugriffen ist bei den drei Unterparametern jeweils false einzustellen. Entscheidet auch darüber, ob bei Eingaben in einem Grid die Suche oder eine Editierung des Datensatzes erfolgen soll (nur in reinen Anzeigedatenmengen ist die Suche möglich)	
ukDelete	Löschen eines Datensatz ist möglich	X
ukInsert	Einfügen von Datensätzen erlaubt	X
ukModify	Datensatz kann geändert werden	X
AutoUpdateOptions	Parameter, welche bei Änderungen von Datensätzen von Bedeutung sind, also wenn UkDelete, ukInsert bzw. ukModify auf true eingestellt sind	
AutoReWriteSqls	Default true. Damit anhand der SelectSQL in Verbindung mit KeyFields und UpdateTableName dynamisch die SQL-Statements für Insert, Update und Delete erzeugt werden können.	X
KeyFields	Angabe der Datenfelder des Primärindex der zugehörigen Datenbanktabelle. Bei zusammengesetzten Indizes per Semikolon getrennt in der definierten Feldreihenfolge. Beispiel: "Kundennr" bei Kundentabelle, "BelegNr;BelegTyp;BelegArt" bei Belegtabelle. Diese Datenfelder müssen auch in der Feldliste der SelectSQL enthalten sein. Hinweis: Konstanten zur Vorbesetzung der Indexfelder bei der Anlage von neuen Datensätzen sind unter **à** Satzarten zu setzen	X
UpdateOnlyModifiedFields	Default true, Bei einem Update-Statement werden nur die Felder angesprochen, welche modifiziert wurden. Bei false würde explizit jedes Feld geschrieben (weniger performant)	
UpdateTableName	Angabe der Tabelle, die bei einem Update/Insert geändert werden soll. Zusammen mit den KeyFields kann somit dynamisch die UpdateSQL erzeugt werden	X
CacheModelOptions	Parameter für das Cache-Verhalten	
BufferChunks	Anzahl Datensätze (wie viele Daten werden "über die Leitung" gezogen und als Puffer vorgehalten). Defaultwert ist 32. Werden Datensätze in einem Grid angezeigt, so sollte der Parameter BufferChunks mindestens der doppelten Zeilenanzahl des Grids entsprechen.	
CacheModelKind	cmkStandard (Default), cmkLimitedBufferSize	
Conditions	Besondere Form eines Filters. Eine Condition wird immer additiv zu einer bereits vorhandenen where-Klausel verwendet.	
DataSource	Nur bis 3.4.1.764: Wird die TsTable als Slave-Datenmenge mit angeschlossenem Grid verwendet, so ist hier der zugehörige Master einzutragen. Sonst ist keine Suche im Grid möglich. Ab 3.4.1.776 sollte der Master bei der Property MasterSource beim Grid hinterlegt werden.	
DeleteSQL	SQL, welche beim Löschen eines Datensatzes ausgeführt wird. Diese SQL wird bei vorhandenen AutoReWriteSqls, KeyFields und UpdateTableName dynamisch erzeugt und muss hier nicht angegeben werden.	
Filter	Filterausdruck, welcher über Filtered = true aktiviert werden kann. Im Maskendesign der Bline werden Filter häufig über eine angeschlossene NavBar eingestellt (also nicht direkt bei der TsTable!).	
Filtered	true, um Filter zu aktivieren	

InsertSQL	SQL, welche beim Einfügen eines Datensatzes ausgeführt wird. Diese SQL wird bei vorhandenen AutoReWriteSqls, KeyFields und UpdateTableName dynamisch erzeugt und muss hier nicht angegeben werden.	
Locking	Default false, bei true werden bei Editieren von Datensätzen Einträge in der Tabelle Locking erzeugt und geprüft (Meldung bei gesperrtem Satz)	X
Name	Name der Komponente in der Maske. Sollte "sinnvoll" vergeben werden, um z.B. die TsTable in GDI-Basic-Code ansprechen zu können, z.B. TA_Kunden, TA_Beleg Tipp: Das Einfügen der TsTable bewirkt gleichzeitig die Anlage einer mit dieser Tabelle verknüpften DataSource. Diese besitzt automatisch den Namen DS_1. Es wird empfohlen diese im Objektinspektor auszuwählen und deren Namen ebenfalls "sinnvoll" und zur TsTable passend zu vergeben, z.B. DS_Kunden, DS_Beleg	X
RechtAktiv	GDI-Rechteverwaltung aktivieren. Damit über das Menü-/Rechtesystem eingestellte Berechtigungen in Masken greifen, muss bei den sTable-Datenobjekten der Maske die Eigenschaft "RechtAktiv" auf "true" eingestellt sein (Defaultwert ist "false").	X
RefreshSQL	SQL, welche zum Aktualisieren eines Datensatzes in der Anzeige ausgeführt wird. Diese SQL wird bei vorhandenen AutoReWriteSqls, KeyFields und UpdateTableName dynamisch erzeugt und muss hier nicht angegeben werden.	
Satzarten	Stringliste mit dem Aufbau Feldname=Wert. Z.B. bei Belegbearbeitung und bei den Warenwirtschafts-Basisdaten (Satzarten in Tabelle GDIDEF) zur Vorbesetzung von Feldern des zusammengesetzen Primärindex verwendet	X
SelectSQL	Wichtigste Eigenschaft. Hier hinterlegt man ein select-Statement mit order by klausel zur Bestimmung der Datenmenge. Nach Möglichkeit nicht mit kompletter Feldliste (*), sondern unter gezielter Angabe der notwendigen Felder z.B. `select Feld1,… Feldn from kunden order by kundennr.` Bei bearbeitbaren Datenmengen muss die SelectSQL die KeyFields-Felder beinhalten. Sofern – wie i.d.R. üblich – eine NavBar zur Steuerung der TsTable verwendet wird, wird die SelectSQL bei der Eigenschaft NavBar.Dialog.Filter hinterlegt.	X
Sessionname	Keine Angabe bei Zugriff auf "eigene" Mandantendatenbank. Sofern im System der Zugriff auf Externe Firebird-Datenbanken definiert wurden, können diese hier ebenfalls verwendet werden.	
ShowMessages	Defaultwert true. Anzeigen oder Unterdrücken von Meldungen bei Satzanlage, Löschen von Datensätzen etc.	X
SQLs	Hier sind alle relevanten SQLs noch einmal in einem Block beieinander stehend zu finden. Diese Parameter entsprechen den Einzelparametern	
UniDirectional	Bei großen Datenmengen kann über true die Performance und Speicherbedarf verbessert werden. Achtung: Nicht anwendbar, wenn die Notwendigkeit besteht, die Datenmenge in beiden Richtungen (abwärts und aufwärts) durchlaufen zu können.	
UpdateSQL	SQL, welche beim Ändern eines Datensatzes ausgeführt wird. Diese SQL wird bei vorhandenen AutoReWriteSqls, KeyFields und UpdateTableName dynamisch erzeugt und muss hier nicht angegeben werden.	

Anwendungsbeispiele

Je nach Anwendungsfall werden mehr oder weniger Einstellungen an der TsTable notwendig. In der Regel kommt man aber mit wenigen Parametern aus.
Ausgangsbasis der folgenden Beispiele sei eine leere Maske, ggfs. sind gliedernde Elemente bereits eingefügt (z.B. ein Panel). Es werden hier nur die essentiellen Einstellungen aufgelistet, um den Blick für das Wesentliche zu erhalten.

§ Beispiel 1: TsTable zur reinen Datenanzeige (Master)
Es sollen die in der Datenbank angelegten Kunden in einem Grid angezeigt werden. Im Grid soll in den Spalten eine Suche möglich sein.

Objekt.Eigenschaft	Einstellung/Wert
TsTable	
TsTable.AllowedUpdateKinds	[] d.h. alle Unterproperties ukModify, ukInsert, ukDelete auf false stellen
TsTable.Name	TA_Kunden
Datasource	
DataSource.Name	DS_Kunden
NavBar	

NavBar.Dialog.Filter	`select * from Kunden`
	`order by Kundennr`
	
	oder
	
	`select * from Kunden`
	`where adressgrp = '200'`
	`order by Kundennr`
	
	Tipp: Sofern sicher ist, welche Felder benötigt werden sollte man statt * aus Performance-Gründen eine Feldliste angeben.

NavBar.DataSource	DS_Kunden (falls Datasource nicht umbenannt wurde DS_1)
NavBar.Options	[nbFile,nbExit]
Grid	
Grid.DataSource	DS_Kunden (falls Datasource nicht umbenannt wurde DS_1)
Grid.InputGrid	false
Grid.LoadDefs	Kunden
Grid.Options	Kontrolle: dgEditing muss aktiviert sein, sonst keine Suche möglich
Grid.Tablename	Kunden

à Hinweis für Umsteiger aus GDILine 2.x: Im Prinzip "wie alt" bei dem Objekttyp TQuery, nur dass man der TsTable explizit über die AllowedUpDateKinds einstellen muss, dass nur ein lesender Zugriff gewünscht ist.

Merksatz: "Soll eine TsTable für rein lesende Zugriffe verwendet werden, beispielsweise um in einem (Slave-) Grid in den Spalten suchen zu können, so sind die drei Unterproperties ukDelete, ukInsert und ukModify bei der TsTable-Eigenschaft AllowedUpdateKinds auf false zu stellen"

§ Beispiel 2: TsTable zur reinen Datenanzeige (Master-Slave)

Erweiterung von Beispiel 1. Anzeige der zugehörigen Belege in einem weiteren Grid. Die bereits vorhandene Kunden-Tabelle stellt also den Master dar, hier die weiteren Objekte samt Einstellungen für den Slave:

Objekt.Eigenschaft	Einstellung/Wert
TsTable	
TsTable.AllowedUpdateKinds	[] d.h. alle Unterproperties ukModify, ukInsert, ukDelete auf false stellen
TsTable.Name	TA_Beleg
DataSource	
DataSource.Name	DS_Beleg
NavBar	

NavBar.Dialog.Filter	`select *  from Beleg`
	`where Adressnr ={kundennr} and Belegtyp = 'V'`
	`order by Belegdat desc`
	
	Felder des Master in geschweifte Klammern!

NavBar.DataSource	DS_Beleg (falls Datasource nicht umbenannt wurde DS_2)
NavBar.MasterSource	DS_Kunden (falls Datasource nicht umbenannt wurde DS_1)
NavBar.MasterField	Kundennr
NavBar.Options	[nbFile,nbExit]
Grid	
Grid.DataSource	DS_Beleg (falls Datasource nicht umbenannt wurde DS_2)
Grid.InputGrid	False
Grid.LoadDefs	BelegVK oder eigene Definition
Grid.MasterSource	DS_Kunden (bzw. falls Datasource nicht umbenannt wurde DS_1)
Grid.Tablename	Beleg
Grid.Range	`Adressnr = {Kundennr} and Belegtyp = 'V'` Felder des Master in geschweifte Klammern! Bei alphanumerischen Feldern die HK um die geschweiften Klammern setzen '{}'

Auch hier gilt **à** im Prinzip "wie alt" bei dem Objekttyp TQuery, nur dass man der TsTable explizit über die AllowedUpDateKinds einstellen muss, dass ein lesender Zugriff gewünscht ist.

Tipp: An dieser Stelle sei noch auf den Abschnitt "*Suchen im Grid, Zusammenfassung der Voraussetzungen*" (Seite 69) hingewiesen.

§ Beispiel 3: TsTable zur Datenbearbeitung (Master)

Die Tabelle enthält die notwendigen Objekte und deren Eigenschaften, damit eine Datenbearbeitung in einer Maske stattfinden kann. Im Beispiel soll wieder ein Zugriff auf die Kunden erfolgen.

Objekt.Eigenschaft	Einstellung/Wert
TsTable	
TsTable.AutoUpdateOptions.UpdateTableName	KUNDEN
TsTable.AutoUpdateOptions.KeyFields	KUNDENNR (wird automatisch anhand der Indizes ermittelt und eingetragen, sollte aber kontrolliert werden)
TsTable.Name	TA_Kunden
DataSource	
DataSource.Name	DS_Kunden
NavBar	
NavBar.DataSource	DS_Kunden (bzw. falls Datasource nicht umbenannt wurde DS_1)
NavBar.Dialog.Filter	`select * from Kunden order by Kundennr` Diese SQL kann auch direkt bei TsTable.SelectSQL hinterlegt werden. Da bei einer bearbeitbaren Datenmenge keine Suche im angeschlossenen Grid möglich ist, wird die SelectSQL nicht durch Eingaben in der Maske/Grid geändert.
NavBar.Options	[nbFile,nbInsert,nbSave,nbExit]
Grid	
Grid.DataSource	DS_Kunden (bzw. DS_1)
Grid.LoadDefs	Kunden
Grid.Options	Siehe unten "Anlage von neuen Datensätzen"
Statt des Grid selbstverständlich auch entsprechende Eingabefelder (TpsDBEdit):	
Eingabefeld	
DBEdit.DataSource	DS_Kunden (bzw. DS_1)
DBEdit.DataField	KundenNr
DBEdit.IndexName	`order by Kundennr` - nur bei dem Feld für die Kundennr, welches gleichzeitig die Primary-Eigenschaft für die Datensatzneanlage besitzt
DBEdit.Primary	true - nur bei dem Feld für die Kundennr, damit hier durch "Leeren des Feldes plus Enter" die Datensatzneuanlage (gemäß Nummernkreiseinstellung) ausgelöst werden kann.
DBEdit.LastField	true - sofern eine Eingabe beim Bestätigen des Feldes sofort gespeichert werden soll

Wichtig bei Anlage von neuen Datensätzen:

- Über das **"+" -Zeichen der NavBar** (nbInsert) erfolgt eine Datensatz-Anlage gemäß der Nummernkreiseinstellungen. Gleiches gilt für die Satzanlage über das mit Primary=true versehene Editfeld (s.o.). Soll ein Grid verwendet und die Satzanlage gemäß Nummernkreiseinstellungen vorgenommen werden, so ist beim Grid dgAppend, dgInsert auf false zu stellen. NavBar.Dialog.Default darf in diesem Falle keine Inkrement-Definition enthalten.

- Bei einer **Datensatzanlage über den Grid** (dgAppend **à** mit Cursor down eine Gridzeile erzeugen, dgInsert **à** mit Einfg-Taste) eine Gridzeile erzeugen) werden Nummernkreiseinstellungen nicht ausgewertet. Eine Satzanlage über einen Grid erfordert die Hinterlegung eines Inkrementes unter NavBar.Dialog.Default, z.B. Kundennr=[max,1]. Dann wird die höchste Kundennr aus den Daten ermittelt und mit dem Inkrement die neue Nummer gebildet (! bei gefilterten Datenmengen kann die Satzanlage ggfs. nicht funktionieren, wenn die neue Nummer bereits in der Tabelle existiert!). Zweckmässigerweise wird bei den Optionen der NavBar nbInsert auf false gesetzt (die Button-Kombination "-" und "+" verschwindet). Sofern über die NavBar das Löschen eines Datensatzes möglich sein soll verwendet man bei den Optionen der NavBar nbDelete (der Button "-" wird aktiviert). Ob über den Grid per <Strg> + <Entf> ein Datensatz gelöscht werden darf entscheidet die Option dgConfirmDelete beim Grid.

  Bei der Satzanlage über den Grid besteht das Problem, dass man bei (dauer-) gedrückter Cursor Down-Taste unzählige Datensätze erzeugen kann. Dies lässt sich aktuell (Stand 3.4.2.1060) nur durch Zuhilfenahme von GDI-Basic auf Maskenebene vermeiden. Folgendes Basic wertet das Post-Ereignis der TA_Kunden aus, welches vor dem Speichern eines Datensatzes eintritt. Befindet sich in diesem Moment der Datensatz/die Tabelle noch im Einfügemodus wird abgebrochen, wenn das Datenfeld Name1 keinen Eintrag besitzt:

```
:TA_Kunden.Post
if (%TA_Kunden.State = 3) then     //3 = Tabelle im Einfügemodus (insert, ap-
pend)
  if %TA_Kunden.FieldByName('Name1') <> '' then
     TA_Kunden_Post_Result := 0;   //Speichern durchführen
  else
     TA_Kunden_Post_Result := 1;   //Speichern verhindern
     %TA_Kunden.Cancel;
   endif;
endif;
exit;
```

Datenbearbeitung mit Filter:
Die zu bearbeitende Datenmenge kann auch gefiltert sein. In Erweiterung von Beispiel 3 sollen nur Kunden mit der Adressgruppe ABC bearbeitet werden. Hier gibt es zwei Varianten.

Variante 1: Man hinterlegt die komplette SQL (mit Filter in where-Klausel) bei NavBar.Dialog.Filter. Damit bei Neuanlage von Datensätzen diese aufgrund des Filters beim Speichern nicht "aus der Datenmenge fallen" ist NavBar.Dialog.Default entsprechend zu setzen. Über Default können Feldinhalte bei der Satzanlage vorgegeben werden

Objekt.Eigenschaft	Einstellung/Wert
NavBar	
NavBar.Dialog.Default	`adressgrp=ABC` Achtung: Ohne Hochkommata bei alphanum. Feldern, keine Leerzeichen vor/nach dem "=" -Zeichen

NavBar.Dialog.Filter	`select * from Kunden`
	`where adressgrp = 'ABC'`
	`order by Kundennr`
	
	"komplette" SQL

Variante 2: Bei dieser Variante wird bei der NavBar nur die Filterbedingung hinterlegt, die eigentliche SelectSQL wird bei der TsTable eingetragen. Hier greift nun - da in NavBar.Dialog.Filter das Schlüsselwort "select" nicht vorkommt - ein spezieller Mechanismus: Im Gegensatz zur Variante 1 wird nicht die SQL an die TsTable durchgereicht, sondern bei der TsTable Condition-Einträge zur Filterung generiert.

Objekt.Eigenschaft	Einstellung/Wert
TsTable	
TsTable.SelectSQL	`select * from Kunden order by Kundennr` SQL ohne where-klausel
NavBar	
NavBar.Dialog.Default	`adressgrp=ABC` Achtung: Ohne Hochkommata bei alphanum. Feldern, keine Leerzeichen vor/nach dem "=" -Zeichen
NavBar.Dialog.Filter	`adressgrp='ABC'` Achtung: Mit Hochkommata bei alphanum. Feldern

§ Beispiel 4: TsTable zur Datenbearbeitung (Master-Slave)

In diesem Beispiel geht es darum, in einer Slave-Datenmenge in einem Grid eine Datenerfassung zu erreichen. Das klassische Beispiel aus dem Standard-Bereich findet man in der Belegbearbeitung bei der Positionserfassung. Im Beispiel sei eine Tabelle kleine Tabelle "Vertrag" angelegt, in der pro Kunden ein oder mehrere Datensätze verwaltet werden sollen:

```
[Vertrag.Felder]
Kundennr=GDI_Long
VertragNr=GDI_Long
Text1=GDI_Char50
Text2=GDI_Char50
Text3=GDI_Char50
Text4=GDI_Char50
--

[Vertrag.Indizes]
KVIndex=*,+,Kundennr,VertragNr
KVIndex_Desc=,-,Kundennr,VertragNr
--
```

Folgende Erweiterung in unserer Testmaske aus Beispiel 1 oder Beispiel 3 wäre erforderlich, wenn die Datensatzanlage über den Grid, also durch Anfügen einer neuen Zeile per Cursor Down "geregelt" werden sollte:

Objekt.Eigenschaft	Einstellung/Wert
TsTable	
TsTable.AutoUpdateOptions.UpdateTableName	VERTRAG
TsTable.AutoUpdateOptions.KeyFields	KUNDENNR;VERTRAGNR (wird automatisch anhand der Indizes ermittelt und eingetragen, sollte aber kontrolliert werden)
TsTable.Name	TA_Vertrag
NavBar	
NavBar.DataSource	DS_Vertrag (bzw. falls Datasourcen nicht umbenannt wurden DS_2)

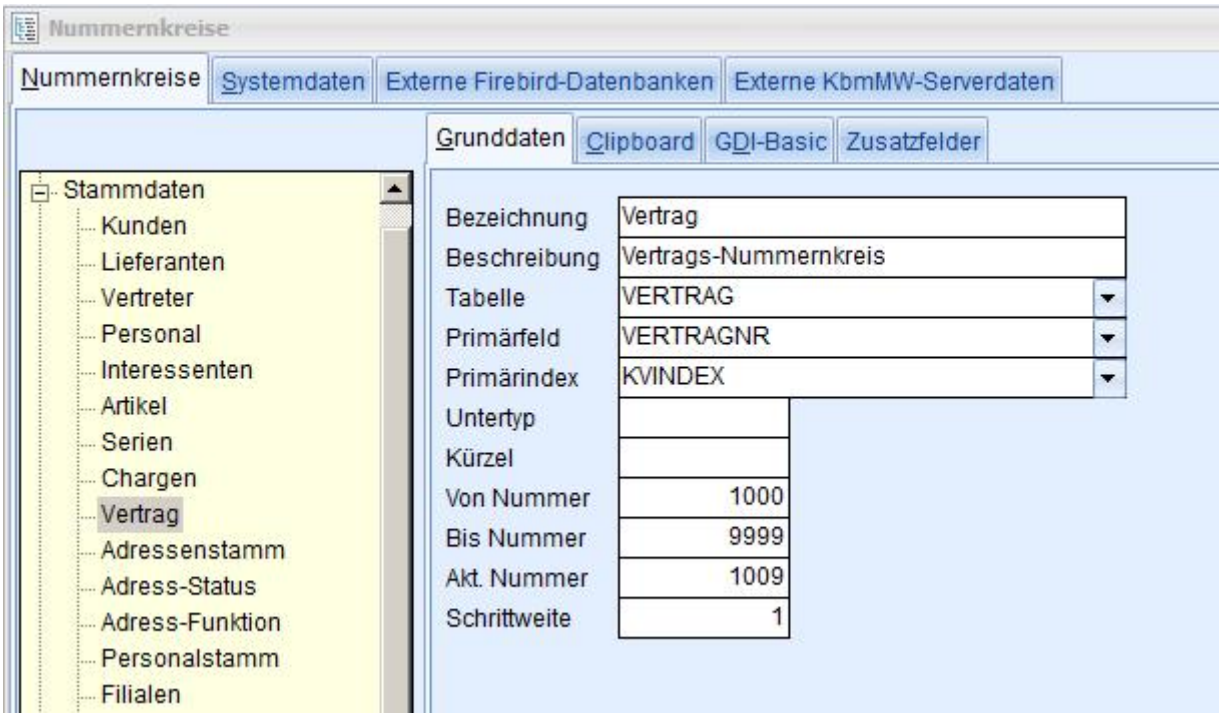
NavBar.Dialog.Default	`Kundennr={Kundennr}`
	`VertragNr=[max,1]`
	
	Zugriff auf Master in geschweifte Klammern

NavBar.Dialog.Filter	`select * from Vertrag`
	`where Kundennr={Kundennr}`
	`order by Kundennr, Vertragnr`
	
	Zugriff auf Master in geschweifte Klammern

NavBar.MasterSource	DS_Kunden (falls Datasource nicht umbenannt wurde DS_1)
NavBar.MasterField	Kundennr
NavBar.Options	[nbFile,nbDelete]
Grid	
Grid.DataSource	DS_Vertrag (bzw. DS_2)
Grid.LoadDefs	Eigener Loaddef
Grid.Options	dgAppend, dgEditing müssen aktiviert sein, dgInsert sollte deaktiviert sein, macht aufgrund der inkrementellen Vergabe der Vertragnr wenig Sinn

Wollte man neue Datensätze über einen Nummernkreis verwalten, wären folgende Modifikationen notwendig:

- Nummernkreis:
  Anlage eines Nummernkreises im gleichnamigen Menüpunkt:



- TsTable:
  TsTable.AutoUpdateOptions.KeyFields nur auf die VertragNr einstellen. Der automatisch gebildete Eintrag ist also zu ändern.

- NavBar:
  NavBar.Dialog.Default darf nur noch `Kundennr={Kundennr}` enthalten
  NavBar.Options ändern auf [nbFile,nbInsert]

- Grid:
  Grid.Options dgAppend und dgInsert deaktivieren

Im Grid lässt sich nun über Cursor Down kein Datensatz mehr anlegen, dafür steht jetzt in der Nav-Bar das "+" zur Verfügung.

In allen aufgezeigten Beispielen zur Datenbearbeitung werden bei der TsTable die UpdateSQL, DeleteSQL … bei den entsprechenden Aktionen in der designten Maske dynamisch erzeugt. Sie müssen beim Design nicht bei der TsTable eingetragen werden.

TpsForm - die unterste Ebene einer Maske

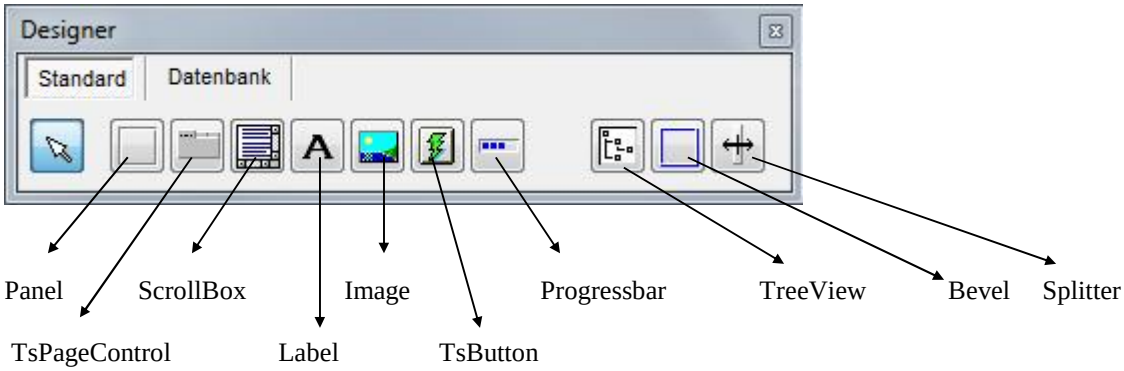
Unmittelbar nach Aufruf des Designers befindet man sich auf der untersten Ebene der Maske, der sog. "Form" (Objekttyp TpsForm) und deren Eigenschaften werden im Property-Pad angezeigt. Als wichtige Eigenschaften sind zu nennen:

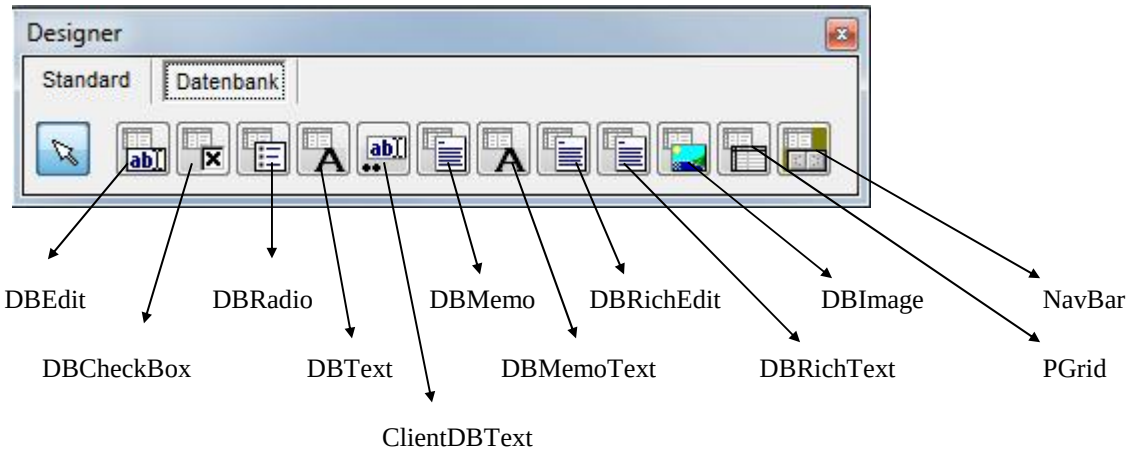
- Caption: Überschrift der Maske
- Constraints: Festlegung von minimaler und maximaler Größe der Maske (in Pixeln)
- Ctl3D: steuert die 3-dimensionale Darstellung von Objekten auf der Maske. Diese Eigenschaft wird auf untergeordnete, d.h. auf der Form liegende Objekte durchgereicht. Setzt man Ctrl3D auf *false* erscheinen beispielsweise Editoren (DBEdit-Eingabefelder) "flach". In den Standardmasken von GDI wird allerdings - weil hierbei Anzeigeprobleme festgestellt wurden - Ctrl3D nicht direkt auf der Form deaktiviert, sondern erst bei daraufliegenden Objekten. Es empfiehlt sich daher bei eigenen Masken zunächst ein Panel auf der Form mit Align *alClient* zu hinterlegen und dort Ctl3D zu deaktivieren
- GDIBasic: Hier kann das sogenannte "Maskenbasic" hinterlegt werden. Über vordefinierte bzw. selbst zu definierende Ansprungpunkte (z.B. :FormShow, :FormClose oder die Click-Ereignisse bei den TsButtons) kann auf Ereignisse reagiert und somit eine Ablaufsteuerung erreicht werden. Zum Zugriff auf die Maske selbst, z.B. um per GDI-Basic die Caption der Maske setzen zu können, kann man die Syntax @Form.Objekteigenschaft (im Beispiel `@Form.Caption`) verwenden
- Height: Höhe der Maske in Pixeln
- Passwort: Wird hier ein Eintrag vorgenommen, wird die Maske beim Speichern verschlüsselt d.h. sie kann von außen nicht mehr gelesen werden (auch bei Speichern als TXT). Will man in dieser Maske den Bildschirm-Designer aufrufen, so wird dieses Passwort abgefragt.
- ProgrammNr: Per Kommata getrennte Liste der Programmnummern (Namenseinträge), für welche diese Maske verwendet werden darf. Ohne Eintrag erfolgt keine Prüfung
- Width: Breite der Maske in Pixeln

Desweiteren sind noch die Properties DefaultValue, FilterValue, ReturnValue und StartValue erwähnenswert. Diese können verwendet werden, um an Masken bestimmte Werte zu übergeben und im umgekehrten Falle beim Schließen von Masken Werte an den Aufrufer zurück zu übernehmen. Sie sind näher im Kapitel "*Kommunikation mit Masken*" beschrieben.

Die Objekte des Object-Pads und ihre wichtigsten Eigenschaften

Objekte: Nachfolgend sind die drei Karteireiter des Object-Pads der Version 3.x abgebildet.
Bis zur Version 3.x gab es noch die beiden Objekttypen BASIC-Button und BASIC-SpeedButton. Diese wurden durch den neuen TsButton ersetzt, das Karteireiter-Objekt Notes durch TsPageControl.





Auf den folgenden Seiten werden die einzelnen Objekte näher beschrieben. Zunächst die "Standard"-Objekte ohne Datenbankanbindung, dann die Navigationsleiste und anschliessend die "Datenbank"-Objekte.

Panel (Objekttyp TPanel)

Ein Panel ist ein Grund-Objekt, auf dem andere Objekte platziert werden können. Dient in der Regel zur optischen Untergliederung einer Maske, in dem man Formatierungseigenschaften entsprechend einstellt (z.B. Rahmen)

- Align: Ausrichtung des Panel bzgl. des ihm übergeordneten Objektes
- BevelInner, BevelOuter: Abschrägungen an der Außenkante des Panels (3D-Effekt), wird in den GDI-Standardmasken meist nicht verwendet und daher auf *bvNone* gesetzt
- Border: Rahmenart
- Ctl3D: 3-D-Darstellung der Komponente, bei den GDI-Masken wird *false* verwendet, dadurch erscheinen beispielsweise auf dem Panel sitzende Eingabefelder "flach". Diese Eigenschaft wird also auf untergeordnete Objekte durchgereicht. Prinzipiell kann man auf der untersten Ebene einer Maske, der sog. Form bereits diese Eigenschaft einstellen. Man konnte aber in solchen Fällen Anzeigeprobleme feststellen. Es empfiehlt sich daher bei eigenen Masken zunächst ein Panel auf der Form (unterste Ebene einer Maske) mit Align *alClient* zu hinterlegen und hier Ctl3D zu deaktivieren.

PageControl (Objekttyp TsPageControl)

Dieses Objekt dient zum Einfügen und Gestalten von Karteikarten (*TabSheets*) auf einer gemeinsamen Ebene (*PageControl*).
Es dient somit zur sauberen Untergliederung einer Maske. Unmittelbar nach Einfügen des Objektes sind noch keine Karteireiter vorhanden / sichtbar, diese müssen zunächst über die Eigenschaft NewTabSheet angelegt werden.
Properties auf PageControl-Ebene:

- ActivePage: gibt an, welche Karteikarte beim Start der Maske aktiv sein soll bzw. beim Designen aktiv ist
- Align: Ausrichtung bzgl. des übergeordneten Objektes
- NewTabSheet: durch diese Property kann eine neue Karteikarte angelegt werden
- Style: Durch Angabe einer Ziffer (0-11) kann der Stil der angezeigten Karteireiter geändert werden

Properties auf TabSheet-Ebene:

- Caption: Karteikarten-Überschrift
- Enabled: wird hier false eingestellt, ist die Karteikarte nicht mehr anwählbar. Sofern das PageControl mit einem TreeView-Objekt verwendet wird (s.u.) verschwindet die Karteikarte aus der Baumstruktur
- PageIndex: Reihenfolge der Karteikarten untereinander.
- Hint: Hilfstext, hier zur Anbindung an TreeView
- TabVisible: Durch die Einstellung *false* kann mühelos eine Karteikarte und damit sämtliche sich auf ihr befindliche Elemente unsichtbar geschaltet werden. *False* wird auch verwendet, wenn man das PageControl-Objekt in Verbindung mit einen **TreeView** einsetzen möchte (s.u.). Der Aufbau der Baumstruktur erfolgt über Angabe des gewünschten Bezeichners bei der Eigenschaft Caption oder bei der Eigenschaft Hint (ist Hint leer wird auf Caption zugegriffen). Will man einen TabSheet komplett aus dem TreeView ausblenden, so sind Hint und Caption leer zu lassen.
  Durch die Baumstruktur wird auch eine zweidimensionale Darstellung möglich. Soll eine Untergliederung stattfinden, so erfolgt die Bezeichnerangabe in der Art "Bezeichnung übergeordnete Karteikarte und Bezeichnung untergeordnete Karteikarte durch Punkt getrennt" Beispiel:
  Vier TabSheets mit den Captions "1 Stammdaten", "1 Stammdaten.1 Zusatz", "1 Stammdaten.2 Bank" und "2 Belege" ergeben folgende Darstellung:

Sofern der Fokus auf dem TreeView-Objekt steht wird dem Bediener die Möglichkeit gegeben, durch Drücken des ersten Zeichens des jeweiligen Zweiges (im o.g. Beispiel die Ziffern) den zugehörigen TabSheet zu aktivieren bzw. zwischen TabSheets zu wechseln.

Hinweis: Man kann im Designmodus per Doppelclick auf einen Karteireiter zu diesem umschalten. Ansonsten kann man natürlich auch auf der PageControl-Ebene die ActivePage-Schaltung benutzen.

TreeView (Objekttyp TpsTreeView)

Das TreeView-Objekt dient zur Ergänzung des Objekttyps PageControl. Es wird benötigt, wenn man auf die Anzeige der einzelnen Karteireiter des PageControl verzichten und stattdessen lieber die "modernere" Anzeigeart als "Baumstruktur" verwenden möchte. Pro Maske ist nur ein TreeView möglich.

- Align: Ausrichtung bzgl. des übergeordneten Objektes
- Color: Hintergrundfarbe. In den GDI-Masken wird *clInfoBk* verwendet
- HideSelection: Ist von *true* auf *false* umzustellen, damit im TreeView angezeigt wird, welcher "Zweig" aktiv ist, auch wenn der TreeView nicht den Fokus besitzt.
- TabSheets: hier muss *true* eingestellt werden. Der TreeView "sucht" dann automatisch nach einem auf der Maske vorhandenen PageControl-Objekt (bei mehreren vorhandenen PageControl-Objekten ist die Erstellungsreihenfolge ausschlaggebend) und nimmt alle TabSheets in die Baumstruktur auf, deren Enabled-Eigenschaft auf *true* steht.

Die Erzeugung der Baumstruktur (Hierarchie) wird über die Angabe der Caption oder Hint bei den Tabsheets geregelt (s.o.). Zur korrekten "Einsortierung" muss bei zweidimensionalen TreeViews der PageIndex der Karteikarten zueinander passen, die untergeordnete Karteikarte muss also einen höheren PageIndex besitzen als die übergeordnete Karteikarte.

ScrollBox (Objekttyp TScrollBox)

Wird benötigt, wenn auf einem Panel oder einer Karteikarte Bildlaufleisten angezeigt werden sollen, wenn das Fenster verkleinert wird bzw. die Größe der Maske nicht ausreicht, um alle Elemente gleichzeitig darzustellen.

- HorzScrollBar: Legt die Eigenschaften des horizontalen Rollbalkens fest.
- VertSrollBar: Legt die Eigenschaften des vertikalen Rollbalkens fest.
    - Increment: Schrittweite bei Mausclick auf einen Pfeil des Scrollbalkens
    - Margin: Mindestabstand des untersten bzw. rechtesten Objekts vom Rand der Scrollbox, ab dem die Laufleiste gezeigt wird.
    - Position: Position des Scrollbalkens in Pixeln vom oberen bzw. linken Rand der Scrollbox
    - Range: Festlegung des zu scrollenden Bereiches in Pixeln (Festlegung der Größe der zu scrollenden Fläche)
    - Tracking: Steht "Tracking" auf true, so ist ein direktes Verschieben des Fensters beim Scrollen möglich. (?)
    - Visible: Ein- und Ausblenden des Srollbalkens. In der Regel wird nur diese Eigenschaft benötigt (visible = true)

Label (Objekttyp TpsLabel)
Dient der Gestaltung von Bezeichnern innerhalb einer Maske.

- AutoSize: Automatische Größenanpassung an den eingegebenen Text
- Caption: Text des Labels
- Font: Schriftart des Labels
- WordWrap: über *true* kann der automatische Zeilenumbruch erreicht werden. Mehrzeilige Labels kann man auch erzeugen, in dem man AutoSize auf *false* setzt und bei der Eingabe der Caption <Strg> + <Enter> für den Umbruch benutzt bzw. über F4 einen Editor zur Eingabe des mehrzeiligen Beschriftungstextes öffnet. Das Label ist dann auf die gewünschte Größe zu ziehen.

Image (Objekttyp TpsImage)
Festes Einbinden eines Bildes in eine Maske (z.B. Firmenlogo). Das Bild ist datensatzunabhängig und wird in der Maske selbst abgelegt.

- Picture: durch diese Property wird der Dialog zum Laden, Speichern und Einbinden eines Bildes geöffnet. Zur Zeit können Dateien vom Typ BMP oder JPG verarbeitet werden.
- AutoSize: Passt die Objektgrösse an die Bildgrösse an. U.U. nur sinnvoll, wenn das TpsImage selbst auf einer ScrollBox platziert ist
- Center: Zentrierte Darstellung des Bildes
- Stretch: Wird diese Property auf *true* gestellt, so wird die Bildgröße automatisch an die Objektgrösse angepasst. Dies kann u.U. zu Verzerrungen führen.

Progressbar (Objekttyp TProgressbar)
Bei der Progressbar handelt es sich um ein Objekt, welches als Fortschrittsanzeige für Programmabläufe eingesetzt wird. Es bietet sich daher z.B. zur Verwendung in eigenen Masken an, in denen per GDI-BASIC ein Datenimport/Datenexport realisiert wird.

- Max: Maximale Positionsnummer
- Min : Minimale Positionsnummer (meist "0")
- Position: aktuelle Positionsnummer
- Smooth: Stufenlose oder segmentierte Darstellung

Zur Ansteuerung der Progressbar über GDI-Basic verwendet man die Syntax, welche allgemein zum Zugriff auf Maskenelemente Verwendung findet: "@Objektname.Objekteigenschaft". Würde beispielsweise die Progressbar den Namen "PBar1" besitzen, so könnte man deren Max-Wert wie folgt auf 10000 setzen: `@PBar1.Max := 10000;`

Bevel (Objekttyp TBevel)
Dieses Objekt dient ähnlich wie ein Panel zur Untergliederung einer Maske. Im Gegensatz zum Panel kann man hier keine Objekte auf dem Bevel platzieren, es handelt sich hier nur um den "optischen Rahmen", mit welchem man signalisieren kann, dass Objekte zusammengehören. Ein Bevel kann einfach über bereits vorhandene Objekte gelegt werden.

- Shape: Darstellungsart des Bevels als Box, Rahmen oder Linien. GDI verwendet meist *bsFrame* (Rahmen)

Splitter (Objekttyp TSplitter)
Das TSplitter-Objekt ist eine Teilerleiste, die den Client-Bereich eines Formulars / einer Maske in einzelne Felder variabler Größe gliedert. Die Teilerleiste kann mit der Maus verschoben werden.

Beispiel: Im Kundenstamm befindet sich ein solcher Teiler zwischen den Grids für die Seriennummern und die Serienpositionen. Eine Teilerleiste ist also immer dann sinnvoll, wenn ein Formular aus mehreren Teilflächen mit variabler Größe bestehen soll. Wichtig ist die Reihenfolge beim Einfügen der "zu teilenden" Objekte. Das erste Objekt wird am Rand ausgerichtet (z.B. alLeft oder alTop), dann das Splitterobjekt angefügt (alLeft oder alTop). Dann können weitere Objekte folgen, wobei das letzte Objekt, das eine Teilfläche bildet (z.B. ein Memofeld oder Panel) immer am Client ausgerichtet (alClient) sein muss, damit es den verbleibenden Platz automatisch ausfüllt.

- Align: Ausrichtung des Teilers. In Voreinstellung ***alLeft***, so dass es sich um einen vertikalen Teiler handelt. Über ***alTop*** wird hieraus beispielsweise ein horizontaler Teiler.
- AutoSnap: Wird hier true eingestellt, so wird die Größe von benachbarten Objekten auf Null setzt, wenn der Benutzer versucht, die Objekte durch die Verschiebung der Teilerleiste kleiner zu machen als in MinSize angegeben. Wenn AutoSnap ***False*** ist, wird eine solche Größenänderung blockiert. Die benachbarten Objekte behalten die in MinSize angegebene Größe.
- MinSize: Bestimmt die minimale Größe in Pixel des Bereichs auf jeder Seite der Teilerleiste, legt also die Mindestgröße der Objekte fest, welche durch die Teilerleiste "geteilt" wurden. MinSize darf max. auf die halbe Client-Breite des übergeordneten Objekts gesetzt werden. Weitere Info siehe AutoSnap.
- ResizeStyle: Bestimmt die Optik während des Verschiebens der Teilerleiste (z.B. ***rsNone*** **à** die Größenänderung wird erst nach Loslassen der Maus wirksam, ***rsUpdate*** **à** die Größenänderung findet während des Verschiebens der Teilerleiste statt)

TsButton (Objekttyp TsButton)

Der TsButton ist der universelle Button seit der Bline 3.x. Er vereint die Funktionen für welche früher verschiedene Button-Typen notwendig waren. Im Bereich des Maskendesigns hat er die Aufgabe GDI-BASIC-Routinen "auf Knopfdruck" abzusetzen.

- Clicked: Dient zum Anstoßen des Buttons über ein GDI-BASIC-Programm, in dem man diese Eigenschaft auf true setzt (z.B. `@TsButton1.clicked := true`). Der TsButton-Button muß dabei nicht sichtbar sein.
- GDIScript: Hinterlegung des BASIC-Programmes. Alternativ kann das Script auch auf der Form hinterlegt werden, siehe hierzu den nachfolgenden Abschnitt "***GDI-Basic in Masken: Click-Ereignisse bei Buttons***" auf der nächsten Seite.
- ModalResult: nach Beenden des beim Button hinterlegten BASIC-Programms wird die Maske geschlossen, wenn man hier den Wert ***mrOK***, ***mrCancel*** oder ***mrAbort*** setzt. Diese textuellen Einträge entsprechen intern den numerischen Werten 1, 2 oder 3. Bis zur Bline-Version 3.6.x konnte lediglich die Ziffer bei der Modalresult-Eigenschaft eingetragen werden. Sofern man das eigentliche Script für den Button auf der Form hinterlegen (siehe nächster Abschnitt ***"GDI-Basic in Masken: Click-Ereignisse bei Buttons"***) und gleichzeitig ModalResult benutzen möchte, muss man beim Button zusätzlich ein (rudimentäres) Script eintragen.
- RunCaption: Beschriftung des Buttons vor und nach dem Programmablauf. Während das Basic-Programm abgearbeitet wird, wird die StopCaption angezeigt. Anmerkung: Die Eigenschaft ***Caption*** kann nicht (wie sonst üblich) für die Beschriftung eingesetzt werden.
- RunGlyphNr: Hinterlegung einer sog. ImageIndex-Nummer zur Darstellung eines Symbols auf dem Button vor und nach dem Programmablauf. Hinweise zur Ermittlung der ImageIndex-Nummer für das gewünschte Symbol **à** siehe unten ab Seite 51 im Abschnitt "***Sonderfall im Grid: ButtonStyle cbsImageList***".
- ShowMessages: Zu- oder Abschalten der Meldung "Programm beendet" nach Ablauf des Basic-Programmes

- SpeedButtonOptions
  - CanBeFocused: Bestimmt, ob der Button den Fokus erhalten kann (TabStop) oder nicht (kein TabStop)
  - Flat: Ist hier true gesetzt, so entfällt die dreidimensionale Darstellung. Der Button wird "flach" dargestellt, wie beispielsweise die Buttons in der Navigationsleiste der Belegbearbeitungsmasken
- StopCaption: Während des Basic-Programmablaufs angezeigte Beschriftung
- StopGlyphNr: Hinterlegung einer sog. ImageIndex-Nummer zur Darstellung eines Symbols auf dem Button während des Programmablaufes. Hinweise zur Ermittlung der ImageIndex-Nummer für das gewünschte Symbol **à** siehe unten ab Seite 51 im Abschnitt "*Sonderfall im Grid: ButtonStyle cbsImageList* ".

GDI-Basic in Masken: Click-Ereignisse bei Buttons

Es ist möglich, per GDI-BASIC auf der Maske festzustellen, ob ein auf der Maske befindlicher Button "gedrückt" wurde. Als Besonderheit ist festzuhalten, dass hierbei jeweils zwei Ereignisse zur Verfügung stehen: Die Ansprungmarken :BUTTON.BeforeClick und :BUTTON.AfterClick können im Basic der Maske gesetzt werden. "BUTTON" ist durch den Objektnamen des betreffenden Buttons zu ersetzen.

- :BUTTON.BeforeClick: Wird ausgeführt, bevor die eigentliche Funktion des Buttons ausgeführt wird. Wird in diesem Programmabschnitt BUTTON_BeforeClick_Result :=1; gesetzt, so wird abgebrochen und die eigentliche Funktion des Buttons einschl. evtl. folgendem AfterClick-Programmcode wird nicht ausgeführt.
- :BUTTON.AfterClick: Wird ausgeführt, nachdem die eigentliche Funktion des Buttons ausgeführt wurde, jedoch nicht, sofern über den Button aufgrund aktivierter Modalresult-Funktionalität bereits das Schließen der Maske ausgelöst wurde.

Somit ist es auf der einen Seite möglich, GDI-BASIC zentral auf der Maske abzulegen und vor/nach Drücken von Buttons auszuführen. Im Falle der "Basic"-Buttons kann die Funktionalität auch dazu benutzt werden, um Programmcode auf der Maske anstelle bei der Eigenschaft "GDIScript" des jeweiligen Buttons abzulegen.

Navigationsleiste (Objekttyp TpsNavBar)

Die Navigationsleiste (kurz: NavBar) erleichtert nicht nur das Durchblättern, Anspringen, Anlegen und Löschen von Datensätzen innerhalb einer Datentabelle. Sie dient grundsätzlich auch zur Steuerung von Master-Slave-Verknüpfungen, wenn mehrere Datentabellen auf einer Maske enthalten sind. Sofern man eine datensensitive Maske gestaltet (also eine Maske, die Tabellenobjekte enthält), sollte man immer eine Navigationsleiste einsetzen.
Folgende Tabelle gibt einen Überblick über die zur Verfügung stehenden Optionen der NavBar, die über die gleichnamige Property "Options" aktiviert werden können:

Option	Symbol/Button	Bedeutung/Hinweis	E	D
nbCopy		Kopieren eines Datensatzes		
nbDelete		Datensatz löschen		X
nbDesigner	Button "Designer" plus nbDruck		E	
nbDialog		Aufruf einer Maske (modal) über die NavBar		X,D
nbDruck	Buttons "Bildschirm", "Drucker" und nbExit		E	
nbExit		Schliessen der Maske		X
nbFile		Blättern/Navigieren durch die Datensätze einschliesslich Refresh-Button		X
nbFilter		Setzen eines Filters zur Laufzeit. Der Filter wird über eine Input-Box abgefragt und als Condition "Navbar-Filter" an die TsTable durchgereicht.		X
nbGridbutton		Diese Option findet im Standard im F4-Auswahlgrid Verwendung. Es kann hierüber - obwohl optisch wie nbSave aussehend - allerdings keine Datenspeicherung ausgelöst werden. Die Positionierung der Buttons ist in der NavBar rechts. Sofern die Maske modal geöffnet ist, wird diese nach Klick auf einen der beiden Buttons geschlossen. Zuschaltbare Optionen: nbDialog, nbFile, nbFilter, nbRefresh.		
nbInsert		Datensatz löschen, Datensatz anlegen (Anlage über Nummernkreis, nur bei numerischen Primärfeldern)		X
nbInsertDelete	Buttons "Einfügen", "Löschen"		E	
nbInsertUpdateDelete	Buttons "Einfügen", "Ändern", "Löschen"		E	
nbPin		Merker setzen, derzeit ohne Verwendung		
nbPlus		Datensatz anlegen (ab 3.6.2.x)		X
nbPrinter		Kann im Rechtesystem über das Systemrecht "ReportNavBarAufruf" abgeschaltet werden		
nbRefresh				
nbSave		Speichern und Verwerfen		X
nbSearch		Öffnen der F4-Auswahltabelle für die zur NavBar gehörenden DataSource (nur bei TsTable, nur bei Master)		X,D
nbUpdateDelete	Buttons "Bearbeiten", "Löschen"		E	
nbUserDef		Noch nicht verwendet		

Die Bedeutung der Spalten "E" (= Exclusiv) und "D" (= im Designer verwendbar):

- Spalte "E": Mit "E" gekennzeichnete Optionen lassen keine Kombination mit anderen Optionen/Buttons zu.
- Spalte "D":
  - ein "X" zeigt an, dass es sich um vorgegebene Funktionen handelt, welche ohne weitere Programmierung einsetzbar sind.
  - steht in dieser Spalte ein "D", so sind zusätzlich Einträge in der Unter-Property "Dialog" notwendig.
  - Besitzt die Spalte keinen Eintrag, so handelt es sich hierbei um Buttons, deren Funktion - sofern sie in den Standardmasken der Bline verwendet werden - im Sourcecode ausprogrammiert ist. Hierzu gehört beispielsweise nbCopy zum Kopieren eines Datensatzes. Was bei Klick auf diesen Button (z.B. im Artikelstamm) passiert ist also nicht über einstellbare Properties innerhalb der Maske definiert, sondern seitens GDI programmiert. Allerdings besteht die Möglichkeit durch Auswerten der Click-Ereignisse per GDI-Basic auf der Maske festzustellen, ob ein Button der Navigationsleiste "gedrückt" wurde und somit eigene Funktionen abbilden (siehe nächster Abschnitt).

GDI-Basic in Masken: Click-Ereignisse bei NavBar-Buttons

Analog zu den Buttonclick-Ereignissen beim TsButton ist es möglich, per GDI-BASIC auf der Maske festzustellen, ob ein Button der Navigationsleiste (NavBar-Button) "gedrückt" wurde. Auch hier ist festzuhalten, dass jeweils zwei Ereignisse zur Verfügung stehen: Die Ansprungmarken `:NAVBAR.BeforeClick` und `:NAVBAR.AfterClick` können im Basic der Maske gesetzt werden. "NAVBAR" ist durch den Objektnamen der in der Maske befindlichen NavBar zu ersetzen.

- `:NAVBAR.BeforeClick`: Wird ausgeführt, bevor die eigentliche Funktion des NavBar-Buttons ausgeführt wird. Wird in diesem Programmabschnitt `NAVBAR_BeforeClick_Result :=1;` gesetzt, so wird abgebrochen und die eigentliche Funktion des Buttons einschl. evtl. folgendem AfterClick-Programmcode wird nicht ausgeführt.
- `:NAVBAR.AfterClick`: Wird ausgeführt, nachdem die eigentliche Funktion des Buttons ausgeführt wurde.

Innerhalb des Basics kann über die Variable "NAVBAR_Button" ermittelt werden, welcher Button der NavBar gedrückt wurde. Ihr Wert enthält den gedrückten Button:

FIRST, PRIOR, NEXT, LAST, DELETE, INSERT, COPY, REFRESH, SAVE, CANCEL, GRID, PRINTER, FILTER, PIN, EXIT, DIALOG, DDELETE, DUPDATE, DINSERT, DDESIGNER, DRUCKER, BILDSCHIRM

und steht direkt in Zusammenhang mit den eingestellten NavBar-Optionen, die ja ggfs. diverse Buttons liefern. Z.B. resultieren aus der Option "nbFile" vier Buttons, die dem FIRST, PRIOR, NEXT und LAST entsprechen. Beispiel:

```
:NB_1.BeforeClick
   Show(NB_1_Button);
   if NB_1_Button = "NEXT" then
      NB_1_BeforeClick_Result := 1;
   endif;
exit;

:NB_1.AfterClick
   Show("Afterclick: " + NB_1_Button);
exit;
```

Durch diese Erweiterung ist es möglich, GDI-BASIC zentral auf der Maske abzulegen und vor/nach Drücken dieser Navigationsleisten-Buttons auszuführen.
Hinweise: Der Afterclick ist bei dem "EXIT"-Button unterbunden. Der Beforeclick beim "EXIT"-Button kommt vor einem `:FormClose` der Maske.

Eigenschaften der NavBar:

- DataSource: Verknüpfte Datenquelle (Datasource aufgrund der TsTable)
- Dialog: Steuerung von speziellen Eigenschaften. Hinweis: Sofern mehrzeilige Eingaben bei den Unterpunkten erforderlich sind kann ein Editor per <F4>-Taste geöffnet werden.
  - Unter-Eigenschaft **Default**: Steuern von Vorgabewerten für Datenfelder beim Anlegen eines Datensatzes in der Form "Feldname=Wert". Die Vorgabewerte sind zeilenweise anzugeben. Wichtig hierbei: Keine Hochkommata für alphanumerische Werte verwenden.

Syntax	Bedeutung	Beispiele	Bemerkung
Feldname=Wert	Zuordnung eines konstanten Wertes	Datei=KUNDEN KZLager=1 KZxyz=1 Adressgrp=200	Konstante Werte ohne Hochkommata (keine doppelten) Diese Art der Zuordnung gilt generell für das Setzen von Defaultwerten bei Datensatzneuanlage (nicht nur in Slave-Tabellen)
Feldname={Feldname}	Zuordnung eines Feldinhaltes aus der Mastertabelle	Referenz={kundennr}	Property Mastersource muß auf eine "passende" Tabelle verweisen (hier: Kunden)
Feldname=[Inkrement] Feldname=[max, Inkrement]	Zuordnung eines automatisch inkrementierenden Wertes	LfdNr=[10] LfdNr=[max, 10]	Bei Eingabe von Werten <= 0 wird Inkrement auf 1 gesetzt

  - Ein bei der Unter-Eigenschaft **Filter** hinterlegter Filtereintrag wird auf die mit der NavBar verbundene DataSource bei Aufruf der Maske "durchgereicht". Je nach Einsatzart des Tabellenobjektes (Master oder Slave) sind verschiedene Konventionen zu beachten. Ein Filter wird zwingend bei der Realisierung einer Master-Slave-Verbindung (1:n) benötigt **à** siehe auch die Beispiele im Abschnitt zur TsTable.

Tabellenobjekt	Einsatz	Syntax/Beispiel	Bemerkung
TsTable	Master	Feldname=Wert Adressgrp='200' Belegtyp = 'V' and Belegart = 'RE' and Belegnr > 200000	Angabe eines "klassischen" Filterausdrucks (Filteranteil einer where-Klausel ohne "where"). Alphanumerische Konstanten mit einfachen Hochkommata (keine doppelten). Führt zur Erzeugung einer Condition mit Namen "NavbarDialogFilter" bei der TsTable. **In diesem Falle muss die TsTable die Select-SQL anderweitig erhalten.** Ist die Option nbSearch aktiviert, wird dieser Filter auch an die F4-Auswahltabelle der NavBar durchgereicht. Ebenso an ein an dieselbe TsTable angeschlossenes EditFeld mit Button-Style ebsGridButton.

TsTable	Master	select * from Kunden Select * from Kunden where Adressgrp = '200'	Select-Statement zur Ermittlung der gewünschten Datenmenge**. Diese SQL wird komplett an die TsTable durchgereicht, aus einer ggfs. vorhandenen where-Bedingung wird zusätzlich eine Condition mit Namen "NavbarDialogFilter" erzeugt.** Ist die Option nbSearch aktiviert, wird die Filterung aufgrund einer where-Klausel auch an die F4-Auswahltabelle der NavBar oder an ein an dieselbe TsTable angeschlossenes EditFeld mit ButtonStyle ebsGridButton durchgereicht.
TsTable	Slave	select * from BelegPos where BelegTyp = 'V' and Adressnr = {KundenNr} order by DatumPr desc	Select-Statement zur Ermittlung der Slave-Datenmenge. Variablen/ Feldnamen aus der Mastertabelle werden in geschweifte Klammern gesetzt, Hochkommata (um die geschweiften Klammern) bei alphanumerischen Feldern

- Die Unter-Eigenschaft **Dialog** wird in Verbindung mit der Option nbDialog benötigt, wenn über die NavBar eine weitere Maske aufgerufen werden soll (siehe auch unten "*Objekttyp TpsDBEdit: Sonderfunktionalität ebsDialog, Property Dialog*"): Hinterlegung des Maskenalias (z.B. TFAKunden), bei eigenen Masken in der Form "ExterneMaske|Maskendateiname". Es können keine Parameter übergeben werden.
- Die Unter-Eigenschaften **Locate**, **Start**, **Return** und **Zusatz** werden in Verbindung mit der Option nbSearch benötigt, wenn über die NavBar auch die F4-Auswahltabelle zur angeschlossenen DataSource geöffnet werden soll. Im Prinzip ergeben sich hier die gleichen Einstellungsmöglichkeiten wie unter "*Objekttyp TpsDBEdit: Sonderfunktionalität ebsGridButton, Property Dialog*" (siehe auch dort):

Unter-Eigenschaft	Syntax/Beispiele	Bemerkung
Locate	true	Hier ist immer der Defaultwert "true" zu verwenden.
Start	Empfängerfeld=Senderfeld Kundennr=Kundennr	Der Auswahltabelle wird ein Startwert übergeben, damit die Auswahltabelle "korrekt" aufsetzt. Hier ist Empfänger die Auswahltabelle. Hinweis: Es kann nur ein Wert übergeben werden, nicht mehrere Werte
Return	Empfängerfeld=Senderfeld Kundennr=Kundennr	Rückgabewert aus der aufgerufenen Auswahltabelle, d.h. in der Maske soll auf den entsprechende Datensatz aufgesetzt werden. Hier ist der Empfänger also die Tabelle der Maske.
Zusatz	Weitere Informationen zur Auswahltabelle TableName=Kunden Caption=Kundenauswahl	Im Prinzip wie bei einem Editfeld als Gridbutton (s.u.). Der TableName muss angegeben werden, die anderen Angaben wie Caption (Überschrift), Loaddefs (Spaltendefinition), Aktivieren des Query-Modus etc... sind optional.

- LoadFromForm: Wird benötigt, sofern man die Maske aus einer anderen Maske aufrufen möchte (siehe auch unten unter "*Objekttyp TpsDBEdit: Sonderfunktionalität ebsDialog, Property Dialog*"). Ist hier True gesetzt, so werden beispielsweise übergebene Filterwerte

(FilterValue der Form) und Rückgabewerte (ReturnValue der Form) durch die NavBar verarbeitet.

- Masterfield: Bei einer "Slave"-Navigationsleiste wird hier der Schlüssel-Feldname der Hauptdatentabelle eingetragen z.B. Artikelnr. Ändert sich beim "Blättern" durch die Hauptdatentabelle der Feldinhalt in diesem Feld, so wird die "Aktualisierung" der Slave-daten angestossen
- Mastersource: Verbindung zur Hauptdatentabelle bei einer "Slave"-Navigationsleiste
- Options: Steuerung der verschiedenen Buttons der NavBar, siehe obige Übersichtstabelle.
- Visible: **Wichtig**: Zur korrekten Funktion muss visible immer auf *true* stehen. Nur wenn die NavBar "gezeichnet" wird, ist sie auch aktiv/in Funktion. Dabei spielt es keine Rolle, ob die NavBar durch die Options gesteuert Symbole/Buttons beinhaltet oder welche Grö-ße sie besitzt.

DBEdit (Objekttyp TpsDBEdit)

Bei diesem Objekttyp handelt es sich um das "Standardeingabefeld" zur Eingabe von Daten in die Datenfelder einer Tabelle. Über die Properties "Buttonstyle" und "Dialog" kann das Eingabefeld mit Sonderfunktionalität ausgestattet werden kann. So kann man z.B. das Feld mit einem integrierten GridButton definieren oder als ComboBox verwenden/gestalten.

Tipp: Für das Designen der Editfelder gibt es einen Hilfsdesigner. Dieser kann im jeweiligen Feld über die Tastenkombination <Strg> + <Alt> + <D> aufgerufen werden, sofern der Maskendesigner geschlossen ist. Nach Beenden des Hilfsdesigners werden die vorgenommenen Änderungen wie gewohnt über "Bildschirm speichern" gespeichert.

Zunächst werden die grundsätzlich notwendigen Properties zur Verwendung als Eingabefeld beschrieben (ButtonStyle = *ebsNone*), anschließend wird auf die Sonderfunktionalität eingegangen. Bei der Festlegung der Properties geht man zweckmäßigerweise in der Reihenfolge DataSource, DataField, IndexName vor.

- ButtonStyle: Regelt Sonderverhalten des Objekts. Vorgabeeinstellung: *ebsNone*, d.h. Standardverhalten
- CheckText: Mit Hilfe dieser Property kann für ein Eingabefeld eine Überprüfung auf gültige Eingabe definiert werden

  Man hinterlegt hier ein SQL-Statement, das einen Datensatz aus einer Relationstabelle lädt. Das Programm vergleicht nun die Eingabe mit einem Feld dieses Datensatzes. Bei Nicht-Übereinstimmung verhält sich der Editor wie folgt: Ist der ButtonStyle des Editors "ebsGridButton" wird die Auswahltabelle geöffnet, ansonsten erfolgt die Fehlermeldung "Fehlerhafte Eingabe".

  Beispiel: Eingabefeld "Preisliste" in der Belegbearbeitungsmaske:
  Property CheckText:
  ```
  select liste ausgabe from rabatte where satzart = 'PL' and Liste= :feldname
  ```
  Der Bediener gibt in diesem Feld eine "8" ein **à** Das Programm ersetzt den Parameter "feldname" durch "8" und setzt folgendes SQL-Statement ab:
  ```
  select liste ausgabe from rabatte where satzart = 'PL' and Liste= 8
  ```
  Existiert die Preisliste 8, so steht in der Ergebnisdatenmenge in der Spalte "ausgabe" ebenfalls eine "8" **à** ein Vergleich liefert "OK"

  Eine Verarbeitung weiterer Übergabewerte ausser dem über den Parameter "feldname" direkt am Editfeld eingegebenen Wert ist möglich. Diese Werte werden in geschweifte Klammern gesetzt. Beispiel: Feld Projektnr in Belegbearbeitungsmaske so steuern, dass nur Projekte des im Beleg stehenden Kunden erfasst werden können:
  select Projektnr ausgabe from Projekt where projektnr = :feldname and kundennr = {adressnr}
  Zusatzlich setzt man bei der Property "Dialog" einen Filter, so dass der F4-Auswahldialog nur diese Projekte anzeigt (kundennr = {adressnr})

- Clicked: Ist der Editor mit einem Button für Sonderverhalten ausgestattet, kann dieses auch über GDI-Basic über @Objektname.Clicked := true ausgelöst werden, z.B. `@TpsDBEdit1.Clicked := true;`
- DataField: Verknüpftes Tabellenfeld (in welchem Feld wird die Eingabe gespeichert)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das DataField)
- DropDownRows: Anzahl der sichtbaren Zeilen in der Auswahlliste bei Verwendung als ComboBox (ButtonStyle ist ebsCombo oder ebsComboList)
- EditMask: Festlegung einer Eingabemaske bzw. Formatierung (z.B. N10.2, D8, D10)
- IndexName: Hier ist eine Eingabe erforderlich, wenn das Editfeld gleichzeitig die Primary-Eigenschaft (s.u.) besitzt, also über die Neuanlage eines Datensatzes "entscheiden" soll (Beispiel: Eingabefeld für die Kundennr im Kundenstamm). Man hinterlegt eine order by-Klausel mit den Schlüsselfeldern des Primärindexes (Beispiel: "order by kundennr").

Dadurch wird es auch möglich mit den Tasten <Bild nach oben> bzw. <Bild nach unten> durch die Datensätze gemäß dieser order-by-Klausel zu blättern.

- LastField: Soll eine Eingabe sofort beim Verlassen des Feldes gespeichert werden, so ist hier *true* zu setzen
- Primary: Wird bei dem Datenbankfeld, welches einen Datensatz eindeutig identifiziert auf *true* gestellt. Bei allen anderen Datenbankfeldern bleibt die Standardeinstellung *false*. Dadurch wird gewährleistet, dass keine doppelten Datensätze angelegt werden und Datensätze (bei numerischen Feldern) automatisch angelegt werden können ("+" in Navigationsleiste oder "Feld leeren plus <Enter>"). Beispiel: Feld für die Kundennr im Kundenstamm. Erfordert zusätzlich eine order by-Klausel in der Property IndexName (s.o).

§ Überblick: Sonderverhalten aufgrund des ButtonStyle

Über die Property "ButtonStyle" kann dem Editor ein Sonderverhalten zugewiesen werden. Stellt man diese Property auf einen Wert <> ebsNone, erscheint am rechten Rand des Editors ein Button, der dieses Sonderverhalten auslöst.

Nachfolgende Tabelle gibt einen Überblick über möglichen Sonderfunktionalitäten bei einem Edit-Feld. Die Spalte Dialog zeigt an, ob zusätzlich zu den oben genannten Property-Einstellungen Einstellungen in der Property "Dialog" notwendig sind:

ButtonStyle	Button	Verhalten	Dialog
ebsNone		Standardverhalten als Eingabefeld	Nein
ebsHttp		Bei Druck auf den Button (bzw. <F4>) wird der in Windows angemeldete Webbrowser gestartet, wobei der Inhalt des Editfeldes als Adresse übergeben wird.	Nein
ebsMail		Bei Druck auf den Button (bzw. <F4>) wird der E-Maileditor gestartet, wobei der Inhalt des Editfeldes als eMail-Adresse übergeben wird. Sofern im DataSet eine CM_Adressnr per Feldname AdressID oder Adressnr vorhanden ist, wird diese verwendet.	Nein
ebsPhone		Bei Druck auf den Button (bzw. <F4>) wird bei installierter TAPI (Telephony Application Programming Interface) – Schnittstelle für Telephonie die Verbindung aufgebaut, wobei der Inhalt des Editfeldes als Nummer übergeben wird.	Nein
ebsDialog		Bei Druck auf den Button (bzw. <F4>) wird eine Maske gestartet. Es erfolgt ein modaler Aufruf der Maske	Ja
ebsGridButton		Bei Druck auf den Button (bzw. <F4>) wird eine Auswahlliste von Datensätzen geöffnet (siehe Beispiele unten).	Ja
ebsDate		Bei Druck auf den Button (bzw. <F4>) wird ein Kalenderfenster geöffnet, wobei der Inhalt des Editfeldes als Tagesdatum übergeben wird.	Nein
ebsCombo		Das Eingabefeld fungiert als ComboBox, wobei die Auswahlliste in der Property "Items" hinterlegt wird. In das Eingabefeld können Werte aus der Auswahlliste oder eigene Eingaben eingetragen werden.	Nein
ebsComboList		Verhalten wie ebsCombo, jedoch können nur Werte aus der Auswahlliste übernommen werden. Eigene Eingaben sind nicht möglich	Nein
ebsOpenFile		Bei Druck auf den Button (bzw. <F4>) wird ein Datei-Lade-Dialog geöffnet	Ja
ebsSaveFile		Bei Druck auf den Button (bzw. <F4>) wird ein Datei-Speichern-Dialog geöffnet	Ja

§ ebsGridButton: Aufruf einer Datensatzauswahl/Suchfenster

 Objekttyp TpsDBEdit: Sonderfunktionalität **ebsGridButton**, Property **Dialog**:

Bei Druck auf den Button (bzw. <F4>) wird eine Auswahlliste von Datensätzen geöffnet

Unter-Property	Typ	Bedeutung
Caption	String	Überschrift der Auswahltabelle. Ohne Angabe wird der Eintrag "Caption=…" aus der Unter-Property Zusatz verwendet (s.u.)
Default	String	Hier ohne Bedeutung
Dialog	String	Hier kann ein Maskenalias eingetragen werden (z.B. TFAKunden, bei eigenen Masken in der Form "ExterneMaske|Maskendateiname". In der Navigationsleiste der F4-Auswahltabelle ist dann ein Icon vorhanden, über welches die angegebene Maske aufgerufen werden kann. Es handelt sich hier um einen einfachen, modalen Maskenaufruf. Es können keine Übergabeparameter gesetzt werden.
Filter	String	Nur sinnvoll in Verbindung mit locate = false. Die Auswahltabelle wird unter Berücksichtigung des Filters geöffnet (Beispiel: Adressgruppenauswahl im Kundenstamm, die Satzart AG der GDIDEF wird über den Filter SATZART='AG' ausgewertet), bei dynamischen Filtern mit geschweiften Klammern (z.B. Adressgrp='{Adressgrp}').
Locate	Boolean	True: Nur sinnvoll, wenn über die Auswahltabelle ein bestimmter Satz der Mastertabelle "gefunden" werden soll (i.d.R. für Primärfeld der Mastertabelle der Maske). Steuert, dass nach Auswahl eines Satzes aus der Auswahltabelle die Datenmenge auf diesen Wert aufgesetzt wird (FindKey-Funktionalität) False: Nach Auswahl eines Satzes aus der Auswahltabelle wird lediglich ein Feld oder mehrere Felder im Datensatz gemäß den unter Return angegebenen Angaben gesetzt
Return	String	Gibt die Felder an, die nach Auswahl eines Satzes in der Auswahltabelle im Datensatz der Mastertabelle gesetzt werden (Syntax: Empfängerfeld=Senderfeld). Zwingend notwendig bei locate = false
Start	String	Zuweisung eines Startwertes, damit die Auswahltabelle korrekt "aufsetzt". Ebenfalls in der Schreibweise Empfängerfeld=Senderfeld, hier also "Feld in Auswahltabelle=Datenfeld des Editfeldes". Nicht notwendig bei locate = true Hinweis: Es kann nur ein Wert übergeben werden, nicht mehrere Werte
Zusatz	String	Steuert den Aufbau der Auswahltabelle. Hierzu können verschiedene Schlüsselwörter gesetzt werden (Hinweis: hier keine Anführungszeichen bei Konstanten notwendig) o Caption **à** Überschrift der Auswahltabelle, sofern nicht über die Unter-Property Caption gesetzt (s.o.). Beispiel: Caption=Kundenauswahl o IndexName **à** bei gleichzeitigem Query=true Angabe einer Sortierung in Form einer order by -Klausel, z.B. IndexName=order by suchname. o TableName **à** Angabe der Datentabelle, die Grundlage für die Auswahltabelle sein soll. Zwingend notwendig bei locate=false oder auch bei locate=true, sofern der Editor an eine lesende Datenmenge angeschlossen ist (z.B. in der 3.x bei einer TsTable ohne gesetzten UpdateTableName). o LoadDefs **à** Angabe der Griddefinition. Wird defaultmässig dem TableName gleichgesetzt o LoadPart **à** weitere Angabemöglichkeit einer Griddefinition o Dialog **à** gleiche Eigenschaft wie Dialog (Angabe eines Maskenalias) o DataField **à** Spalte in der Auswahltabelle, welche den Fokus erhält o Query **à** Query=true bewirkt eine für große Datenmengen optimierte Verfahrensweise, indem die SQL für die Auswahltabelle inkl. einer where-Klausel gebildet wird. Bei Query=false wird stattdessen ohne where-Klausel gearbeitet und anschliessend der "Start"-Datensatz per locate positioniert. o SessionName **à** Ermöglicht für die Datenauswahl einen Zugriff auf andere FireBird-Datenbanken (z.B. SepaMandat-Auswahl im Beleg).

Hinweis: Sofern bei den Unter-Properties mehrzeilige Eingaben erforderlich sind (wie z.B. bei Zusatz) kann zur leichteren Erfassung ein Editor per F4-Taste geöffnet werden

Sonderfall Gridbutton ohne "Datenbankanschluß":
Soll der Gridbutton nur eine Datenauswahl ermöglichen ohne dass das Editfeld an eine DataSource angeschlossen ist, z.B. in einer eigenen Maske zur Erfassung einer Selektion, so ist dies auch möglich. Als Beispiel aus der Bline sei die Auswahl der Belegnummer in der Fibuübergabe-Maske genannt.

Beispiele für Gridbuttons:
Hier die zwingend notwendigen Einstellungen bei drei Editfeldern mit GridButtons in der Standard-Kundenmaske:
Hinweis: In der Praxis ist es möglich, dass im Parameter Zusatz noch weitere Einträge vorhanden sind, entweder zur Feineinstellung oder als "Relikt" früherer Versionen.

- Eingabefeld Kundennr: Da es sich hier um das Primärfeld der Maske handelt, sind im Prinzip ausser der defaultmässig gegebenen Einstellung Locate = True keine weiteren Angaben notwendig.

Unter-Property	Wert
Default	
Dialog	
Filter	
Locate	`True`
Return	
Start	
Zusatz	

- Eingabefeld Suchname: Hier wird Return gesetzt, da man über den Button ja einen Kunden suchen möchte. Deshalb ist auch locate = true gesetzt.

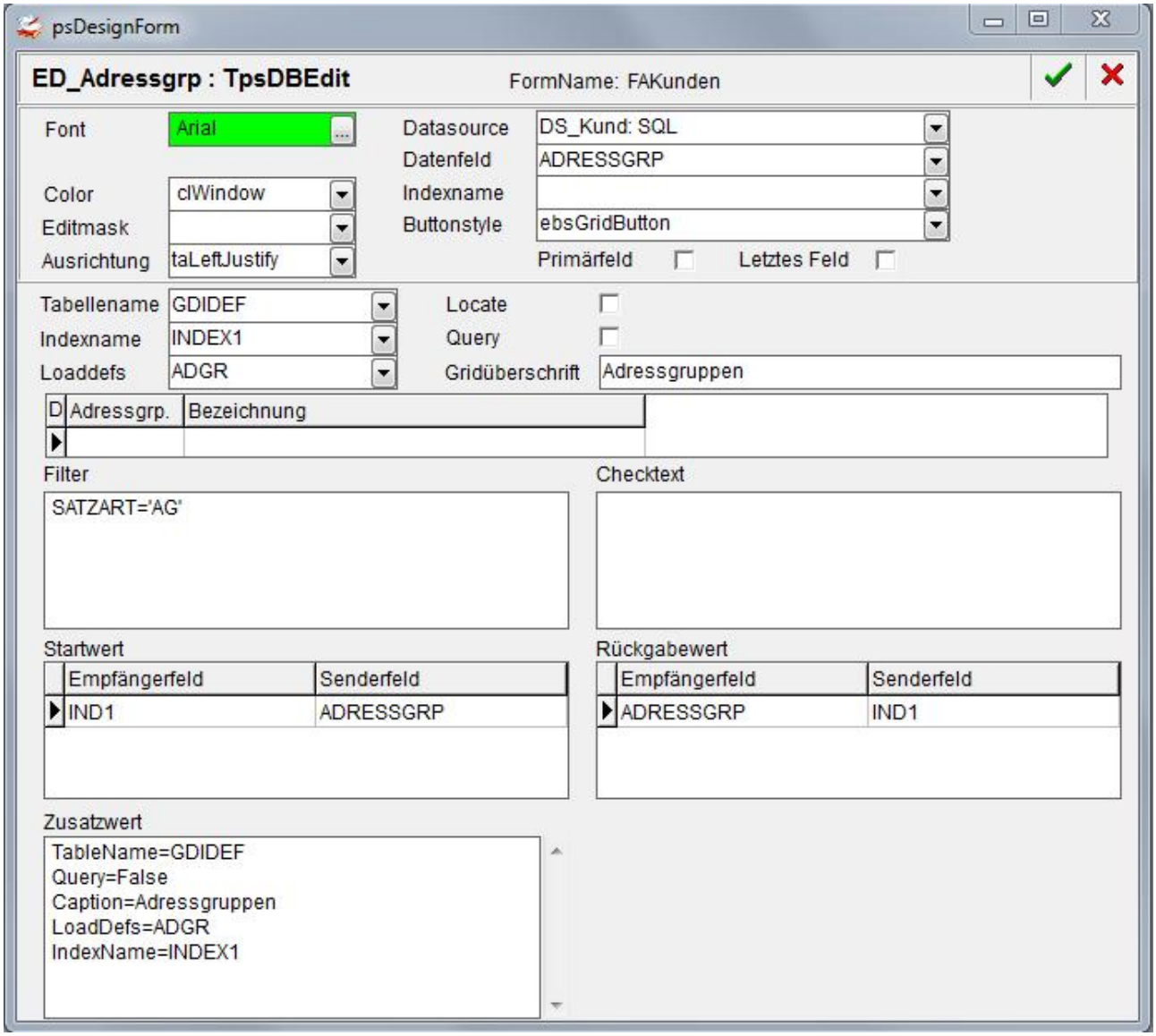
Unter-Property	Wert
Default	
Dialog	
Filter	
Locate	`True`
Return	`KUNDENNR=KUNDENNR`
Start	
Zusatz	`Caption=Suchname` `Datafield=SUCHNAME`

- Eingabefeld Adressgruppe: Dieser Button dient zur Auswahl einer Adressgruppe und deren Zuweisung an den Kundenstammsatz. Es soll also kein Kundensatz zur Bearbeitung gesucht werden (kein locate in Mastertabelle), deshalb ist Locate auf false gesetzt. Hieraus ergibt sich die Notwendigkeit, Start- und Return-Werte zu setzen. Der Filter wird benötigt, da die GDIDEF-Tabelle diverse Satzarten enthält. Query=false ist gesetzt, weil es sich um eine "kleine" Datenmenge handelt, so dass man im Gridbutton problemlos alle Datensätze anzeigen kann.

Unter-Property	Wert
Default	
Dialog	
Filter	`SATZART='AG'`
Locate	`false`
Return	`ADRESSGRP=IND1`
Start	`IND1=ADRESSGRP`

Zusatz	Tablename=GDIDEF Query=False Caption=Adressgruppen Loaddefs=ADGR

Hier die Ansicht im Hilfsdesigner (<Strg> + <Alt> + <D>):



§ ebsDialog: Aufruf einer Maske

Objekttyp TpsDBEdit: Sonderfunktionalität **ebsDialog**, Property **Dialog**:

Bei Druck auf den Button (bzw. <F4>) wird eine Maske gestartet. Es erfolgt ein modaler Aufruf der Maske, d.h. andere Masken können erst nach Schließen der aufgerufenen Maske bearbeitet werden. Bei dem Editfeld muss die DataSource (des Senders) angegeben sein, eine Angabe des DataFields ist nicht erforderlich.
Hinweis: Zum Aufruf von Masken bietet sich grundsätzlich die Funktion *Startdialog* in GDI-BASIC an. Deren Einstellungsmöglichkeiten/Parameter entsprechen weitestgehend der hier gezeigten Property Dialog beim DBEditFeld. Per Startdialog sind auch nicht-modale Maskenaufrufe möglich. Siehe separate GDI-Basic-Dokumentation.

Unter-Property	Typ	Bedeutung
Default	String	Hier können der aufzurufenden Maske Defaultwerte zugewiesen werden, die dort bei Neuanlage eines Datensatzes berücksichtigt werden, Beispiel: Adressgrp=200 Sinnvollerweise müssen die Unter-Properties Default und Filter (s.u.) aufeinander abgestimmt sein, damit bei Neuanlage der neue Satz nicht gleich "herausgefiltert" wird und somit nicht sichtbar ist.
Dialog	String	Hier gilt prinzipiell dieselbe Regel wie bei Aufruf einer Maske über die GDI-Basic-Funktion *Startdialog*: Was im Menüdesigner über Programmaufruf und Parameter 1-4 gestaltet wird, wird per Pipe-Zeichen getrennt in einem String angegeben: o Soll eine System-Maske gestartet werden, so wird hier der (interne) Programmaufruf eingetragen (z.B. TFAKunden). Diesen kann man bei einer System-Maske leicht ermitteln: Er wird im Property-Pad in der Überschrift der Property-Tabelle angezeigt, wenn man den Bildschirmdesigner aufgerufen hat. Alternativ kann man den Menüdesigner heranziehen. o Bei Aufruf einer Belegbearbeitungsmaske muss hier zusätzlich die Belegart angegeben werden in der Form "Belegartkürzel|Text". Beispiel: TFABelegVK||RE|Rechnungen o Soll eine eigene Maske gestartet werden, so muss hier das Schlüsselwort "ExterneMaske", gefolgt von dem Dateinamen der Maske gesetzt werden (durch Pipe getrennt), also in der Syntax "ExterneMaske|Maskendateiname", z.B. "ExterneMaske|Test.txt".
Filter	String	Filterwert, der bei Verwendung einer Navigationsleiste in der aufzurufenden Maske (**LoadFromForm** muss dort auf **True** gesetzt sein) für die Haupttabelle dieser Maske gesetzt wird, Beispiel Adressgrp='200' Hinweis: In den Stammdaten (Kunden, Liefer, Personal, Vertreter, Interessenten, Artikel) LoadFromForm in der Navbar auf true gesetzt.
Locate	Boolean	Hier ohne Bedeutung
Return	String	Rückgabewert in der Form Empfängerfeld=Senderfeld. Der Rückgabewert ergibt sich dann aus einem String des Aufbaues "Empfängerfeld=Wert". Bei Verwendung einer Navigationsleiste in der aufzurufenden Maske muss **LoadFromForm** auf **True** gesetzt sein. Nähere Hinweise – insbesondere bei Aufruf von Masken ohne Datenbankanschluss – siehe Abschnitt "ButtonStyle cbsMaske" für Aufruf einer Maske im Grid.
Start	String	Startwert für die Maske in der Form Empfängerfeld=Senderfeld, Beispiel Belegnr=2100003 oder Artikelnr=Ersatzart. Als Senderfeld sind also Feldnamen aus der unter DataSource angegebenen Datentabelle möglich. Hinweis: Es kann nur ein Wert übergeben werden, nicht mehrere Werte.
Zusatz	String	

§ ebsOpenFile, ebsCloseFile: Datei-Lade/Speicher-Dialog

 Objekttyp TpsDBEdit: Sonderfunktionalität **ebsOpenFile** und **ebsCloseFile**, Property **Dialog**:

Bei Druck auf den Button (bzw. <F4>) wird ein Datei-Lade-Dialog bzw. Datei-Speichern-Dialog geöffnet

Unter-Property	Typ	Bedeutung
Default	String	Vorgabe-Dateityp-Suffix. Wird an den zurückgegebenen Dateinamen angehängt, sofern im Dialog kein gültiges Suffix angegeben oder über den Filter erzeugt wurde. Hier ist nur das Suffix ohne Punkt anzugeben, z.B. dbf für dBase-Dateien
Dialog	String	Hier ohne Bedeutung
Filter	String	Filter für den Datei-Öffnen-Dialog. Hier werden die ladbaren Dateitypen angegeben. Beschreibung und Dateisuffix werden durch "|" ("|" = ASCII-Zeichen 124) voneinander getrennt aneinandergereiht. Beispiel: dBase-Dateien|*.dbf|Paradox-Dateien|*.db Wird im Datei-Öffnen-Dialog keine Dateiendung angegeben, so wird die Endung des aktuell im Dialog eingestellten Dateityps angehängt
Locate	Boolean	Hier ohne Bedeutung
Return	String	Hier ohne Bedeutung
Start	String	Startverzeichnis. Der Dialog stellt sich automatisch auf das angegebene Verzeichnis
Zusatz	String	Zusatzoptionen für den Dialog. Diese werden durch Kommata getrennt aneinandergereiht oder der Zahlenwert entsprechend addiert und angegeben. Die wichtigsten sind nachfolgend aufgeführt, eine vollständige Liste ist in der GDI-BASIC-Syntax-Dokumentation enthalten o OfHideReadOnly, Wert: 4 **à** Entfernt das Kontrollkästchen "Schreibgeschützt" aus dem Dialog o OfNoChangeDir , Wert: 8 **à** Verzeichnis wird nach Schliessen des Dialoges auf den Wert zurückgesetzt, den es vor dem Öffnen des Dialogfeldes hatte o OfPathMustExist, Wert: 256 **à** Überprüfung Pfad muss existieren o OfFileMustExist, Wert: 512 **à** Überprüfung Datei muss existieren (nur für Öffnen-Dialog) o OfEnableSizing, Wert: 524288 **à** Größe des Dialogfeldes kann geändert werden Beispiel: Options=[OfHideReadOnly,OfPathMustExist,OfFileMustExist] Analog mit gleichem Ergebnis möglich: Options=772 In Voreinstellung sind die beiden Optionen OfHideReadOnly und OfEnableSizing gesetzt. Hinweis: Die Funktion einzelner Optionen kann vom Betriebssystem abhängen.

Als Rückgabewert erhält man den Pfad einschl. Dateinamen in das Editfeld. Eine alleinige Rückgabe des Pfades ist derzeit nicht möglich.

Sofern man mehr Möglichkeiten benötigt (z.B. dynamische Vorgabe eines Verzeichnisses, Ermittlung des Dateinamenes aus dem Rückgabewert) wird der Einsatz von GDI-Basic in der Maske unter Zuhilfenahme der Funktionen *OpenDialog* bzw. *SaveDialog* empfohlen.

DBCheckBox (Objekttyp TpsDBCheck)

Bei der CheckBox handelt es sich um ein Ankreuzfeld, es gibt nur zwei mögliche Zustände (ja/nein). Den je nach Zustand in den Datensatz zurückgeschriebenen Wert kann man frei bestimmen. Einige Anwendungsfälle für DBCheckBoxen findet man beispielsweise in der Artikelstamm-Standardmaske zum Kennzeichen für Lagerartikel, Skontofähigkeit etc...

- Caption: Text zum Ankreuzfeld
- DataField: Verknüpftes Tabellenfeld (in welchem Feld soll der Eingabewert gespeichert werden)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Tabellenfeld)
- DefaultValue: Voreingestellter Zustand (cbChecked, cbGrayed (?), cbUnchecked)
  à Hinweis: Voreinstellungen für die Anlage neuer Datensätze sollten nicht über diese Property, sondern über die Angabe von Default-Werten bei der NavBar der Maske realisiert werden!
- ValueChecked: Wert oder per Semikoli getrennte Aufzählung von Werten, bei denen die CheckBox in "angekreuztem" Zustand dargestellt wird. Der Wert/der erste Wert der Aufzählung ist der Rückgabewert, welcher bei Ankreuzen im Datenfeld gespeichert wird.
- ValueUnchecked: wie vor, jedoch für den "nicht angekreuzten" Zustand der CheckBox

Hinweis: Bzgl. der Auswertung eines solchen Datenfeldes muss beachtet werden, dass in der Datenbank mindestens drei verschiedene Werte vorliegen können (z.B. kein Wert oder "0" **à** nicht aktiviert, "1" **à** aktiviert)

DBRadio (Objekttyp TpsDBRadio)

Radiobuttons dienen ebenfalls wie CheckBoxen zum Abspeichern eines fest definierten Wertes, wenn sie aktiv geschaltet werden. Im Gegensatz zu den CheckBoxen schließen sich Radiobuttons gegenseitig aus, d.h. innerhalb einer RadioButton-Gruppe kann nur ein Button aktiv geschaltet sein. Wird ein Radiobutton eingefügt, so erhält man in der Maske zunächst einen mit einer Umrandung versehenen Bereich, in welchen durch die Definition der Properties die eigentlichen Auswahlbuttons eingezeichnet werden.

- Caption: Text zur RadioButton-Gruppe
- Columns: Anzahl der Spalten für die Items-Einträge: 1 **à** einspaltige Anordnung untereinander (default), n **à** n-spaltige Anordnung nebeneinander und - sofern mehr als n Items - auch untereinander
- DataField: Verknüpftes Tabellenfeld (in welchem Feld soll der Eingabewert gespeichert werden)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Tabellenfeld)
- DefaultValue: Voreingestellter Zustand.
  à Hinweis: Voreinstellungen für die Anlage neuer Datensätze sollten nicht über diese Property, sondern über die Angabe von Default-Werten bei der NavBar der Maske realisiert werden!
- ItemIndex: Hierüber kann - sofern das Objekt keinen Datenbankanschluss besitzt – der aktuelle Wert ausgelesen/gesetzt werden (z.B. per GDI-Basic über @Objektname.Itemindex). Der Wertebereich reicht dabei von 0 (**à** erster Eintrag) bis Anzahl der Items -1 (letzter Eintrag). Sofern gar kein Eintrag/Button angewählt ist, besitzt ItemIndex den Wert "-1".

- Items: Per Mausklick kann hier ein Memofeld geöffnet werden, in welchem man die Bezeichnung der einzelnen Buttons, die zur Verfügung stehen sollen zeilenweise eingibt. Nach Bestätigen mit "ok" werden die Buttons gezeichnet.
- Values: Per Mausklick kann hier ein Memofeld geöffnet werden, in welchem man die Rückgabewerte der einzelnen Buttons zeilenweise eingibt (Reihenfolge wie bei der Property "Items").

Hinweis: Wie bei der DBCheckBox muss hier bzgl. der Auswertung eines solchen Datenfeldes beachtet werden, dass in der Datenbank die angegebenen Werte plus mindestens ein weiterer Wert vorliegen können.

GDI-Basic in Masken: Toggle-Ereigniss bei CheckBox und Radiobutton

Das sogenannte Toggle-Ereignis ist auswertbar für CheckBoxen (TpsDBCheck) und RadioButtons (TpsDBRadio). Die Syntax für die jeweils auf der Maske zu hinterlegenden Ansprungpunkte lautet:

:Objektname.Toggle

Ein unter diesem Ansprungpunkt auf der Maske hinterlegtes GDI-Basic-Programm wird ausgeführt, wenn die Checkbox oder ein Eintrag der Radiobox angeklickt oder per Tastaturbedienung (Space-Taste) der Wert geändert wird.

Beispiel:
Das folgende Beispiel ist zur Hinterlegung in der Maske des Bestellwesens (Menüpunkt Einkauf > Bestellwesen) gedacht und soll lediglich die Funktionsweise verdeutlichen. In dieser Maske gibt es sowohl Checkboxen als auch eine RadioButton-Gruppe:

```
:RA_Bestellmenge.Toggle
@Form.Caption := "Toggle Radiobutton " + System("now") + " - " +
@RA_Bestellmenge.Itemindex;
exit;

:CH_Bestell.Toggle
@Form.Caption := "Toggle Checkbox " + System("now") + " - " + @CH_Bestell.State;
exit;
```

GDI-Basic in Masken: Change-Ereigniss bei Radiobutton

Bei RadioButtons (TpsDBRadio) ist auch das sogenannte Change-Ereignis auswertbar. Die Syntax für den jeweils auf der Maske zu hinterlegenden Ansprungpunkt lautet:

:Objektname.Change

Ein unter diesem Ansprungpunkt auf der Maske hinterlegtes GDI-Basic-Programm wird ausgeführt, wenn der Wert der Radiobox per Mausklick oder per Tastaturbedienung geändert wird. Folgendes Beispiel ist wie oben beim Toggle-Ereignis für einen Test in der Bestellwesen-Maske gedacht:

```
:RA_Bestellmenge.Change
@Form.Caption := "Change " + System("now") + " - " + @RA_Bestellmenge.Itemindex;
exit;
```

DBText (Objekttyp TpsDBText)

Ein DBText-Objekt dient zur reinen Anzeige des Feldinhaltes eines Datenfeldes. Dabei befindet sich das Datenfeld befindet sich in der "Mastertabelle" der designten Maske (z.B. wäre die Anzeige des Kundennamens in den Kundenstammdaten mit Hilfe eines DBText-Objektes möglich). Anmerkung: Das DBText-Objekt findet in den Masken der Auslieferversion der Business-Line kaum Verwendung. Werden Feldinhalte angezeigt, so handelt es sich meist um MasterDBText-Objekte (s.u.).

- DataField: Verknüpftes Tabellenfeld (in welchem Feld steht der anzuzeigende Wert)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Tabellenfeld)
- EditMask: Formatierung des angezeigten Wertes (z.B. N10.2, D10)

ClientDBText (Objekttyp TpsClientDBText)

Wie das "normale" DBText-Objekt dient das ClientDBText-Objekt zur Anzeige von Feldinhalten von Datenfeldern aus der Datenbank. Beim ClientDBText befindet sich der anzuzeigende Wert jedoch nicht in der "Mastertabelle" der designten Maske, sondern in einer anderen Tabelle der Datenbank (z.B. wird im Kundenstamm die Branchennummer abgespeichert, über ein ClientDBText wird die dazugehörige Branchenbezeichnung aus der GDIDEF-Tabelle angezeigt). Die Beziehung zwischen Mastertabelle und Slavetabelle wird feldbezogen direkt beim ClientDBText-Objekt definiert.

- ClientMask: Formatierung der Datenausgabe (z.B. N10.2)
- DataField: Verknüpftes Master-Tabellenfeld (in welchem Feld steht der Schlüssel für den anzuzeigenden Wert) z.B. Feld Branche im Kundenstamm.
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Tabellenfeld, z.B. Kundentabelle)
- EditMask: Maskierung des ausgelesenen Wertes der Mastertabelle vor Übergabe an das SQL-Statement (s.u.). Wird im Normalfall nicht benötigt. Lediglich dort, wo das Referenzfeld in den beiden Tabellen unterschiedliche Formatierungen aufweist (z.B. Gross/Kleinschreibung unterschiedlich), kann eine solche Maske Sinn machen
- SessionName: Ab Version 3.5.2.x: Ermöglicht den Zugriff auf eine Externe Firebird-Datenbank über die Angabe der sog. Session. Die SQL (s.u.) wird dann auf diese Datenbank angewendet. Für den Zugriff auf die "eigene" Mandantendatenbank - was den Regelfall darstellt - wird keine Session benötigt.
- SQL: SQL-Statement zur Ermittlung der Daten. Das Ergebnis muss "Ausgabe" heißen, das Schlüsselfeld wird als "Feldname" übergeben. Beispiel:
  `"select F1 Ausgabe from gdidef where satzart = "BR" and W0 = :Feldname"`
  zur Anzeige der Branchenbezeichnung im Kundenstamm

  `"select Feldxyz Ausgabe from kunden where Kundennr = :Feldname"`
  zur Anzeige des Feldinhalts des Feldes Feldxyz aus dem Kundenstamm in einer Verkauf-Belegbearbeitungsmaske.
  Ab Version 3.5.2.x können weitere Parameter/Variablen aufgrund des "Masters" gesetzt werden, indem die Feldnamen aus der Mastertabelle als Parameternamen verwendet werden, z.B.:
  ```
  "select OPHaus Ausgabe from Beleg where BelegNr = :Feldname and BelegTyp
  = :BelegTyp and BelegArt = :BelegArt"
  ```

Hinweis bis Version 3.5.2.x: Der Objekttyp "ClientDBText" kann nicht eingesetzt werden, wenn in der where-Klausel mehrere Variablen benötigt würden. In einem solchen Falle müsste ein "normaler" DBText unter Hinzuziehung einer weiteren Table verwendet werden, wobei die Beziehung der Tabellen über einer NavBar definiert wird (Beispiel Master-Slave zwischen Beleg und Belegpos und Anzeige ausgewählter Daten der Belegpos in DBText-Feldern, also nicht in einem Grid).

DBMemo (Objekttyp TpsDBMemo)
DBMemo-Objekte sind mit DBEdit-Feldern (s.o.) vergleichbar. Der Unterschied gegenüber den DBEdit-Feldern liegt darin, dass eine mehrzeilige Erfassung möglich ist.

- DataField: Verknüpftes Tabellenfeld (in welchem Feld sollen die Eingabewerte gespeichert werden)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Memofeld)

DBMemoText (Objekttyp TpsDBMemoText)
Wie es analog zum DBEdit den DBText gibt, wurde zum DBMemo passend das DBMemoText-Objekt geschaffen. Somit lässt sich die Anzeige mehrzeiliger Texte rein lesend gestalten.

- DataField: Verknüpftes Tabellenfeld (in welchem Feld sollen die Eingabewerte gespeichert werden)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Memofeld)

DBRichEdit (Objekttyp TpsDBRichEdit)
Dieses Objekt ist mit dem Objekt DBMemo vergleichbar. Es besteht hier jedoch die Möglichkeit, RichText zu verarbeiten. Die Basis dieses Objekttyps beruht auf den RichText-Funktionen von Windows. Drückt man im DBRichEdit-Feld die <F4>-Taste, so wird der Richtexteditor eingeblendet. Vorteil: Formatierter Text kann hinterlegt werden (vgl. Zahlungszieltexte in den Basisdaten). Nachteil: Ein Zugriff auf Feldinhalte per SQL bzw. per GDI-Basic ist wegen der störenden Formatierungsanweisungen nicht oder nur eingeschränkt möglich.

- DataField: Verknüpftes Tabellenfeld (in welchem Memofeld sollen die Eingabewerte gespeichert werden)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Memofeld)
- PlainText: Wird hier *true* eingestellt, wird die Richtextfunktion ausgeschaltet. Es ergibt sich dadurch in etwa die Funktion des DBMemo. Hat man im Editor (<F4>-Taste) mit Formatanweisungen gearbeitet, so gehen diese beim Schliessen des Editors verloren. Es wird nur der reine Text gespeichert.

DBRichText (Objekttyp TpsDBRichText)
Hier handelt es sich ebenfalls um eine RichText-Komponente, allerdings beruht diese auf einer speziellen Komponente (WPTools), welche weitergehende Möglichkeiten besitzt (z.B. Einfügen von Tabelle und Grafiken). Vor- und Nachteile wie beim DBRichEdit.

- DataField: Verknüpftes Tabellenfeld (in welchem Memofeld sollen die Eingabewerte gespeichert werden)
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Memofeld)
- Die Umschaltung zwischen der Verarbeitung von formatiertem und unformatiertem Text (PlainText) erfolgt über die beiden Eigenschaften TextLoadFormat und TextSaveFormat: Wird hier anstelle von *AUTO* händisch *ANSI* eingetragen wird die Richtextfunktion ausgeschaltet. Es ergibt sich dadurch in etwa die Funktion des DBMemo. Hat man im Editor (<F4>-Taste) mit Formatanweisungen gearbeitet, so gehen diese beim Schliessen des Editors verloren. Es wird nur der reine Text gespeichert.

DBImage (Objekttyp TpsDBImage)

Mit Hilfe eines DBImage-Objektes können Bilder (BMP oder JPG-Format) datensatzabhängig abgespeichert werden (Bsp. Artikelbilder im Artikelstamm). Das DBImage wird also im Gegensatz zum Image-Objekt mit einem Datenbankfeld verknüpft. Der Dialog zum Laden, Speichern und Einbinden eines Bildes wird hierbei in der fertig designten Maske durch Doppelklick auf das DBImage-Feld geöffnet und nicht im Design-Modus wie bei Image-Objekten.

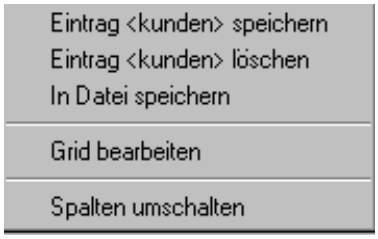
- DataField: Verknüpftes Tabellenfeld. Der Datenfeld-Typ des Tabellenfeldes entscheidet über den Ort der Speicherung und dem damit möglichen Handling:
  - GDI_BINARY **à** Das Bild wird komplett in der Firebird-Datenbank abgelegt. Diese Variante ist beispielsweise im Standard beim Bild im Artikelstamm gegeben.
  - GDI_CHAR **à** nur der Pfad/Dateiname zum Bild wird in der Datenbank gespeichert. Diese Variante ist vorteilhaft, wenn entweder viele Datensätze ein und dasselbe Bild besitzen (z.B. bei Symboldarstellungen) oder wenn das Bild hin und wieder mit anderen Anwendungen bearbeitet/aktualisiert werden soll. Als Standardpfad wird der Ordner Mandantenpfad\Bilder verwendet. Liegt das Bild in diesem Ordner wird im Tabellenfeld nur der Dateiname gespeichert, wird das Bild aus einem anderen Ordner geladen so wird der Pfad einschließlich Dateiname verwendet (benötigt ein dementsprechend "größeres" Datenbankfeld und ist problematischer bei einem Wechsel der Systemumgebung).
    Hinweis: Der Ordner Bilder unterhalb des Mandantenpfades ist ggfs. manuell anzulegen.
- DataSource: Anbindung an die Tabelle (in welcher Tabelle befindet sich das Tabellenfeld)
- Stretch: Wird diese Property auf *true* gestellt, so wird die Bildgröße automatisch in den zur Verfügung stehenden „Rahmen“ angepaßt. Dies kann u.U. zu Verzerrungen führen.

Hinweis: Bilder welche nicht nativ als BMP oder JPG vorliegen können mit dem DBImage geladen werden, werden aber dabei ins JPG-Format gewandelt. D.h. man kann beispielweise auch PNG-Dateien laden und in der Datenbank speichern, aber anschließend nicht mehr das Ursprungsbild aus der Datenbank "holen". Eine "verlustfreie" Behandlung gibt es nur bei BMPs, bei JPG-Bildern wurden Veränderungen bzgl. Komprimierung und DPI-Wert beobachtet. Allgemein unproblematisch ist die reine Anzeige von Bildern mit dem DBImage, sofern die "Pfad"-Variante per Anbindung an ein GDI_CHAR-Datenfeld gewählt wird und die Bilder nicht über den per Doppelclick geöffneten Lade/Speichern-Dialog abgespeichert werden.

PGrid (Objekttyp TpGrid)

Unter Grid versteht man allgemein die tabellarische Darstellung von Datensätzen. Dabei besteht die Möglichkeit auch ohne Bildschirmdesigner den Tabellen-/Spaltenaufbau einstellen zu können (falls die Property *LoadDefs* einen Eintrag enthält).

- Columns: Öffnet ein Hilfsfenster über welches sich die Spalten einstellen lassen, welche der Grid anzeigen soll. Bei Anwahl einer Spalte werden deren Eigenschaften im Objektinspektor zur Bearbeitung angezeigt. Die Beschreibung dieser Eigenschaften finden Sie auf der übernächsten Seite. Sofern noch keinerlei Spaltendefinition getroffen wurde zeigt das Hilfsfenster alle Datenfelder der zugrundeliegenden DataSource an. Beachten Sie, dass in diesem Zustand keine Spalten geändert werden können. Man beginnt mit der Definition der gewünschten Spalten über den "Neu-Hinzufügen-Button" des Hilfsfensters. Sofern bei dem Grid ein LoadDefs-Eintrag gesetzt wurde kann auch außerhalb des Bildschirmdesigners die Bearbeitung der Gridspalten erfolgen (rechte Maustaste auf Titelzeile des Grids, Eintrag "Grid bearbeiten" im Kontextmenü). Nähere Beschreibung siehe unten.
- DataSource: Verknüpfte Datenquelle
- FixedColor: Farbe von fixen, nicht bearbeitbaren Spalten
- FixedCounter: Anzahl der fixen, nicht bearbeitbaren Spalten, welche beim horizontalen Scrollen "stehen" bleiben (von links an gezählt).
- InputGrid: Wenn über den Grid Eingaben in die Datenbank möglich sein sollen, ist die Eigenschaft auf true zu stellen. Suchgrids dagegen werden auf false gestellt
- LoadDefs: Wird hier eine Eintragung vorgenommen (z.B. kunden), so kann das rechte Maustastenmenü des Grids aufgerufen werden.



Spaltenaufbau in Datenbank speichern (für Bedienergruppe)
Spaltenaufbau aus Datenbank löschen
Spaltenaufbau in gdf-Datei speichern (allgemeingültig)

Spaltenbearbeitungsfenster aufrufen

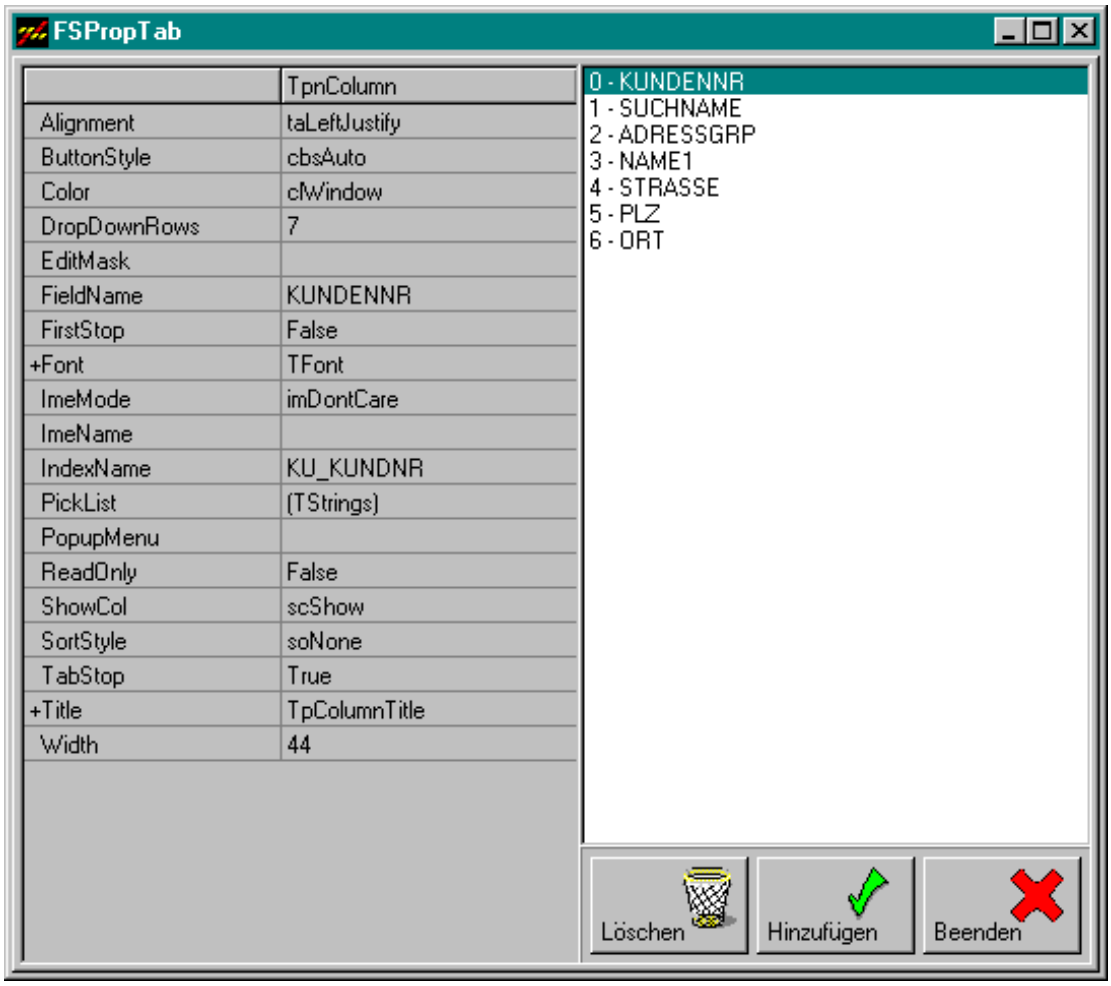
"scSecond-Spalten" ein-/ausblenden

- LoadPart: Weitere Ebene zum Abspeichern einer Spaltendefinition über das rechte Maustastenmenü, übersteuert einen LoadDef-Eintrag (vgl. Beleggrids in der Belegbearbeitung)
- MasterField: nicht verwendet
- Mastersource: Notwendig bei Slave-Suchgrids in 3.x: Stellt den Bezug zur Mastertabelle bei Suchgrids her, damit die über den Range (s.u.) in geschweiften Klammern angegebenen Parameter gefüllt werden können.
- Options: Regelt Detailfunktionen des Grids:
  - dgAlwaysShowEditor **à** erlaubt direktes Editieren beim Anspringen der Zelle (ohne F2 oder zusätzlichen Mausklick)
  - dgAlwaysShowSelection **à** ?
  - dgAppend **à** erlaubt Anfügen von Datensätzen
  - dgCancelOnExit **à** wird ein im Insertmodus befindlicher, noch "leerer" Datensatz verlassen, wird die Gridzeile entfernt
  - dgColLines **à** aktiviert die senkrechten Gitterlinien
  - dgColumnResize **à** erlaubt/verhindert die Änderung der Spaltenbreite (Spaltenbreite verschieben)
  - dgConfirmDelete **à** erlaubt/verhindert das Löschen von Datensätzen
  - dgEditing **à** erlaubt Änderungen an Datensätzen. Muss auch bei Suchgrids auf true stehen.

- dgIndicator **à** Anzeige der ersten Spalte
- dgInsert **à** erlaubt Einfügen von Datensätzen
- dgMultiSelect **à** Selektieren mehrerer Datensätze, per GDI-Basic auswertbar (Bookmarks)
- dgRowLines **à** aktiviert die waagrechten Gitterlinien
- dgRowSelect **à** ganze Zeile anwählen und markieren. In diesem Falle wird dgEditing automatisch auf false gestellt, es ist somit keine Datensatzbearbeitung, -Neuanlage und auch keine Suche möglich
- dgTabs **à** mit Tab von Zelle zu Zelle springen oder mit Tab den Grid verlassen (nächsten Tab-Stop-Element auf der Maske anspringen)
- dgTitles **à** Anzeige der Titelzeile

- Range: Hier wird bei Slave-Grids, die auf einer TsTable beruhen der Anteil der where-Klausel definiert, der immer Bestandteil der SQL sein muss. Dadurch bleibt auch bei einer Suche oder einem Umsortieren der Daten im Grid die "Grundfilterung" erhalten. Beispiel: `BelegTyp = 'V' and Adressnr = {KUNDENNR}`
- ReadOnly: Wird hier true eingestellt, hat der Grid nur "reine" Anzeigefunktion.
- SaveColumns: Der Defaultwert ist ***true***, d.h. beim Abspeichern einer Maske über den Bildschirmdesigner wird der Spaltenaufbau in der Maske selbst abgelegt. Das bedeutet, dass ein später über das rechte Maustastenmenü eingestellter und gespeicherter Gridaufbau nicht herangezogen wird. Wird SaveColumns auf ***false*** gesetzt, so wird beim Speichern der Bildschirmmaske der Spaltenaufbau des Grids nicht mitgespeichert. Alternativ kann man auch alle Gridspalten in der Griddefinition löschen und die Maske erneut speichern (TPassThroughColumn)
- ShowAll: Ist hier true gesetzt, so werden die scSecond-Spalten angezeigt (siehe auch Eigenschaft LoadDef, "Spalten umschalten" im Kontextmenü). Per GDI-Basic über `@Gridname.ShowAll := true;` "von aussen" schaltbar.
- TableName: Sofern im Grid gesucht werden soll, so benötigt er die Angabe der passenden Daten-Tabelle (Tabellenname aus der Datenbank) zur Ermittlung der Feldliste. Nur dann ist ein Suchen im Grid möglich

Die **Eigenschaften von Gridspalten** können im Bildschirmdesigner über die Property Columns beim Grid oder außerhalb des Designers per rechter Maustaste auf der Gridspaltenüberschrift (Kontextmenüpunkt "Grid bearbeiten", sofern der Grid einen Eintrag bei der Eigenschaft Loaddefs besitzt) festgelegt werden. Nachfolgende Hardcopy zeigt den per Kontextmenü aufgerufenen Hilfsdesigner:



Für jede Spalte des Grids (in der rechten Fensterhälfte zu markieren) können wichtige Einstellungen getroffen werden:

- Alignment: Text-Ausrichtung in der Spalte
- ButtonStyle: normalerweise ist hier "cbsAuto" einzustellen. Es besteht die Möglichkeit in Verbindung mit der Eigenschaft "PickList" (s.u.) andere Einstellungen zutreffen, z.B. um aus der Spalte heraus eine ComboBox oder eine Auswahltabelle zu öffnen (cbsPicklist **à** ComboBox, cbsGridButton **à** F4-Auswahltabelle, cbsBoolean **à** CheckBox, cbsMemo **à** Anzeige mehrzeiliger Datenfelder, cbsDate **à** Datumsauswahl, cbsImage **à** Anzeige von Bildern im Grid, cbsMaske **à** Aufruf einer Maske, cbsImageList **à** Anzeige kleiner Bilder aus den systemweit verfügbaren Imagelisten)
- EditMask: Formatierung des Feldinhaltes, Z.B. N10.2, D8 (8-stelliges Datum, D10 (10-stelliges Datum), zusätzlich zu den üblichen Formatierungen gibt es Formatierungsmöglichkeiten der Art R###,##0.00 (entspricht der Printmaske im Reportdesigner)
- FieldName: Name des Datenfeldes, dessen Werte in der Spalte dargestellt werden sollen

- IndexName: In Spalten, in welchen in Suchgrids eine Suche möglich sein soll muss hier ein Eintrag vorhanden sein. Sofern beim Grid die Eigenschaft LoadDefs gefüllt ist, setzt das Programm automatisch als "Dummy"-Eintrag ein Fragezeichen "?" ein
- PickList: Wird in Verbindung mit der Eigenschaft "ButtonStyle" benötigt
- ShowCol: In diesem Feld kann festgelegt werden, ob und wann eine Spalte angezeigt wird. Standardeinstellung ist "ScShow", d.h. die Spalte soll angezeigt werden. Wird für eine Spalte "ScSecond" eingestellt, so wird diese Spalte und alle weiteren Spalten mit höherer Spaltennummer erst nach Anwählen des Eintrags "Spalten umschalten" im Kontextmenü angezeigt (s.o.). Vgl. auch Eigenschaft ShowAll beim Grid
- ReadOnly: Sind in einem Grid Eingaben erlaubt (InputGrid = true), so kann über diese Eigenschaft dennoch das Editieren einzelner Spalten unterbunden werden (ReadOnly = false)
- TabStop: Regelt das Anspringen der Spalte beim Spaltenwechsel mit dem Cursor. Hinweis: In der Belegbearbeitung nicht/nur bedingt verwendbar, da hier das Programm das Anspringen der Spalten regelt
- Title: Einstellungen für die Spaltenüberschrift
- Button "Hinzufügen": Erweitern des Grids um eine weitere Spalte
- Button "Löschen": Löschen der aktuell markierten Spalte
- Button "Beenden": Schließt das Spaltenbearbeitungs-Fenster

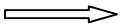
Achtung: Hierarchie beim Laden des Spaltenaufbaues:

1. Spaltenaufbau aus der Maske
2. Spaltenaufbau aus der Datenbank, bedienergruppen- und mandantenspezifisch, Tabelle GRIDDEFS
3. Spaltenaufbau aus Datei, mandantenspezifisch (im Mandantenverzeichnis\Grids unter LoadDefs.gdf abgelegte Spaltenaufbau-Beschreibung)
4. Spaltenaufbau aus Datei, programmspzifisch (im Programmverzeichnis\Grids unter LoadDefs.gdf abgelegte Spaltenaufbau-Beschreibung). In der Bline 3.x wird diese Hierachieebene für die Standardgrids verwendet, d.h. GDI liefert für jeden Grid eine GDF-Datei mit
5. Nur in der GDILine 2.x: Aus Resource innerhalb der GDILine (Quellcode). In der 2.x waren die Standardgrids Bestandteil der GDILine.exe

Anmerkung: Die von GDI vorgesehenen Spaltenaufbauten werden auch verwendet wenn das Flag "Standardgrid laden" aktiviert ist (3.x im Menüpunkt Einstellungen > Bediener-Einstellungen, 2.x unter Basisdaten | Firmendaten > Einstellungen)

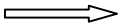
Anzeige der Herkunft der Griddefinitionen im Grid:
Wie oben erwähnt kann der Spaltenaufbau eines Grids an diversen Stellen hinterlegt sein. Gerade wenn man einen Spaltenaufbau ändern möchte, steht man somit vor der Frage: Woher wird aktuell der Spaltenaufbau geladen? Hierzu wurde eine Anzeigemöglichkeit geschaffen:
Über ein Zeichen wird in der linken oberen Ecke des Grids angezeigt, woher die Spaltendefinition geladen wurde (Sourcecode, Datenbanktabelle GRIDDEFS, GDF-Datei, MSK-Datei (Das ist auch die Ecke, mit der man die Anzahl der Zeilen im Grid umschalten kann...). Folgende Anzeigen sind derzeit möglich:

M	Griddefintion steht in der Masken-Datei (nicht möglich wenn SaveColumns auf false steht)
g	Tabelle GRIDDEFS, Spalte LoadPart (z.B. VREGRID) **à** im Standard nur bei Belegen
G	Tabelle GRIDDEFS, Spalte LoadDef (z.B. VERKAUF)
p	LoadPart.GDF im Mandantenverzeichnis (z.B. VREGRID.GDF)) **à** nur bei Belegen
P	LoadPart.GDF im Programmverzeichnis (z.B. VREGRID.GDF)) **à** nur bei Belegen
d	LoadDef.GDF im Mandantenverzeichnis (z.B. VERKAUF.GDF)
D	LoadDef.GDF im Programmverzeichnis (z.B. VERKAUF.GDF)
(leer)	Griddefintion wurde aus der Resource geladen (Grid ist unverändert, nur bei 2.x) oder es wurde kein Eintrag bei der Property LoadDefs beim TpGrid angegeben
F	Es ist zwar ein Eintrag bei der Property LoadDefs beim TpGrid vorhanden, aber es wurde keine gespeicherte Spaltendefinition gefunden
*	Die Option "Standardgrid laden" ist in den Bedienereinstellungen gesetzt.



	Kundennr.	Suchname	AGR	Name	Stra
▶	10000	COMP3000	100	Computerstudio 3000	In d
	11000	WEISS	100	Weiss Computer	Lind
	12000	GLASER	200	Katharina Glaser	Ro
	13000	MÜLLERCO	200	Müller & Co	Tal

Unveränderter Gridaufbau bei einer GDILine 2.x



G	Kundennr.	Suchname	AGR	VTNR	Name
▶	10000	COMP3000	100	1	Computerstudio 3000
	11000	WEISS	100	1	Weiss Computer
	12000	GLASER	200	1	Katharina Glaser
	13000	MÜLLERCO	200	3	Müller & Co

Geänderter Gridaufbau, in der Datenbank gespeichert ("G")

.

§ Sonderfall im Grid: ButtonStyle cbsGridButton à F4-Auswahltabelle

Im Grid lässt sich über F4 oder per Mausklick ein sogenannter Gridbutton (Auswahltabelle) öffnen. Darin kann nach Datensätzen gesucht und aus einem gewählten Datensatz ein oder mehrere Felder übernommen werden, sofern es sich grundsätzlich um eine bearbeitbare Datenmenge handelt (funktional also so ähnlich wie die Artikelauswahl in der Belegerfassung).
Hier ist die Property ButtonStyle bei der Grid-Spalte auf "cbsGridButton" zu stellen und unter der Eigenschaft "PickList" die weitere Definition zu erfassen.
Nachfolgend Mustereinträge für die Property "Picklist".

Beispiel zum Testen der Funktion im Positionsgrid der Belegbearbeitung: Hier würde eine Kundenauswahl erscheinen, eine gewählte *Kundennr* würde in das Feld *Kommnr* der Position übernommen werden (dieses Beispiel hat keinen Praxisbezug, ist aber schnell mal ausprobiert, wenn man die Kommnr als Spalte in den Positionsgrid aufnimmt, cbsGridButton einstellt und die Picklist füllt):

```
Dialog.Start=Kundennr=Kommnr
Dialog.Return=Kommnr=Kundennr
Dialog.Return=Text=Name1
Dialog.Filter=Adressgrp='200'        à Filter kann gesetzt werden!
Dialog.Zusatz=Tablename=Kunden
Dialog.Zusatz=Caption=Kundentest
Dialog.Zusatz=Query=true
```

Beispiele für die Auswahl von Kostenart, Kostenstelle und Kostenträger mit Zugriff auf die GDI-Kostenrechnung. Diese Definitionen findet man im Standard im Positionsgrid der Belegerfassung in der Bline 3.2.x. Der Zugriff auf die (externe) Firebird-Datenbank der Kostenrechnung erfolgt über die Angabe des Sessionnames (Definition im Menüpunkt "Nummernkreise" bzw. ab Version 3.2.2.222 im "GDIScriptEditor" bzw. ab Version 3.4.2.x in "TGDIServerConfig"):

```
Dialog.Start=KoArtNr=KArt
Dialog.Return=KArt=KoArtNr
Dialog.Zusatz=Tablename=KOART
Dialog.Zusatz=Caption=Kostenarten der GDI-Kostenrechnung
Dialog.Zusatz=Loaddefs=KoArt
Dialog.Zusatz=Sessionname=GDI$KORE
```

```
Dialog.Start=KstNr=Kst
Dialog.Return=Kst=KstNr
Dialog.Zusatz=Tablename=KST
Dialog.Zusatz=Caption=Kostenstellen der GDI-Kostenrechnung
Dialog.Zusatz=Loaddefs=KST
Dialog.Zusatz=Sessionname=GDI$KORE
```

```
Dialog.Start=KtrNr=Ktr
Dialog.Return=Ktr=KtrNr
Dialog.Zusatz=Tablename=KTR
Dialog.Zusatz=Caption=Kostenarten der GDI-Kostenrechnung
Dialog.Zusatz=Loaddefs=KTR
Dialog.Zusatz=Sessionname=GDI$KORE
```

Die von Eingabefeldern (TpsDBEdit) bekannte CheckText-Funktion kann auch bei Gridspalten gesetzt werden. Das entsprechende SQL-Statement ist mit einem Semikolon abzuschliessen. Beispiel: Definition einer Spalte für die Projektnr im Grid der Belegerfassung mit ButtonStyle cbsGridButton, in der Picklist folgende Einträge:

Dialog.Start=Projektnr=Projektnr
Dialog.Return=Projektnr=Projektnr
Dialog.Filter=Kundennr={adressnr}
Dialog.Zusatz=Tablename=Projekt
Dialog.Zusatz=Caption=Projekte für Kunde {adressnr}
Dialog.Zusatz=Query=true
CheckText=select projektnr ausgabe from projekt where Projektnr = :feldname and kundennr = {adressnr};

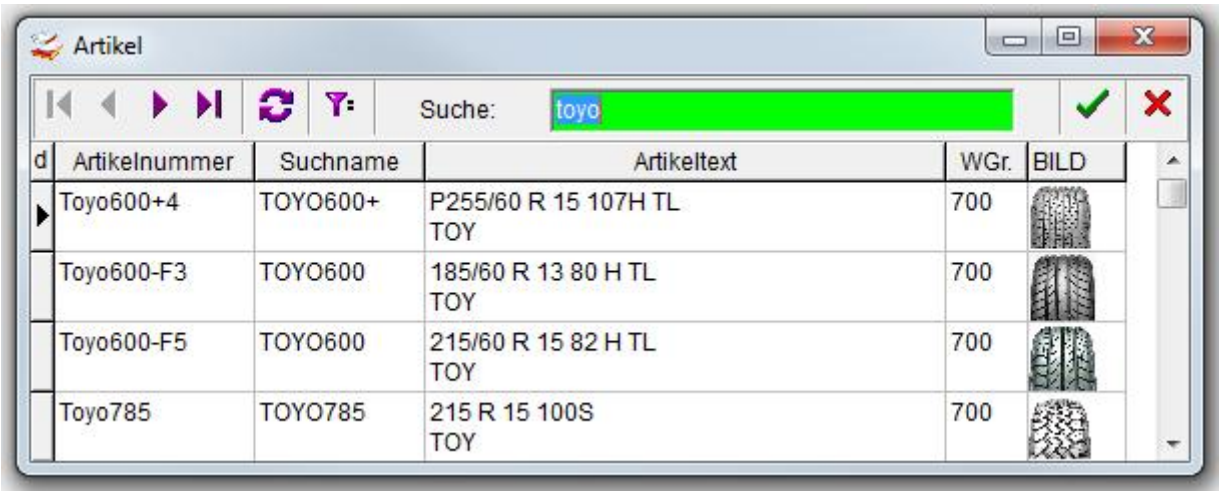
à es können nur Projekte angegeben werden, die bereits für den Kunden im System angelegt sind. Bei einer fehlerhaften Eingabe öffnet sich durch die CheckText-Funktion die Auswahl.

Hinweise für die Artikelauswahl in der Belegerfassung:

1. Der Buttonstyle ist dort cbsEllipsis und darf nicht geändert werden.
2. **Filter setzen**: Ein statischer Filter kann wie bei ebsGridbutton gesetzt werden, z.B.
   Dialog.Filter=WGR='300'
   Dynamische Filter sind jedoch nicht möglich.
3. Sonderfall **Multiselect bei Artikelauswahl** im Beleg: Normalerweise kann über die F4-Artikelauswahl im Belegpositionsgrid nur ein Artikel gewählt und in die Belegposition übernommen werden. Zur Version 3.1.0.196 wurde es ermöglicht, dass durch Hinterlegung des Eintrages
   Dialog.Zusatz=Multiselect=true
   in der Picklist der Artikelnr-Spalte auch der sog. Multiselect aktiviert und ausgewertet wird. D.h. man kann mehrere Artikel auswählen und "in einem Rutsch" in den Beleg übernehmen.

§ Sonderfall im Grid: ButtonStyle cbsImage à Bild im Grid:

Dieser Buttonstyle bietet die Möglichkeit Bilder aus der Datenbank im Grid darzustellen. Sofern ein entsprechendes Datenfeld im Grid vorhanden ist, genügt es diesen Buttonstyle einzustellen.
Beispiel: Erweitern des Artikelauswahl-Grids um die Anzeige des Artikelbildes (Datenfeld Bild)



§ Sonderfall im Grid: ButtonStyle cbsImageList à Symbolanzeige Grid:
Über diesen Spaltenstil können in der Gridspalte kleine Symbole angezeigt werden. In der Picklist der Spalte wird dabei die Zuordnung des jeweiligen Datenbank-Feldwertes zum sog. ImageIndex vorgenommen. Verfügbar ab Version 3.1.0.194.

ImageIndex	Imageliste	Interner Name
1 - 999	ImageListe CMTools	FSMenu.sTool.IL_Image
1000 - 1999	ImageListe Menü, kleine Symbole	FSMenu.IL_SmallHaupt
2000 - 2999	ImageListe Menü, große Symbole	FSMenu.IL_LargeHaupt
3000 - 3999	Eigene ImageListe 16*16 ab 3.1.0.194	FSMenu.sTool_IL_User16
4000 - 4999	Eigene ImageListe 32*32 ab 3.1.0.194	FSMenu.sTool_IL_User32

Wie ermittelt man die notwendige ImageIndex-Nummer für das gewünschte Symbol?

- Menüsymbole --> in der Menü-/Rechteverwaltung bei einem Menüpunkt die Auswahl für das "Symbol" öffnen und die Nummer des gewünschten Symboles ablesen. Zu dieser Nummer entweder 1000 (kleine Symbole) oder 2000 (große Symbole) addieren
- CMTools-Symbole --> kann im Designer für den neuen VGrid eingesehen werden --> z.B. Adress-Stamm öffnen, den Designer für VG_Adresse aufrufen, bei einer beliebigen Zeile die ComboBox "Properties > ImageIndex" öffnen und dort die Nummer des gewünschten Symboles ablesen
- Eigene Symbole --> werden in der Menü-/Rechteverwaltung unter "Benutzerdefinierte Icons" definiert. Die dort vergebene Nummer plus 3000 bzw. 4000 ergibt den ImageIndex

Werte in der Property "Picklist":

- Stretch=False

  à Verhindert, dass das Image auf die Zellengröße gestretcht wird
  à Defaultwert ist Stretch=True (=Image stretchen)

- Default=nnn

  à ImageIndex für Feldwerte, die in der Picklist nicht explizit aufgeführt sind

- Feldwert=Imageindex

  à Zuordnung eines Feldwertes (numerisch oder alphnumerisch) zu einem Image
  Bei alphanumerischen Feldwerten findet keine Unterscheidung nach Groß-/Kleinschreibung statt
  à Eine Zuordnung "=Imageindex" ist möglich und erlaubt. Damit kann dem Feldwert "leer" ein Image zugeordnet werden

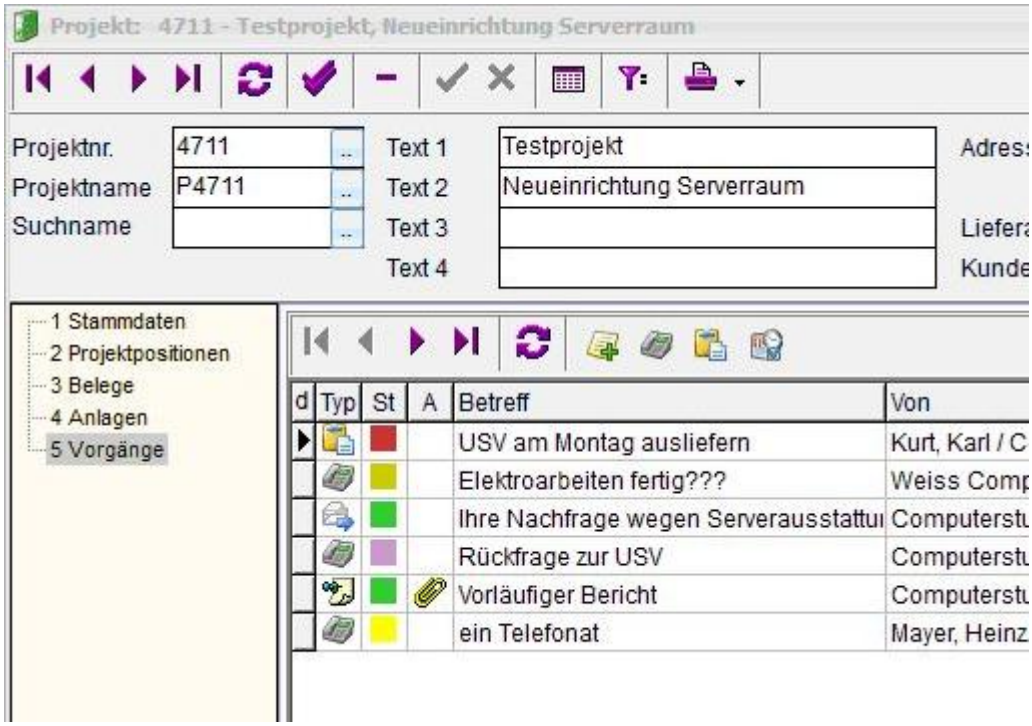
Besonderheiten:

- Gridspalten- Basic mit $GDI-Basic ist in diesem Spaltenstil nicht möglich Ein in dieser Spalte hinterlegtes Basic wird ignoriert
- Die Spalten-Property "Alignment" (taLeftJustify, taRightJustify, taCenter) wirkt sich auf die Position der auszugebenden Images aus.

Beispiele:
Grid für Vorgangsanzeige im Projekt bzw. im Beleg (CMPROJEKTVorgang.GDF bzw. CMBELEGVorgang.GDF, siehe auch unten angefügte Hardcopy aus der Projektverwaltung):
- Darstellung des Status eines Vorganges (Erledigt, Nicht begonnen, etc.) über Farbsymbole:
Stretch=False
0=150
1=153
2=148
3=149
4=155

- Darstellung des Dokumententyps durch die betreffenden Symbole:
Stretch=False
1=25
2=129
3=130
4=38
5=5
6=158
7=0
8=0
9=138
10=156

- Darstellung des KZAnhang durch das betreffende Symbol (gelbe Büroklammer):
Stretch=False
1=119



§ Sonderfall im Grid: ButtonStyle cbsBoolean à CheckBox im Grid:

Die typischen "Ankreuzfelder" können auch im Grid dargestellt werden. Dies wirkt optisch bessser als die Anzeige von "0" oder "1". Hierfür wird die Property Buttonstyle bei der betreffenden Gridspalte auf cbsBoolean gestellt. In der Property Picklist wird die Anzeige für den Grid geregelt, in dem man die Datenbankwerte den Anzeigewerten gegenüberstellt

Beispiel: Picklisteinträge für Anzeige KZLager im Grid

```
0=0
1=1
=0
```

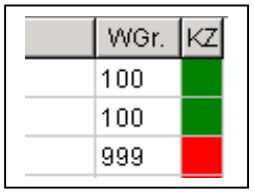
Artikelnummer	Suchname	Artikeltext	WGr.	KZLAGER
ibm32160	IBM32160	IBM DAQA 32160, 2.1 GB, EIDE	100	☑
ibm333240	IBM33240	IBM DAQA 33240, 3,2 GB, EIDE	100	☑
porto	PORTO	Paketdienst	999	☐

Es werden dabei auch mehrere "0"- bzw. "1"-Werte unterstützt: In der Picklist kann z.B. eingetragen werden:

```
0=0
=0
1=1
2=1
7=1
```

- à Dann werden für die ANZEIGE die Werte "0" und "" als False, die Werte "1", "2", "7" als True interpretiert.
- à Beim SPEICHERN (sofern es sich um einen Input-Grid handelt) wird der jeweils erste Wert der "0"- bzw. "1"- Werte verwendet.

Zusätzlich lässt sich eine Farbanzeige anstelle der Anzeige der CheckBox realisieren: In der Picklist kann zusätzlich durch Komma abgetrennt eine Farbangabe stehen, z.B.:




```
0=0,clRed
=0
1=1,clGreen
2=1
7=1
```

- à Dann werden für die ANZEIGE die angegebenen Farben anstelle der CkeckBox-Darstellung verwendet. Es genügt, die Farbe bei je einem zutreffenden Wertepaar anzugeben. Die Farbangabe kann als unter Windows verfügbare Systemfarbe textuell ("clRed") oder als Farbzahl (z.B. ein helles Gelb wäre 13172735 oder $C8FFFF) angegeben werden.
- à Das SPEICHERN (sofern es sich um einen Input-Grid handelt) funktioniert wie gehabt.

Hinweis: Für solche Anzeigen bietet sich ab der 3.x der ButtonStyle cbsImageList (s.o.) mit Zugriff auf die systeminternen Imagelisten an.

§ Sonderfall im Grid: ButtonStyle cbsMaske à Aufruf einer Maske im Grid

Aus einer Zelle des Grids heraus kann auch eine Maske modal gestartet werden. Hierfür wird die Property Buttonstyle bei der betreffenden Gridspalte auf cbsMaske gestellt. Über die Property Picklist wird definiert, welche Maske aufgerufen werden soll. Ebenso werden Start- und Returnwerte festgelegt. Die Einstellungsmöglichkeiten/Parameter entsprechen weitestgehend der GDI-BASIC-Funktion Startdialog. Da man sich im Grid auf einem Datensatz befindet, können logischerweise dessen Daten als Startparameter übergeben und - sofern es sich um eine bearbeitbare Datenmenge handelt - auch Rückgabewerte in diesen Datensatz übernommen werden.

Property Picklist, Einträge bei ButtonStyle cbsMaske:

Syntax/Beispiel	Bemerkung
Dialog.Default=Feldname=Wert	Hier können der aufzurufenden Maske Defaultwerte zugewiesen werden, die dort bei Neuanlage eines Datensatzes berücksichtigt werden. Sinnvollerweise müssen die Unter-Properties Default und Filter (s.u.) aufeinander abgestimmt sein, damit bei Neuanlage der neue Satz nicht gleich "herausgefiltert" wird und somit nicht sichtbar ist. Hinweis: Keine Hochkommata bei Konstanten erforderlich
Dialog.Default=Adressgrp=200	
Dialog.Default=Adressgrp={Adressgrp} Dialog.Default=Liefersp=1	Geschweifte Klammer bei dynamischen Feldinhalten
Dialog.Dialog=Maskenalias	Je nach Maske sind hier verschiedene Fälle zu unterscheiden: o Soll eine System-Maske gestartet werden, so wird hier der Maskenalias eingetragen (z.B. "TFAKunden" oder "TFABelegVK"). o Soll eine eigene Maske gestartet werden, so muss hier das Schlüsselwort "ExterneMaske", gefolgt von dem Namen der Maskendatei gesetzt werden (durch Pipe getrennt).
Dialog.Dialog=TFAKUNDEN	Aufruf der Kundenstamm-Maske
Dialog.Dialog=ExterneMaske|Test.txt	Aufruf der eigenen Maske "Test.txt"
Dialog.Filter=Feldname=Wert	Filterwert, der bei Verwendung einer Navigationsleiste in der aufzurufenden Maske (LoadFromForm muss dort auf True gesetzt sein) für die Haupttabelle dieser Maske gesetzt wird.
Dialog.Filter=Adressgrp='200'	
Dialog.Filter=Adressgrp={Adressgrp} and Liefersp = '1'	Zusammengesetzter Filterausdruck, geschweifte Klammer bei dynamischen Feldinhalten
Dialog.Start=Empfängerfeld=Senderfeld	Startwert für die Maske in der Form Empfängerfeld=Senderfeld, Hinweis: Bei Masken mit Datenbankanschluss kann nur ein Wert übergeben werden, nicht mehrere Werte, Hochkommata bei alphanumerischen Konstanten.
Dialog.Start=Kundennr=Adressnr	Bei Start wird hier der Inhalt des Feldes Adressnr als Kundennr an die aufgerufene Maske übergeben
Dialog.Start=Belegnr=Belegnr	
Dialog.Start=Artikelnr="ibm32160"	Übergabe der Artikelnr "ibm32160"
Dialog.Return=Empfängerfeld=Senderfeld	Rückgabewert in der Form Empfängerfeld=Senderfeld. Der Rückgabewert ergibt sich dann aus einem String des Aufbaues "Empfängerfeld=Wert". Bei Verwendung einer Navigationsleiste in der aufzurufenden Maske muss LoadFromForm auf True gesetzt sein.
Dialog.Return=Kundennr=Kundennr	à bei Auswahl des Datensatzes 15000 wird "Kundennr=15000" zurückgegeben. Sofern das Feld Kundennr im Datensatz der an den Grid angeschlossenen Tabelle vorhanden ist, wird der Wert eingetragen.
Dialog.Return=Kommnr=Kredlimit	Hier würde Kreditlimit in das Feld Kommnr übernommen werden
Dialog.Zusatz=Belegartkürzel,Text	Dialog.Zusatz wird bei Aufruf einer Belegbearbeitungsmaske benötigt. Hier muss die Belegart angegeben werden in der Form "Belegart-Kürzel,Text". Syntax für Rechnungsmaskenaufruf
Dialog.Zusatz=RE,Rechnungen	

GDI-Basic in Masken: DoppelClick-Ereignis beim Grid

Per GDI-Basic ist das Doppelclick-Ereignis im Grid auswertbar. Hierzu ist die Ansprungmarke `:Grid.DblClick` im Basic der Maske zu setzen. "GRID" ist durch den Objektnamen des betreffenden Grids zu ersetzen. Sollte programmseitig bereits ein Doppelclick verarbeitet werden, so erfolgt die Abarbeitung des `:Grid.DblClick` im zeitlichen Anschluß.

Beispiel: Aufruf des Artikelstammes aus der Belegerfassung. Der Erfassungsgrid besitzt den Objektnamen GR_POS:

```
:GR_POS.DblClick
   show("HALLO");
   A := %TA_Pos.FieldByName("Artikelnr");
   Start := "Artikelnr=" + A;
   Startdialog("TFAArtikel",true,Start);
exit;
```

Erweitertes Beispiel: Falls der Doppelclick bei der Spalte "Menge" ausgelöst wurde, öffnet sich die Positionsinfo, ansonsten der Artikelstamm:

```
:GR_POS.DblClick
   show("HALLO");
   A      := %TA_Pos.FieldByName("Artikelnr");
   Start  := "Artikelnr=" + A;
   Spalte := @GR_POS.Column.AktCol.FieldName;
   show("Spalte: " + Spalte);
   if Spalte = "MENGE" then
      @SB_Position.clicked := true;
   else
      Startdialog("TFAArtikel",true,Start);
   endif;
exit;
```

In diesem Beispiel wird also zusätzlich bestimmt, in welcher Spalte man sich befindet **à** siehe auch nächstes Kapitel.

GDI-Basic in Masken: Gridansteuerung

Über den Bildschrimdesigner lässt sich ein Grid "statisch" designen und speichern. In den meisten Fällen reicht dies völlig aus, es besteht aber hin und wieder die Anforderung, per GDI-Basic auf den Grid und seine Spalten Einfluß zu nehmen. Aus diesem Grunde wurde die Möglichkeit geschaffen, einzelne Eigenschaften der Gridspalten eines Grids (Objekt TpGrid) von außen per GDI-BASIC anzusteuern. Dadurch kann man beispielsweise

- die Anzahl der Gridspalten bestimmen
- die Eigenschaften einzelner Gridspalten auslesen
- die Eigenschaften einzelner Gridspalten setzen

Die nachfolgende Tabelle gibt einen Überblick über die zur Verfügung stehenden Funktionen:

Funktion/Syntax-Beispiel	Bemerkung
@Grid.<Property> @GR_Pos.ShowAll := true;	**Zugriff auf eine Eingenschaft des Grids** Einblenden aller für die Anzeige vorgesehenen Gridspalten, so dass auch die scSecond-Spalten angezeigt werden
@Grid.ColCount Anz := @GR_Wieder.ColCount;	**Liefert die Anzahl aller Gridspalten (sichtbare und ausgeblendete)**
@Grid.VisibleColCount Anz := @GR_Wieder.VisibleColCount;	**Liefert die Anzahl aller sichtbaren Gridspalten** (verfügbar ab Version 3.5.2.x)
@Grid.AktCol @GR_Wieder.AktCol := 3; i := @GR_Wieder.AktCol;	**Liefert oder setzt die Nummer der aktuellen Spalte.** (nur möglich bei angezeigten/sichtbaren Spalten) Setzen der aktuellen Spalte auf die dritte Spalte "Merken" der aktuellen Spaltennummer
@Grid.Column.AktCol.<Property> S := @GR_Wieder.Column.AktCol.FieldName; S := @GR_Wieder.Column.AktCol.Title.Caption; @GR_Wieder.Column.AktCol.Title.Caption := "Hallo";	**Erlaubt Zugriff auf die Eigenschaft <Property> der aktuellen Spalte des Grid** Auslesen des Feldnamen der aktuellen Spalte Auslesen der Überschrift der aktuellen Spalte Setzen der Spaltenüberschrift der aktuellen Spalte
@Grid.Column.<ColName>.FieldNr i := @GR_Wieder.Column.Adressnr.FieldNr	**Liefert die Spaltennummer der Gridspalte mit dem Feldnamen <ColName>** Nummer der Spalte mit Feldnamen "Adressnr"
@Grid.Column.<ColName>.<Property> S := @GR_Wieder.Column.Belegtyp.Title.Caption; @GR_Wieder.Column.Belegtyp.Title.Caption := "TYP"; S := @GR_Pos.Column.KZDruck.Picklist;	**Erlaubt Zugriff auf die Eigenschaft <Property> der Gridspalte mit dem Feldnamen <ColName>** Überschrift der Spalte mit Feldnamen "Belegtyp" lesen Überschrift der Spalte "Belegtyp" neu setzen. Auslesen der Picklist im Positionsgrid der Belegerfassung

Sofern nicht anders angegeben beziehen sich die aufgeführten Syntax-Beispiele auf den Grid im Menüpunkt "Autom. Belegerstellung > Belege bearbeiten". Als "aktuelle Spalte" gilt diejenige Spalte, auf welcher der Cursor positioniert ist, wenn der Grid den Fokus besitzt. Der Wertebereich der Spaltennummern liegt zwischen 1 und Anzahl der Spalten [1..ColCount].
Die Grid-Option dgRowSelect sollte auf false stehen, ansonsten kann nicht die aktuelle (markierte) Spalte ermittelt werden.

Beispiel 1: Auslesen der angezeigten Spalten und deren Überschrift
Aufgezeigt am Positionsgrid in der Belegerfassung. Dieser hat den Objektnamen "GR_Pos", der Zugriff erfolgt also über @GR_Pos.

```
S := "";
@GR_Pos.ShowAll := true;
Anz := @GR_Pos.ColCount;
i := 1;
while i <= Anz do
   @GR_Pos.AktCol := i;
   Feld  := @GR_Pos.Column.AktCol.FieldName;
   Titel := @GR_Pos.Column.AktCol.Title.Caption;
   S := S + Feld + " - " + Titel + CHR(13,10);
   i := i + 1;
endwhile;
@GR_Pos.ShowAll := false;
show(S);
```

Hinweis: Da über die Anweisung `@GR_Pos.AktCol := i;` der Fokus auf die aktuelle Spalte gesetzt wird, kann dieses Programm nur dann eingesetzt werden, wenn alle Spalten des Grids sichtbar sind. Im Standard sind einige Spalten wie KART, KST und KTR in Voreinstellung "verborgen". Deshalb wird im Basic-Programm vor der while-Schleife `@GR_Pos.ShowAll` auf true und danach wieder auf false gesetzt. Dieser "Trick" würde allerdings nicht ausreichen, wenn der Grid auch scNo-Spalten enthält.

Ab Version 3.5.2.x kann zur Ermittlung der Spaltenanzahl statt ColCount auch VisibleColCount verwendet werden. Dann müssten scSecond-Spalten nicht zwangsweise eingeblendet werden und scNo-Spalten würden "keinen Ärger" bereiten.

Beispiel 2: Verschiedene Artikelfilter bei Artikelauswahl im Beleg
In der Belegbearbeitung sollen bei der Artikelauswahl verschiedene Filter gesetzt werden können, beispielsweise um die Auswahl auf bestimmte Sortimente oder Warengruppen einzuschränken. Per Griddesign kann zwar ein Filter in der Picklist bei der Artikelnr-Spalte gesetzt werden, jedoch ist über reines Griddesign nur ein Filter möglich (statisch). Im Beispiel wird eine Lösung mit drei Basic-Buttons dargestellt (zwei Filter-Buttons plus ein "Filter aus"-Button), welche in die Picklist verschiedene Filter eintragen. Vergibt man diesen Buttons ShortCuts, so ist eine schnelle Umschaltung zwischen den Filtermöglichkeiten gegeben.

Button 1: **à** Warengruppe 100

```
WGR    := "100";
WGRTXT := " - Festplatten";
Zeile1 := "Dialog.Filter=WGR=" + CHR(39) + WGR + CHR(39);
Zeile2 := "Dialog.Zusatz=Caption=Auswahl mit Filter WGR: " + WGR + WGRTXT;
@GR_Pos.Column.Artikelnr.Picklist := Zeile1 + CHR(13,10) + Zeile2;
```

```
exit;
```

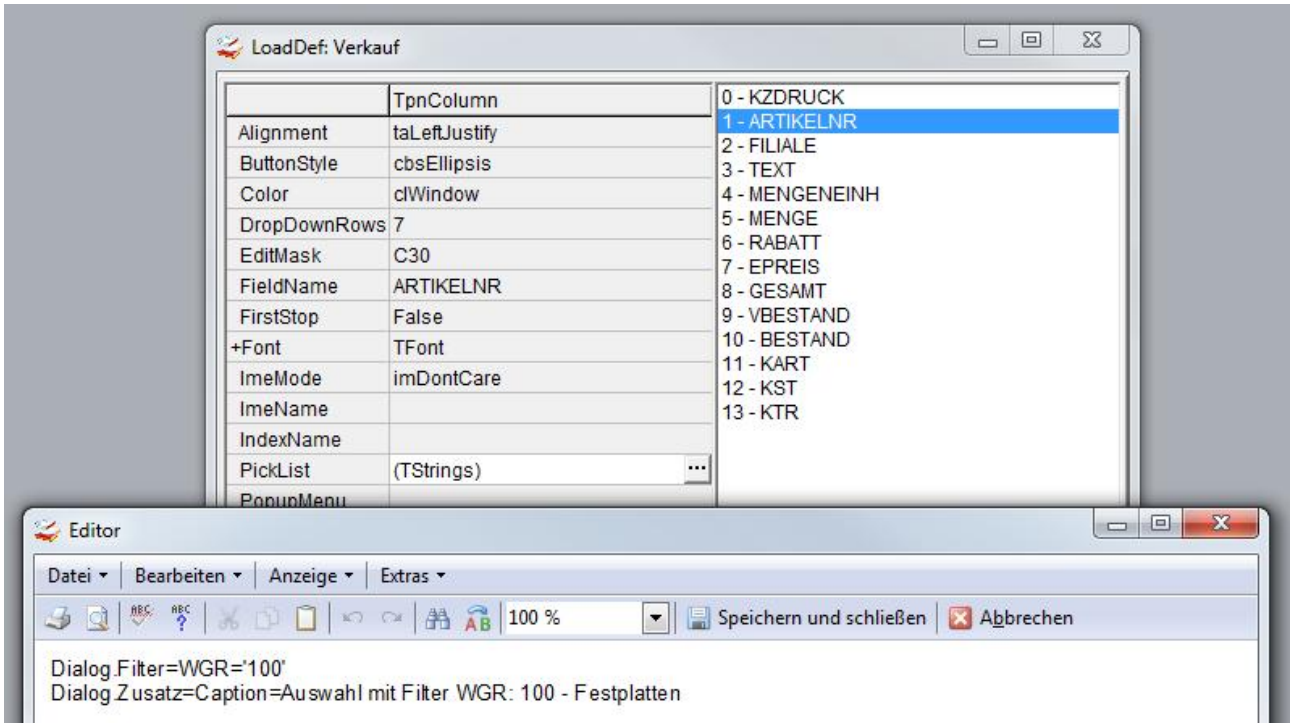
Button 2: **à** Warengruppe 200

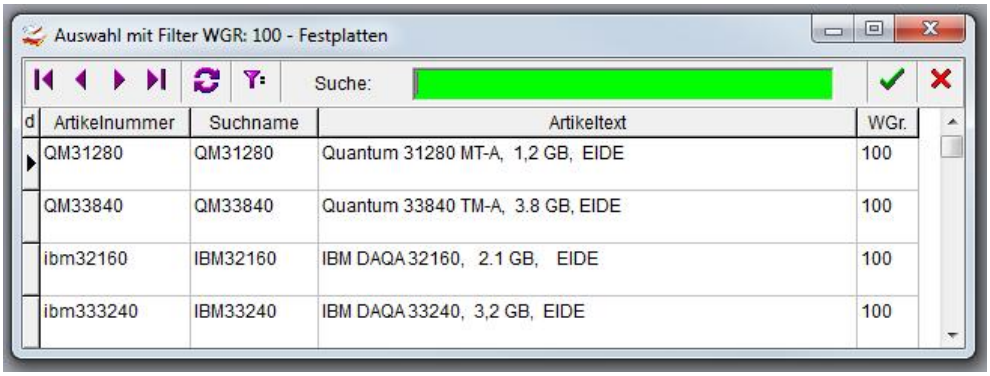
```
WGR    := "200";
WGRTXT := " - Monitore/TFTs";
Zeile1 := "Dialog.Filter=WGR=" + CHR(39) + WGR + CHR(39);
Zeile2 := "Dialog.Zusatz=Caption=Auswahl mit Filter WGR: " + WGR + WGRTXT;
@GR_Pos.Column.Artikelnr.Picklist := Zeile1 + CHR(13,10) + Zeile2;
exit;
```

Button 3: **à** Filter aus

```
@GR_Pos.Column.Artikelnr.Picklist := "";
exit;
```

Wird beispielsweise Button 1 gedrückt, so findet man anschließend in der Picklist der Artikelnr-Spalte folgenden Inhalt und die Artikelauswahl erfolgt dementsprechend gefiltert:





GDI-Basic in Masken: GDI-Basic im Grid

GDI-BASIC kann in der Picklist einer Gridspalte hinterlegt und somit für jede dargestellte Zelle abgearbeitet werden. Mit einem solchen Basic kann man

- den Zellinhalt **à** Result_Text
- die Zellfarbe **à** Result_Color
- und Attribute des Fonts in einer Zelle
  - Fontfarbe **à** Result_FontColor
  - durchgestrichen **à** Result_StrikeOut
  - kursiv **à** Result_Italic
  - fett **à** Result_Bold
  - unterstrichen **à** Result_Underline

übersteuern. Zu diesem Zweck ist die Zelle/Gridspalte an ein Datenfeld anzuschließen (ohne Datenanschluß geht es nicht, auch wenn die Zelle u.U. durch Übersteuerung per Result_Text später nicht den Inhalt des angeschlossenen Datenfeldes anzeigen wird) und in der Property Picklist ein Basic-Programm zu hinterlegen, welches mit dem Schlüsselwort **$GDI-BASIC** beginnt.

Besonderheiten:

- Sofern in der betreffenden Gridspalte aufgrund des Buttonstyle der Picklist- Eintrag anderweitig benötigt wird (cbsGridButton, cbsMaske, cbsBoolean, cbsPicklist) so ist das "$GDI-Basic" darunter in der Picklist einzutragen. Allerdings ist $GDI-BASIC nicht in Kombination mit dem ButtonStyle cbsImageList möglich.
- Wird Result_Text nicht gesetzt, wird der "originale Inhalt" des Datenfeldes angezeigt. Möchte man eine leere Zelle anzeigen, so ist Result_Text auf ein Leerzeichen/Blank zu setzen: `Result_Text := " ";`
- Um die Ausgabe zu formatieren, ist die Mask-Funktion zu verwenden. Die Angabe einer EditMask bei der Gridspalte greift nicht.
- Sofern es sich um einen Gridbutton oder Suchgrid handelt: Eine Suche in Spalten mit errechneten Werten sucht gemäß dem Inhalt des angeschlossenen Datenfeldes, gleiches gilt für die Berücksichtigung bei der Volltextsuche.

Datenzugriffe

Hier sind verschiedene Fälle zu unterscheiden:

- In einer Maske eingebundener Grid:
  Datenzugriffe erfolgen in gewohnter Weise über den Objektnamen des an den Grid angeschlossenen Tabellenobjektes, nach Möglichkeit sollte die Funktion FieldByName eingesetzt werden. Beispielsweise ist der Belegpositionsgrid in den Belegbearbeitungsmasken mit der DataSource "DS_Pos" verknüpft, welche wiederum auf das Tabellenobjekt "TA_Pos" verweist. Beispiel:

  ```
  KZDruck  := %TA_Pos.FieldByName("KZDruck");
  ```

- F4-Auswahlgrid (GridButton) und TableBrowser (GDI-BASIC-Funktion "Startdialog" mit Option "Tablebrowser"):
  Das diesen Auswahldialogen zugrundeliegende Tabellenobjekt besitzt den Namen STAB_Grid, so dass dieses den Zugriff mit der Funktion FieldByName ermöglicht. Beispiel:

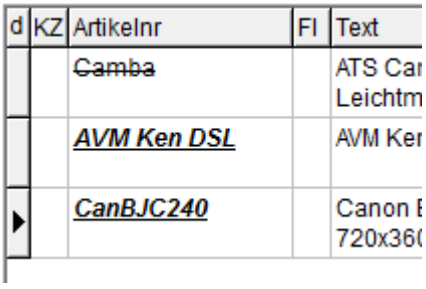
  ```
  Kundennr := %STab_Grid.FieldByName("Kundennr");
  ```

  Die im Basic angesprochenen Datenfelder müssen in der Datenmenge vorhanden sein, beim Gridbutton sind diese ggfs. als Spalten in den jeweiligen Gridaufbau auf zu nehmen.

Allgemeiner Hinweis: Sofern in dem $GDI-BASIC eigene Tabellen- oder SQL-Objekte für Datenzugriffe notwendig sind (CreateSQuery, CreateSTable), kann sich dies negativ auf die Performance auswirken (das Basic wird beim Zeichnen jeder Zelle ausgeführt).

Beispiel 1: Verschiedene Attribute für den Font
Aufgezeigt am Positionsgrid in der Belegerfassung. Der Zugriff auf die Daten der Position erfolgt über das Tabellenobjekt TA_Pos.

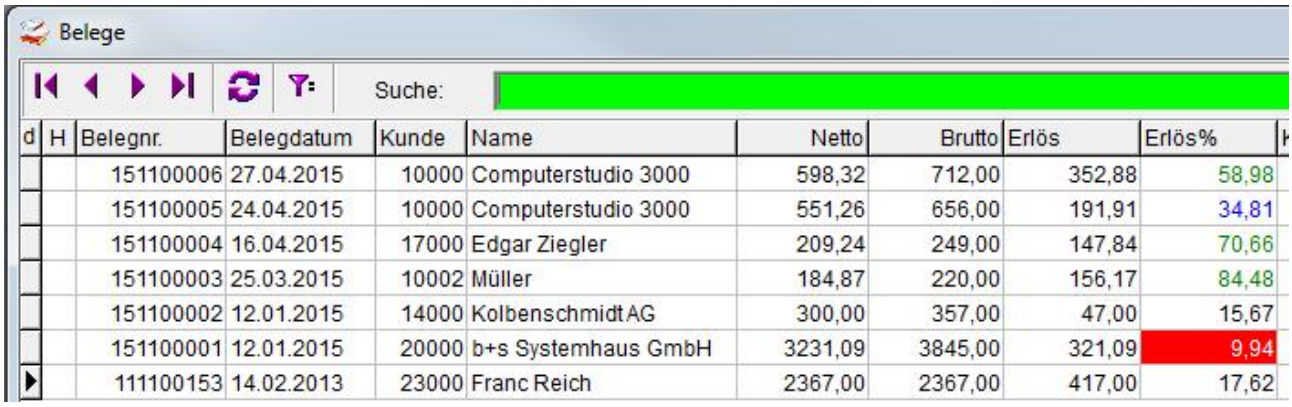
```
$GDI-Basic
   Artikelnr := %TA_Pos.FieldByName("Artikelnr");
   if Artikelnr = "Camba" then
      Result_StrikeOut := True;
   else
      Result_StrikeOut := False;
      Result_Underline := True;
      Result_Italic    := True;
      Result_Bold      := True;
   endif;
exit;
```



Beispiel 2: Erlösanzeige in F4-Belegauswahl
In der Belegbearbeitung werden auf der Karteikarte "Zahldaten" diverse Belegsummen angezeigt, u.a. auch der Erlös absolut und in Prozent. Diese beiden Werte sollen im F4-Auswahlgrid für Belege angezeigt werden.
Das Problem hierbei: Nur der absolute Erlös ist in der Datenbank in der Belegtabelle gespeichert, der prozentuale Wert wird - sofern er wie z.B. in genannter Anzeige auf der Zahldaten-Kartei benötigt wird - im Programm errechnet. Somit muss auch im F4-Auswahlgrid dieser Wert errechnet werden. Es ist zu beachten, dass man auf normierte Werte zurückgreift, der VKWert des Beleges wird deshalb aus der Summen von Warenwert und NKWarenW (Nebenkosten) bestimmt, der Erlös ist (bereits normiert) im Feld EKPreis (!) zu finden. Da diesem Beispiel der Gridbutton zugrunde liegt erfolgt der Datenzugriff über %STab_Grid.FieldByName. Zusätzlich werden unterschiedliche Erlös-Spannen durch eine Farbsteuerung hervorgehoben.

```
$GDI-Basic
Warenwert := %STab_Grid.FieldByName("Warenwert");
NKwarenW  := %STab_Grid.FieldByName("NKWarenW");
EKPreis   := %STab_Grid.FieldByName("EKPreis");  // = Erlös  in HW und Netto!
NettoHW   := Warenwert + NKwarenW;               // = VKWert in HW und Netto
EProz := 0.00;
if NettoHW <> 0 then
   EProz := Runden(100/NettoHW * EKPreis,2);
   Result_Text := Mask("N6.2",EProz);
   if EProz > 30 and EProz < 50 then
      Result_FontColor := "clBlue";
```

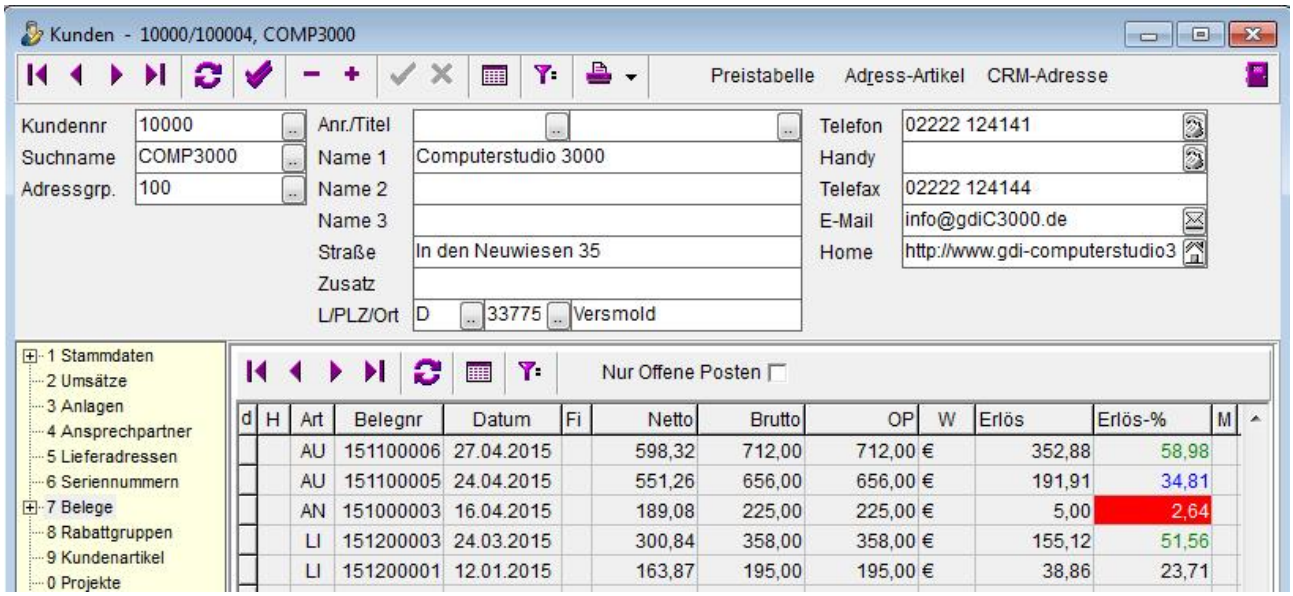
```
   endif;
   if EProz >= 50 then
      Result_FontColor := "clGreen";
   endif;
   if EProz < 10 then
      Result_Color := "clRed";
      Result_FontColor := "clWhite";
   endif;
else
   EProz := 0;
   Result_Text := " ";
endif;
exit;
```



Beispiel 3: Erlösanzeige in der Beleganzeige im Kundenstamm
Inhaltlich entspricht dieses Beispiel dem vorherigen. Hier geht es nun um die analoge Anzeige von Erlös und Erlös-Proz. in der Beleganzeige im Kundenstamm.
Der Knackpunkt: Bei dieser Beleganzeige handelt es sich um eine eingebettete Maske. Sie wird in der Bline zum einen eigenständig als "Belegtabelle" mit dem Spaltenaufbau/LoadDef "BelTab" verwendet, als auch eingebunden in den Kundenstamm mit Spaltenaufbau/LoadDef "BelTabKund" bzw. in den Lieferantenstamm mit Spaltenaufbau/LoadDef "BelTabLief". Um herauszufinden, welches Tabellenobjekt angesprochen werden muss ist die Maske des Menüpunktes "Verkauf > Zusatzfunktionen > Belegtabelle" zu betrachten. Per Bildschirmdesigner findet man darin, dass der Grid an die DataSource DS_Beleg angeschlossen ist, welche wiederum auf die TsTable QU_Beleg verweist. Folglich ist für den Datenzugriff im Grid %QU_Beleg.FieldByName zu verwenden. Für die Umsetzung selbst reicht es den Grid-Spaltenaufbau "BelTabKund" zu erweitern.

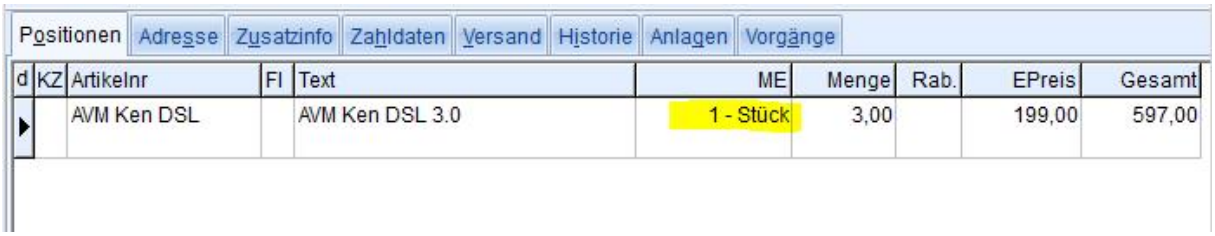
```
$GDI-Basic
Warenwert := %QU_Beleg.FieldByName("Warenwert");
NKwarenW  := %QU_Beleg.FieldByName("NKWarenW");
EKPreis   := %QU_Beleg.FieldByName("EKPreis");  // EKPreis=Erlös in HW u. Netto!
NettoHW   := Warenwert + NKwarenW;              // VKWert in HW und Netto
EProz := 0.00;
if NettoHW <> 0 then
   EProz := Runden(100/NettoHW * EKPreis,2);
   Result_Text := Mask("N6.2",EProz);
   if EProz > 30 and EProz < 50 then
      Result_FontColor := "clBlue";
   endif;
   if EProz >= 50 then
      Result_FontColor := "clGreen";
   endif;
   if EProz < 10 then
      Result_Color := "clRed";
      Result_FontColor := "clWhite";
   endif;
```

```
else
   EProz := 0;
   Result_Text := " ";
endif;
exit;
```



Beispiel 4: Anzeige der Mengeneinheitsbezeichnung im Positionsgrid der Belegerfassung.
Hier wird aus Platzgründen standardmässig nur die Nummer der Mengeneinheit angezeigt, die Bezeichnung findet man unten in der Statuszeile der Belegbearbeitungs-Maske.
Die Bezeichnung ist nicht in der Belegposition gespeichert, man muss diese per Zugriff auf die Tabelle GDIDEF aus der Datenbank "holen". Die ME-Nummer wird per %TA_Pos.FieldByName aus der Position ermittelt, für den Zugriff auf die Tabelle GDIDEF wurde ExecSQLVariant gewählt (Datenzugriff, bei dem im Programmcode kein Tabellenobjekt erzeugt werden muss).

```
$GDI-Basic
ME := %TA_Pos.FieldByName("Mengeneinh");
if ME > 0 then
   SQL := "select F1 from GDIDEF where Satzart = :Satzart and W0 = :W0";
   Erg := ExecSQLVariant(SQL,"Satzart;W0","ME",ME);
   Bez := GetFieldValue(Erg,"F1");
   Result_Text := IntToStr(ME) + " - " + Bez;
else
   Result_Text := " ";
exit;
```



Beispiel 5: Tablebrowser
Beim Tablebrowser handelt es sich um den Aufruf einer dem Gridbutton vergleichbaren (Auswahl-) Tabelle, bei welchem die Daten mit einer eigendefinierten SQL ermittelt werden. Im Beispiel wurde der Zugriff auf die Tabelle GDIDEF formuliert, dabei auf die Datensätze der Adressgruppen-Definitionen eingeschränkt (Satzart "AG"). Über den Zusatzparameter wurde eine eigene Spaltenaufbaudefinition (LoadDefs=GDIDEFDATEN) gewählt und dort über einen Picklisteintrag bei der Spalte des Feldes IND1 eine Farbsteuerung realisiert.

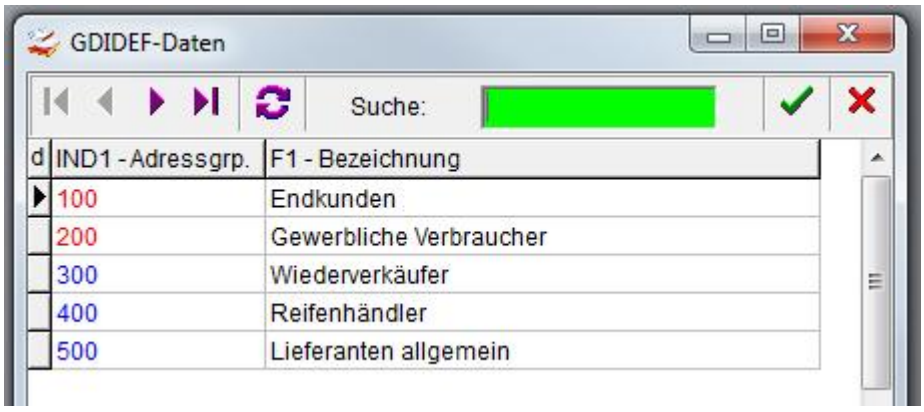
Hinweis: Man kann auch den in der Bline im Standard vorhandenen Spaltenaufbaus für die Adressgruppenauswahl (Loaddef "ADGR") verwenden. Das würde dann dazu führen, dass bei den entsprechenden Standardauswahlen im Programm die Farbsteuerung ebenfalls vorhanden wäre.

à GDI-Basic zum Aufruf des Tablebrowsers:

```
SQL := "select IND1, F1 from gdidef " +
       "where Satzart = :Satzart " +
       "order by IND1";
%Test.CreateSTable(SQL);
%Test.ParamByName("Satzart","AG");
%Test.Open;
Return := "IND1=IND1" + CHR(13,10) +
          "F1=F1";
T      := "Test";
Zusatz := "LoadDefs=GDIDEFDATEN" + chr(13,10) +
          "Caption=GDIDEF-Daten";
S      := StartDialog("TableBrowser",True,"","",Return,"",Zusatz,T);
Show(S);
%Test.Close;
%Test.FreeSTable;
```

à $GDI-Basic in der Picklist des Grids bei der Spalte IND1:

```
$GDI-BASIC
X := StrToInt(%sTab_Grid.FieldByName("IND1"));
if X > 200 then
   Result_FontColor := "clBlue";
else
   Result_FontColor := "clred";
endif;
exit;
```



Kommunikation mit Masken

Oftmals ist es notwendig, an Masken bestimmte Werte zu übergeben und im umgekehrten Falle beim Schließen von Masken Werte an den Aufrufer zurück zu übernehmen. Der Aufrufer kann hierbei

- ein Menüpunkt
- eine Maske (Button mit GDI-Basic, Editor mit *ebsDialog*, Gridspalte mit *cbsDialog*, NavBar mit *nbDialog*)
- ein GDI-Basic-Programm darstellen (GDI-Basic-Funktion *Startdialog*).

sein.
Beispiele: Kundenaufruf aus der Belegbearbeitung **à** Es wird der Kundenstammsatz passend zum Beleg aufgerufen.

Wichtig: Die Übergabe von Startwerten ist unabhängig von der Art des Maskenaufrufes möglich, Rückgabewerte erhält man grundsätzlich nur bei einem **modalen** Masken-Aufruf.

Grundsätzlich sind zwei Anwendungsfälle zu unterscheiden:

- Die aufgerufene Maske besitzt "Datenbankanschluss" aufgrund eines Tabellenobjektes (TsTable) auf der Maske (im Prinzip sind dies alle Masken, welche Datenbankbearbeitung erlauben, z.B. Kundenstamm, Artikelstamm, eigene Masken mit TsTable-Objekten....).
- Die aufgerufene Maske besitzt keinen "Datenbankanschluss" und es sollen dennoch Werte bearbeitet werden.

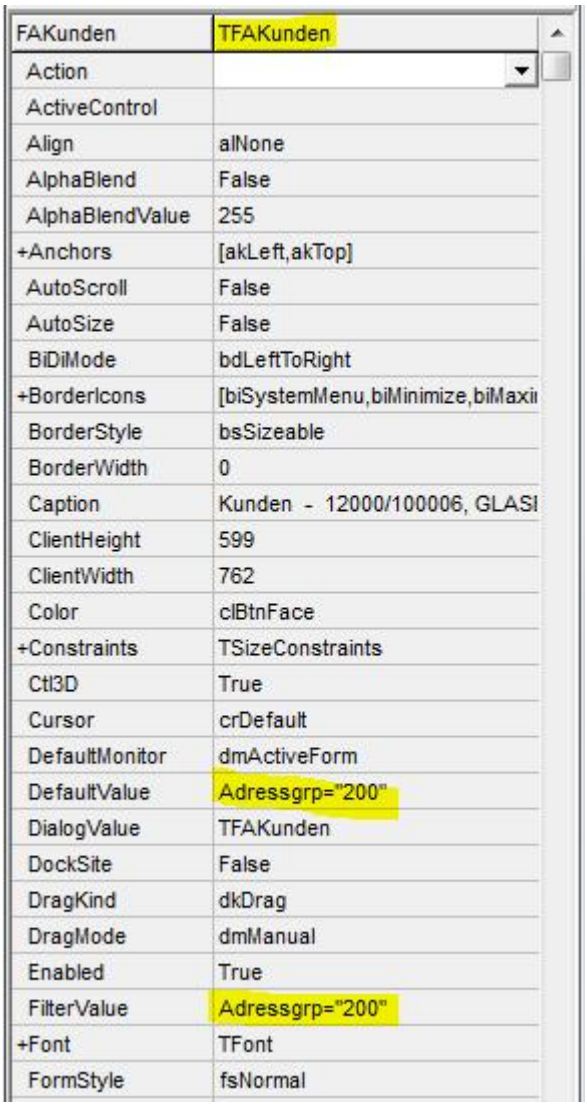
Die "Kommunikation" der Masken erfolgt per Übergabe der Start- und Rückübergabe der Return-Parameter. Beim Start werden die Properties "StartValue" und "ReturnValue" auf der Form (unterste Ebene) der aufgerufenen Maske mit den zugehörigen Parameter-Listen gefüllt.

Sofern es sich um eine Maske mit Datenbankanschluss handelt, holt sich die NavBar der Maske aus diesen Properties die Informationen und wertet diese aus.
Hinweis zu den Properties DefaultValue, FilterValue und ReturnValue einer Form: Diese werden nur bei LoadFrom-Form=true der NavBar ausgewertet/geschrieben. Der StartValue wird "immer" übergeben und ausgewertet, sofern eine NavBar mit Datenbankanschluss in der aufgerufenen Maske existiert.
D.h. die Startwerte bewirken das Setzen auf den angegebenen Datensatz und beim Schließen der Maske wird die ReturnValue- Parameterliste mit entsprechenden Werten gefüllt, so dass die aufrufende Instanz diese auswerten kann.

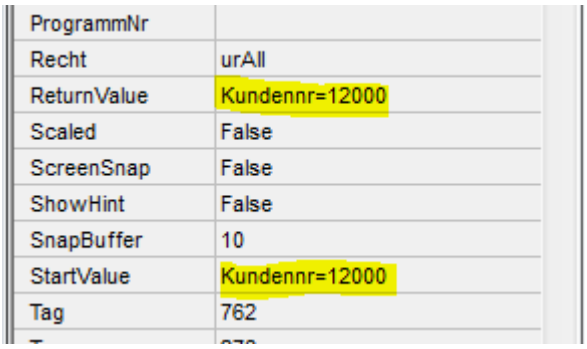
Beispiel für den Aufruf einer Maske mit Datenbankanschluss: Aufruf des Kundenstammes per GDI-Basic-Funktion Startdialog. Hier wird auf den Kunden mit der Kundennr 12000 positioniert und gleichzeitig ein Filter und ein Default bzgl. der Adressgruppe übergeben. Somit können nur Kunden dieser Adressgruppe bearbeitet/gewählt werden.

```
Dialog  := "TFAKunden";
Modal   := True;
Start   := "Kundennr=12000";
Default := "Adressgrp=""200""";   //"Adressgrp=" + CHR(34) + "200" + CHR(34);
Return  := "Kundennr=Kundennr";
Filter  := "Adressgrp=""200""";
Zusatz  := "";
S := StartDialog(Dialog, Modal, Start, Default, Return, Filter, Zusatz);
Show(S);
```

Startet man den Bildschirmdesigner in der aufgerufenen Maske, so findet man die übergebenen Werte bei den entsprechenden Properties der Form wieder (sofern man in der geöffneten Maske auf einen anderen Kunden wechselt würde sich der ReturnValue entsprechend anpassen):



FAKunden	TFAKunden
Action	
ActiveControl	
Align	alNone
AlphaBlend	False
AlphaBlendValue	255
+Anchors	[akLeft,akTop]
AutoScroll	False
AutoSize	False
BiDiMode	bdLeftToRight
+BorderIcons	[biSystemMenu,biMinimize,biMaxi
BorderStyle	bsSizeable
BorderWidth	0
Caption	Kunden - 12000/100006, GLASI
ClientHeight	599
ClientWidth	762
Color	clBtnFace
+Constraints	TSizeConstraints
Ctl3D	True
Cursor	crDefault
DefaultMonitor	dmActiveForm
DefaultValue	Adressgrp="200"
DialogValue	TFAKunden
DockSite	False
DragKind	dkDrag
DragMode	dmManual
Enabled	True
FilterValue	Adressgrp="200"
+Font	TFont
FormStyle	fsNormal



ProgrammNr	
Recht	urAll
ReturnValue	Kundennr=12000
Scaled	False
ScreenSnap	False
ShowHint	False
SnapBuffer	10
StartValue	Kundennr=12000
Tag	762

Bei einer Maske ohne Datenbankanschluss muss die Verarbeitung der übergebenen Werte in den Properties StartValue, ReturnValue etc. per GDI-BASIC, z.B. im :FormShow bzw. :FormClose der Maske erfolgen.

Beispiel: Aus einem Grid, welcher Kundenstammdaten anzeigt, soll eine eigene Maske (Dateiname " KommunikationsTestmaske.txt") ohne Datenbankanschluss geöffnet werden, welche eine Nachbearbeitung von Kundenfeldern ermöglichen soll. Im Grid wird bei einer Spalte als ButtonStyle cbsMaske eingestellt, in der Property Picklist werden folgende Zeilen hinterlegt:

```
Dialog.Dialog=ExterneMaske|KommunikationsTestmaske.txt
Dialog.Start=Kundennr=Kundennr
Dialog.Start=Suchname=Suchname
Dialog.Start=Name1=Name1
Dialog.Start=Ort=Ort
Dialog.Return=Kundennr=Kundennr
Dialog.Return=Suchname=Suchname
Dialog.Return=Name1=Name1
Dialog.Return=Ort=Ort
```

Die "KommunikationsTestmaske" selbst besitzt ein DBText-Feld für die Kundennr (unveränderlich) und drei Eingabefelder (DBEdit), dazu zwei TsButtons, welche die Eigenschaft Modalresult=1 besitzen (somit kann über diese Buttons die Maske geschlossen werden), aber nur der "OK"-Button bewirkt das Verändern einer Variable, die im FormShow initiiert und im FormClose ausgewertet wird. In der Maske wird folgendes Basic hinterlegt:

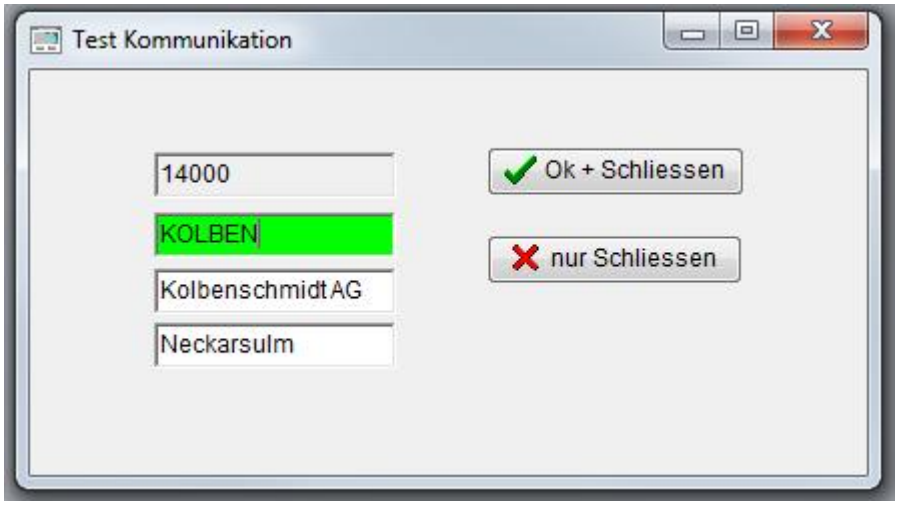
```
:FormShow
   S := @Form.Startvalue;    //Auslesen StartValue
   @TpsDBText1.Text := GetStringValue(S,"Kundennr");
   @TpsDBEdit1.Text := GetStringValue(S,"Suchname");
   @TpsDBEdit2.Text := GetStringValue(S,"Name1");
   @TpsDBEdit3.Text := GetStringValue(S,"Ort");
   Ok := false;
exit;


:TsButton1.BeforeClick
   OK := true;          // nur bei true wird im Formclose Return neu gesetzt
exit;


:FormClose
if OK = false then
   exit;
endif;
S := @Form.ReturnValue;    //Auslesen ReturnValue
$LReturn.CreateStringlist;
$LReturn.SetText(S);
i := 0;
while i < $LReturn.Count do
   Zeile := $LReturn.Element(i);
   if GetLValue(Zeile) = "Kundennr" then
         $LReturn.SetElement(i, "Kundennr=" + @TpsDBText1.Text);
   endif;
   if GetLValue(Zeile) = "Suchname" then
         $LReturn.SetElement(i, "Suchname=" + @TpsDBEdit1.Text);
   endif;
   if GetLValue(Zeile) = "Name1" then
         $LReturn.SetElement(i, "Name1=" + @TpsDBEdit2.Text);
   endif;
   if GetLValue(Zeile) = "Ort" then
         $LReturn.SetElement(i, "Ort=" + @TpsDBEdit3.Text);
   endif;
   i := i + 1;
endwhile;
@Form.ReturnValue := $LReturn.GetText;   //Neusetzen ReturnValue
$LReturn.FreeStringlist;
exit;
```

Alternativ könnte man das wie folgt formulieren, sofern man von einer festen Anzahl von Rückgabewerten ausgeht (also nicht berücksichtigt wird, welche Rückgabewerte der Aufrufer "angefordert" hat):

```
:FormClose
if OK = false then
   exit;
endif;
@Form.ReturnValue := "Kundennr=" + @TpsDBText1.Text  + CHR(13,10)
                   + "Suchname=" + @TpsDBEdit1.Text  + CHR(13,10)
                   + "Name1="    + @TpsDBEdit2.Text  + CHR(13,10)
                   + "Ort="      + @TpsDBEdit3.Text;
exit;
```



Hinweis: Im gezeigten Beispiel wurde auf sprechende Objektnamen verzichtet, für den praktischen Einsatz sollte man beim Design Objektnamen vergeben, z.B. das Objekt TpsDBEdit1 umbenennen in ED_Suchname.

Datenformate/EditMask

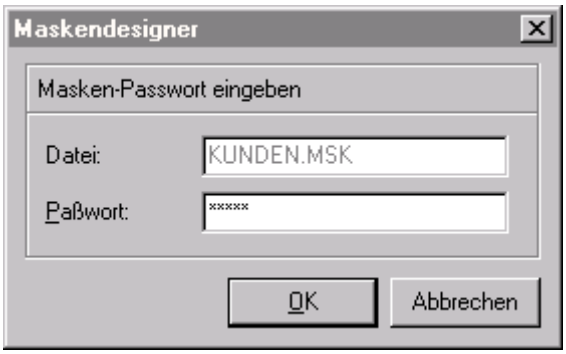
Bei Eingabefeldern findet man die Property EditMask. Mit dieser kann festgelegt werden, in welcher Form Eingaben in einem Feld möglich sind und in welchem Format Daten angezeigt werden sollen. Ohne Angabe einer EditMask wird die Definition des Datenfeldes herangezogen.

EditMask	Beispiel	Bemerkung
C[n]	C20	alphanumerisch, n = beliebige Zahl
L[n]	L20	nur Kleinbuchstaben, n = beliebige Zahl
U[n]	U10	nur Großbuchstaben, n = beliebige Zahl
M[mmmm]	M##,##	genaueste Maskenvorgabe, vgl. das Feld "Selektion" im Artikelstamm
D[8,10]	D10	Datumsformat, entweder D8 oder D10
N[n.n]	N7.2	Numerisch mit Vor- und Nachkommastellen, n = beliebige Zahl
n[n.n]	n7.2	Wie vor, jedoch werden auch Nullwerte entsprechend formatiert ausgegeben
A[nzzzzzzz]	A201+3KL	n Zeichen aus der angegebenen Zeichenauswahl, 1 <= n <= 9 bei n = 0 beliebige Anzahl von Zeichen eingebbar
P[n]	P5	Paßwort, 1 <= n <= 9, ermöglicht verdeckte Eingaben

Maskenschutz

Es besteht die Möglichkeit, designte Masken zu schützen. Hierfür wurden auf Maskenebene zwei Properties eingeführt:

- PASSWORT: Wird hier ein Eintrag vorgenommen, wird die Maske beim Speichern verschlüsselt d.h. sie kann von aussen nicht mehr gelesen werden (auch bei Speichern als TXT). Will man in dieser Maske den Bildschirm-Designer aufrufen, so wird dieses Passwort abgefragt.
- PROGRAMMNR: Per Kommata getrennte Liste der Programmnummern (Namenseinträge), für welche diese Maske verwendet werden darf. Ohne Eintrag erfolgt keine Prüfung.



Suchen im Grid, Zusammenfassung der Voraussetzungen

- Suchen im "Master"-Grid, Zusammenfassung der Voraussetzungen:
  - Verwendung einer Query bei 2.x bzw. einer TsTable bei 3.x als Tabellenobjekt. Bei der TsTable sind die AllowedUpdateKinds auf false zu stellen (= nur lesender Zugriff)
  - Verwendung einer NavBar mit dem "grundlegenden" SQL-Statement
  - Verwendung eines Grids mit:
    - Angabe eines TableName (damit die Feldliste ermittelt werden kann)
    - Verwendung einer Loaddefs-Definition (ohne LoadDefs muss in den Suchspalten der Eintrag "?" bei der Property IndexName hinterlegt sein)
    - die Eigenschaft InputGrid muss auf false eingestellt sein (es sind keine Eingaben in die Datenbank möglich)
    - bei den Options des Grids muss dgEditing auf true stehen
    - der grundlegende Anteil der where-Klausel der bei der NavBar hinterlegten SQL muss bei der Eigenschaft Range hinterlegt werden (aber ohne das Schlüsselwort "where")

- Suchen im "Slave"-Grid, Zusammenfassung der Voraussetzungen:
  - Verwendung einer Query bei 2.x bzw. einer TsTable bei 3.x als Tabellenobjekt. Bei der TsTable sind die AllowedUpdateKinds auf false zu stellen (= nur lesender Zugriff)
  - Nur 2.x: Die Slave-Query muss ebenso über die Eigenschaft DataSource mit der Mastertabelle verbunden sein.
  - Verwendung einer NavBar mit dem "grundlegenden" SQL-Statement und Anbindung an die Mastertabelle über MasterSource und MasterField
  - Verwendung eines Grids mit:
    - Angabe eines TableName (Name der Datenbank-Tabelle) für die Feldlisten-Ermittlung
    - Verwendung einer Loaddefs-Definition (ohne LoadDefs muss in den Suchspalten der Eintrag "?" bei der Property IndexName hinterlegt sein)
    - die Eigenschaft InputGrid muss auf false eingestellt sein (es sind keine Eingaben in die Datenbank möglich)
    - bei den Options des Grids muss dgEditing auf true stehen
    - der grundlegende Anteil der where-Klausel der bei der NavBar hinterlegten SQL muss bei der Eigenschaft Range hinterlegt werden (aber ohne das Schlüsselwort "where")
    - der Grid muss über die Eigenschaft MasterSource mit der Mastertabelle verbunden sein

Rechtesteuerung in (eigenen) Masken

Damit über das Menü-/Rechtesystem eingestellte Berechtigungen in Masken greifen, muss bei den TsTable-Datenobjekten der Maske die Eigenschaft "RechtAktiv" auf "true" eingestellt sein (Defaultwert ist "false"). Dann ergibt sich folgende Funktionsweise:

- Die über das Menü-/Rechtesystem vorgebene Berechtigung wird beim Start/Aufruf der Maske an diese durchgereicht und ist bei der Property "Recht"der Form zu finden:
  - urAll **à** alle Rechte (hellgrün)
  - urNoDelete **à** Einfügen/Ändern (dunkelgrün)
  - urUpdate **à** Ändern (gelb)
  - urReadOnly **à** nur Lesen (türkis)
  - urNone **à** keine Rechte (rot) = Maske wird erst gar nicht geöffnet
- Sofern die Property "RechtAktiv" bei der TsTable aktiviert ist, wird die Berechtigung ausgewertet

Diese Verfahrensweise gilt grundsätzlich bei eigenen Masken und findet auch in GDI-Standard-Masken Anwendung, jedoch gibt es insbesondere in der Belegerfassung spezielle (über Programmcode) realisierte Steuerungen (z.B. wenn ein Beleg durch Ausdruck, FIBU-Übergabe oder Abschluss als nicht bearbeitbar gelten soll).

In diesem Zusammenhang ist auch die Property "AllowedUpdateKinds" bei der TsTable zu beachten:
Diese regelt, was die TsTable grundsätzlich "darf", unabhängig vom Rechtesystem und somit von der Property "RechtAktiv" der TsTable. Benötigt man beispielsweise eine reine Anzeige von Daten und/oder eine Suche in einem Grid, so sollte man bei der TsTable alle drei AllowedUpdateKinds - Unterpunkte

- ukInsert
- ukDelete
- ukModify

auf "false" stellen.

Zusatzhinweis: Es ist derzeit nicht möglich, in einer Maske bei an sich voller Bearbeitbarkeit einzelne Eingabe-Felder für die Bearbeitung zu sperren. Die Editoren besitzen zwar die "ReadOnly"-Eigenschaft, diese ist allerdings für diesen Zweck nicht funktional.

Probleme/Sonstiges/Tipps zum Bildschirmdesigner

- Beschriftung in Titelleiste einer Maske **à** Dies erfolgt auf der untersten Ebene der Maske, der sog. "Form". Direkt nach Aufruf des Designers befindet man sich auf dieser Ebene. Hier kann unter Caption der Titel der Maske eingetragen werden.
- GDI-BASIC auf Maskenebene **à** Dies erfolgt ebenfalls auf der untersten Ebene der Maske, der sog. "Form" bei der Property "GDIBasic". Direkt nach Aufruf des Designers befindet man sich auf dieser Ebene.
  Zugriff auf Daten per GDI-BASIC: **à** Sofern in der Maske TsTables verwendet werden, genügt die Syntax "Tabellenname.Feldname" zu Zugriff auf den aktuellen Datensatz (Beispiel: `KD := Kunden.Kundennr`) oder "Objektname.Feldname" (z.B. `KD := TA_Kund.Kundennr`). I.d.R. können auch die üblichen Tabellen-Funktionen wie gewohnt über ein vorangestelltes %-Zeichen verwendet werden (`%TA_Kund.Edit, KD := %TA_Kund.FieldByName("Kundennr")`). Ein Erzeugen eines Tabellenobjektes per GDI-BASIC ist nicht erforderlich.
- Zugriff auf Maskenelemente (Maskenoberfläche) per GDI-BASIC: **à** Objekte können über die Syntax "@Objektname.Objekteigenschaft" angesprochen werden.
  Beispiel:
  ```
  @TpsLabel1.Caption := "Hallo";
  ```

  Beim Setzen von (konstanten) Properties muss man beachten, dass hier Hochkommata gesetzt werden müssen.
  Beispiel:
  ```
  if Bedingung = "1" then
     @PN_Kopf.Color := "clMaroon";
  else
     @PN_Kopf.Color := "clBtnFace";
  endif;
  ```

- Auslesen/Setzen von Booleschen Properties, z.B. bei CheckBoxen **à** erfolgt am einfachsten nicht über Strings, d.h. hier lässt man die Hochkommata weg (dann muss man sich nicht um Groß-/Kleinschreibung kümmern):
  Auslesen:

  ```
  if @CH_Hinweis.Checked = True then
     show("WAHR");
  else
     show("FALSCH");
  endif;
  ```

  alternativ auch in Kurzform möglich:

  ```
  if @CH_Hinweis.Checked then
     show("WAHR");
  else
     show("FALSCH");
  endif;
  ```

  Setzen:
  ```
  if Bedingung = "1" then
     @CH_Hinweis.Checked := True;
  else
     @CH_Hinweis.Checked := False;
  endif;
  ```

- Zugriff auf Eigenschaften der Maske: Über @Form.Objekteigenschaft kann auf die unterste Maskenebene zugegriffen werden, ganz gleich welchen Namen sie besitzt. (z.B. `@Form.StartValue := "Belegnr=" + 2600005;`).
  Ein Zugriff über den eigentlichen Maskennamen ist zwar auch möglich, wird aber nicht empfohlen. Hintergrund: Wird eine Maske mehrfach aufgerufen, so erhält nur die zuerst geöffnete Maske den originalen Namen z.B. im Einkauf FABelegEK, bei allen weiteren wird dynamisch der Maskenname erweitert **à** FABelegEK_1, FABelegEK_2 etc. Beim Speichern der Maske wird der Maskenname mitgespeichert. Dies kann somit zu Problemen führen, wenn auf Eigenschaften der Maske per GDI-Basic zugegriffen werden soll (z.B. `@FABelegEK.StartValue := "Belegnr=" + 2600005;`)
- Vorsicht beim Designen von Masken mit eingeschränkten Rechten:
  - Sind Elemente einer Maske zum Zeitpunkt des Speicherns deaktiviert (z.B. aufgrund eingeschränkter Rechte über das Codewortsystem oder bei einer Belegmaske, weil der aktuelle Beleg bereits gedruckt /an die FIBU übergeben ist), so wird dieser Zustand mit gespeichert. Selbst wenn man die Maske mit allen Zugriffsrechten öffnet, bleiben die Elemente inaktiv! **à** immer Designen, wenn man Vollzugriff hat.
- Meldung "Ein deaktiviertes oder unsichtbares Fenster kann nicht den Fokus erhalten":
  - Diese Meldung erhält man, wenn vom Programm beim Aufruf der Maske ein Element den Fokus erhalten soll, dieses aber nicht angezeigt wird. Dies tritt beispielsweise auf, wenn man Masken designt, die "Assistentenfunktion" besitzen und mehrere Karteireiter umfassen. Als Beispiel sei hier die Belegübernahme genannt. Per Assistent werden mehrere Karteireiter abgearbeitet. Beim Speichern muss man darauf achten, dass man sich auf der Startseite des Assistenten befindet.
- Fehlermeldung (in älteren Versionen): "Eine Komponente mit der Bezeichnung GDI_STABLE_xxx existiert bereits" (xxxx = Name des Tabellenobjektes). Die Ursache für dieses Problem kann darin liegen, dass die Maske ein GDI-BASIC-Programm enthält, welches das Tabellenobjekt xxx anlegt. Beim Speichern wird dieses Objekt, sofern zuvor kein %xxx.FreeSTable angewendet wurde mit der Maske gespeichert. Beim nächsten Aufruf und Abarbeiten des GDI-BASIC-Programmes erhält man diese Meldung. (Kann auch in Belegbearbeitungsmasken auftreten, die über den Bildschirmdesigner gespeichert wurden, nachdem eine Mengeneinheit (ME) mit CreateSTable-Befehl in der Erfassung getestet wurde.) **à** Abhilfe: Maske als TXT-Datei speichern und Objekt über Editor entfernen (Tabellenobjekte findet man immer ganz unten/am Schluss)
- Designen von "Unter-Masken": Es gibt Teilbereiche von Masken, welche dynamisch in andere, übergeordnete Masken einbezogen werden. Als Beispiel sei die Stücklistenbearbeitung im Artikelstamm genannt (Artikel | Artikel > Stücklisten). Hier verhält es sich so, dass sie nicht Bestandteil der Artikelmaske ist. Deshalb sind an dieser Stelle Design-Änderungen zwecklos. Wenn man etwas ändern möchte, muss dies unter Artikel | Stücklisten vorgenommen werden. Diese Änderungen findet man dann auch wieder im Artikelstamm unter "Stücklisten".
  Ein weiteres Beispiel stellt die Belegansicht im Kunden- oder Lieferstamm dar **à** bei Anpassungen, welche über Änderungen des Gridspaltenaufbaus hinausgehen wäre hier die Belegtabelle zu designen (Verkauf | Zusatzfunktionen | Belegtabelle)
  - Der Zugriff auf Eigenschaften der MasterForm bei solchen dynamisch eingebundenen Masken erfolgt über die Syntax @MasterForm.Objekteigenschaft, z.B. `@Masterform.Caption`
- Desginen von Masken, welche nicht über einen eigenständigen Menüpunkt aufgerufen werden **à** siehe oben im Kapitel *Maskenaufrufe in der Bline 3.x (Factur-Bereich, MSK bzw. TXT)* im Abschnitt "Verbergen von Menüpunkten"

Der Grid-Designer im CRM-Bereich

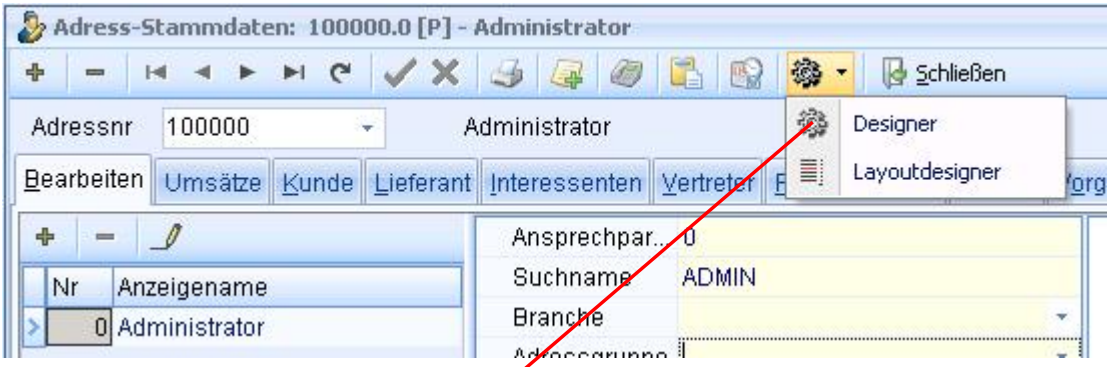
Beim Gridobjekt des CRM-Bereiches gibt es die Unterscheidung zwischen:
Vertikaler Grid (VGrid)
Horizontaler Grid (HGrid)
Die Funktionen dieser beiden Grids sind sehr umfangreich. Hier wurde nach Sichtung/Prüfung eine Beschreibung der wichtigsten (praxisrelevanten) Funktionen vorgenommen. Sie finden diese auf den folgenden Seiten.

Drop-Down-Menüs zum Speichern von Gridbuttons (GBN) und Masken (MSK2) im CRM-Bereich der GDI-Business-Line

In der GDI-Business-Line wurden Drop-Down-Menüs zum Speichern und Löschen von Masken und Griddefinitionen implementiert. Deren Funktion wird im Folgendem beschrieben:

MSK2-Masken-Speichern

In den Masken des CRM-Bereiches gibt es ein Drop-Down-Menü über das "Zahnrad"-Symbol, um den Designer aufrufen zu können. Über den Designer zudem angezeigt, welche MSK2-Definitionsdatei zugrunde liegt und woher sie geladen wurde.



Dieses Fenster zeigt im oberen Bereich die designbaren Objekte an. Per Doppelclick auf einen Eintrag oder über den Designen-Button kann der eigentliche Designer gestartet werden.

Über den unteren Bereich wird eine Maske gespeichert. Es wird angezeigt, ob und woher eine gespeicherte Maske geladen wurde.

Geladen/gesucht wird in folgender Hierarchie:

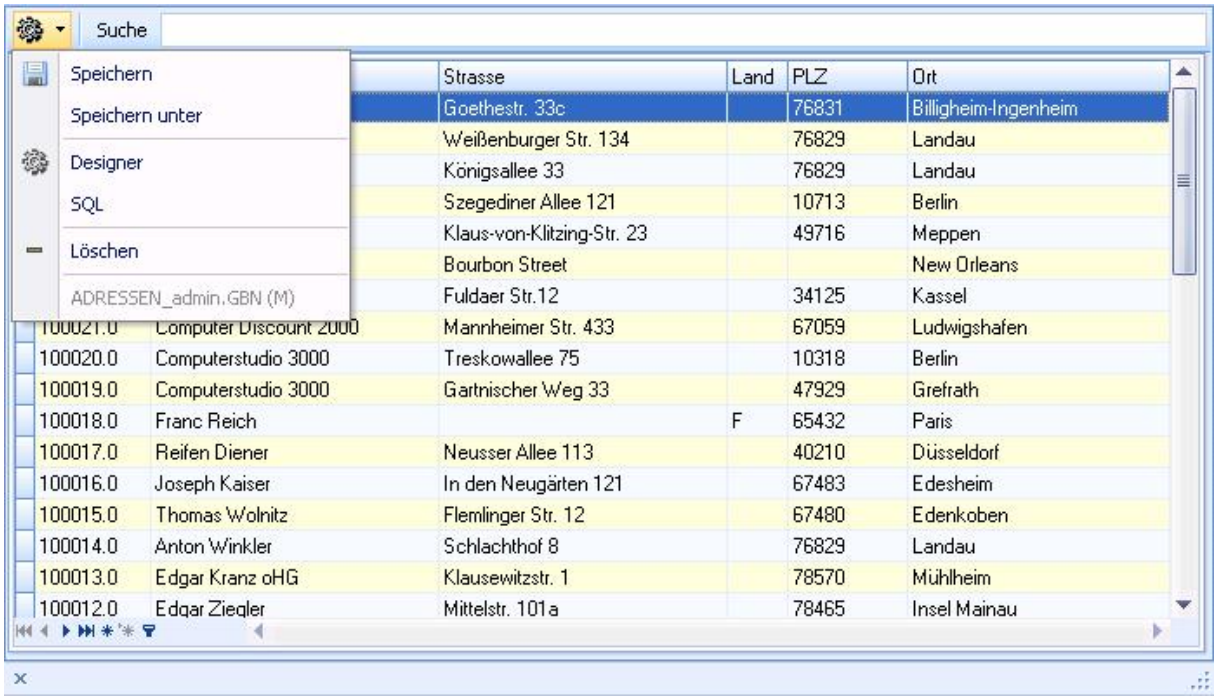
- MSK2-Definition aus Mandanten, bedienerbezogen
- MSK2-Definition aus Mandanten, allgemein
- MSK2-Definition aus Programm (mandantenübergreifend), allgemein

Hinweise:

- Das Laden einer bedienerbezogenen Maske aus der Programmebene ist nicht vorgesehen
- Nur der Bediener admin besitzt die Berechtigung, eine MSK2-Maske in der Programmebene (mandantenübergreifend) zu speichern oder aus der Programmebene zu löschen
- Die Masken aus dem CRM-Bereich besitzen als Dateiendung/Suffix "MSK2" zur Abgrenzung von Masken aus dem Factur-Bereich ("MSK"). Die Masken aus dem Factur-Bereich der Bline besitzen wie früher als Dateiendung ".MSK" oder ".TXT und werden im selben Ordner wie die MSK2-Masken abgelegt.

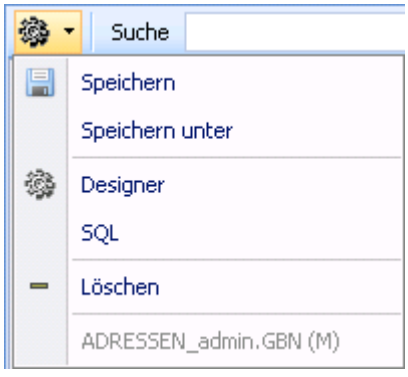
GBN-Gridbutton-Speichern

Wurde ein Gridbutton gedrückt und die Auswahltabelle ist sichtbar, so kann dort das Drop-Down-Menü über das "Zahnrad"-Symbol geöffnet werden, um die erforderlichen Einstellungen vorzunehmen und die GBN-Datei zu speichern. Gleichzeitig wird dort angezeigt, welche Griddefinitionsdatei zugrunde liegt und woher sie geladen wurde.



Geladen/gesucht wird eine GBN-Definition in folgender Hierarchie:
- Griddefinition aus Mandanten, bedienerbezogen
- Griddefinition aus Mandanten, allgemein
- Griddefinition aus Programm, allgemein

Hinweis: Das Laden einer bedienerbezogenen Griddefinition aus Programmebene ist nicht vorgesehen.



- Bedienerbezogen speichern in Verzeichnis Grids unterhalb des Mandanten z.B. "ADRESSEN_admin.GBN"
- Allgemein speichern z.B. als ADRESSEN.GBN über Speichern-Dialog mit Vorschlag Verzeichnis Grids unterhalb des Mandanten
- Aufruf Griddesigner
- Aufruf SQL-Bearbeitung
- Löschen der aktuell geladenen Definition (GBN-Datei). Nicht verfügbar, wenn Grid aus Programmebene gezogen wurde.
- Anzeige der aktuell geladenen Definition über GBN-Dateiname plus Herkunftsangabe in Klammern. (P) = Programmebene, (M) = Mandantenebene

Anpassung des neuen Grid (Masken-Griddesign (VGrid, HGrid) und F4-Griddesign)

Masken-Griddesign
In der NavBar befindet sich das "Zahnrad" zum Aufruf des Designers. Hier den Eintrag "Designer" wählen. Es startet das Fenster mit der Auswahl der designbaren Objekte und stellt sich - falls in der Maske mehrere Grids designt werden können - immer auf den aktuell in der Maske markierten Grid ein. Ein Doppelclick oder der Designen-Button startet den eigentlichen Designer.

Über eine Radiobox mit den möglichen Werten "**Grid**", "**Felder**" und "**Zeilen**" bzw. **"Spalten"** lassen sich die drei Grundbereiche des Griddesigns auswählen:

Grid: Designen des Grids
Nach Aufruf des Designers ist rechts oben in der Radio-Box der Eintrag "Grid" eingestellt. Das bedeutet, dass im linken Designer-Bereich grundsätzliche, für den gesamten Grid geltende Einstellungen getroffen werden können. Hierzu Beispiele:

Vertikaler Grid (VGrid)

Property "Layoutstyle": Legt fest ob der Grid Daten einspaltig oder mehrspaltig darstellen soll.

- IsBandsView: Ein Datensatz wird dargestellt und gegebenenfalls automatisch auf mehrere Spalten umgebrochen, wenn der Platz nicht ausreicht. Beispiel einer so eingestellten Standardmaske: Grid Bearbeiten im Adress-Stamm.
- IsSingleRecordView: Es wird nur ein Datensatz dargestellt, aber ohne automatischen Umbruch. Zur vollständigen Anzeige des Datensatzes wird die Aktivierung der Scrollbars notwendig (OptionsView > Scrollbars). Beispiel einer so eingestellten Standardmaske: Grid im Firmenstamm.
- IsMultiRecordView: Darstellung mehrerer Datensätze nebeneinander. Beispiel einer so eingestellten Standardmaske: Grid für den Bereich Kunden im Adress-Stamm.

Hinweis: Im Adress-Stamm ist diese Umschaltung auch über das rechte Maustasten-Kontextmenü möglich

Property "OptionsBehavior > CellHints":
Steuert, ob beim Überfahren einer Gridzelle mit der Maus, sofern der Zellinhalt nicht komplett angezeigt werden kann, die sog. Hint-Anzeige erscheint. Defaultwert ist true.

Property "OptionsBehavior > RowSizing":
Legt fest, ob man im Grid grundsätzlich die Zeilenhöhe per Ziehen mit der Maus ändern kann (auch ohne aktiven Designer). Sollen einzelne Zeilen unveränderlich bleiben / davon ausgenommen sein, so ist bei diesen die Eigenschaft "Options > CanResized" auf false zu stellen.

Horizontaler Grid (HGrid)

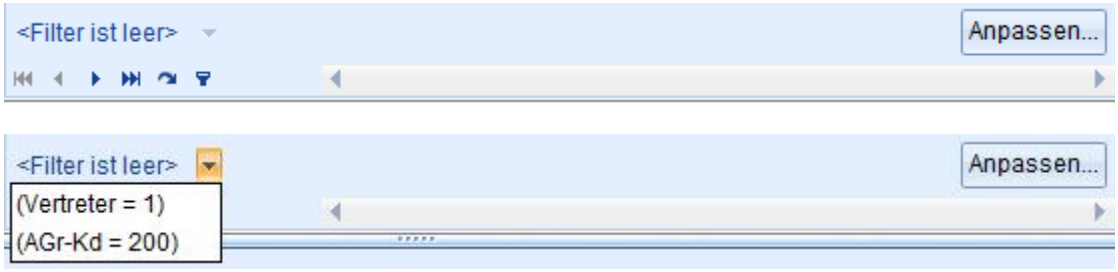
Property "FilterRow"
Blendet als oberste Gridzeile eine sogenannte Filterzeile ein. Ermöglicht eine gezielte, auf die jeweiligen Spalten bezogene Filterung.
Hinweis: In der Adress-Tabelle ist die Aktivierung der Filterzeile per rechter Maustaste möglich

Property "OptionsView > Navigator" und Property "NavigatorButtons"

Erstere blendet unterhalb des Grids den Navigator ein, dessen Buttons über die zweitgenannte im Detail festgelegt werden (ein-/ausblenden). Im Standard ist dieser kleine Navigator z.B. in der Adress-Tabelle zu finden, wo er in erster Linie zum Einblenden der sog. Filterbox dient.

Property "FilterBox":
Alternativ lässt sich die FilterBox auch fest über die gleichnamige Einstellung aktivieren, indem man dort visible auf "fvAlways" einstellt.

Hinweis: Die FilterBox am unteren Gridrand wird immer angezeigt, sofern mindestens ein gespeicherter Filter vorliegt (sofern mit der Maske ein Filter gespeichert wurde ist dieser sofort sichtbar, ansonsten steht dort <Filter ist leer>). Gespeicherte Filter können dann über das Pulldown ausgewählt werden.



Property "OptionsView.ColumnAutoWidth":
Defaultwert = true. Diese Funktion steuert, dass immer alle visible-eingestellten Spalten ohne horizontale Scrollbar angezeigt werden. Wird beispielsweise durch Vergößern/Verkleinern der Maske die Gridbreite geändert, so werden alle Spalten in der Breite dynamisch angepasst. Stellt man diese Eigenschaft auf false, so wird stattdessen ein Scrollbalken - sofern erforderlich - eingeblendet.

Property "OptionsCustomize > ColumnsQuickCustomization":
Soll der Spaltenaufbau des Grids nicht durch den Bediener veränderbar sein (Spalten zu- oder abschalten verhindern), so ist diese Einstellung auf false zu stellen.

Felder: Aktivieren eines (neuen) Datenfeldes
Zum Einfügen eines Feldes zunächst rechts oben in der Radio-Box von „Grid“ auf „Felder“ wechseln. Hier weder alle in der jeweiligen Datentabelle zur Verfügung stehenden Felder angeboten. Beim gewünschten Feld das Häkchen zur Aktivierung setzen, das Feld ist nun zum Design im Grid „verfügbar“.

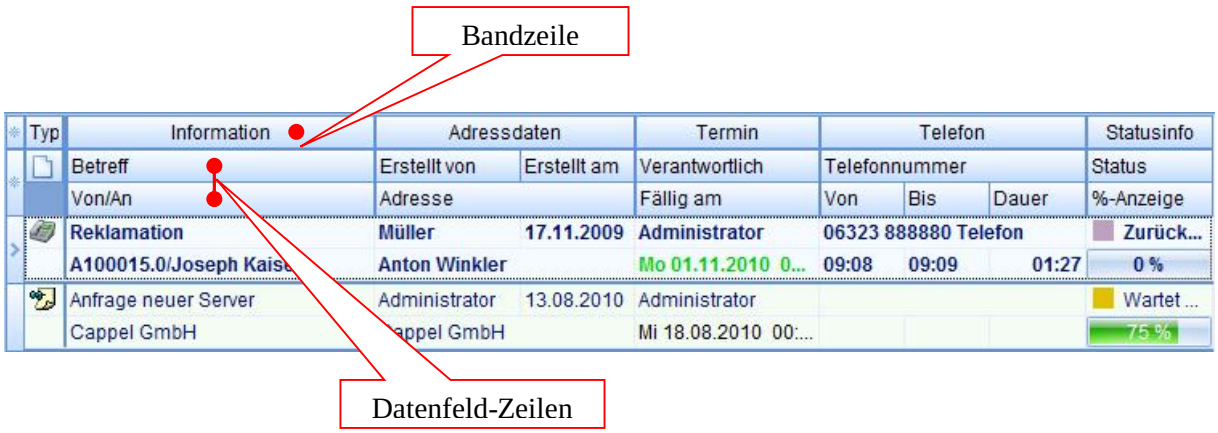
Zeilen (bei VGrid) bzw. Spalten (bei HGrid): Designen eines Datenfeldes im Grid
Zur weiteren Einstellung rechts oben in der Radio-Box von "Felder" auf "Zeilen" bzw. "Spalten" wechseln. Nun werden alle zuvor unter "Felder" aktivierten Datenfelder zum Design innerhalb des Grids angeboten. Bei beiden Arten von Grids (horizontaler Grid /vertikaler Grid) steuert die Eigenschaft "visible", ob ein Datenfeld im Grid aktuell angezeigt wird. Zusätzlich kann bestimmt werden, ob ein Datenfeld außerhalb des Designers zu-/abgeschaltet werden kann:

Horizontaler Grid:
Property "VisibleForCustomization": Bestimmt, ob ein Datenfeld in der über die linke Ecke des Grids zu öffnende Feldliste erscheint und somit durch den Benutzer zu-/abgeschaltet werden kann.

Vertikaler Grid:
Property "Options > ShowInCustomizationForm": Bestimmt, ob ein Datenfeld im Layout-Designer sichtbar ist. Nur Felder bei denen diese Eigenschaft auf true steht, können durch den Bediener beliebig aus dem Grid genommen oder wieder eingefügt werden, sofern der Griddesigner selbst über das Rechtesystem abgeschaltet ist und somit dem Bediener nicht zur Verfügung steht.

Zweizeiliger HGrid – Einzeiliger HGrid

In der Dokumentenverwaltung kommt eine besondere Form des horizontalen Grids zum Einsatz, welches mehrzeilige Datensatzanzeigen ermöglicht. D.h. zur Ausgabe eines einzigen Datensatzes stehen mehrere Zeilen des Grids zur Verfügung. Somit hat man auf einen Blick - ohne im Grid nach rechts/links scrollen zu müssen - mehr Information.
Die Besonderheiten dieses Grid gegenüber dem "Standard"-Grid sind folgende:

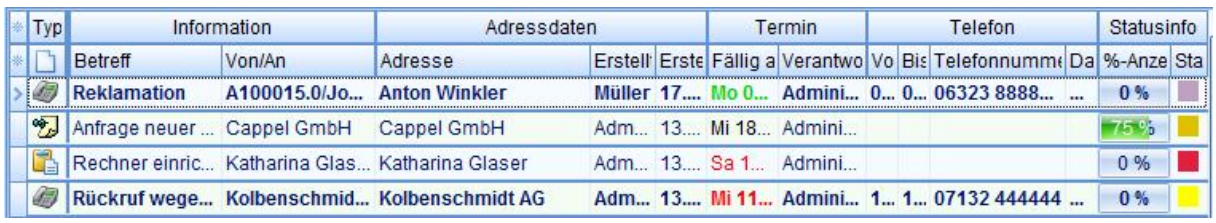


Die oberste angezeigte Zeile des Grids stellt eine Bandzeile (rein textuelle Überschriftszeile) dar. Diese Bandzeile kann mehrere Einträge (im Designer unter der Eigenschaft "Bands" beim Grid einstellbar) umfassen, um die Überschriftszeile in einzelne Blöcke aufzuteilen. In der Standardmaske sind dies folgende Überschriften:

Typ, Information, Adressdaten, Termin, Telefon und Statusinfo.

Da es sich um reine Texte handelt, können die Überschriften nicht für Sortierungen oder Gruppierungen verwendet werden. Das bei den Datenfeldern gewohnte Kontextmenü ist hier nicht verfügbar. Schaltet man eine Überschrift in der Ansicht ab (ganz links über das "*"-Symbol in der Bandzeile), so werden alle unter ihr angeordneten Datenfelder ausgeblendet.

Unterhalb der Bandzeile folgen wie beim Standardgrid die auszugebenden Datenfelder. Als Besonderheit können Datenfelder nicht nur horizontal, sondern auch vertikal verschoben werden. Somit wird die mehrzeilige Darstellung ermöglicht. Die Standardmaske umfasst eine zweizeilige Darstellung, beispielsweise werden die Felder "Betreff" und "Von/An" untereinander ausgegeben.
Wünscht man eine einzeilige Darstellung der Datensätze, so müssen alle Datenfelder vertikal auf gleicher Höhe stehen. Für folgende Abbildung wurden alle Datenfelder der zweiten Zeile in die obere verschoben, der Grid wurde einzeilig:



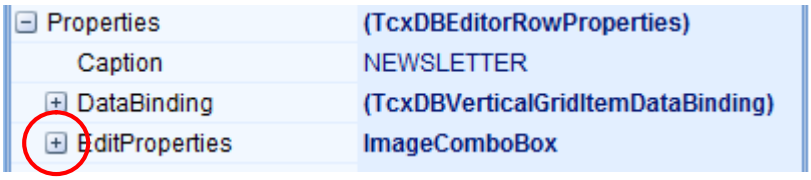
Wichtig beim Einfügen weiterer Felder (Spalten) in den mehrzeiligen HGrid: Die Einordnung eines Feldes erfolgt über die Angabe seiner Position. Im Designer kann dies bei der ausgewählten Spalte unter der Eigenschaft "Position" über BandIndex, CollIndex und RowIndex eingestellt werden. Außerhalb des Designers ist ein Verschieben im Grids per Drag & Drop möglich. Da ein neues Feld in Voreinstellung mit BandIndex=0 eingefügt wird, findet man dieses unter der ersten Überschrift - in der Dokuverwaltung also unter "Typ". Da diese Spalte sehr schmal ist, empfiehlt es sich den BandIndex gleich nach dem Einfügen auf 1 zu setzen. Das Feld befindet sich dann in der breiteren "Adressdaten"-Spalte und kann leichter per Maus an die gewünschte Stelle im Grid verschoben werden.

à Übersicht über die "Objekttypen" im CRM-Bereich

Folgende Tabelle zeigt die Gestaltungsmöglichkeit von Feldern im DevExpress-Grid. Ein "X" in der dritten Spalte (B) zeigt an, dass weiter unten im Text eine ausführliche Beschreibung folgt:

Funktion	Einstellung/Beschreibung	B
Normales Feld	Bei einem "normalen" Datenfeld sind keine weiteren Einstellungen notwendig. Über den Grid selbst ist das Datenfeld bereits mit der Datenbank verbunden.	
Checkbox	EditProperties: CheckBox, meist ist es sinnvoll anschließend EditProperties > ValueChecked auf "1", EditProperties > ValueUnchecked auf "0" einzustellen. EditProperties > NullStyle ist auf "nssUnchecked" zu stellen.	
ComboBox veränderlich	EditProperties: ComboBox, danach unter Items die gewünschten Einträge erfassen. Angezeigter und bei einer Auswahl in der Datenbank gespeicherter Wert sind bei einer ComboBox identisch.	
ComboBox unveränderlich	wie oben, jedoch zusätzlich die Eigenschaft EditProperties > DropDownListStyle von IsEditList auf IsFixedList stellen. Somit wird die Eingabe von "freiem" Text unterbunden.	
RadioGroup	EditProperties: RadioGroup, danach unter Items die gewünschten Einträge mit Caption und zugehörigem Value erfassen	
ImageComboBox	EditProperties: ImageComboBox Hier handelt es sich um eine Combobox, bei der pro Listeneintrag zwischen in der Auswahl angezeigtem und in die Datenbank übernommenen Wert unterschieden wird. Zusätzlich lassen sich Grafiksymbole anzeigen.	X
Mehrzeiliges Feld	Editproperties: psMemo Normalerweise springt der Cursor bei Bestätigen mit Enter in die nächste Gridzelle und ein Umbruch im Text kann per Strg + Enter erzeugt werden. Möchte man dies nicht, so ist EditProperties > WantReturns auf true zu stellen	
Bild im Grid	EditProperties: Image. Unter GraphicClassName angeben, ob Bilder im BMP- (à Eintrag TBitmap) oder JPG- (à Eintrag TJPEGImage) Format verwendet werden. Es können nur Bilder angezeigt werden, die in der Datenbank in einem Binärfeld abgespeichert sind.	
"ClientDBText"	EditProperties: psEdit, psMemo, psPopUp, psRichEdit	X
F4-Auswahlgrid	EditProperties: psPopUp	X
Datumsfeld	EditProperties: DateEdit	X
Button für Script ("Basic-Button")	EditProperties: psEdit	X

Wichtig: Sobald bei der Eigenschaft EditProperties die gewünschte Funktion eingestellt ist, werden bei dieser passende Untereigenschaften aktiviert. Wie in der folgenden Abbildung gezeigt wird das "+" –Zeichen zum Öffnen der Unterpunkte sichtbar.



à Nachbildung eines ClientDBText per GDIAnzeigeSQL

In einer Gridzelle wird normalerweise der Inhalt des angeschlosssen Datenfeldes ("DataBinding > FieldName") angezeigt. Beim sog. ClientDBText-Objekt des Designers des Facturbereiches befindet sich der anzuzeigende Wert nicht in der "Mastertabelle" der designten Maske, sondern in einer anderen Tabelle der Datenbank (z.B. wird im Kundenstamm die Branchennummer abgespeichert, über ein ClientDBText wird die dazugehörige Branchenbezeichnung aus der GDIDEF-Tabelle angezeigt). Die Beziehung zwischen Mastertabelle und Slavetabelle wird feldbezogen direkt beim ClientDBText-Objekt definiert. Zur Nachbildung dieser Funktionalität im Designer des CRM-Bereiches wurde die Property "GDIAnzeigeSQL" bei verschiedenen EditProperty-Typen implementiert. Grundsätzlich sind zwei Varianten zu unterscheiden: Die ursprüngliche Variante arbeitet rein auf SQL-Basis und ist nach wie vor einsetzbar. Ab Version 3.7.1.x wurde eine erweiterte Variante ergänzt, bei der GDI-Basic zum Einsatz kommt:

I) ALTE VARIANTE:
Diese Variante steht bei bei den EditProperty-Typen psEdit und psPopUp zur Verfügung.
In der Property "GDIAnzeigeSQL" wird eine SQL hinterlegt, deren Ergebnis-Feld "Ausgabe" in der Gridzelle hinter dem eigentlichen Feldwert angezeigt wird, und zwar fix in spitzen Klammern eingeschlossen "<...>". Die Referenz auf den eigentlichen Feldwert wird in Form des Parameters ":Feldname" übergeben.
Im Standard vorhandene Beispiele: Man findet diese Variante an vielen Stellen in der Adress-Stamm-Maske, meist bei Zellen in welchen F4-Auswahlgrids über die EditProperty psPopUp realisiert sind. Über die GDIAnzeigeSQL wird hinter dem im Datensatz gespeicherten Wert in spitzen Klammern ein erklärender Klartext angezeigt. Z.B. auf der Kartei "Bearbeiten" bei den Feldern für den Adress-Status, die Adress-Funktion oder das Land oder auf der Kartei "Kunde" bei den Feldern für Adressgruppe, Zahlart, Zahlungsziel, Vertreter etc.

Weiteres Beispiel:
In einem Grid in welchem die AdressID angezeigt wird soll zusätzlich Vormane und Nachname angezeigt werden. Die Editproperty wird auf psEdit gestellt und in der Unter-Property GDIAnzeigeSQL eine entsprechende SQL hinterlegt:

```
select Vorname || ' ' || Name Ausgabe from CM_Adressen where AdressID =
:FeldName
```

--> Die Anzeige in der Gridzelle sieht dann z.B. so aus: "123456 <Uwe Meierhoff>"

II) NEUE VARIANTE (ab 3.7.1.x, November 2017)
Sie steht bei den EditProperty-Typen psEdit, psPopUp, psMemo und psRichEdit zur Verfügung.

- Zusätzlich zur Property "GDIAnzeigeSQL" wurde eine neue Property "GDIAnzeigeMask" eingeführt, in der eine EditMask zur Formatierung der Ausgabe hinterlegt werden kann.
  Wichtig: Um die neuen, erweiterten Möglichkeiten nutzen zu können, muss immer ein Wert in dieser Property hinterlegt werden. Ist diese Eigenschaft leer, erfolgt die Ausgabe in der "alten" Variante.

Zu beachten ist außerdem, dass man hier immer eine für die auszugebenden Werte geeignete EditMask hinterlegen muss. Möchte man z.B. ausschließlich einen numerischen Wert ausgeben, kann man z.B. "N9.2" angeben, oder für ein Datumsfeld die Maske "D10". Sollen jedoch zusammengesetzte Werte ausgegeben werden, macht es im Allgemeinen nur Sinn, die universelle EditMask "C" anzugeben.

- In der Property "GDIAnzeigeSQL" kann man statt einer SQL, wie sie in der "alten" Variante beschrieben ist, und die natürlich unverändert funktioniert, auch ein GDI-Basic bzw. ein PascalScript- Fragment laufen lassen, dessen Ergebnis zur Anzeige gebracht wird.
    a. GDI-Basic/PascalScript direkt in der Property hinterlegt: Hier muss ganz am Anfang des hinterlegten Textes das Schlüsselwort "$GDI-Script" vorkommen. Ob der auszuführende Code in GDI-Basic oder PascalScript verfasst ist, wird über das Vorhandensein des Schlüsselwortes "#LANGUAGE" im Programmcode des Scriptes gesteuert.
    b. GDI-Basic/PascalScript über den Dateinamen referenziert: Dazu ist ganz am Anfang des Textes das Schlüsselwort "$GDI-FileName=" zu hinterlegen, gefolgt vom Dateinamen des GDI-Basic- oder Pascal-Scriptes. Enthält der Filename einen Pfad, wird dieser direkt verwendet und die Datei von der angegebenen Stelle geladen. Ist nur der Dateiname inkl. Endung angegeben, so wird die Datei zuerst im Mandanten-Unterordner "Script", und, sofern dort nicht vorhanden im gleichnamigen Programm-Unterordner gesucht.

  Allgemeines zu den Scripten:
  - Der anzuzeigende Text muss aus dem Script über die Variable "Result" zurück gegeben werden.
  - Im Gegensatz zum Ergebnis der Anzeige-SQL in der "alten" Variante muss das GDIScript den gesamten anzuzeigenden Inhalt über die Result-Variable zurückgeben. Sofern in der Anzeige im Grid gewünscht muss man also auch den eigentlichen Feldwert zurückliefern.
  - Der Feldwert des Datenfeldes wird vom Programm an das Script über die vordefinierte Variable "FieldValue" übergeben.

Beispiel: Anzeige des Kunden-Kreditlimits im Adress-Stamm auf der Kartei "Bearbeiten". Der dortige Grid ist an die Tabelle CM_Adressen angeschlossen. Man fügt eine neue Gridzeile hinzu, verbindet diese mit dem Datenfeld AdressID und stellt die EditProperty psEdit ein. Dann werden die Unter-Properties gefüllt:

- EditProperties > GDIAnzeigeMask: Zur Aktivierung der neuen Variante muss hier ein Eintrag hinterlegt werden. Da lediglich ein Zahlenwert ausgegeben wird bietet sich die Maske R###,###,##0.00 an.
- EditProperties > GDIAnzeigeSQL: Hinterlegung des eigentlichen Scriptes:

```
$GDI-Script
  Result := '';
  SQL := 'select Kredlimit as Ausgabe from Kunden where AdressNr = :AdressID';
  Res := ExecSQLVariant(SQL, 'AdressID', FieldValue);
  Result := GetFieldvalue(Res, 'Ausgabe');
exit;
```

Alternativ könnte man dieses Basic in eine Datei z.B. Kreditlimitanzeige_AdrStamm.prg packen, im Unterordner "Script" zum Mandanten ablegen und deren Aufruf bei der Property GDI-AnzeigeSQL hinterlegen:

```
$GDI-FileName=Kreditlimitanzeige_AdrStamm.prg
```

Nachfolgende Hardcopy zeigt das Ergebnis einer solchen Design-Erweiterung. Zum Vergleich wurde auch das Ergebnis der "alten" Variante mit in die Hardcopy aufgenommen. In der oberen Zeile wurde lediglich die SQL in der GDIAnzeigeSQL-Property hinterlegt. Es wird die AdressID und dazu Kreditlimit in spitzen Klammern ausgegeben. Die untere Zeile ist das Ergebnis der "neuen" Variante. Es wird nur das Kreditlimit, aufgrund der GDIAnzeigeMask mit Tausender- und Dezimaltrenner formatiert ausgegeben.

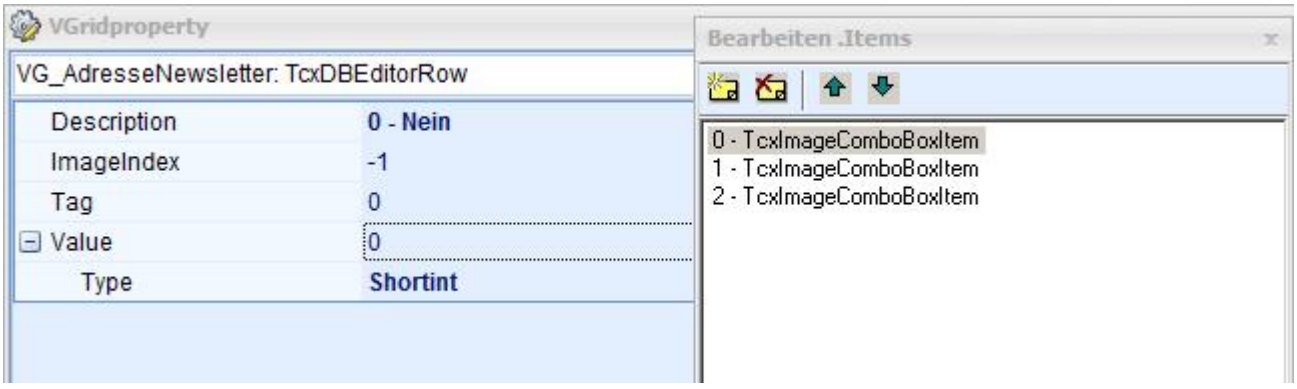
ADRESSID --> Kredlimit über AnzeigeSQL	100004 <20000>
ADRESSID --> Kredlimit AnzeigeSQL mit Basic	20.000,00

à ImageComboBox (ImageComboBox)

Bei der ImageComboBox können für jede Zeile der sich öffnenden DropDown-Liste drei Werte hinterlegt werden: Anzeigewert, Datenbankwert und optional eine Grafik.

- EditProperties > DropDownRows: Max. Anzahl sichtbarer Zeilen bei geöffneter ComboBox
- EditProperties > ImageAlign: Anzeige der Grafik links oder rechts
- EditProperties > Images: Falls Grafik-Symbole in der Box angezeigt werden sollen, wird hier die gewünschte Imageliste eingestellt, aus der diese geladen werden. "Gängige" Imagelisten sind ".FSMenu.sTool.IL_Image" (diese Liste enthält derzeit 160 verschiedene Grafiken, welche meist in Navigationsleisten der Masken des CRM-Bereiches verwendet werden) und ".FSMenu.IL_SmallHaupt" (Liste mit den Grafiksymbolen des Menüs). Desweiteren kann auf die individuellen Imagelisten zugegriffen werden.
- EditProperties > Items: Hier wird "(TcxImageComboBoxItems)" angezeigt. Per Mausklick auf diesen Eintrag öffnet sich ein Hilfsfenster, in welchem sich die Zeilen für die DropDown-Liste definieren lassen. Über Symbole können Zeilen eingefügt, gelöscht oder auch in ihrer Reihenfolge verschoben werden. Markiert man eine Zeile in diesem Fenster, zeigt der Grid des Designers die Detaileinstellungen an.
    - Description: Text für die Anzeige (was der Bediener bei einer Auswahl in der Box sieht)
    - ImageIndex: Nummer einer Grafik aus einer ImageListe. Der Eintrag "-1" bedeutet "keine Grafik".
    - Tag: nicht von Bedeutung
    - Value: Wert, der in der Datenbank gespeichert werden soll. Zusätzlich wird eingestellt, um welchen Datentyp es sich handelt

Die nachfolgende Abbildung zeigt Einstellungen der ImageComboBox für das Feld "Newsletter" im Adress-Stamm.



à F4-Auswahlgrid (psPopUp)

Für das F4-Auswahlgrid sind mehrere Schritte notwendig: Zum einen die Definition des Feldverhaltens als "Gridbutton" - dies erfolgt im Designer der Maske, zum anderen muss der sich öffnende Auswahlgrid designt werden. Das erfolgt bei geöffnetem Auswahlgrid über den dort zur Verfügung stehenden Griddesigner.

Grid in der Maske: EditProperties: psPopUp
Die weiteren Schritte spezifizieren die Einstellungen, zur Verdeutlichung am Beispiel einer Adressgruppenauswahl im CM_Adress-Stamm

- Properties > EditProperties > GDIAnzeigeSQL: Entspricht der aus der GDILine gewohnten SQL zur Anzeige von Daten in einem ClientDBText-Objekt

  ```
  select F1 ausgabe from gdidef where satzart = "AG" and
  IND1=:Feldname
  ```

- Properties > EditProperties > GDIGridname: Entspricht der LoadDefs-Angabe aus der GDILine. Unter diesem Namen wird der Grid gespeichert/geladen.

  ADRESSGRUPPEN

- Properties > EditProperties > GDIReturnwert: Feldinhalt, welcher bei der Auswahl übernommen werden soll in der Syntax Empfängerfeldname=Senderfeldname (vgl. GDILine Property Dialog.Return beim EditFeld mit Buttonstyle ebsGridbutton)

  ADRESSGRP=IND1

- Properties > EditProperties > GDIStartWert: Feldinhalt, welcher an den Auswahlgrid beim Start übergeben werden soll, ebenfalls in der Syntax Empfängerfeldname=Senderfeldname (vgl. GDILine Property Dialog.Start beim EditFeld mit Buttonstyle ebsGridbutton)

  IND1=ADRESSGRP

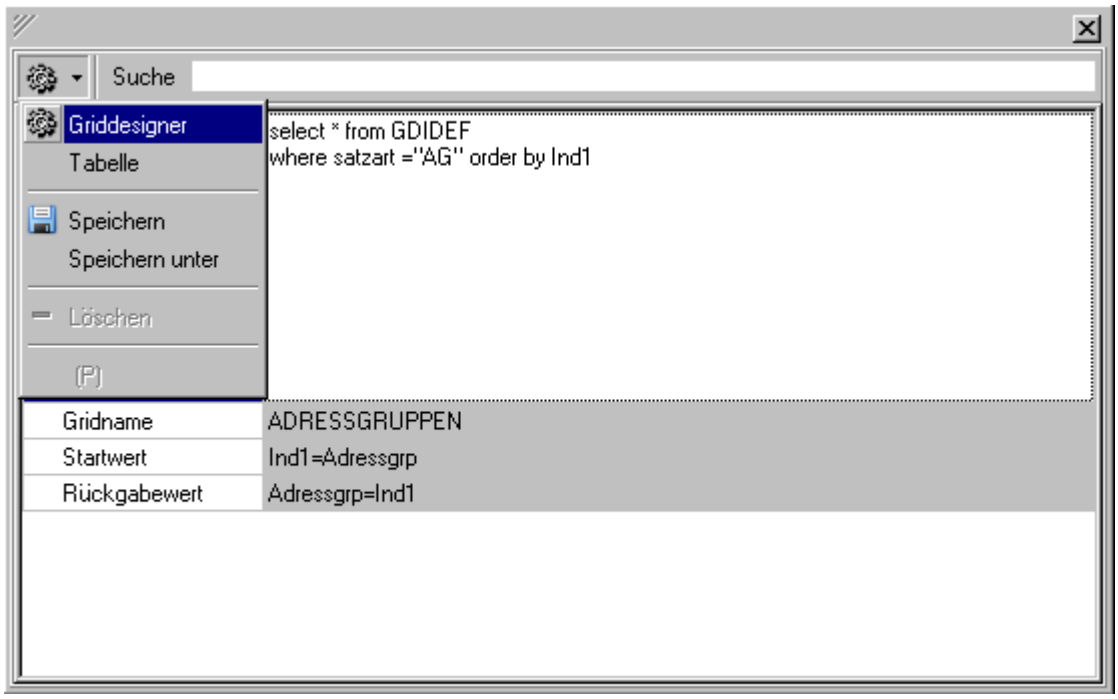
- Properties > Options > ShowEditButtons: Hier sollte man *eisbAlways* einstellen, damit der keline Button in der Gridzelle immer sichtbar ist

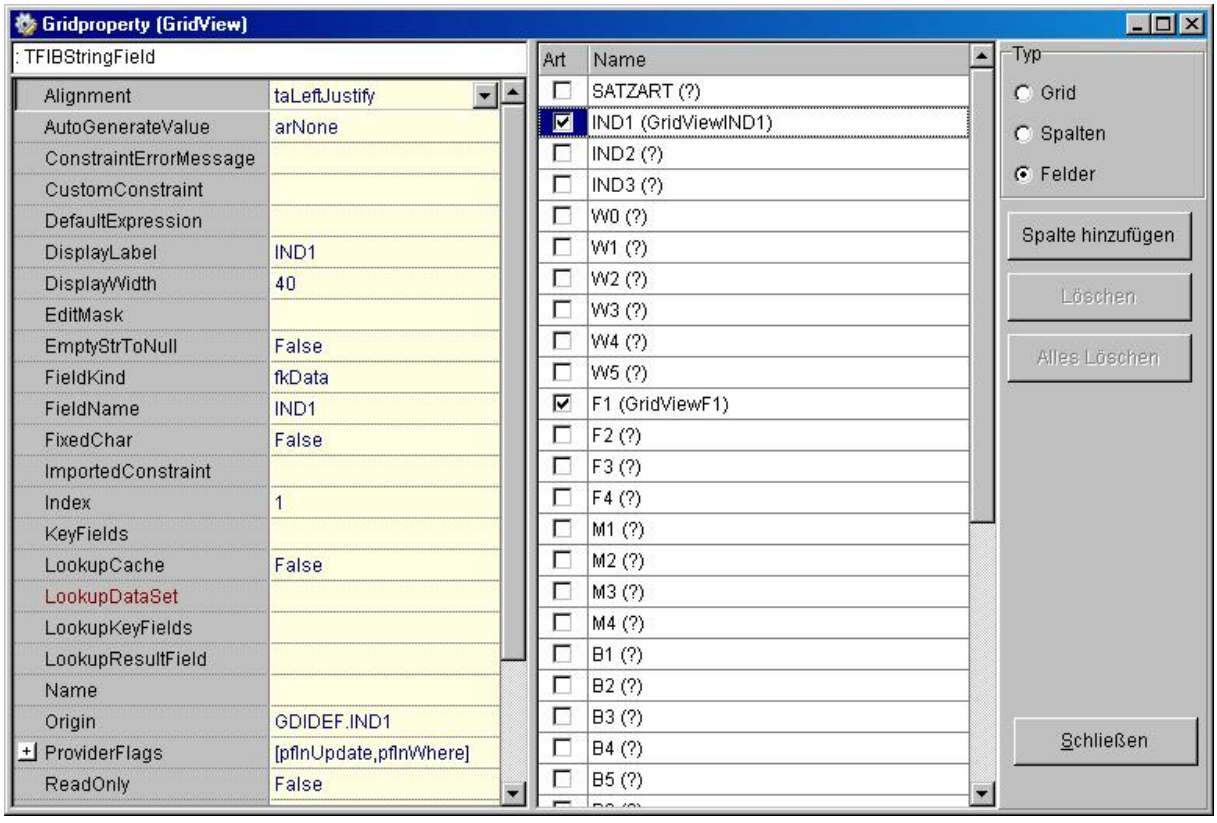
Nach diesen Definitionen kann der Griddesigner der Maske geschlossen werden. Bei Click oder F4 öffnet sich jetzt bereits der Auswahldialog, allerdings sogleich in der "Designeransicht" zur Hinterlegung der SQL:

Zur Information werden Gridname, Rückgabewert (Returnwert) und Startwert angezeigt. Es wird eine entsprechende select- SQL, welche die zur Auswahl anzubietenden Daten liefert hinterlegt: (Hinweis: Eingabefeld mit <Tab> anspringen)

```
select * from GDIDEF
where satzart ="AG" order by Ind1
```

Damit ist die Grunddefintion zum "Ziehen" der Daten aus der Datenbank gelegt, es fehlt noch die Definition des Grids an sich, d.h. welche Spalten die Auswahltabelle anzeigen soll. Man wechselt hierzu über das Zahnrad zum Griddesigner:





Nach Einstellen der gewünschten, anzuzeigenden Spalten und Speichern der Griddefinition über Zahnrad > Speichern bzw. Speichern unter ist das Designen der F4-Auswahltabelle fertiggestellt.

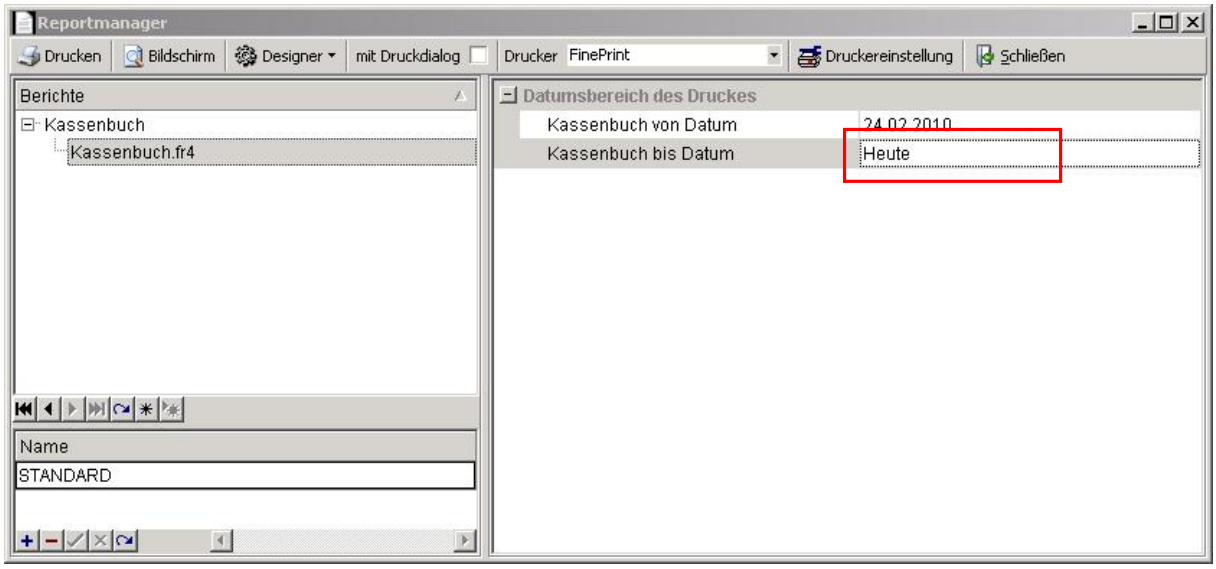
à Datumsfelder im CRM (DateEdit)

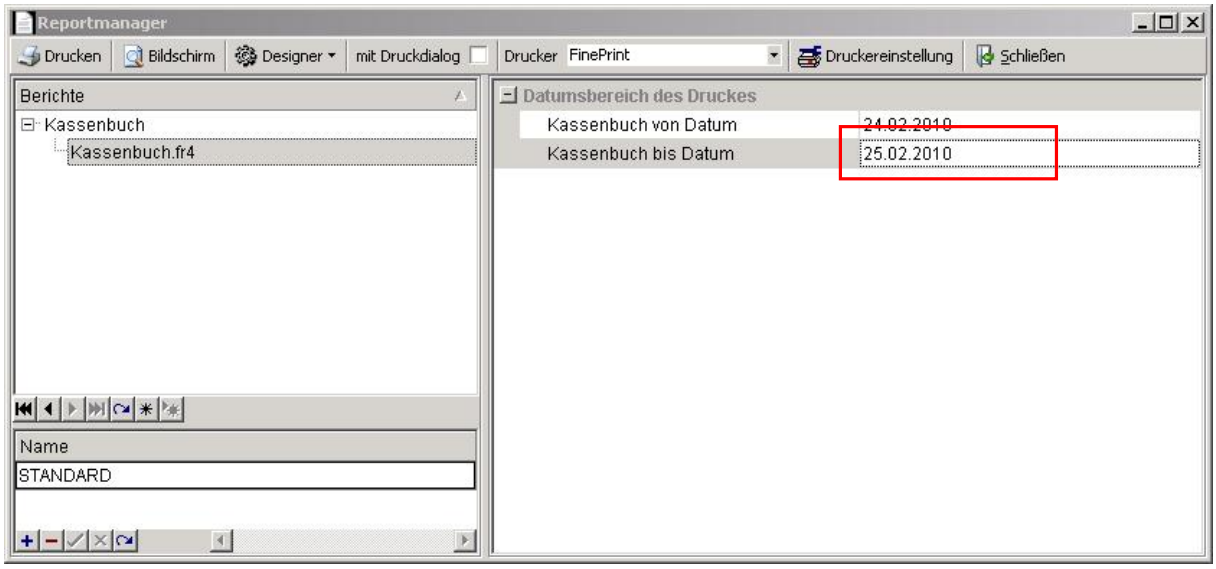
Für die Anzeige vom Datumswerten ohne Uhrzeit sind folgende Eigenschaften wichtig:

- EditProperties > AssignedValues > DisplayFormat: True
- EditProperties > DisplayFormat: dd.mm.yyyy

Bei den Datumsfeldern des CRM besteht die Möglichkeit einer Schnelleingabe. Man muss nicht zwingend das Datum mit Ziffern eingeben oder per Mausklick/F4 den Kalender öffnen, sondern kann gezielt durch Eingabe weniger Zeichen ein Datum bilden. Aufgrund einer Mindesteingabe erfolgt eine automatische Wort-Vervollständigung. Beim Verlassen des Eingabefeldes wird aus dem textuellen Begriff ein korrektes Datum erzeugt. Für diese Funktion reichen normalerweise die Voreinstellungen. Zur Sicherheit hier die Voraussetzungen: Die Eigenschaft EditProperties des Eingabefeldes steht auf DateEdit, die Eigenschaft EditProperties > InputKind auf ikRegExpr. Hinweis: Letzteres nicht verwechseln mit EditProperties > AssignedValues > InputGrid, hier ist der Defaultwert false zu belassen.

Beispiel: Eingabe von "h" ergibt "heute" und somit das Tagesdatum.





Auch "rechnende" Eingaben sind möglich. So ergibt die Eingabe von "h - 7" das Datum des genau eine Woche zurückliegenden Tages.

Übersichtstabelle (groß- und fettgedruckte Buchstaben = Mindesteingabe):

Eingabe	Ergebnis
BOM	erster Tag des Monats
EOM	letzter Tag des Monats
ERster	erster Tag der aktuellen Woche (beginnend bei Montag)
Zweiter	zweiter Tag der aktuellen Woche
DRitter	dritter Tag der aktuellen Woche
Vierter	vierter Tag der aktuellen Woche
FÜnfter	fünfter Tag der aktuellen Woche
SEchster	sechster Tag der aktuellen Woche
SIebenter	siebenter Tag der aktuellen Woche
Heute	aktueller Tag
MORgen	nächster Tag
Gestern	vorheriger Tag
Jetzt	aktueller Tag und Zeit (nur bei Kind = ckDateTime)
MONtag	Datum des nächsten Montags
DIenstag	Datum des nächsten Dienstags
MIttwoch	Datum des nächsten Mittwoch
DOnnerstag	Datum des nächsten Donnerstags
FReitag	Datum des nächsten Freitags
SAmstag	Datum des nächsten Samstags
SOnntag	Datum des nächsten Sonntags

à Button für Script ("Basic-Button")

Voraussetzung: Lizenz für Maskenaufruf oder Lizenz Autom. Belegerstellung
Eine Gridzelle des vertikalen Grids kann als Button definiert und zur Ausführung von GDI-Basic verwendet werden. Die Haupteinstellung wird durch Festlegung der EditProperties auf psEdit festgelegt, dadurch werden die entscheidenden Einstellungen zugänglich.

- Caption: wie üblich die Beschriftung links im "festen" Teil des Grids

- Properties > EditProperties: psEdit

- Properties > EditProperties > Buttons: Es kann ein Zusatzdialog zum Definieren des Buttons geöffnet werden. Dort sind folgende Einstellungen von Belang:
  - Caption: Beschriftung des Button
  - Glyph: Auswahl einer Bitmap
  - Kind: Soll die Caption angezeigt werden, ist hier bkText zu setzen. Soll eine Bitmap gesetzt werden, so ist bkGlyph zu setzen.

- Properties > EditProperties > ViewStyle: Regelt die Breite des Buttons.

  vsButtonsAutoWidth

- Properties > Options > ShowEditButtons: Diese Einstellung sorgt dafür, dass der Button angezeigt wird.

  eisbAlways

- Properties > EditProperties > GDIMaskenAufruf: Hier wird hinterlegt, welches Script aufgerufen werden soll. Nach dem Schlüsselwort "GDIScript=" wird der Name einer gespeicherten Scriptdatei angegeben. Bei dieser Datei kann es sich um ein GDI-Basic oder um ein Pascalscript handeln. Sie wird im Ordner Script (beim Programm oder Mandanten) abgelegt.

  GDIScript=Test.prg

Ein (lesender) Zugriff auf die aktuellen Daten z.B. um aus dem Basic-Programm heraus Übergabeparameter zu bilden und andere Masken aufzurufen ist bereits möglich. Beispielsweise gibt es folgende Datenobjekte in der Maske Adress-Stamm:

- TA_Ansprech für CM_Adressen-Zugriff Ansprechpartner-Satz
- TA_Adressen für CM_Adressen-Zugriff Hauptdaten-Satz
- TA_Kunden für Kundenzugriff
- TA_Liefer für Lieferantenzugriff

Beispiel-Programm für den Aufruf der Beleg-Adress-Artikel-Maske:

```
Kundennr := IntToStr(TA_Kunden.Kundennr);
if Kundennr = "0" then
   exit;
endif;
Name1    := TA_Kunden.Name1;
Ort      := TA_Kunden.Ort;
CR       := CHR(13,10);
Start := "Adressnr=" + Kundennr + CR + "Name1=" + Name1 + CR + "Ort=" + Ort;
show(Start);
Startdialog("TFAdrArtBeleg",true,Start);
exit;
```

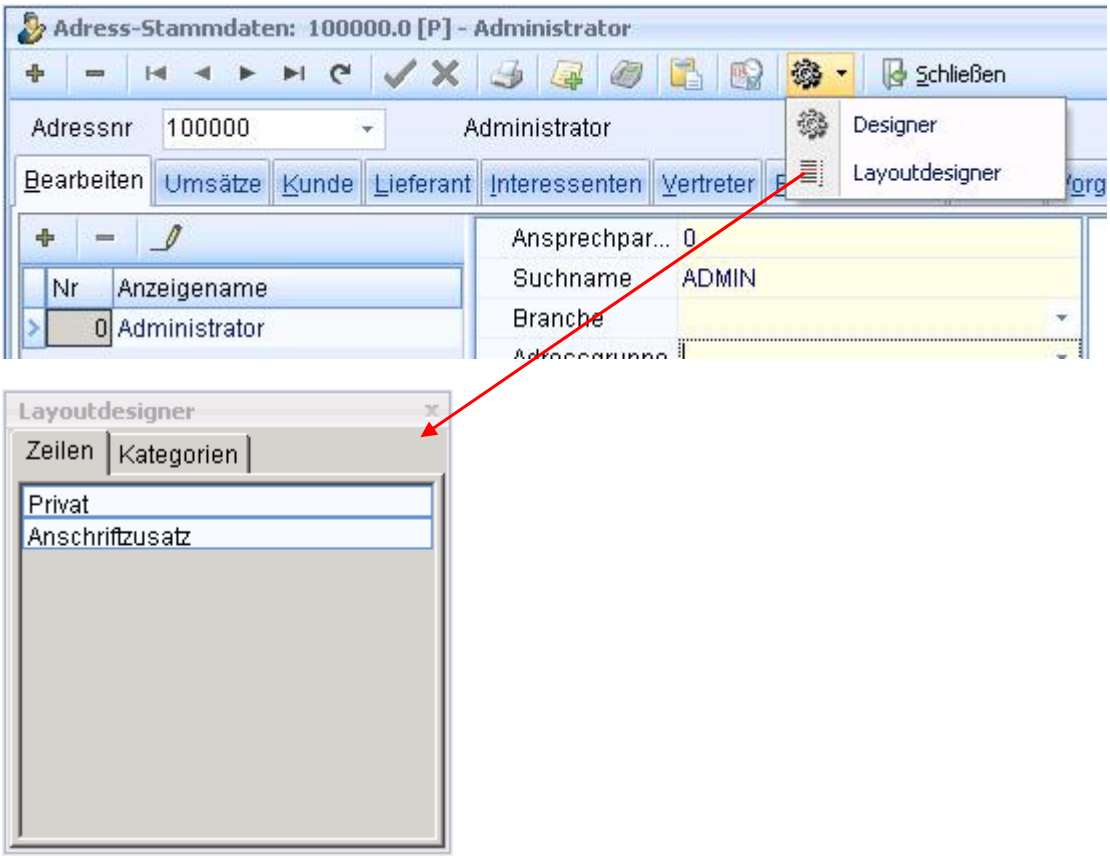
Weitere Buttons im Designer

Im Designer gibt es auf rechts fünf Buttons, diese sind nur auf bei Einstellung der Radiobox auf "Zeilen" aktiv:

- Editor hinzufügen **à** Über diesen Button lässt sich eine einzelne Gridzeile einfügen. Die Funktion entspricht der oben beschriebenen Vorgehensweise, dass man zunächst auf der Einstellungsseite "Felder" das gewünschte Datenfeld aktiviert und dann nach Wechsel auf die Einstellungsseite "Zeilen" weitere Eigenschaften vornimmt.
- Kategorie hinzufügen **à** Über diesen Button kann eine sog. Kategorie-Zeile (Überschrift) eingefügt werden. Diese Funktion ist auch über den Layoutdesigner (s.u.) verfügbar.
- **Multieditor hinzufügen à** Über diesen Button kann man eine Gridzeile zur Darstellung mehrerer Datenfelder "vorbereiten":
  Nach Anwahl des Buttons ist über die Eigenschaft Properties > Editors ein Hilfsfenster zu öffnen. Man kann dort die Felder eintragen, die nebeneinander im Grid dargestellt werden können. Nach Markieren eines Feldes im Hilfsfenster lassen sich im Griddesigner die gewohnten Definitionen zur Gestaltung als "normales" Eingabefeld, Checkbox, F4-Auswahlgrid etc. vornehmen (EditProperties etc... siehe obige Tabelle).
  Wichtig: Standard-Multieditoren können per Design nicht verändert werden. Bei Bedarf sind diese zu in der Anzeige zu deaktivieren und durch eigene zu ersetzen.
- Löschen **à** Löscht eine Zeile aus dem Grid
- Alles Löschen **à** Löscht den kompletten Grid

Layout-Designer

Diesen Designer gibt es nur bei einem vertikalen Grid. Der Layout-Designer muss aktiviert werden, wenn die Position eines Feldes innerhalb des Grids verschoben werden soll. Er dient weiterhin zum Gliedern von Datenfeldern innerhalb des Grids durch sogenannte Kategorien (=Überschriften).



Eigene Buttons und eigene Karteireiter im Adress-Stamm

In der Maske TAdrStamm (Adress-Stamm) gibt es erweiterte Design-Möglichkeiten: Sofern die Maskendesigner/Maskenaufruf-Lizenz gegeben ist, lassen sich eigene Buttons und Karteireiter realisieren. Hierzu sind bereits Objekte im Standard vorbereitet, die jedoch visible=false definiert sind. Die Objekte lassen sich freischalten und designen.

Eigene Buttons im Adress-Stamm

In der Adress-Stamm-Maske wurden auf der Navigationsleiste fünf Buttons für eigene Erweiterungen ergänzt und im Designer (Zahnrad-Symbol) zugänglich gemacht. Jedem Button kann ein individuelles Script (GDI-Basic, Pascal-Script) zugeordnet werden. In Voreinstellung sind die Buttons nicht sichtbar. Wird ein Button benötigt, so ist er über die Tag-Eigenschaft zu "aktivieren" und die weiteren Einstellungen vorzunehmen (aus technischen Gründen kann die naheliegende visible-Eigenschaft hierfür nicht verwendet werden!). Die Buttons reihen sich zwischen dem Button für den Terminaufruf und dem Designer-Button ein.

Eigenschaften der vordefinierten Buttons BT_User1 – BT_User5 (Objekttyp TdxBarButton)

- **Caption**: Beschriftung. Wird nur angezeigt, wenn ImageIndex = -1 eingestellt ist (kein Image) oder zusammen mit einem Image, wenn PaintStyle auf psCaptionGlyph eingestellt wurde
- **Description**: Hier wird hinterlegt, welches Script aufgerufen werden soll. Nach dem Schlüsselwort "GDIScript=" wird der Name einer gespeicherten Scriptdatei angegeben. Bei dieser Datei kann es sich um ein GDI-Basic oder um ein Pascalscript handeln. Sie wird beim Mandanten bzw. auf Programmebene im Ordner Script abgelegt
- **Hint**: Beschreibungstext, welcher angezeigt wird, wenn man sich mit der Maus über den Button bewegt
- **ImageIndex**: Nummer einer Grafik aus der internen (nicht änderbaren) ImageListe. Die Liste enthält rund 160 verschiedene Grafiken, welche meist in Navigationsleisten der Masken des CRM-Bereiches verwendet werden. Nicht zu verwechseln mit den Nummern für die Grafiksymbole des Menüs
- **Name**: Name des Buttons. **Darf nicht geändert werden**
- **PaintStyle**: Beim Defaultwert psStandard übersteuert die Angabe einer Grafiknummer bei der ImageIndex die ggf. vorhandene Caption. Bei Einstellung von psCaptionGlyph wird Caption und Image zusammen angezeigt
- **ShortCut**: Hinterlegung eines Tastenkürzels zum Start des Buttons
- **Tag**: Aktivieren des Button. 0 = ausgeblendet, 1 = aktiviert, der Button wird bei Neuaufruf der Maske sichtbar
- **Visible**: Im Designer muss hier **immer ivNever** eingestellt sein! Die Sichtbarkeit über über die Tag-Eigenschaft gesteuert

Es besteht ein (lesender) Zugriff auf die aktuellen Daten z.B. um aus dem Basic-Programm heraus Übergabeparameter zu bilden und andere Masken aufzurufen. Beispielsweise gibt es folgende Datenobjekte in der Maske Adress-Stamm:

- TA_Ansprech für CM_Adressen-Zugriff Ansprechpartner-Satz
- TA_Adressen für CM_Adressen-Zugriff Hauptdaten-Satz
- TA_Kunden für Zugriff Kunden-Tabelle
- TA_Liefer für Zugriff Lieferanten-Tabelle
- TA_Vertret für Zugriff Vertreter-Tabelle
- TA_Interess für Zugriff Interessenten-Tabelle
- TA_Beleg für Zugriff Beleg-Tabelle

- TA_Pos für Zugriff Belegposition
- TA_Doku für Zugriff auf Vorgang

Wichtig: Ein korrekter Zugriff ist i.d.R. nur möglich, wenn die zugehörige Karteikarte aktiv ist. Nur dann ist gewährleistet, dass die zugehörige Datenmenge geöffnet und "positioniert" ist. Das Karteikarten-Objekt der AdrStamm-Maske besitzt den Namen "PG_Adr", so dass sich über `@PG_Adr.ActivePage` leicht ermitteln lässt, welche Karteikarte aktiv ist.

§ Beispiel: Button zum Aufruf der Beleg-Adress-Artikel-Maske:

Im gezeigten Beispiel wurde der Button BT_User1 so gestaltet, dass sich die aus dem Kunden- bzw. Lieferantenstamm bekannte Funktion "Aufruf Beleg-Adress-Artikel" aufrufen lässt. Somit entfällt der Schritt, dass man aus der AdrStamm-Maske zunächst die entsprechende Stamm-Maske (Kunden, Lieferant) aufrufen muss, um an diese Auswertung zu gelangen.

Hier das Beispiel-Programm für den Aufruf der Beleg-Adress-Artikel-Maske, welches unter dem Namen " CMAdrArtBeleg.prg" auf Platte abgelegt ist:

```
//********  AdrArtBeleg-Aufruf aus CM-Adress-Stamm **********************
//  ab Bline 3.1.0.192  18.03.2011
//  Dieses Basic kann bei einem der Buttons BT_User1 - BT_User5 hinterlegt
//  werden (Eigenschaft "Description" --> "GDIScript=CMAdrArtBeleg.prg")
//  Letzte Änderung 18.03.2011  GL
//*********************************************************************
Adressnr := "0";
CR       := CHR(13,10);
Kartei := @PG_Adr.ActivePage;
//show(Kartei);
if SameText(Kartei,"TS_Liefer") then
   Adressnr := IntToStr(%TA_Liefer.FieldByname("Liefernr"));
   Name1    := %TA_Liefer.FieldByname("Name1");
   Ort      := %TA_Liefer.FieldByname("Ort");
endif;
if SameText(Kartei,"TS_Kunden") then
   Adressnr := IntToStr(%TA_Kunden.FieldByname("Kundennr"));
   Name1    := %TA_Kunden.FieldByname("Name1");
   Ort      := %TA_Kunden.FieldByname("Ort");
endif;
//show(Adressnr);
if Adressnr <> "0" then
   Start := "Adressnr=" + Adressnr + CR + "Name1=" + Name1 + CR + "Ort=" + Ort;
   show(Start);
   Startdialog("TFAdrArtBeleg",true,Start);
else
   show("Funktion Beleg-AdressArtikel nur für Kunde/Lieferant verfügbar");
endif;
exit;
```

Eigene Karteireiter im Adress-Stamm

In der Adress-Stamm-Maske (TAdrStamm) wurden zwei - im Standard nicht sichtbare - Karteireiter ergänzt und als TS_Zusatz1 und TS_Zusatz2 im Designer (Zahnrad-Symbol) zugänglich gemacht. Wird ein Karteireiter benötigt, so ist er über die TabVisible-Eigenschaft zu "aktivieren". Die Karteireiter werden somit sichtbar.
Jedem der Karteireiter entspricht ein vordefinierter Programmaufruf, welcher über das Menü-/Rechtesystem einem eigenen Menüpunkt zugeordnet werden kann:

Der Kartei "TS_Zusatz1" entspricht "TAdrStamm_Zusatz1"
Der Kartei "TS_Zusatz2" entspricht "TAdrStamm_Zusatz2"

Bei diesen beiden Programmaufrufen handelt es sich um Masken der "alten" Art, d.h. hier ist der Maskendesigner vorhanden.

Somit kann man zwei Masken designen, welche dynamisch in die AdrStamm-Maske eingebunden sind (vgl. beispielsweise auch die Konstruktion der Beleganzeige im alten Kunden- bzw. Lieferanten-Stamm, die letztlich einer dynamischen Einbindung der Maske "Belegtabelle" entspricht). Die Besonderheit dieser beiden Masken liegt nun darin, dass bereits ein sTable-Datenobjekt "TA_Zusatz1" bzw. "TA_Zusatz2" (**Objektnamen bitte nicht ändern!**) vorhanden ist. Sofern die hinterlegte SelectSQL das Datenfeld AdressID enthält, wird automatisch ein Filter aufgebaut, der diese Datenmenge (quasi als Slave) mit der im Adress-Stamm aktiven Adresse (Master) synchronisiert.

In beiden Fällen lautet die in Voreinstellung hinterlegte SelectSQL

```
Select * from CM_Adressen where Lfdnr = 0 order by Adressnr
```

Würde man bei Designen des "TAdrStamm_Zusatz1" einfach nur einen Grid und eine Navigationsleiste auf die Maske platzieren und beide mit DataSource "DS_Zusatz1" verbinden, so hätte das folgenden Effekt:

- bei Aufruf über den separaten Menüpunkt zeigt der Grid alle Adressen an
- bei Anklicken des Reiters für Zusatz1 im AdrStamm zeigt der Grid genau eine Adresse an - nämlich die aktuelle des AdrStammes

Hinweis: Die Navigationsleiste (NavBar) mit Anschluss an DS_Zusatz1 ist nur für den separaten Aufruf relevant. Für den in der AdrStamm-Maske eingebunden Aufruf ist diese nicht erforderlich. Dennoch empfiehlt sich die NavBar zum leichteren Designen der Maske. Zum Öffnen der Datenmenge genügt es, wenn die NavBar ohne besonderen "Schnickschnack" irgendwo klein auf der Maske liegt. Alternativ würde auch ein Basic-Button mit der Anweisung @TA_Zusatz1.active := true; zum Öffnen der Datenmenge genügen.

Die SelectSQL kann - vorausgesetzt die Datenmenge enthält das Datenfeld AdressID - beliebig geändert werden. Ein

```
Select * from Beleg order by BelegNr, BelegTyp, BelegArt
```

würde bei analoger Vorgehensweise ergeben:

- bei Aufruf über den separaten Menüpunkt zeigt der Grid alle Belege an
- bei Anklicken des Reiters für Zusatz1 im AdrStamm zeigt der Grid die Belege der AdrStamm-Adresse an.

Ausgehend von der TA_Zusatz1 bzw. TA_Zusatz2 lassen sich weitere Slave-Datenmengen "anschließen" (z.B. Anzeige von Belegpositionen). Dieses erfolgt in gewohnter Weise unter Zuhilfenahme von Navigationsleisten (NavBar).

Das select-Statement sollte aus Performance-Gründen grundsätzlich mit geeigneter "order by klausel" und Möglichkeit nicht mit kompletter Feldliste (*), sondern unter gezielter Angabe der notwendigen Felder z.B. select Feld1,… Feldn from kunden order by kundennr definiert sein.

Ergänzung 14.09.2012:

Für den Sonderfall, dass die Datenmenge auf der Zusatzmaske bearbeitbar ist, sollte die Tabelle TA_Zusatz1 nicht direkt für die Verarbeitung herangezogen werden, sondern lediglich als "Master" für die eigentliche Verarbeitungstabelle (Slave) dienen.
TA_Zusatz1 würde dann gegenüber dem Standard unverändert bleiben (SelectSQL: " Select * from CM_Adressen where Lfdnr = 0 order by Adressnr ") und man benötigt dann eine sTable für die Slave-Tabelle und eine Navbar mit
- MasterSource = DS_Zusatz1
- DataSource = DataSource der Slave-Tabelle
- Dialog.Filter = AdressID={AdressID}
- Dialog.Default = AdressID={AdressID}

Diese Variante gibt Ihnen viele Möglichkeiten, Daten aus dem Adressensatz mit in den Slave zu übernehmen, also nicht nur die AdressID...

§ Abbildungen zur Adress-Stamm-Erweiterung

Die folgenden Abbildungen zeigen exemplarisch die Einrichtung und Gestaltung der oben erwähnten Beispiele.

Abb. 1: Aktivierung des Zusatzkarteireiters über den Designer

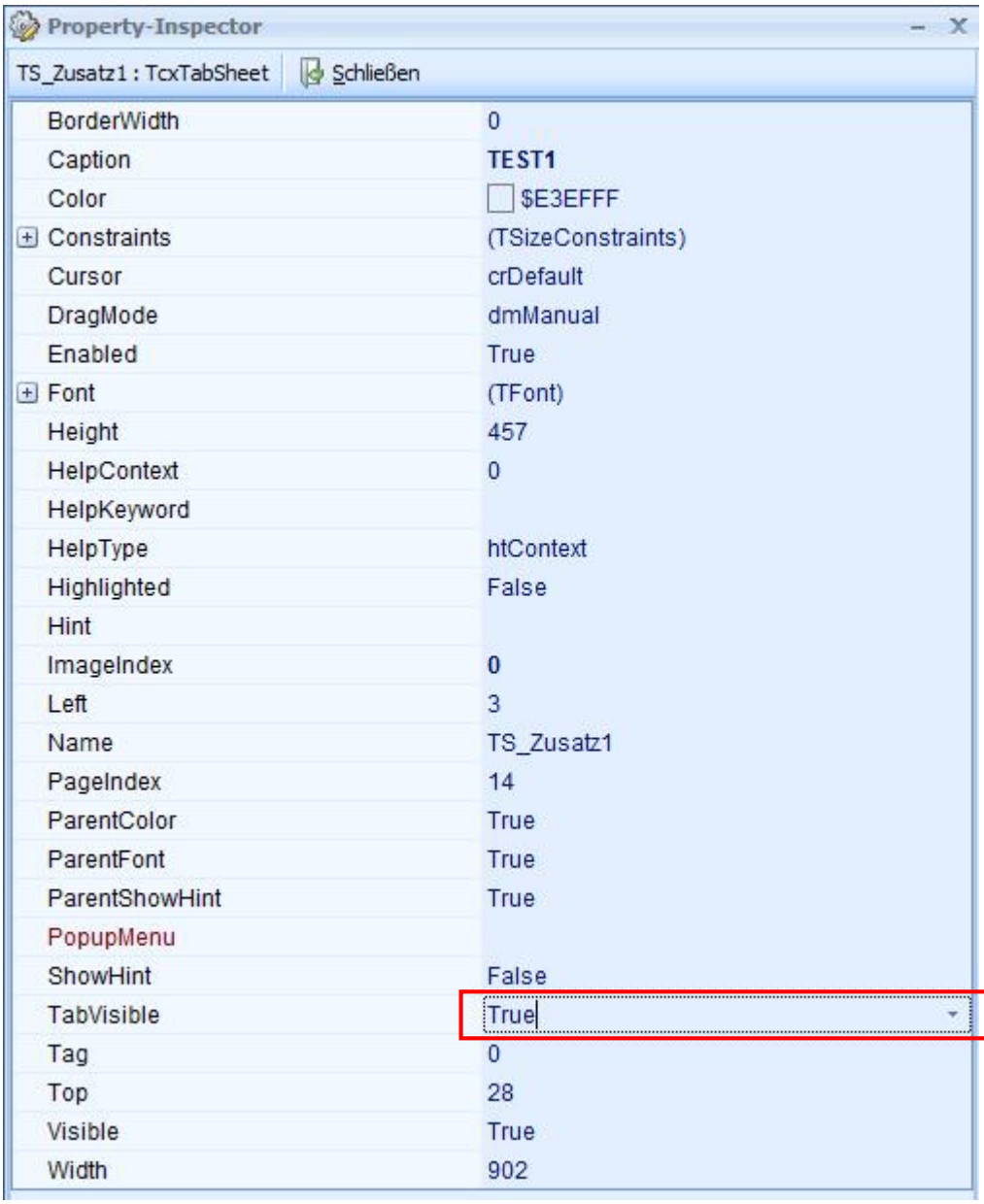


Abb. 2: Einrichtung des Menüpunktes zum Design der Zusatzmaske:
Zum Designen muss dieser Menüpunkt "sichtbar" sein, d.h. die Maske hierüber aufgerufen werden können. Für Bedienergruppen, welche die Maske lediglich in der Zusatzkartei im Adress-Stamm erhalten sollen, ist ein "versteckter" Menüpunkt anzulegen (siehe Technische Doku. zur Bline).

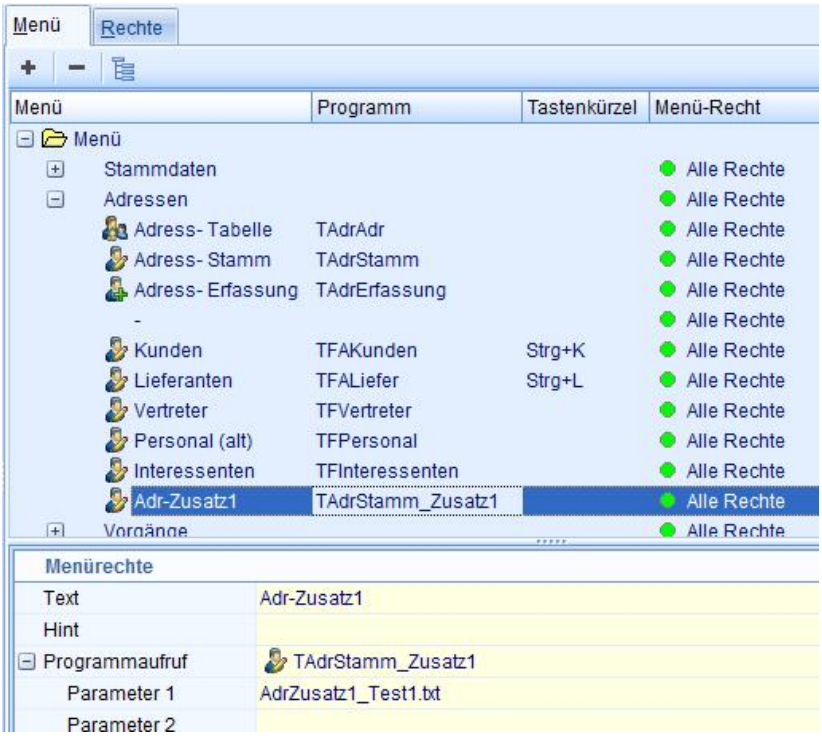


Abb. 3: Designen/Aufruf der Zusatz-Maske über den neuen Menüpunkt. Hier wurde lediglich eine NavBar und ein Grid an DS_Zusatz1 angeschlossen. Unter Verwendung der vorgegebenen SelectSQL mit Zugriff auf CM_Adressen zeigt der Grid alle CM_Adressen an.

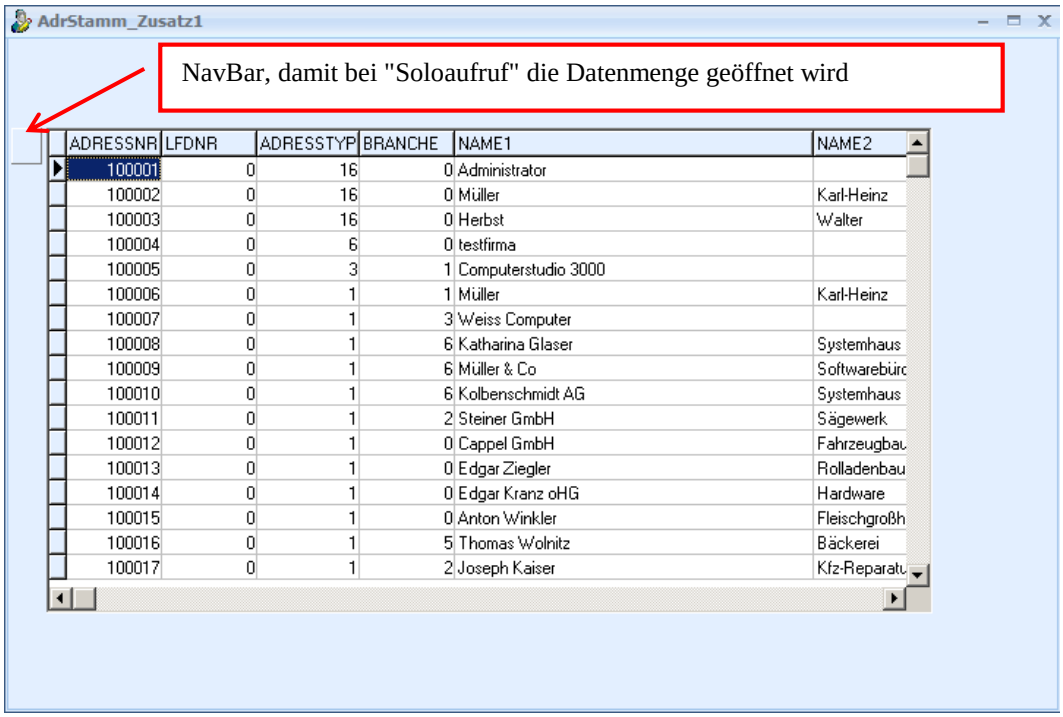


Abb. 4: Aufruf der Zusatz-Maske über den Karteireiter TEST1 im Adress-Stamm. Die Datenmenge wird automatisch mit der Master-Datenmenge des Adress-Stammes synchronisiert, so dass der Grid hier denselben Datensatz anzeigt.

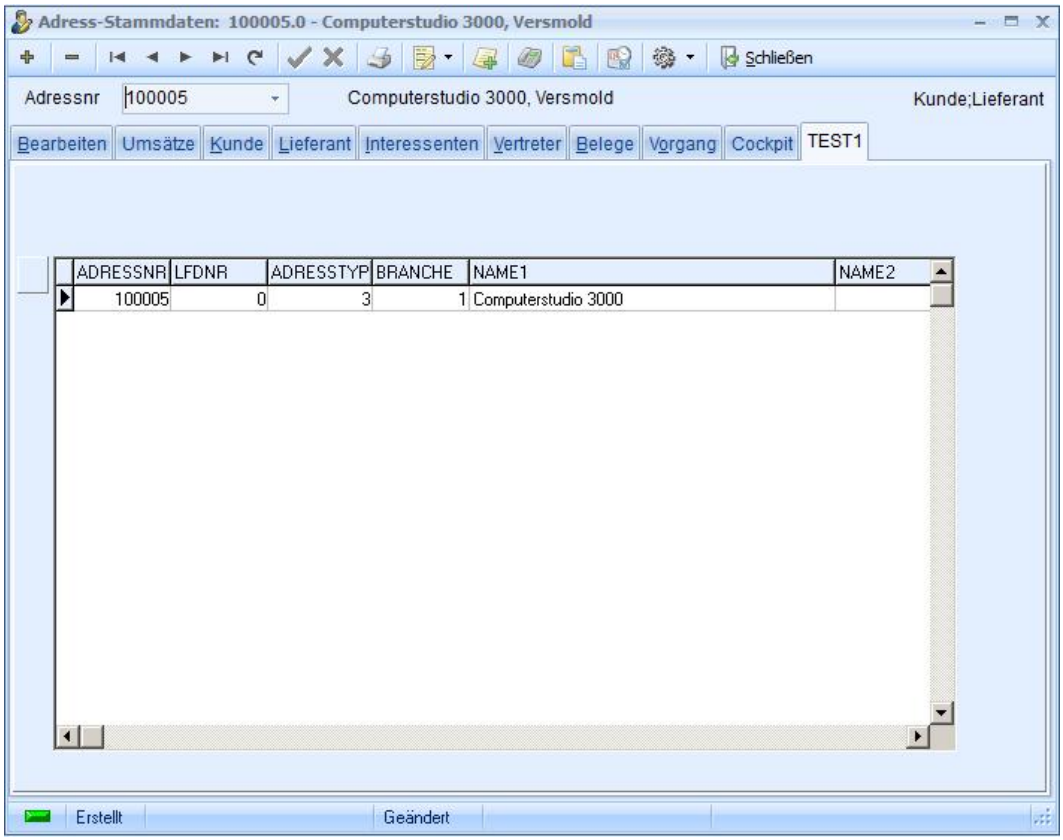


Abb. 3a: Designen/Aufruf der Zusatz-Maske über den neuen Menüpunkt. Wie bei Abb. 3, jedoch wurde hier die SelectSQL von TA_Zusatz1 so geändert, dass jetzt Belege angezeigt werden. Beim "Solo-Aufruf" zeigt der Grid alle Belege an.

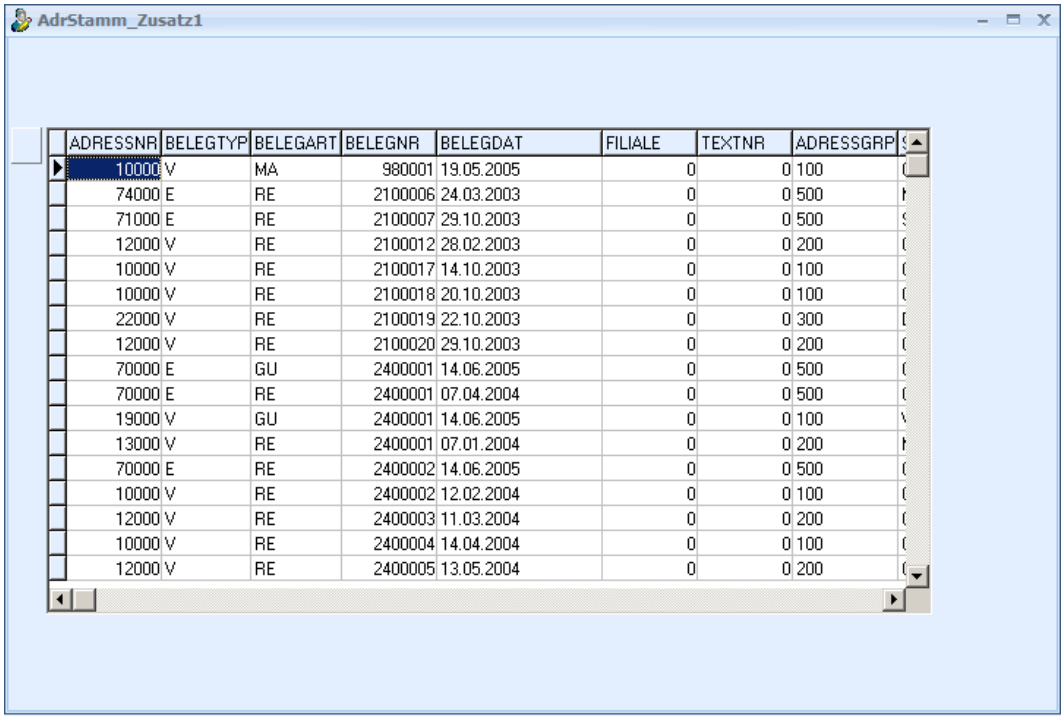
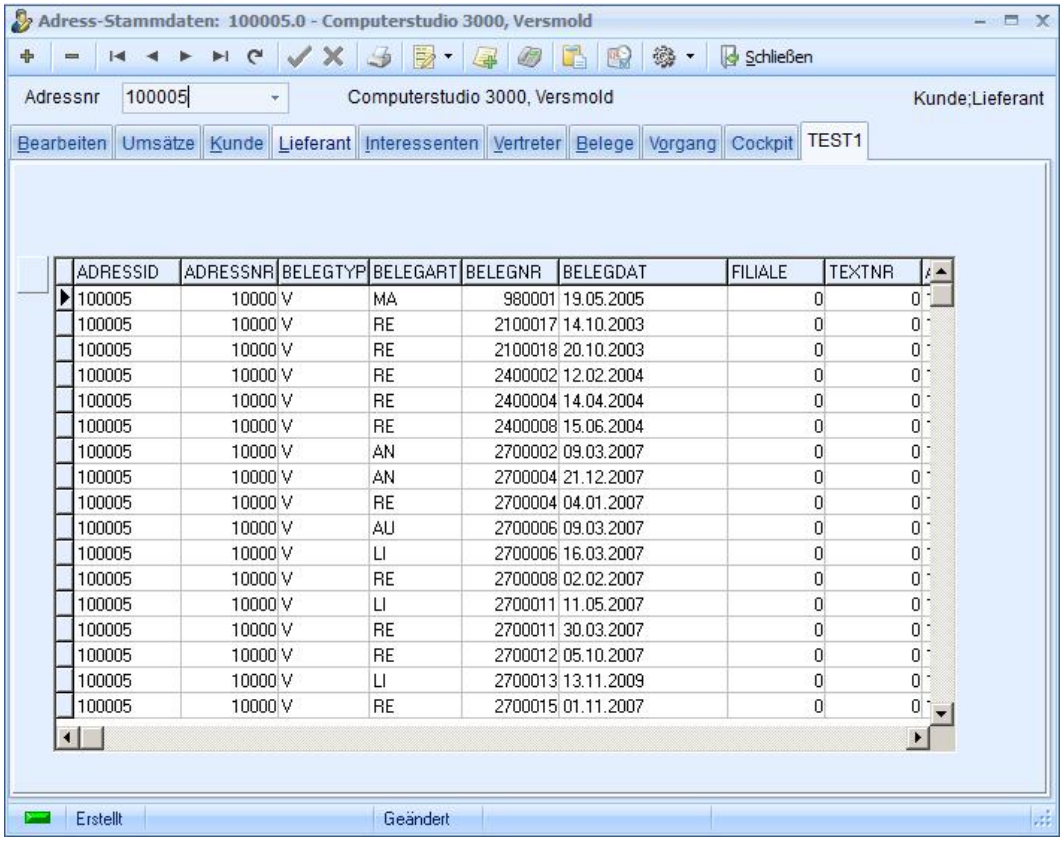


Abb. 4a: Aufruf der Zusatz-Maske über den Karteireiter TEST1 im Adress-Stamm. Die Datenmenge wird automatisch mit der Master-Datenmenge des Adress-Stammes synchronisiert, so dass der Grid nur noch Belege mit passender AdressID (zur Verdeutlichung als erste Spalte eingestellt) anzeigt.



ADRESSID	ADRESSNR	BELEGTYP	BELEGART	BELEGNR	BELEGDAT	FILIALE	TEXTNR
100005	10000	V	MA	980001	19.05.2005	0	0
100005	10000	V	RE	2100017	14.10.2003	0	0
100005	10000	V	RE	2100018	20.10.2003	0	0
100005	10000	V	RE	2400002	12.02.2004	0	0
100005	10000	V	RE	2400004	14.04.2004	0	0
100005	10000	V	RE	2400008	15.06.2004	0	0
100005	10000	V	AN	2700002	09.03.2007	0	0
100005	10000	V	AN	2700004	21.12.2007	0	0
100005	10000	V	RE	2700004	04.01.2007	0	0
100005	10000	V	AU	2700006	09.03.2007	0	0
100005	10000	V	LI	2700006	16.03.2007	0	0
100005	10000	V	RE	2700008	02.02.2007	0	0
100005	10000	V	LI	2700011	11.05.2007	0	0
100005	10000	V	RE	2700011	30.03.2007	0	0
100005	10000	V	RE	2700012	05.10.2007	0	0
100005	10000	V	LI	2700013	13.11.2009	0	0
100005	10000	V	RE	2700015	01.11.2007	0	0