# ISA Dialog Manager

## OBJEKTREFERENZ

A.06.03.b

In diesem Handbuch sind alle Objekte des ISA Dialog Managers beschrieben. Zu jedem Objekt sind seine vordefinierten Attribute und Methoden sowie die unterstützten Ereignisse und Kindobjekte aufgeführt.


**ISA Informationssysteme GmbH**

Meisenweg 33

70771 Leinfelden-Echterdingen

Deutschland

# Darstellungskonventionen

DM wird in diesem Handbuch synonym zu "Dialog Manager" verwendet.

Die Bezeichnung UNIX schließt generell alle unterstützten UNIX-Derivate ein - außer in den explizit angegebenen Fällen.

Dort wo für geläufige englische Fachbegriffe keine gängigen deutschen Übersetzungen existieren, wird zur Vermeidung von Unklarheiten der englische Begriff verwendet.

| | |
|---|---|
| < > | muss durch einen entsprechenden Wert ersetzt werden |
| **color** | Schlüsselwort ("keyword") |
| .bgc | Attribut |
| { } | optional (0 oder einmal) |
| [ ] | optional (0 oder n-mal) |
| <A> \| <B> | entweder <A> oder <B> |

### Beschreibungsmodus

Alle Schlüsselwörter sind fett und unterstrichen, z.B.

**variable** **integer** **function**

### Indizierung von Attributen

Syntax für indizierte Attribute:

[I]

[I,J] bzw. [row,column]

### Identifikatoren

Identifikatoren müssen mit einem Großbuchstaben oder einem "Unterstrich" ('_') beginnen. Die weiteren Zeichen können Groß-, Kleinbuchstaben, Zahlen oder Unterstriche sein.

Der Bindestrich ('-') ist für die Benennung von Identifikatoren als Zeichen ***nicht*** zugelassen!

Die maximale Länge eines Identifikators beträgt *31* Zeichen.

*Beschreibung der zugelassenen Identifikatoren in Backus-Naur-Form*

<Identifikator> ::= <erstes Zeichen>{<Zeichen>}

<erstes Zeichen> ::= _ | <Großbuchstabe>

<Zeichen> ::= _ | <Kleinbuchstabe> | <Großbuchstabe> | <Ziffer>

<Ziffer> ::= 1 | 2 | 3 | … 9 | 0

<Kleinbuchstabe> ::= a | b | c | … x | y | z

<Großbuchstabe> ::= A | B | C | … X | Y | Z

# Inhalt

# 1 Einführung

In diesem Handbuch finden Sie eine Übersicht über die Definition der Objekte mit ihren Attributen im Dialog Manager.

In der Dialogdefinition werden folgende **Objekte** verwendet:

***application***

Dieses Objekt für die Verarbeitung verteilter Anwendungen enthält Informationen über die Anwendung, deren Funktionen und die Art der Kommunikation. Diese Anwendungen können einzeln gestartet und wieder beendet werden.

***canvas***

Die Canvas dient zum Darstellen von dynamischen Informationen. Bei diesem Objekt stellt der DM nur die "Hülle" des Objektes; der eigentliche Inhalt muss dann von der Anwendung bereitgestellt werden.

***checkbox***

Mit Hilfe von Checkboxes kann man eine Einstellung vornehmen, so z.B. die des Layouts. Dabei ist es grundsätzlich möglich, aus mehreren gebotenen Alternativen von Angaben auch mehrere Angaben auszuwählen.

**control**

Das Control-Objekt definiert einen Client oder Server für die Kommunikation mit externen Objekten über das OLE-Protokoll von MICROSOFT WINDOWS.

***datetime***

Das ***datetime***-Objekt (Klassenname ***dmw_datetime***) stellt eine einfache und intuitive Eingabemöglichkeit für Datums- und Zeitwerte zur Verfügung.

Das Objekt ist als benutzerdefiniertes Widget (USW) implementiert.

***dialog***

Das Dialog-Objekt ist das erste Objekt in jedem Dialogskript. Es bildet die Wurzel der Objekthierarchie in diesem Skript.

**document**

Das Document-Objekt ist der Behälter für ein XML-Dokument, welches als DOM-Baum gespeichert wird.

### doccursor

Das Doccursor-Objekt verweist auf einen Knoten des DOM-Baums in einem XML-Dokument, dessen Kind der Doccursor ist, und kann durch seine Methoden im DOM-Baum bewegt werden.

### *edittext*

Mit Hilfe eines editierbaren Textes schaffen Sie die Möglichkeit, dass zur Laufzeit ein neuer Text an der entsprechenden Stelle eingegeben werden kann.

### *filereq*

Das Objekt Filerequestor ist ein modales Dateiauswahlfenster. Es unterscheidet drei Modi:

- Auswahl einer Datei zum Öffnen.
- Angabe einer Datei zum Speichern.
- Auswahl eines Verzeichnisses.

### *groupbox*

Die Groupbox dient als Gruppierungselement innerhalb des Fensters. Alle Objekte, die sich innerhalb einer Groupbox befinden, liegen auf derselben logischen Ebene.

### *image*

Hier können Sie ein Bild hineinladen, das zuvor mit Hilfe eines Grafikprogramms erstellt wurde.

### *import*

Die Verwendung eines Moduls in einem anderen Modul oder Dialog wird mit einem Import angekündigt.

**Siehe auch**

Kapitel „Modularisierung“ im Handbuch „Programmiertechniken“

### *layoutbox*

Die Layoutbox kann ihre Kinder selber anordnen. Sie wird daher genommen, wenn eine (unbekannte) Anzahl von gleichartigen Objekten dargestellt werden soll.

### *listbox*

In eine Listbox können Informationen in Form von schriftlichen Einträgen platzsparend und übersichtlich untereinander aufgereiht sowie zur Auswahl bzw. Selektion bereitgestellt werden.

### *listview*

Das ***listview*** (Klassenname ***dmw_listview***) ist ein erweitertes Listenobjekt, das verschiedene Darstellungsarten unterstützt.

Das Objekt ist als benutzerdefiniertes Widget (USW) implementiert.

### mapping

Mit einem Mapping-Objekt wird eine semantische Aktion definiert, die während einer Transformation für einen bestimmten Knoten eines XML-Dokuments oder einer IDM-Objekthierarchie durchgeführt wird.

***menubox***

Das Menü wird zur Dialogführung verwendet; mit seiner Hilfe können Aktionen ausgeführt oder Befehle aufgerufen werden.

***menuitem***

Ein Menüeintrag ist das eigentliche Objekt innerhalb eines Menüs, das vom Benutzer angewählt werden kann.

***menusep***

Neben dem Menüeintrag ist der Menüseparator das zweite Element innerhalb eines Menüs. Er dient lediglich zur optischen Trennung von Menüeinträgen.

***messagebox***

Mit Hilfe dieses Objektes können die im Fenstersystem verwendeten Standard-Messageboxes benutzt werden.

***module***

Ein Modul ist ein Teildialog, eine in sich geschlossene Einheit. Solch ein Dialogteil kann z.B. dienen als Bibliothek

- für Standardressourcen, die in einem Unternehmen eingesetzt werden sollen (Umsetzung firmenspezifischer Style-Guides!)
- für dialog- und projektübergreifende Vorlagen oder Modelle
- für immer wieder benötigte Funktionalitäten (z.B. Suche, Anmelden bei Datenbank...)

**Siehe auch**

Kapitel „Modularisierung“ im Handbuch „Programmiertechniken“

***notebook***

Das Notebook symbolisiert ein Notizbuch mit verschiedenen Seiten, die Notepages. Es ist in unterschiedliche Abschnitte unterteilt, die durch Reiter, sog. „Tabs“, markiert sind.

***notepage***

Dies ist die Seite eines Notebooks.

***poptext***

Mit Hilfe dieses Objektes können Informationen platzsparend dargestellt werden, wenn der Benutzer aus verschiedenen Informationen immer jeweils eine benötigt.

Die Combobox ist eine Kombination aus Eingabefeld und Auswahlliste. Über Attribute kann eingestellt werden, wie sich das Objekt verhält, ob z.B. die Auswahlliste ständig sichtbar ist oder erst bei Bedarf erscheint.

***progressbar***

Die Progressbar wird benutzt, um einen Fortschritt einer Verarbeitung anzuzeigen. Dazu können ihr Werte gesetzt werden, die den relativen Fortschritt einer Verarbeitung gegenüber der Gesamtaktionen bedeuten.

***pushbutton***

Der Pushbutton dient dazu, bestimmte Aktionen innerhalb des Dialoges zu bestätigen oder auszuführen.

***radiobutton***

Radiobuttons dienen dazu, eine "entweder-oder"-Entscheidung zu treffen, d.h. aus mehreren Alternativen darf immer nur eine ausgewählt werden.

***record***

Mit Records lassen sich Informationsobjekte definieren, die verschiedene, logisch zusammengehörende, Informationsarten enthalten.

***rectangle***

Das Rechteck dient lediglich als Objekt zur graphischen Abgrenzung von verschiedenen Objekten. Im Grenzfall (Höhe oder Breite = 1) kann das Rechteck als Linie eingesetzt werden.

***scrollbar***

Die Scrollbar ist ein Anzeigeobjekt, mit dessen Hilfe man bestimmte Einstellungen vornehmen bzw. anzeigen kann.

***setup***

Mit Hilfe dieses Objektes kann aus dem Dialog heraus die Systemkonfiguration abgefragt werden.

***spinbox***

Die Spinbox stellt eine Kombination aus Eingabefeld und Pushbuttons dar. Über die Pushbuttons (Pfeil nach oben, Pfeil nach unten) kann der Inhalt innerhalb eines Wertebereiches durchgescrollt werden.

***splitbox***

Die Splitbox ist ein Objekt, um einen Bereich in einem Fenster in beliebig vom Benutzer veränderbare Teilbereiche zu unterteilen.

***statictext***

Mit dem statischen Text definieren Sie einen Text, der zur Laufzeit nicht editierbar ist, sondern lediglich als feste Information dient.

***statusbar***

Die Statusbar wird am unteren Rand des Fensters dargestellt und dient zur Anzeige von Informationen.

**subcontrol**

Ein Subcontrol repräsentiert ein OLE-Kindobjekt, das direkt oder indirekt von einem OLE-Server verwaltet wird.

***tablefield***

Mit diesem Objekt können beliebig formatierte Listen ausgegeben werden. Darin übersichtlich dargestellte Daten können dort auch bearbeitet werden.

***timer***

Mit Hilfe dieses Objektes können Sie Aktionen zu fest definierten Zeitpunkten mit fest definierten Abständen durchführen lassen.

***toolbar***

Die Toolbar ist eine Funktionsleiste, in der Funktionen dem Benutzer angeboten werden können. Die Funktionen werden in der Regel durch die Anwahl eines Bildes aktiviert.

**transformer**

Das Transformer-Objekt ermöglicht es ein XML-Dokument oder eine IDM-Objekthierarchie zu durchlaufen und dabei an einzelnen Knoten semantische Aktionen auszuführen.

***treeview***

Mit Hilfe dieses Objektes können hierarchische Informationen dargestellt werden. Durch Selektion können Teilinformationen ein- und ausgeblendet werden.

***window***

Innerhalb eines Fensters können weitere Objekte angelegt werden; das Fenster dient somit als optischer sowie sinngemäßer Rahmen für andere Objekte.

Objekte können mit oder ohne eigenen Identifikator definiert werden. Objekte ohne eigenen Identifikator erben den Identifikator von ihrer Vorlage, werden aber weiterhin von Objekten mit eigenem Identifikator unterschieden. Objekte mit eigenem Identifikator werden als **benannte Objekte**, Objekte mit ererbtem Identifikator werden als **unbenannte Objekte** bezeichnet.

Innerhalb eines Vaters darf der Identifikator eines benannten Objektes für maximal ein benanntes Objekt verwendet werden. Es dürfen beliebig viele unbenannte Objekte mit demselben ererbten Identifikator definiert werden. Zusätzlich darf der ererbte Identifikator der unbenannten Objekte noch maximal einmal als Identifikator für ein benanntes Objekt verwendet werden.

Werden im Dialogskript innerhalb eines Vaterobjektes mehrere benannte Objekte mit demselben Identifikator definiert, erhält nur das erste Objekt diesen Identifikator. Bei allen weiteren Objekten wird beim Laden eine Warnung herausgeschrieben und der Identifikator ignoriert. Diese Objekte werden

also wie unbenannte Objekte behandelt. Beim späteren Herausschreiben ist der Identifikator nicht mehr vorhanden.

Werden Objekte über Pfadnamen referenziert, kann man mit *Parent.Child* auf das benannte Objekt *Child* zugreifen, das als Kind des Objektes *Parent* definiert wurde (davon existiert maximal eines, vgl. oben).

Existiert kein benanntes Kind-Objekt mit dem Identifikator *Child*, wird auf das erste unbenannte Kind-Objekt von *Parent* zugegriffen, das den Identifikator *Child* ererbt hat. Mit *Parent.Child[4]* kann auf das vierte Kind von *Parent* zugegriffen werden, das den Identifikator *Child* als unbenanntes Objekt von seiner Vorlage ererbt hat. Existieren innerhalb eines Vaterobjektes sowohl ein benanntes als auch unbenannte Objekte mit dem Identifikator *Child*, muss auch das erste unbenannte Kind indiziert angesprochen werden, also mit *Parent.Child[1]*.

Beim Schreiben einer Dialogdatei wird die Indizierung automatisch richtig vergeben.

### Siehe auch

Kapitel „Übersicht“ im Handbuch „Entwicklungsumgebung“ zum Aufbau des Dialog Managers

# 2 Konventionen bei der Objektbeschreibung

In den nachfolgenden Kapiteln werden die im Dialog Manager verfügbaren Dialog Manager-Objekte beschrieben.

Nach dem Keyword und der Syntax folgen Erläuterungen zum Objekt bzw. zu dessen Attributen. Des weiteren werden die für das Objekt erhältlichen Ereignisse und Menüs genannt sowie die möglichen Kindobjekte und Vaterobjekte.

### Anmerkung

Die Schlüsselwörter **export** und **reexport** bedeuten, dass das entsprechende Objekt auch außerhalb des Moduls, in dem es definiert ist, referenziert werden kann. **export** und **reexport** sind nur zulässig, wenn das Objekt innerhalb eines Moduls definiert ist.

Bei den im Folgenden aufgeführten Objekten ist ein Exportieren nur möglich, wenn auch das entsprechende Vaterobjekt exportiert ist.

*Siehe auch*

Kapitel „Modularisierung“ im Handbuch „Programmiertechniken“

Falls Sie noch weitere Informationen zu einem Attribut benötigen, steht Ihnen die „Attributreferenz“ zur Verfügung, in dem alle verfügbaren Attribute alphabetisch aufgelistet sind.

Bei jedem Objekt können objektspezifische Attribute aufgeführt sein. Dieses sind Attribute, die nur genau dieser Objekttyp hat oder die bei diesem Objekttyp eine spezielle Bedeutung haben.

# 3 Radmaus bzw. Radmausunterstützung unter Microsoft Windows

Das Scrollen eines Objektes mit Hilfe einer Scrollbar ist in der Regel klar: Es wird das Objekt gescrollt an dem sich die Scrollbar befindet. Um die Scrollbar zu bedienen, muss sich die Maus über der Scrollbar befinden.

Auch beim Scrollen über Tastatur ist es jedem relativ klar, dass das Objekt, das den Fokus besitzt gescrollt wird.

Leider gibt es diese Klarheit beim Scrollen mit dem Rad einer Radmaus nicht. Es kann entweder das Objekt, über dem sich die Maus befindet, oder das Objekt, das den Fokus besitzt, gescrollt werden. Die "Windows User Experience Guidelines" von Microsoft sagen leider nichts darüber aus, welches Objekt von einer Radmaus gescrollt werden soll. Und so gibt es in der Realität tatsächlich auch beide Modelle. Welches Modell verwendet wird, ist hierbei nicht von der Anwendung sondern vom verwendetem Maustreiber abhängig (hierzu später mehr).

Befragt man Computernutzer, so wird das eine oder das andere Modell für logisch erklärt, je nachdem welches der einzelne Nutzer gewohnt ist.

### Weshalb bestimmt der Maustreiber das Verhalten?

In der Onlinehilfe von Microsoft steht zwar zu der speziellen Radmaus Nachricht WM_ MOUSEWHEEL:

> *The WM_MOUSEWHEEL message is sent to the focus window when the mouse wheel is rotated. The DefWindowProc function propagates the message to the window's parent. There should be no internal forwarding of the message, since DefWindowProc propagates it up the parent chain until it finds a window that processes it.*

Das heißt, Microsoft sieht hier das Modell vor, dass das Fokusobjekt und nicht das Objekt unter der Maus gescrollt wird. Der IDM unterstützt dies auch, in dem er diese Nachricht wie gefordert behandelt.

Allerdings gab es die ersten Radmäuse bevor es die WM_MOUSEWHEEL Nachricht gab. Die Treiber dieser Radmäuse nutzten die Tatsache aus, dass ein Objekt scrollt wenn es die WM_VSCROLL oder WM_HSCROLL Nachricht erhält. Das heißt, es wird eine WM_VSCROLL oder WM_HSCROLL Nachricht und keine spezielle WM_MOUSWHEEL Nachricht versendet. Bei neueren Treibern lässt es sich sogar einstellen, wie diese arbeiten sollen.

### Fazit

Das Verhalten des IDM ist korrekt, da die WM_MOUSEWHEEL Nachricht richtig verarbeitet wird. Wenn der Maustreiber diese Nachricht gar nicht erzeugt, sondern lediglich normale WM_VSCROLL

und WM_HSCROLL versendet, dann liegt es am Treiber oder an dessen Einstellungen.

Leider ist uns kein offizieller Style Guide bekannt, der eine Aussage darüber trifft, welches Objekt beim Betätigen des Mausrades gescrollt werden muss.

# 4 Hinweis zu Mnemonics und Fokusrahmen unter Microsoft Windows

In Microsoft Windows kann eingestellt werden, ob Fokusrahmen und Mnemonics (Buchstaben mit Unterstrich) auch bei reinen Mausaktionen dargestellt werden soll oder nicht. Letzteres ist die Grundeinstellung von Windows.

Der Dialog Manager für Windows beachtet diese Einstellung, so dass in der Grundeinstellung keine Unterstriche und Fokusrahmen dargestellt werden. Sobald die `Alt`-Taste gedrückt wird, werden die Unterstriche wieder dargestellt.

Implizit wird dadurch auch die Darstellung des Fokusrahmens beeinflusst, was der Dialog Manager ebenfalls beachtet. Nach dem ersten Betätigen einer Navigationstaste (z.B. `Tab`) wird der Fokusrahmen wieder dargestellt.

Diese Einstellungen gelten immer nur für ein Fenster. Beim Öffnen eines neuen Fensters überprüft Microsoft Windows das letzte Eingabeereignis. Bei einem Mausklick werden, je nach Systemeinstellung, Unterstriche und Fokusrahmen gegebenenfalls nicht gezeichnet. Werden nach entsprechenden Tastaturereignissen Fokusrahmen oder Unterstriche wieder dargestellt bleibt diese Einstellung erhalten, bis das Fenster neu angelegt wird.

Unter Windows kann diese Systemeinstellung wie folgt geändert werden:

- *„Eigenschaften der Anzeige“* (Desktop) öffnen
  *„Start“* → *„Systemsteuerung“* → *„Anzeige“*
- Reiter *„Darstellung“* wählen
- Schaltfläche *„Effekte…“* auswählen
- Häkchen bei *„Unterstrichene Buchstaben für Tastaturnavigation ausblenden (mit Alt-Taste einblenden)“* entfernen.

# 5 Hinweis zu Objekten mit virtueller Größe (.vheight/.vwidth)

Das Clipping der Kindern von Objekten (z.B. groupbox, window, notepage oder toolbar) mit virtueller Größe (.vwidth/.vheight gesetzt) ist unter Windows und Motif unterschiedlich. Motif clipped die Kindobjekte entsprechend der virtuellen Größe, MS Windows entsprechend der Größe des Vaterobjektes.

Dies hat eine visuelle Auswirkung, wenn die virtuelle Größe kleiner als die des Vaterobjektes ist und ist beim Design von Dialogen zu beachten.

# 6 Besonderheiten unter Motif

## 6.1 Verwendung von Widget statt Gadget

Unter Motif kann bei einigen Objekten mittels des Werts *opt_use_widget* des Attributs .options die Verwendung des Widgets anstelle des Gadgets gewählt werden.

Für die Objektklassen ***pushbutton***, ***checkbox***, ***radiobutton***, ***menuitem*** und ***statictext*** nutzt der ISA Dialog Manager entweder „leichtgewichtige“ Gadget-Klassen oder „schwergewichtige“ Widget-Klassen von Motif. Die Nutzung der Widget-Klassen kann explizit durch *.options[opt_use_widget] := true* oder implizit durch bestimmte Attribute bzw. Bedingungen erzwungen werden. Die Verwendung der Widget-Klassen ist notwendig, wenn die genannten Objekte ein **Popup-Menü** besitzen oder mindestens eins der Attribute *.cursor*, *.statushelp* und *.toolhelp* gesetzt ist.

Gadgets und Widgets unterscheiden sich in der Nutzung von Farben. Gadgets verwenden für nicht gesetzte Farben normalerweise die entsprechenden Farben des Vaters. Unter Umständen passt Motif dabei ungesetze Farben an, zum Beispiel führt ein gesetzter schwarzer Hintergrund bei ungesetzter Vordergrundfarbe in der Regel zu einem weißen Vordergrund (Textfarbe) um die Lesbarkeit zu gewährleisten. Dagegen haben Widgets eigene Default-Farben, die für ungesetzte Farben verwendet werden.

Grundsätzlich gibt es in Motif keine Möglichkeit eine gesetzte Farbe zu entfernen. Das vom IDM erlaubte, dynamische Setzen einer Farbe auf *null* ist daher problematisch.

Der IDM versucht dem Standardverhalten von Motif Rechnung zu tragen. Bei *.options[opt_use_widget] = false* oder wenn die Nutzung von Widget implizit verursacht wird, benutzt der IDM für nicht gesetzte Farben die aktuelle jeweilige Farbe des Vaters. Das heißt, in diesen Fällen entspricht das Verhalten den Gadget-Klassen von Motif, wobei das spätere Setzen einer anderen Farbe beim Vater gewollt unberücksichtigt bleibt. Bei explizit erzwungener Benutzung von Widgets durch *.options[opt_use_widget] = true* entspricht die Nutzung von Farben den Widget-Klassen von Motif.

## 6.2 Positionierung und Sichtbarkeit von Objekten und ihre Fokussierbarkeit

Unter Motif können nur Objekte, die vollständig sichtbar sind oder in den sichtbaren Bereich gescrollt werden können, den Fokus erhalten. Dies ist das typische Verhalten von Motif-Anwendungen, im Unterschied zu MS Windows. Dabei ist zu beachten, dass ein Kindobjekt in einem Gruppierungsobjekt eventuell vollständig sichtbar ist, aber trotzdem per Tastatur unerreichbar und nicht fokussierbar ist, weil das Gruppierungsobjekt nicht vollständig sichtbar ist.

Aus Gründen der Kompatibilität zwischen den Plattformen ermöglicht es der ISA Dialog Manager auch unter Motif, Kindobjekte durch negative Positionsangaben für die x- und y-Koordinaten der linken oberen Ecke in nicht sichtbaren Bereichen zu positionieren. Motif erlaubt dies eigentlich nicht, sodass der IDM dafür Prüfmechanismen umgehen muss. Als Konsequenz wird dadurch meistens

ausgeschlossen, dass die Objekte per Tastaturnavigation erreichbar sind und den Fokus erhalten können.

### 6.2.1 Empfehlungen

- Der beste Weg, um die Erreichbarkeit von Objekten per Tastaturnavigation und ihre Fokussierbarkeit per Tastatur und Maus zu gewährleisten, besteht darin, sie immer vollständig im sichtbaren Bereich zu positionieren und negative Werte für ihre x- und y-Koordinaten zu vermeiden.
- Wenn dies nicht möglich ist, kann versucht werden, die Tastaturnavigation und Fokussierbarkeit mithilfe von *.options[opt_scroll_on_focus]*zu verbessern, die ab IDM-Version A.05.02.i bei Objekten mit virtueller Größe (Groupbox, Layoutbox, Notepage, Window) und dem Treeview vorhanden ist. Mit ihr kann eingestellt werden, ob ein Kindobjekt, das den Fokus erhält, in den sichtbaren Bereich gescrollt wird.

### 6.2.2 Option .options[opt_scroll_on_focus] (ab IDM-Version A.05.02.i)

Die Option *.options[opt_scroll_on_focus]* dient in erster Linie dazu, die Erreichbarkeit von Objekten per Tastaturnavigation zu verbessern und zu ermöglichen, dass sie in bestimmten Konstellationen per Tastatur oder Mausklick den Fokus erhalten können. Damit kann ein konsistenteres Verhalten zwischen Motif- und Windows-Anwendungen erreicht werden.

Allerdings kann durch die Option – aufgrund von Unterschieden zwischen den Plattformen bei der Fokus-Behandlung – weder eine vollständige Konsistenz erzielt werden, noch die Erreichbarkeit von Objekten per Tastatur und ihre Fokussierbarkeit in allen Situationen gewährleistet werden.

Datentyp der Option ist Boolean mit folgender Wirkung der Werte:

- *true* (Standard)
  Der IDM versucht bei Gruppierungsobjekten (Groupbox, Layoutbox, Notepage, Window) mit virtuellem Bereich bzw. beim Treeview, ein Kindobjekt bzw. einen Eintrag, die den Fokus erhalten sollen, in den sichtbaren Bereich zu scrollen.
- *false*
  Der IDM versucht **nicht**, zu fokussierende Kindobjekte bzw. Einträge in den sichtbaren Bereich zu scrollen.
  Zwar können nicht oder nur teilweise sichtbare Kindobjekte bzw. Einträge bei abgeschaltetem Scrollen den Fokus nicht erhalten, aber Motif erlaubt zumindest, Kindobjekte und Einträge zu fokussieren, die sich bereits vollständig im sichtbaren Bereich befinden.

## 6.3 Z-Ordnung von Fenstern und Dialogfeldern

Moderne Desktops und Fenstermanager haben immer wieder Probleme, das Vorschieben eines Applikationsfensters vor ein modales Fenster (z. B. Filerequestor, Messagebox) zu unterbinden, wenn mehrere Applikationsfenster vorhanden sind. Der IDM für Motif versucht dies über die korrekte Angabe der Modalität (applikationsmodal) und durch eine Heuristik zur Korrektur der Z-Ordnung von Fenstern sicherzustellen. Allerdings kann der IDM nicht alle Situationen korrigieren, weil einige

moderne Desktops und Fenstermanger in speziellen, modalen Situationen keine VisibilityNotify-Ereignisse weitergeben.

# 7 application

Mit Hilfe der Netzwerkversion können Dialog Manager-Anwendungen in zwei Teile oder in beliebig viele Teile aufgeteilt werden.

Die Teilung von Anwendungen in mehrere Anwendungsteile ist von großer Bedeutung, da es dadurch möglich ist, die Display-Rechner besser auszulasten und den Server zu entlasten. Des Weiteren ist es dadurch möglich, beliebige Aktionen wie Zeichnen von Grafik, Abfragen von externen Schnittstellen usw. auf dem Display-Rechner durchzuführen.

Dieses Objekt für die Verarbeitung verteilter Anwendungen enthält Informationen über die Anwendung, deren Funktionen und die Art der Kommunikation. Diese Anwendungen können dann einzeln gestartet und wieder beendet werden.

### Definition

```
{ export | reexport } application  { <Bezeichner> }
{
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

*finish*

*start*

### Kinder

document

Funktion

***record***

transformer

### Vater

***dialog***

***module***

### Menü

keins

## 7.1 Attribute

active

certificatefile

codepage

connect

dialog

document[l]

errorcode

exec

external

external[l]

extevent

firstrecord

index

label

lastrecord

local

options[enum]

password

privatekeyfile

publickeyfile

record[l]

recordcount

systemerror

transformer[l]

transport

username

## 7.2 Spezifische Attribute

Das Objekt ***Application*** besitzt keine Display-Attribute, sondern nur die folgenden Attribute:

| Attribut | Beschreibung |
|---|---|
| active | Mit Hilfe dieses Attributs kann definiert und abgefragt werden, ob die Anwendung gerade läuft. Eine Änderung dieses Attributes von *false* auf *true* bewirkt, dass die Anwendung gestartet wird. Wird dieses Attribut von *true* auf *false* geändert, so wird die Anwendung wieder beendet. |
| certificatefile (ab A.06.02.g) | In diesem Attribut wird die Zertifikatsdatei definiert, die für eine SSL-Verbindung verwendet wird. |
| codepage (ab A.06.01.d) | Dieses Attribut definiert die Codepage, mit der Applikationsfunktionen aufgerufen werden. |
| connect | Dieses Attribut definiert, dass sich die Anwendung mit einem laufenden Server-Prozess in Verbindung setzen soll, der auf dem Host gestartet worden ist (definiert durch Name oder IP-Adresse und der Portnummer). Bitte beachten Sie auch Kapitel „Abhängigkeit der Attribute .connect und .exec von der Transportschicht“.<br>Das Attribut kann nur geändert werden, wenn *.active = false* ist. |
| errorcode (ab A.05.01.d) | Das Attribut zeigt die Fehlerart an, welche bei der Aktivierung/Deaktivierung bzw. der Nutzung von Applikationsfunktionalitäten aufgetreten ist. Zurückgesetzt wird dieses Attribut intern bei der Aktivierung einer Applikation über das *.active*-Attribut (näheres siehe „Attributreferenz“). |
| exec | Dieses Attribut definiert, dass die Anwendung einen Prozess auf dem Host starten soll, der durch den Pfad gegeben ist (definiert durch Name oder IP-Adresse). Der Wert des Attributes beinhaltet den Programmnamen, Pfad, Rechnernamen und sonstige Informationen. Bitte beachten Sie auch Kapitel „Abhängigkeit der Attribute .connect und .exec von der Transportschicht“ und „Hinweis bei Verwendung von .exec zum Starten des Servers“.<br>Das Attribut kann nur geändert werden, wenn *.active = false* ist. |
| label | Interner Name der Anwendung, Identifikator. |
| local | Dieses Attribut definiert, ob die Anwendung lokal oder über ein Netzwerk laufen soll.<br>Das Attribut kann nur geändert werden, wenn *.active = false* ist. |

| Attribut | Beschreibung |
|---|---|
| options[enum] | Optionen für eine SSL-Verbindung.<br>» *opt_cert_required*<br>» *opt_no_ssl_v2*<br>» *opt_verify_peer*<br>**Verfügbarkeit**<br>Ab IDM-Version A.06.02.h |
| password<br>(ab A.06.02.g) | In diesem Attribut kann das Passwort definiert werden, das zum Starten der Anwendungsseite beim RSH- oder SSH-Protokoll verwendet wird. |
| privatekeyfile<br>(ab A.06.02.h) | In diesem Attribut wird die Datei mit dem privaten Schlüssel definiert, der für das SSH-Protokoll verwendet wird. |
| publickeyfile<br>(ab A.06.02.h) | In diesem Attribut wird die Datei mit dem öffentlichen Schlüssel definiert, der für das SSH-Protokoll verwendet wird. |
| systemerror<br>(ab A.05.01.d) | Dieses Attribut wird beim Auftreten eines Fehlers mit dem Fehlerstring, den das System liefert, gesetzt und kann dann im Fehlerfall abgefragt werden. |
| transport | Mit Hilfe dieses Attributes wird definiert, welchen intern vorhandenen Transportmechanismus die Kommunikationsschicht verwenden soll. Dabei gibt es folgende Möglichkeiten:<br>» *"tcpip"*<br>» *"dynlib"*<br>Das Attribut kann nur geändert werden, wenn *.active = false* ist. |
| username<br>(ab A.06.02.g) | In diesem Attribut kann der Benutzername definiert werden, das zum Starten der Anwendungsseite beim RSH- oder SSH-Protokoll verwendet wird. |

### 7.2.1 Abhängigkeit der Attribute .connect und .exec von der Transportschicht

Die Attribute *.connect* und *.exec* sind vom verwendeten Transport abhängig, d.h. zukünftige neue Arten einer Transportschicht können andere Arten des Verbindungsaufbaus enthalten.

Im Falle des Protokolls *"tcpip"* sieht die Syntax von *.connect* wie folgt aus:

```
"<host>:<portnumber>"
```

Im Parameter *host* wird der Rechnername, im Parameter *portnumber* die zu verwendende Portnumber angegeben.

Im Falle des Protokolls *"tcpip"* sieht die Syntax von *.exec* wie folgt aus:

```
"<host>[%<username>[%<password>]]:<path>"
```

Im Parameter *host* wird der Rechnername angegeben, auf dem das Programm gestartet werden soll.

Im Parameter *username* kann der Name des Users angegeben werden, unter dem der Benutzer auf dem Host-Rechner eingeloggt werden soll.

Im Parameter *password* kann das dort gültige Password des Benutzers angegeben werden. Diese beiden Parameter sind optional.

Im Parameter *path* wird der Pfad des zu startenden Programms angegeben.

Ab IDM-Version A.05.02.i unterstützt der DISTRIBUTED DIALOG MANAGER (DDM) das IPv6-Protokoll auf allen Architekturen, die IPv6 **nativ** unterstützen.

### 7.2.2 Hinweis bei Verwendung von .exec zum Starten des Servers

Bei Verwendung von *.exec* wird auf dem Anwendungsserver mittels **rexec**-Protokoll eine IDM-Anwendung gestartet, und der Darstellungsrechner wartet darauf (entsprechend **-IDMlisten**), dass sich diese Anwendung verbindet. Die Anwendung auf dem Anwendungsserver öffnet dann eine neue Verbindung (entsprechend *.connect*) um sich mit dem Darstellungsrechner zu verbinden. Dies wird im Normalfall eine „normale“ Verbindung sein, unabhängig davon, wie die Verbindung zum **rexecd**-Service aufgebaut wurde.

## 7.3 Beispiel

```
application TestAppl
{
  .active  false;
  .connect "localhost:4711";

  /* function definitions */
  function callback CheckFilename() for deselect, modified;

  function boolean  FillListbox(object,  string,
                                integer, integer,
                                string input output);

  function integer FillContinue(object, integer, integer,
                                string input output);

  function void  QueryListbox (object, string);
}
```

## 7.4 Funktionen von dynamischen Bibliotheken anbinden

Über das Applikationsobjekt besteht auch die Möglichkeit, Funktionen einer dynamischen Bibliothek direkt in der Regelsprache anzubinden. Dieses Funktionalität ist aber nur auf UNIX-Plattformen mit dl- oder shl-Support möglich, oder unter MICROSOFT WINDOWS (als DLL). Erlaubt sind nur Funktionen ohne ***record***-Parameter.

Unter UNIX muss die IDM-Bibliothek **libIDMdl.a** mit **–lIDMdl** (vor **libIDMinit.a**) sowie die dl-Systembibliothek dazu gelinkt werden, um das dynamische Anbinden zu ermöglichen, unter WINDOWS ist dies nicht notwendig.

Um zu Kennzeichnen, dass die unter dem Applikations-Objekt angegebenen Funktionen zu einer dynamischen Bibliothek gehören, ist das transport-Attribut auf den Wert *"dynlib"* zu setzen.

Im *.exec*-Attribut hinterlegt man den Pfad zur Bibliothek. Die Suchreihenfolge ist systemabhängig und kann folgendermaßen erklärt werden:

| Microsoft Windows | Unix |
|---|---|
| 1. Verzeichnis der Applikation<br>2. Arbeitsverzeichnis<br>3. 32bit-Systemverzeichnis (GetSystemDirectory)<br>4. 16bit-Systemverzeichnis<br>5. Windows-Verzeichnis<br>6. In den Verzeichnissen der PATH-Variable | Enthält der Pfad ein ‚/'-Zeichen:<br>» Absolut, oder relativ zum Arbeitsverzeichnis<br>Ansonsten:<br>1. In den zur Linkzeit angegebenen Verzeichnissen für dynamische Libraries<br>2. In den Verzeichnissen der LD_LIBRARY_PATH-Variable |

Die Anbindung der Funktionen erfolgt erst beim Aktivieren der Applikation über das *.active*-Attribut. Falls die Bibliothek nicht gefunden wurde, ist das Attribut trotz Setzung auf *true* danach *false*. Es können mehrere Applikationsobjekte auf ein und dieselbe Bibliothek zugreifen, dabei wird aber die Bibliothek auch nur ein mal geladen.

### Beispiel

*Dialog-Teil*

```
dialog D

application Appl
{
  .transport "dynlib";
  .exec "./fib.so";
  function integer Fib(integer);
}

on dialog start
{
```

```
  case setup.opsys_type
  in os_nt:
    Appl.exec := "fib.dll";
  endcase

  Appl.active := true;
  if Appl.active then
    print "Fibonacci-Zahl von 7 ist: "+Fib(7);
  else
    print "Bibliothek "+Appl.exec+" nicht ladbar.";
  endif
  exit();
}
```

*C-Teil*

```
#include "IDMuser.h"

#if defined(WIN32)
  __declspec(dllexport) DM_Integer DML_default DM_ENTRY Fib(DM_Integer i);
#endif

DM_Integer DML_default DM_ENTRY Fib(DM_Integer i)
{
  if (i <= 2)
    return 1;
  else
    return Fib(i-1)+Fib(i-2);
}
```

Unter Windows müssen bei der Generierung einer DLL die Funktionen noch über `__declspec (dllexport)` als exportiert gekennzeichnet werden, damit sie dynamisch vom IDM angebunden werden können.

### Zu beachten

Um innerhalb einer DLL auf DM-Funktionen konsistent mit der Anwendung zugreifen zu können, muss die Anwendung wie auch die DLL die DM-Bibliothek als dynamische Bibliothek (**dmdll.lib**) dazu gelinkt haben.

# 8 canvas

Hochspezialisierte und bestehende graphische Anwendungen (z.B. Prozessvisualisierungen), die direkten Zugriff auf das verwendete Fenstersystem brauchen, können das Dialogobjekt ***Canvas*** als „privaten“ Zeichenbereich nutzen.

Dieses Objekt stellt der DM der Applikation zur Verfügung. Der DM kümmert sich nicht um den Inhalt dieses Objektes, allein die Applikation ist für den Inhalt verantwortlich. Aus diesem Grund darf eine ***Canvas*** auch keine Kinder enthalten.

### Definition

```
{ export | reexport } { model } canvas  { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

Die Handhabung der im Kapitel „Attribute“ aufgelisteten Attribute übernimmt der DM in gewohnter Manier. Typischerweise wird dieses Objekt dann verwendet, wenn die Applikation dynamische Grafiken oder Schaubilder darstellen muss.

### Ereignisse

*cut*

*extevent*

*focus*

*help*

*paste*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 8.1 Attribute

acc_label

acc_text

accelerator

bgc

bordercolor

borderraster

borderstyle

borderwidth

canvasfunc

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstrecord

focus

font

function

groupbox

height

help

index

label

lastrecord

layoutbox

mapped

member[I]

membercount

menu

model

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

sizeraster

statushelp

toolhelp

toolbar

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 8.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a). |
| canvasfunc | Spezifiziert eine Funktion, die bei einem bestimmten Ereignis aufgerufen werden soll. Diese Funktion muss von der Anwendung bereitgestellt werden und muss vom Typ **canvasfunc** sein. |
| index | Index des Objekts im Kindvektor des Vaterobjekts. |
| options[enum] | Siehe Kapitel „Bedeutung des Attribut .options“.<br>Zusätzlich mögliche Indizes:<br>» *opt_addevents* (nur IDM für Qt)<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_graphicsview* (nur IDM für Qt) |

### 8.2.1 Bedeutung des Attribut .options

Mit Motif 1.1 können Sie nicht mittels der Tastaturnavigation über ein Objekt navigieren, wenn dieses Objekt ein „Composite-Widget“ ist oder dieses Widget keine Kinder hat. Es kann dabei auch zu einem Programmabsturz kommen.

Darum wurde für das Objekt Canvas die Option *opt_accept_child* verfügbar gemacht, welches entscheidet, ob ein DrawingArea-Widget („composite“, d.h. akzeptiert Kindobjekte) oder ein DrawnButton-Widget („primitive“, d.h. akzeptiert keine Kindobjekte) verwendet werden soll.

Das DrawnButton-Widget wird als Default verwendet.

Anwendungen, die eigene Widgets oder spezielle Widgets in der Canvas beinhalten, müssen die Option *opt_accept_child* setzen.

Wenn *opt_accept_child* auf *true* gesetzt wird, wird ein DrawingArea-Widget (Composite) für die Canvas verwendet. Default ist ein DrawnButton-Widget.

Die folgenden *enum*-Optionen werden nur ausgewertet, wenn *opt_accept_child* nicht gesetzt ist.

Wenn *opt_focus_frame* auf *false* gesetzt wird, wird die sog. „location-cursor-border“ (Tastaturfokus) nicht gezeichnet, wenn die Canvas den Fokus nicht hat. Der Default ist *true*.

| option_index | Bedeutung |
|---|---|
| *opt_accept_child* | Canvas akzeptiert Kind-Widgets (d.h. kein Fokus). |
| *opt_focus_frame* | Fokusrahmen wird nicht gezeichnet. |
| *opt_motif_shadow* | Canvas zeichnet Motif-Schattenrand. |

## 8.3 Beispiel

```
model canvas MCanvas
{
  .canvasfunc CanvasEvent;
  .xauto 0;
  .xleft 0;
  .xright 0;
  .yauto 0;
  .ytop 0;
  .ybottom 5;
  .fgc DiaColor20;
  .bgc Background;
  .posraster true;
  .sizeraster true;
  .font DiaFont;
}
```

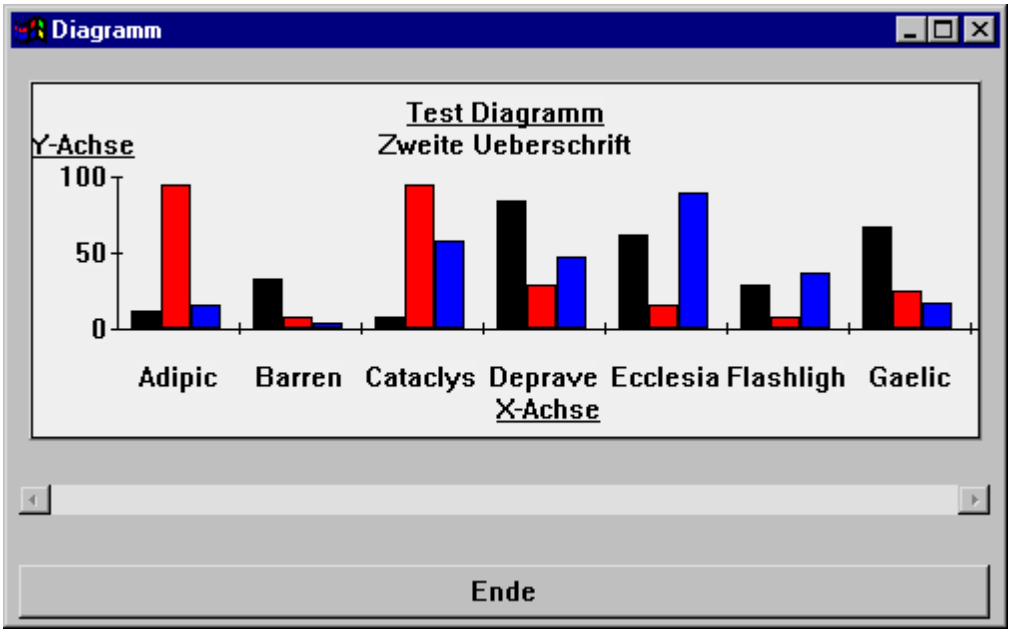



***Abbildung 1:*** *Canvas*

# 9 checkbox

Die ***Checkbox*** (Kontrollkästchen, Markierungsfeld) ist ein Knopftyp, der die Betätigungszustände „an“ und „aus“ kennt. Jedem Zustand ist eine eigene Darstellung der Checkbox zugeordnet, sodass stets erkennbar ist, welchen Zustand eine Checkbox gerade hat. So repräsentiert je nach Fenstersystem ein ausgefülltes Quadrat oder ein Quadrat mit einem Häkchen darin den Zustand „an“ und ein leeres Quadrat den Zustand „aus“.

### Definition

```
{ export | reexport } { model } checkbox { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Wichtige Anmerkung zur Tab-Steuerung

Checkboxen werden von den Fenstersystemen `Tab`-Gruppen zugeordnet. Die Gruppe wird dabei meist durch das Vater-Objekt bestimmt. Alleinstehende Checkboxen sind mit der `Tab`-Taste einzeln erreichbar. Bei Checkboxen innerhalb einer Gruppe kann nur ein Gruppenelement per `Tab`-Taste erreicht werden. Innerhalb der Gruppe wird mit den Cursortasten navigiert.
Über das Attribut .navigation, kann die `Tab`-Steuerung beeinflusst werden. Es legt fest, ob eine Checkbox bei der `Tab`-Steuerung zu einer Gruppe von Elementen gehört oder als alleinstehendes Element behandelt wird.

### Ereignisse

*activate*

*cut*

*deactivate*

*extevent*

*focus*

*help*

*key*

*paste*

*select*

**Kinder**

document

***record***

transformer

**Vater**

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup Menü**

## 9.1 Attribute

acc_label

acc_text

accelerator

active

bgc

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstrecord

focus

font

function

groupbox

height

help

index

label

lastrecord

layoutbox

mapped

member[l]

membercount

menu

model

navigation

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[l]

recordcount

scope

sensitive

sizeraster

state

statushelp

style

text

toolhelp

toolbar

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 9.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| active | Definiert bei *.style* = *checkbox* (*2*) ob das Objekt den Zustand „an“ (*true*) oder „aus“ (*false*) haben soll.<br>Siehe auch Kapitel „Checkbox als Tristatebutton“. |
| height | Höhe des Objekts.<br>Beim Wert *0* berechnet der IDM die Höhe automatisch aufgrund des aktuellen Zeichensatzes. |
| navigation | Legt fest, ob eine Checkbox bei der `Tab`-Steuerung zu einer Gruppe von Elementen gehört oder als alleinstehendes Element behandelt wird. Alleinstehende Checkboxen sind mit der `Tab`-Taste einzeln erreichbar. Bei Checkboxen innerhalb einer Gruppe kann nur ein Gruppenelement per `Tab`-Taste erreicht werden. Innerhalb der Gruppe wird mit den Cursortasten navigiert.<br>» *navi_default* (Standard)<br>» *navi_grouped*<br>» *navi_tab* |

| Attribut | Beschreibung |
|---|---|
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_use_widget* (nur Motif)<br>» *opt_push_like* (nur Microsoft Windows)<br>» *opt_center_toolhelp* (nur Microsoft Windows) |
| state | Aktueller Aktivierungszustand des Objekts.<br>Mögliche Werte:<br>» *state_unchecked*<br>» *state_checked*<br>» *state_indeterminate*<br>Siehe auch Kapitel „Checkbox als Tristatebutton“. |
| style | Legt die Darstellung und das Verhalten des Objekts fest.<br>Mögliche Werte:<br>» *checkbox* (2, default)<br>» *tristate* (3)<br>Siehe auch Kapitel „Checkbox als Tristatebutton“. |
| text | Angezeigter Text (Beschriftung). |
| width | Breite des Objekts.<br>Beim Wert *0* berechnet der IDM die Breite automatisch aufgrund des aktuellen Zeichensatzes. |

## 9.3 Checkbox als Tristatebutton

Checkbox und Tristatebutton werden im ISA Dialog Manager über dasselbe Objekt realisiert. Das Attribut .style steuert, welche Art von Objekt angezeigt wird. Die Darstellung der Tristatebuttons hängt vom jeweiligen Fenstersystem ab.

Die Zustände der Checkbox (als Tristatebutton) können über das Attribut .state abgefragt und gesetzt werden.

Bei einem Tristatebutton gibt das Attribut .active immer *false* zurück. Daher ist es nicht geeignet, den Zustand der Checkbox abzufragen. Dies ändert sich auch dann nicht, wenn die Checkbox zur Laufzeit dynamisch von *.style = tristate* (*3*) auf *.style = checkbox* (*2*) umgestellt wird.

Die Ereignisse activate und deactivate sind bei einem Tristatebutton nicht vorhanden.

Ob bei interaktiver Selektion des Tristatebuttons zwischen allen drei Zuständen oder nur zwischen „an“ (*state_checked*) und „aus“ (*state_unchecked*) gewechselt wird, sowie die Reihenfolge beim Wechsel zwischen den drei Zuständen, wird durch das jeweilige Fenstersystem definiert. Der IDM hat darauf keinen Einfluss.

## 9.4 Hinweiszur Checkbox unter Microsoft Windows

Beachten Sie zur Hintergrundfarbe die Anmerkungen beim Attribut .bgc in der „Attributreferenz“.

## 9.5 Beispiel

Vier navigierbare Checkboxen innerhalb eines Fensters, die ersten zwei innerhalb einer Gruppe:

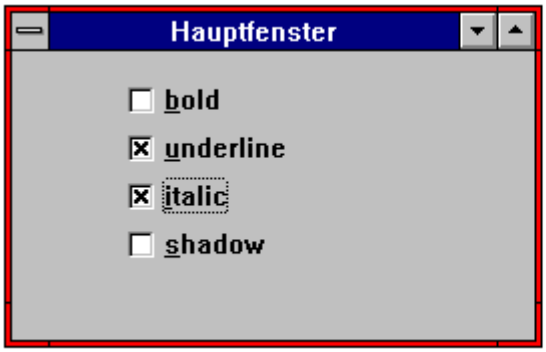
```
window MAIN
{
  /* tab navigation via grouping.
   * within the group each checkbox
   * can be reached using cursor keys */
  groupbox
  {
    .xleft 10;
    child checkbox Bold
    {
      .ytop 10;
      .text "bold";
      .visible true;
      .active false;
    }

    child checkbox Underline
    {
      .ytop 40;
      .text "underline";
      .visible true;
      .active true;
    }
  }

  /* tab navigation via .navigation.
   * each checkbox can be reached
   * individually using Tab key */
  child checkbox Italic
  {
    .xleft 10;
    .ytop 70;
    .text "italic";
```

```
    .visible true;
    .active true;
    .navigation navi_tab;
  }

  child checkbox Shadow
  {
    .xleft 10;
    .ytop 100;
    .text "shadow";
    .visible true;
    .active false;
    .navigation navi_tab;
  }
}
```




Hier sind die Checkboxen underline und italic ausgewählt worden.

***Abbildung 2:*** *Checkbox*

# 10 control

Die Aufgaben des Control-Objektes sind abhängig davon, in welcher Art das Control Objekt benutzt wird. Dabei gibt es zwei prinzipielle Möglichkeiten:

» Wenn das Control-Objekt als OLE-Client eingesetzt wird, dann wird mit Hilfe dieses Objektes die Kommunikation mit dem Server durchgeführt. Zusätzlich definiert dieses Objekt dann den Bereich, in welchem der Server innerhalb des Clients erscheinen soll. Alle Aufrufe an den Server zum Setzen und Abfragen von Properties oder zum Aufruf von Methoden gehen dann über dieses Control-Objekt.

» Wird das Control-Objekt als OLE-Server eingesetzt, dient dieses Objekt zur Definition der Schnittstelle und zur Kommunikation mit dem Client. Der Dialog Manager kann nicht all seine Objekte, Methoden, Ressourcen, Attribute etc. als Schnittstelle anbieten. Dies würde den Rahmen der Interfaces in OLE sprengen, und es ist auch nicht sinnvoll, die Gesamtheit eines Dialogs dem fremden Programm darzubieten. Die Attribute, Methoden und Ereignisse des Control-Objektes beschreiben das Interface des OLE-Servers zu seinen Clients.

### Definition

```
{ export | reexport } { model } control { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Rasterattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

*finish*

*help*

*key*

*paste*

*select*

*start*

### Kinder

***canvas***

***checkbox***

***edittext***

***groupbox***

***image***

***listbox***

***menubox***

***menuitem***

***menusep***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***rectangle***

***scrollbar***

***spinbox***

***statictext***

***tablefield***

***treeview***

***window***

**Vater**

***dialog***

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 10.1 Attribute

acc_label

acc_text

accelerator

active

bgc

bordercolor

borderwidth

child[I]

childcount

class

connect

cursor

cut_pending

cut_pending_changed

dialog

external

external[I]

fgc

firstchild

firstrecord

firstsubcontrol

focus

font

function

groupbox

height

help

index

label

lastchild

lastrecord

lastsubcontrol

layoutbox

license_key

mapped

member[I]

membercount

menu

message[I]

mode

model

name

notepage

parent

picture

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_xraster

real_y

real_yraster

record[I]

recordcount

reffont

scope

sensitive

sizeraster

statushelp

subcontrol[I]

subcontrolcount

toolhelp

userdata

uuid

visible

width

window

xauto

xleft

xraster

xright

yauto

ybottom

yraster

ytop

# 11 datetime

Das ***datetime***-Objekt (Klassenname ***dmw_datetime***) stellt eine einfache und intuitive Eingabemöglichkeit für Datums- und Zeitwerte zur Verfügung. Es kann Datums- oder Zeitwerte oder beides in unterschiedlichen Formaten anzeigen. Neben der Eingabe per Tastatur kann der Anwender die Werte ändern, indem er zum Beispiel ein Datum aus einem Kalender auswählt (oberes ***datetime***-Objekt in „Abbildung 3“) oder mit Spinbuttons einstellt (unteres ***datetime***-Objekt in „Abbildung 3“). Außerdem kann dem Anwender die Möglichkeit gegeben werden, den Wert mithilfe einer Checkbox auf „unbestimmt“ zu setzen (rechter Teil von „Abbildung 3“).

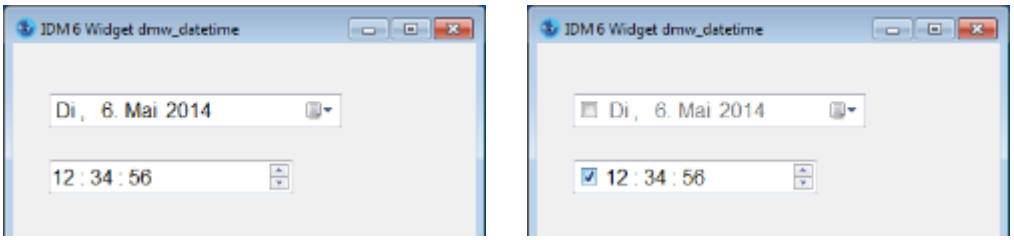



***Abbildung 3:*** *datetime-Objekte*

„Abbildung 4“ zeigt ein ***datetime***-Objekt mit aufgeklapptem Kalender. Die Darstellung des Kalenders kann durch mehrere Attribute angepasst werden. Beispielsweise werden im rechten Teil von „Abbildung 4“ die Nummern der Kalenderwochen und das heutige Datum im Kalender angezeigt.

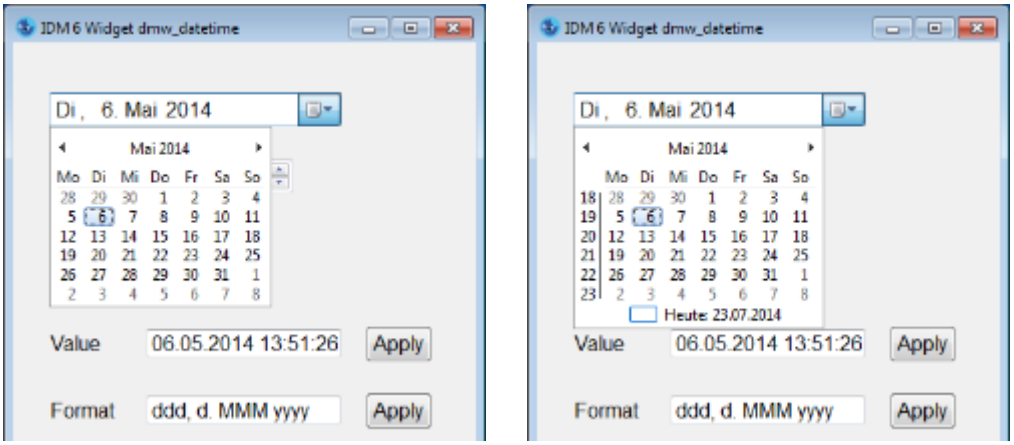



***Abbildung 4:*** *datetime-Objekt mit aufgeklapptem Kalender*

Die Darstellung der Datums- und Zeitwerte kann über einen Formatstring festgelegt werden. Man kann vordefinierte Systemformate nutzen oder mit den entsprechenden Formatierungszeichen eigene Anzeigeformate definieren. In beiden Fällen werden die Sprach- und Regionseinstellungen des Systems berücksichtigt (zum Beispiel bei den Monatsnamen). „Abbildung 5“ zeigt ***datetime***-Objekte mit Formaten, die außer Teilen des Datums oder der Zeit noch zusätzlichen, nicht editierbaren, Text enthalten.

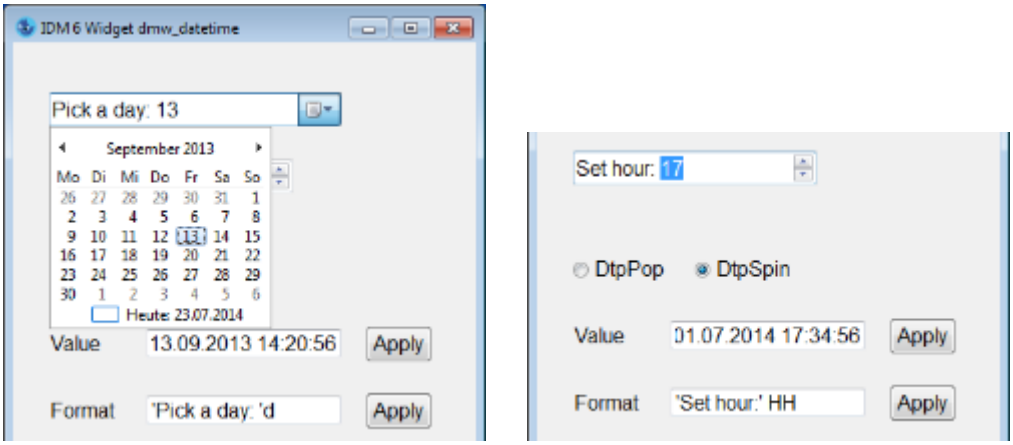



***Abbildung 5:*** *datetime-Objekte mit unterschiedlichen Formaten*

### Verfügbarkeit

- Nur unter MICROSOFT WINDOWS.
- Kann nur mit der USW-Option des ISA Dialog Managers verwendet werden.
- Die **idmwidgets.dll** muss sich im USW-Klassenpfad befinden (Standard: *<IDM-Installationsverzeichnis>\uswclasses*).

## 11.1 Definition

```
{ export | reexport } { model } dmw_datetime { <Bezeichner> }
{
  <Standardattribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

| | | | |
|---|---|---|---|
| **Ereignisse** | *changed* | *deselect* | *extevent* |
| | *focus* | *help* | *select* |
| **Kinder** | Keine | | |
| **Vater** | ***groupbox*** | ***layoutbox*** | ***notepage*** |
| | ***splitbox*** | ***toolbar*** | ***window*** |
| **Menü** | Popup Menü | | |

## 11.2 Beschreibung der Ereignisse

### 11.2.1 deselect

Das *deselect*-Ereignis tritt auf, wenn das ***datetime*** den Fokus verloren hat. Dies ist ein Hinweis darauf, dass der Anwender die Eingabe oder Auswahl von Datum oder Zeit abgeschlossen hat. Dabei ist allerdings zu beachten, dass dieses Ereignis nicht nur durch direkte Anwenderaktionen ausgelöst wird, sondern auch als Nebeneffekt anderer Aktionen auftreten kann:

- Der Anwender setzt den Fokus mit der Maus oder Tastatur auf ein anderes Objekt.
- Der Anwender aktiviert eine andere Anwendung oder eine andere Anwendung drängt sich in den Vordergrund und zieht den Fokus auf sich.

**Hinweis**

Das Öffnen eines Menüs oder Popups setzt nicht unbedingt den Fokus um. Daher kann nicht vorhergesagt werden, ob bei diesen Aktionen ein *deselect*-Ereignis auftritt. Zum Beispiel kann es sein, dass ein *deselect*-Ereignis erst auftritt, während die *select*-Ereignisregel eines ***menuitems*** ausgeführt wird, wenn es überhaupt auftritt.

### 11.2.2 select

Das *select*-Ereignis tritt jedes Mal auf, wenn sich der Wert des Attributs *.value* ändert. Das Ereignis kann bei unterschiedlichen Aktionen des Anwenders ausgelöst werden:

- Der Anwender bearbeitet den Wert im Eingabefeld des ***datetime***.
- Der Anwender navigiert im Kalender, um zu einem bestimmten Datum zu gelangen.
- Der Anwender wählt mit Maus oder Tastatur einen Wert aus.
- Der Anwender scrollt mit der Spinbox des ***datetime***.

Das heißt, in der Regel treten mehrere *select*-Ereignisse auf, bevor der Anwender den endgültigen Wert eingegeben, eingestellt oder ausgewählt hat. Daher sollten in der Ereignisregel des *select*-Ereignisses keine umfangreichen Aktionen ausgeführt werden.

Mit dem Attribut *.real_modified* kann geprüft werden, ob sich der Wert des ***datetime*** durch die Anwenderaktionen tatsächlich verändert hat, und zwar im Vergleich zum Wert des ***datetime***, als es den Fokus erhalten hat.

Das *select*-Ereignis kann auch bei einer Verletzung von *.minvalue* oder *.maxvalue* auftreten. Dies hängt vom Systemobjekt ab, wobei auch hierbei eine Wertänderung von *.value* vorliegen muss.

## 11.3 Geerbte Attribute

| Attribut | Datentyp | Hinweise |
|---|---|---|
| .accelerator | object | |
| .bgc | object | wird ignoriert |
| .borderraster | boolean | |
| .control | object | |
| .count[attribute] | anyvalue | |
| .cursor | object | |
| .cut_pending | boolean | |
| .cut_pending_changed | boolean | |
| .dialog | object | |
| .document[integer] | object | |
| .export | boolean | |
| .fgc | object | wird ignoriert |
| .firstrecord | object | |
| .focus | boolean | |
| .focus_on_click | boolean | |
| .font | object | |
| .function | object | |
| .function[integer] | object | |
| .groupbox | object | |
| .height | integer | |
| .help | string | |
| .index | index | |
| .label | string | |
| .lastrecord | object | |
| .mapped | boolean | |

| Attribut | Datentyp | Hinweise |
|---|---|---|
| .menu[integer] | object | |
| .model | object | |
| .module | object | |
| .navigable | boolean | |
| .notepage | object | |
| .parent | object | |
| .posraster | boolean | |
| .real_height | integer | |
| .real_path[string] | string | |
| .real_sensitive | boolean | |
| .real_visible | boolean | |
| .real_width | integer | |
| .record[integer] | object | |
| .recordcount | object | |
| .reexport | object | |
| .scope | object | |
| .sensitive | boolean | |
| .sizeraster | boolean | |
| .source | object | |
| .statushelp | string | |
| .target | object | |
| .toolbar | object | |
| .toolhelp | object | |
| .transformer[integer] | object | |
| .type[anyvalue] | datatype | |

| Attribut | Datentyp | Hinweise |
|---|---|---|
| .userdata | anyvalue | |
| .visible | boolean | |
| .width | integer | |
| .window | object | |
| .xauto | integer | |
| .xleft | integer | |
| .xright | integer | |
| .yauto | integer | |
| .ybottom | integer | |
| .ytop | integer | |

## 11.4 Spezifische Attribute

| Attribut | Kurzbeschreibung | Datentyp | Standard | Zugriff | | C | V |
|---|---|---|---|---|---|---|---|
| | | | | get | set | | |
| .allowundefined | Zulassen unbestimmter Werte | *boolean* | *false* | x | x | x | x |
| .calendaralignment | Ausrichtung des Kalenders | *integer* (*-1*, *0*, *1*) | *1* | x | x | x | x |
| .format | Anzeigeformat | *string* | "" | x | x | x | x |
| .maxvalue | Maximalwert | *string* | "" | x | x | x | x |
| .minvalue | Minimalwert | *string* | "" | x | x | x | x |
| .real_modified | zeigt tatsächliche Änderung von *.value* an | *boolean* | | x | – | – | – |
| .shortdaynames | Anzeige kurzer Wochentagsnamen | *boolean* | *false* | x | x | x | x |

| Attribut | Kurzbeschreibung | Datentyp | Standard | Zugriff | | C | V |
|---|---|---|---|---|---|---|---|
| | | | | get | set | | |
| .style | Werteingabe über aufklappbaren Kalender oder Spinbox | *class* (*poptext*, *spinbox*) | *poptext* | x | x | x | x |
| .today | Anzeige des aktuellen Datums | *boolean* | *true* | x | x | x | x |
| .todaymarker | Markierung des aktuellen Datums | *boolean* | *true* | x | x | x | x |
| .trailingdates | Anzeige von Tagen des vorherigen oder folgenden Monats | *boolean* | *true* | x | x | x | x |
| .value | Datums- und/oder Zeitwert | *string* | "" | x | x | x | x |
| .weeknumbers | Anzeige der Kalenderwoche | *boolean* | *false* | x | x | x | x |

C *changed*-Ereignis bei Änderung

V Attribut wird vererbt

# 12 dialog

In zumindest einer DM-Datei muss der Name des Dialoges definiert werden. Hierzu dient das Schlüsselwort **dialog**, gefolgt von dem Dialognamen und den Attributen des Dialogs.

### Definition

```
dialog { <Bezeichner> }
{
  <Standardattribute>
  <Rasterattribute>
  <Objektspezifische Attribute>
}
```

Bei größeren Projekten ist die Verteilung des Programmcodes auf mehrere Dateien, eine Modularisierung, sinnvoll. Weitere Informationen hierzu finden Sie beim Objekt import und im Kapitel „Modularisierung“ des Handbuchs „Programmiertechniken“.

### Ereignisse

*deactivate*

(wenn Timeout auftritt)

*extevent*

*finish*

*help*

*key*

*start*

### Kinder

document

***import***

***messagebox***

***record***

transformer

***window***

### Vater

keiner

**Menü**

keins

## 12.1 Attribute

child[I]

childcount

class

document[I]

external

external[I]

firstchild

firstrecord

label

lastchild

lastrecord

member[I]

membercount

msgboxtext[enum]

options[enum]

record[I]

recordcount

reffont

timeout

xraster

yraster

## 12.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| msgboxtext [enum] | Legt die Beschriftung der Messageboxbuttons fest (nur Motif). |

| Attribut | Beschreibung |
|---|---|
| options[enum] | Index: *opt_et_margin*.<br>Gibt an ob unter Windows ein Rand innerhalb des Eingabetextes erscheinen soll |
| reffont | Gibt den Namen des Zeichensatzes an, auf den sich das X- und Y-Raster beziehen.<br>Siehe auch Kapitel „Rasterattribute“. |
| timeout | Definiert, nach welcher Zeit (in Sekunden) ein Timeout wirksam werden soll, wenn der Benutzer nicht mit dem Programm arbeitet. |
| xraster | Dieses Attribut bildet die zugrundeliegende Einheit in X-Richtung.<br>Siehe auch Kapitel „Rasterattribute“. |
| yraster | Dieses Attribut bildet die zugrundeliegende Einheit in Y-Richtung.<br>Siehe auch Kapitel „Rasterattribute“. |

### 12.2.1 Rasterattribute

Mit Hilfe der im Dialog definierten Rasterwerte können die in ihm enthaltenen Fenster relativ zu einer fiktiven Größe oder zu einem festen, im Attribut reffont angegebenen, Zeichensatz positioniert werden. Über die Attribute xraster und yraster kann dabei ein Raster fest vorgegeben werden.

# 13 doccursor (XML-Cursor)

Das ***doccursor***-Objekt ist immer ein Kind eines XML-Dokuments. Es verweist auf einen Knoten des DOM-Baums, der im XML-Dokument (dem Vater) gespeichert ist.

Mittels Methoden kann der Verweis auf einen anderen Knoten des DOM-Baumes gesetzt werden. Verständlicher formuliert bedeutet dies, dass der XML-Cursor mittels Methoden im DOM-Baum bewegt werden kann. Er bleibt natürlich ein Kind des XML-Dokuments.

### Definition

```
{ export | reexport } { model } doccursor { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

Außer den „normalen“ Dialog Manager Attributen, besitzt der XML-Cursor Attribute, um auf Eigenschaften wie Name, Wert oder Attribute des DOM-Knotens zuzugreifen. Da dies Laufzeitattribute sind, sind diese nicht vererbbar. Außerdem sind viele dieser Attribut auch nur lesbar, da die entsprechende Eigenschaft des DOM-Knotens nicht geändert werden kann.

Der XML-Cursor ist zunächst ungültig, wird jedoch beim ersten Zugriff automatisch auf die Wurzel des DOM-Baums positioniert. Achtung, dies geschieht auch, wenn der XML-Cursor durch irgendeine Aktion ungültig geworden ist. Das Attribut *.mapped* liefert Auskunft darüber, ob der XML-Cursor gültig ist.

### Ereignisse

keine

### Kinder

document

***record***

transformer

### Vater

document

### Menü

keins

### Methoden

:add()

:delete()

:match()

:reparent()

:select()

:transform()

## 13.1 Attribute

attribute[I]

attribute[string]

data

dataselect[attribute]

dataselectattr[attribute]

dataselectcount[attribute]

dataselecttype[attribute]

document[I]

external

external[I]

firstrecord

idispatch

ixmldomnode

ixmldomnodelist

label

lastrecord

mapped

model

name

nodetype

parent

path

publicid

record[I]

recordcount

scope

specified

systemid

target

text

transformer[l]

userdata

value

xml

## 13.2 Objektspezifische Attribute

**attribute[l]**
**attribute[string]**

Das Attribut attribute dient je nach Indizierung zum Abfragen des Namens oder des Wertes eines Attributs des DOM-Knotens.

Ist der Index eine Zahl, dann wird der Name des entsprechenden Attributs des DOM-Knotens geliefert. Es ist zu beachten, dass Attribute eines DOM-Knotens primär unsortiert sind.

Ist der Index ein String, dann stellt der Index den Namen eines Attributs dar und es wird der Wert des Attributs zurückgeliefert. Eine Zuweisung auf das mit einem String indizierte Attribut attribute legt ein entsprechendes Attribut am DOM-Knoten an. Eine Zuweisung eines Leerstrings löscht das entsprechende Attribute des DOM-Knotens.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**data**

Dient zum Setzen und Abfragen der Daten des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_cdata_section* oder *nodetype_processing_instruction* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**dataselect[attribute] (ab IDM A.06.01.b)**

Mit diesem Datenmodellattribut werden gleichzeitig das als Index angegebene Datamodel-Attribut und ein als Wert zugewiesenes Selektionsmuster für Knoten eines XML-Dokuments definiert.

**dataselectattr[attribute] (ab IDM A.06.01.b)**

Dieses Datenmodellattribut definiert, mit welchem Knotenattribut das als Index angegebene Datamodel-Attribut verknüpft ist.

**dataselectcount[attribute] (ab IDM A.06.01.b)**

Dieses Datenmodellattribut definiert die Kardinalität des als Index angegebenen Datamodel-Attributs.

### dataselecttype[attribute] (ab IDM A.06.01.b)

Dieses Datenmodellattribut definiert den Datentyp, in den die Werte des als Index angegebenen Datamodel-Attributs konvertiert werden.

### idispatch

Über das idispatch Attribut kann unter Microsoft Windows auf den IDispatch COM Interface Pointer des XML-Cursors zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

### ixmldomnode

Über das ixmldomnode Attribut kann unter Microsoft Windows auf den IXMLDOMNode COM Interface Pointer des XML-Cursors zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

### ixmldomnodelist

Über das ixmldomnodelist Attribut kann unter Microsoft Windows auf den IXMLDOMNodeList COM Interface Pointer des XML-Cursors zugegriffen werden. Über diesen Interface Pointer kann auf die direkten Kinder des XML-Cursors zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**mapped**

Ist *true*, wenn der XML-Cursor auf einen Knoten im DOM-Baum zeigt. Es ist zu beachten, dass ein XML-Cursor, der auf keinen Knoten im DOM-Baum verweist, automatisch auf die Wurzel des DOM-Baums positioniert wird, wenn auf eines der objektspezifischen Attribute zugegriffen wird oder eine der objektspezifischen Methoden aufgerufen wird.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**name**

Name oder auch Tag des DOM-Knotens.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**nodetype**

Dient zur Abfrage des Typs des DOM-Knoten.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**path**

Liefert eine String-Repräsentation für die Position des XML-Cursors im DOM-Baum. Mit diesem String kann die select Methode aufgerufen werden, um eine XML-Cursor auf den Knoten im DOM-Baum zu positionieren.

Wird der Wert des path Attributs an anderer Stelle gespeichert (zum Beispiel im userdata Attribut), dann ist zu beachten, dass dieser gespeicherte Wert nicht angepasst wird, wenn die Struktur des DOM-Baums verändert wird. Ein anschließender Aufruf der Methode select mit diesem gespeicherten Wert, wird demzufolge den XML-Cursor auf einen falschen DOM-Knoten zeigen lassen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**publicid**

Öffentliche Kennung des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_entity* oder *nodetype_notation* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**specified**

Gibt an, ob ein Attribut eines DOM-Knotens explizit angegeben wurde oder von einem Standardwert ererbt wurde. Das Attribut ist immer *true*, außer für den nodetype *nodetype_attribute*.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**systemid**

Systemkennung des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_entity* oder *nodetype_notation* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**target**

Name der Instruktion eines DOM-Knotens. Der Wert entspricht dem Wert des name Attributes. Das Attribut ist nur verfügbar, wenn der nodetype *nodetype_processing_instruction* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**text**

Werte aller Unterknoten des DOM-Knotens. Es wird ein String geliefert, der den Text aller Unterknoten repräsentiert. Das Attribut ist hauptsächlich hilfreich, wenn man den Text eines XML-Elements benötigt und nicht erst zu dem Kindknoten, der den Text enthält, navigieren möchte.

Es ist zu beachten, dass das Setzen dieses Attributs alle Kindknoten löscht und einen neuen Textknoten anfügt.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**value**

Wert des DOM-Knotens. Das Attribut ist nur verfügbar, wenn der nodetype entweder *nodetype_attribute*, *nodetype_text*, *nodetype_cdata_section*, *nodetype_processing_instruction* oder *nodetype_comment* ist.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

**xml**

String-Darstellung des DOM-Knotens und all seiner Unterknoten.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

## 13.3 Objektspezifische Methoden

**:add()**

Fügt einen Kindknoten als letzten Knoten an den aktuellen DOM-Knoten an. Der XML-Cursor wird auf das neue Element gesetzt.

**:delete()**

Löscht den DOM-Knoten mit allen Kindknoten. Der XML-Cursor wird auf den Vaterknoten positioniert. XML-Cursor, die in den Unterbaum des gelöschten DOM-Knotens zeigen, werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

Werden die Werte des path Attributs an anderer Stelle gespeichert (zum Beispiel im userdata Attribut), dann ist zu beachten, dass diese gespeicherten Werte nicht angepasst werden, wenn die Struktur des DOM-Baums verändert wird. Ein anschließender Aufruf der Methode select mit einem dieser gespeicherten Werte, kann demzufolge den XML-Cursor auf einen falschen DOM-Knoten zeigen lassen.

**:match()**

Testet, ob der DOM-Knoten dem angegebenen Muster entspricht (siehe Kapitel „Muster für die Methoden :match() und :select()“).

**:reparent()**

Hängt den DOM-Knoten mit allen Kindknoten um.

Werden die Werte des path Attributs an anderer Stelle gespeichert (zum Beispiel im userdata Attribut), dann ist zu beachten, dass diese gespeicherten Werte nicht angepasst werden, wenn die Struktur des DOM-Baums verändert wird. Ein anschließender Aufruf der Methode select mit einem dieser gespeicherten Werte, kann demzufolge den XML-Cursor auf einen falschen DOM-Knoten zeigen lassen.

**:select()**

Bewegt den XML-Cursor in der angegebene Richtung oder bewegt den XML-Cursor auf den ersten DOM-Knoten, der dem angegebenen Muster entspricht (siehe Kapitel „Muster für die Methoden :match() und :select()“).

**:transform()**

Transformiert den XML-Cursor mit dem angegebenen Schema. Wenn das Ziel ein XML-Dokument ist, wird der gespeicherte DOM-Baum gelöscht und ein neuer DOM-Baum aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

Alternativ kann das Ziel der Transformation auch ein Text oder eine Datei sein. Ist das Resultat der Wandlung kein legales XML-Format, dann muss direkt in einen Text oder eine Datei gewandelt werden, da das Resultat nicht einem XML-Dokument zugewiesen werden kann. Dies ist zum Beispiel der Fall, wenn zu HTML gewandelt wird.

## 13.4 Muster für die Methoden :match() und :select()

Ein Muster ist ähnlich zu einem Dialog Manager Bezeichner. Das Muster bildet einen Pfad von Elementnamen. Der Pfad beginnt bei der Wurzel des DOM-Baums. Jede Hierarchiestufe wird mit dem entsprechenden Teil des Pfades verglichen. Zudem können noch bestimmte Eigenschaften wie Vorhandensein eines Attributs oder die Position innerhalb der Kinder bzw. des Vaters angegeben werden.

Im Muster ist prinzipiell jedes Zeichen mit Ausnahme der Zeichen **.**, **[**, **"**, **]**, **<**, **>**, **=**, ****, Tab, Leerzeichen und Zeilenumbruch zulässig. Soll eines der oben erwähnten Zeichen, .mit Ausnahme des Zeilenumbruchs, verwendet werden, dann muss ein **** vorangestellt werden. Zwischen zwei Anführungszeichen (**"**), also innerhalb eines Strings, sind zusätzlich auch die Zeichen **.**, **[**, **]**, **<**, **>**, **=**, Tab und Leerzeichen erlaubt. Es ist zu beachten, dass im Dialog Skript das Zeichen **** in einem String schon Escape-Zeichen ist, so dass im Dialog Skript immer **** anzugeben ist, wo ein **** benötigt wird. Ebenso muss im Dialog Skript **\"** angegeben werden, wo ein **"** benötigt wird.

```
{ <Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} }
  [ .<Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} ]
```

**<Name>**

Wird mit dem name Attribut des XML-Cursors verglichen. Der XML-Cursor muss den nodetype *nodetype_element* besitzen. Alternativ kann hier auch * angegeben werden, dann wird das name Attribute nicht beachtet.

Der Name eines XML-Elements beginnt mit einem Buchstaben oder einem Unterstrich und er kann Buchstaben, Ziffern, Bindestriche, Unterstriche und Punkte enthalten. Die genaue Definition für einen XML-Elementnamen kann in der XML-Spezifikation (**www.w3c.org/XML**) nachgelesen werden.

**<Attr>**

Der DOM-Knoten, auf den der XML-Cursor zeigt, muss das angegebene Attribut besitzen.

Der Name eines XML-Attributs beginnt mit einem Buchstaben oder einem Unterstrich und er kann Buchstaben, Ziffern, Bindestriche, Unterstriche und Punkte enthalten. Die genaue Definition für einen XML-Attributnamen kann in der XML-Spezifikation (**www.w3c.org/XML**) nachgelesen werden.

**<Op>**

Vergleichsoperator, um das in <Attr> angegebene Attribut mit dem <Value> zu vergleichen. Es kann auf gleich **=** und ungleich **<>** verglichen werden.

**<Value>**

Der Wert gegen den das <Attr> verglichen wird.

**<Idx>**

Der DOM-Knoten muss an dieser Position innerhalb der Kindknoten seines Vaterknotens stehen. Das erste Kind besitzt die Position 1.

Der <Idx> besteht nur aus Ziffern (**0** – **9**), die als Zahl interpretiert werden.

### Besonderheiten

**.**

Beginnt das Muster mit einem Punkt, dann ist es relativ zum aktuellen DOM-Knoten. Der Pfad beginnt also beim aktuellen DOM-Knoten.

**..**

Zwei aufeinander folgende Punkte überspringen beliebig viele Hierarchiestufen.

**[<Idx>]**

Ein Index ohne weitere Angaben selektiert den DOM-Knoten an dieser Position. Der nodetype des Knotens bleibt dabei unberücksichtigt. Jeder DOM-Knoten kann somit durch einen Ausdruck der Form {**[**<Idx>**]**[.**[**<Idx>**]**]} eindeutig referenziert werden (path Attribut).

### Erweiterung unter Microsoft Windows

Es kann auch XPath als Mustersyntax verwendet werden. Hierzu muss das Muster entweder mit / oder ./ beginnen. Die Verwendung von XPath Mustern ist wird nicht auf jeder Plattform unterstützt und ist somit nicht portabel.

# 14 document (XML-Dokument)

Das document Objekt ist der Behälter für ein XML-Dokument. Ein XML-Dokument wird als DOM-Baum gespeichert. Dieser DOM-Baum kann mit Hilfe eines Doccursor-Objekts, das ein Kind des Document-Objekts sein muss, durchwandert werden.

### Definition

```
{ export | reexport } { model } document { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

### Ereignisse

keine

### Kinder

doccursor

document

***record***

transformer

### Vater

***application***

***canvas***

***checkbox***

***dialog***

doccursor

document

***edittext***

***groupbox***

***image***

***import***

***layoutbox***

***listbox***

***menubox***

***menuitem***

***menusep***

***messagebox***

***module***

***notebook***

***notepage***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***statusbar***

***tablefield***

***timer***

***toolbar***

transformer

***treeview***

***window***

**Menü**

keins

**Methoden**

:load()

:save()

:transform()

:validate()

## 14.1 Attribute

doccursor[I]

document[I]

external

external[I]

firstrecord

idispatch

ixmldomdocument2

label

lastrecord

model

parent

real_version[enum]

record[I]

recordcount

scope

transformer[I]

userdata

version[enum]

xml

## 14.2 Objektspezifische Attribute

### doccursor[I]

Über das doccursor Attribut kann auf den XML-Cursor des XML-Dokuments zugegriffen werden. Das Attribut wird mit dem Objektindex indiziert (ähnlich zu child).

### idispatch

Über das idispatch Attribut kann unter Microsoft Windows auf den IDispatch COM Interface Pointer des XML-Dokuments zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

#### ixmldomdocument2

Über das ixmldomdocument2 Attribut kann unter Microsoft Windows auf den IXMLDOMDocument2 COM Interface Pointer des XML-Dokuments zugegriffen werden.

In der Regelsprache darf das Attribut nur demselben Attribut eines anderen Dialog Manager Objekts zugewiesen werden. In den Programmierschnittstellen ist zu beachten, dass das COM Objekt nur so lange gültig ist, wie der Dialog Manager dieses verwendet. Eine Anwendung sollte deshalb den Referenzzähler sofort erhöhen (COM Methode: IUnknown->AddRef). Wenn das Objekt nicht mehr gebraucht wird, muss der Zähler wieder heruntergezählt werden (COM Methode: IUnknown->Release). Es darf aber auf gar keinen Fall der Zähler öfter erniedrigt als erhöht werden, da sonst das COM Objekt freigegeben wird. Der Dialog Manager kann diese Situation nicht erkennen und wird abstürzen. Der Dialog Manager wird ebenso abstürzen, wenn der angegebene Zeiger nicht auf ein COM Interface zeigt.

Dieses Attribut wird nicht vererbt, da es sich auf eine Laufzeiteigenschaft bezieht.

#### xml

Über dieses Attribut kann man die String-Darstellung des XML-Dokuments erfragen. Wenn ein neuer Wert gesetzt wird, wird der gespeicherte DOM-Baum gelöscht und ein neuer DOM-Baum aus dem gesetzten Wert aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

## 14.3 Objektspezifische Methoden

#### :load()

Lädt ein XML-Dokument von der angegebenen Datei oder URL. Der gespeicherte DOM-Baum wird gelöscht und ein neuer DOM-Baum aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

#### :save()

Speichert XML-Dokument in Datei oder URL ab.

#### :transform()

Transformiert das XML-Dokument mit dem angegebenen Schema. Wenn das Ziel ein XML-Dokument ist, wird der gespeicherte DOM-Baum gelöscht und ein neuer DOM-Baum aufgebaut. Alle bestehenden XML-Cursor werden ungültig. Bei einem ungültigen XML-Cursor besitzt das Attribut *.mapped* den Wert *false*.

Alternativ kann das Ziel der Transformation auch ein Text oder eine Datei sein. Ist das Resultat der Wandlung kein legales XML-Format, dann muss direkt in einen Text oder eine Datei gewandelt werden, da das Resultat nicht einem XML-Dokument zugewiesen werden kann. Dies ist zum Beispiel der Fall, wenn zu HTML gewandelt wird.

**:validate()**

Überprüft das XML-Dokument, ob es dem im XML-Dokument angegebenen Dokumenttyp entspricht. Ist kein Dokumenttyp enthalten, wird ein Fehler gemeldet.

# 15 edittext (editierbarer Text)

Zur Handhabung von vom Benutzer editierbaren Texten dient das Schlüsselwort **edittext**.

### Definition

```
{ export | reexport } { model } edittext { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Scrollbarattribute>
  <Objektspezifische Attribute>
}
```

Innerhalb des Objektes Edittext kann der Textcursor mit den Cursortasten der Tastatur bewegt werden.

Die `Backspace`-Taste löscht den links vom Textcursor stehenden Buchstaben, die `Delete`-Taste löscht den rechts vom Textcursor stehenden Buchstaben.

Alle Buchstaben- und Zahlentasten werden als „eingegebenes Zeichen“ verarbeitet.

### Ereignisse

*activate*

*charinput*

*cut*

*deselect*

*deselect_enter*

*extevent*

*focus*

*help*

*key*

*modified*

*paste*

*select*

**Kinder**

document

***record***

transformer

**Vater**

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 15.1 Attribute

acc_label

acc_text

accelerator

active

alignment

bgc

borderraster

borderstyle

borderwidth

class

content

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

editable

endsel

external

external[I]

fgc

firstrecord

focus

font

format

formatfunc

function

groupbox

height

help

hinttext

hsb_visible

index

label

lastrecord

layoutbox

mapped

maxchars

member[I]

membercount

menu

model

multiline

navigable

notepage

options[enum]

parent

posraster

real_height

real_modified

real_sensitive

real_visible

real_width

real_x

real_y

record[l]

recordcount

scope

sensitive

sizeraster

startsel

statushelp

textwidth

toolhelp

toolbar

userdata

visible

vsb_visible

width

window

xauto

xleft

xmargin

xright

yauto

ybottom

ymargin

ytop

## 15.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| alignment (nur IDM FÜR WINDOWS) | Ausrichtung des Textes im Objekt. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. Nur aktiv, wenn *.multiline = true*. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a). Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |
| content | Inhalt (Text) des Objekts. Siehe auch Kapitel „Edittext als Kind einer Spinbox“. |
| editable | Steuert die Editierbarkeit des Objekts. |
| endsel | Ende eines selektierten Teilstücks in einem Editierbereich. Siehe auch Kapitel „Markieren von Text und Cursorposition“. |
| format | Definiert ein Format für den vom Objekt angezeigten String. |
| formatfunc | Gibt die zum Objekt gehörende Formatfunktion an. |
| hinttext | Platzhalter- oder Hinweistext des Objektes. Siehe auch Kapitel „Hinweise zum Attribut .hinttext“ |
| hsb_visible | Sichtbarkeit der horizontalen Scrollbar. Siehe auch Kapitel „Multiline-Edittext und Scrollbarattribute“. |
| maxchars | Maximale Anzahl möglicher Zeichen im Eingabestring. |
| multiline | Steuert, ob das Objekt ein- oder mehrzeilig dargestellt wird. Siehe auch Kapitel „Multiline-Edittext und Scrollbarattribute“. |
| navigable | Steuert die Fokussierbarkeit (Erreichbarkeit des Objekts mittels Tastatur). |

| Attribut | Beschreibung |
|---|---|
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_html* (nur IDM für Qt)<br>» *opt_rtf* (nur IDM für Windows, siehe Kapitel „Editierbarer Text mit Formatierungen (RTF-Edittext)“) |
| real_modified (nicht unter Motif) | Zeigt an, ob der Inhalt seit dem Fokuserhalt tatsächlich geändert wurde. |
| startsel | Anfang eines selektierten Teilstücks in einem Editierbereich, siehe auch Kapitel „Markieren von Text und Cursorposition“. |
| textwidth (nur IDM FÜR WINDOWS) | Textbreite eines RTF-Edittextes (siehe auch Kapitel „Attribut .textwidth“). |
| vsb_visible | Sichtbarkeit der vertikalen Scrollbar.<br>Siehe auch Kapitel „Multiline-Edittext und Scrollbarattribute“. |
| xmargin | Horizontaler Abstand zwischen Umrandung und Text des Objekts in Pixeln, siehe auch Kapitel „Hinweise zu den Attributen .xmargin und .ymargin“. |
| ymargin (nur Motif) | Vertikaler Abstand zwischen Umrandung und Text des Objekts in Pixeln, siehe auch Kapitel „Hinweise zu den Attributen .xmargin und .ymargin“. |

### 15.2.1 Hinweise zu den Attributen .xmargin und .ymargin

Der Abstand des Inhalts zum rechten und linken Rand bestimmt das Attribut xmargin (unter Microsoft Windows muss am Dialog das Attribut *.options[opt_et_margin] = false* gesetzt sein (Defaultwert), außerdem kann bei überlangen Texten (auch abhängig vom Windows-Patchlevel) und einem einzeiligen Edittext unter Umständen der Abstand zum rechten Rand ignoriert werden.).

Zusätzlich kann unter Motif noch der Abstand zum oberen und unteren Rand mittels ymargin gesetzt werden.

Sowohl *.xmargin* als auch *.ymargin* können unter bestimmten Bedingungen die Größe des Objekts beeinflussen, näheres dazu in der „Attributreferenz“ bei den Detailbeschreibungen.

### 15.2.2 Hinweise zum Attribut .hinttext

Dieses Attribut wird nur von WINDOWS und QT unterstützt.

Beim ***edittext***-Objekt unter WINDOWS wird .hinttext nur unterstützt, wenn die Attribute .multiline und .options[opt_rtf] den Wert *false* besitzen.

### 15.2.3 Multiline-Edittext und Scrollbarattribute

Ein editierbarer Text kann nur dann mit Scrollbarattributen hsb_visible und vsb_visible versehen werden, wenn das Attribut multiline auf *true* gesetzt ist. (Die Attribute *.multiline* sowie *.vsb_visible* und *.hsb_visible* sind unter Motif unabhängig voneinander.)

### 15.2.4 Markieren von Text und Cursorposition

Um den Cursor innerhalb eines Edittextes zu setzen oder Bereiche eines Edittextes zu markieren müssen die Attribute startsel und endsel benutzt werden. Werden Sie auf denselben Wert gesetzt, so geben sie die Cursorposition an. Werden Sie auf unterschiedliche Werte gesetzt, geben sie den Bereich an, der selektiert erscheinen soll. In der Regel bestimmt das Attribut .endsel die Cursorposition (zu Ausnahmen siehe startsel und endsel in der „Attributreferenz“). In Normalfall wird dieser Selektionsbereich aber nur angezeigt, wenn der Edittext den Fokus besitzt.

### 15.2.5 Edittext als Kind einer Spinbox

Ist ein Edittext ein Kindobjekt eines ***spinbox*** Objekts, darf das Attribut content nicht genutzt werden. Die Wertsetzung erfolgt ausschließlich über die entsprechenden Attribute des ***spinbox*** Objekts.

## 15.3 Hinweis zu veralteten Attributen

Die veralteten Attribute .charwidth, .charheight und .focusable werden nicht mehr unterstützt und erzeugen eine ignoring-Warnung beim Setzen und Zurücksetzen (setinherit) des jeweiligen Attributs sowie ein FAIL beim Auslesen dieser Attribute.

## 15.4 Editierbarer Text mit Formatierungen (RTF-Edittext)

Unter MICROSOFT WINDOWS verfügt der Edittext über einen zusätzlichen Modus zur Unterstützung von formatiertem Text im Rich Text Format (RTF). Dieser Modus wird eingeschaltet, indem *.options [opt_rtf] = true* gesetzt wird.

RTF ist ein von Microsoft spezifiziertes Textformat, das Formatierungsanweisungen enthält. RTF-Dateien können zum Beispiel mit dem Windows-Programm WordPad, Microsoft Office Word und dem Writer von OpenOffice bzw. LibreOffice bearbeitet werden.

### 15.4.1 Besonderheiten einiger Attribute

Die Bedeutung der Attribute des RTF-Edittextes entspricht weitgehend ihrer Bedeutung bei einem Edittext ohne Formatierungen. Allerdings sind bei manchen Attributen Besonderheiten zu beachten, die damit zusammenhängen, dass sich Content-String – also der Wert des Attributs *.content* – und angezeigter Text unterscheiden.

Der Content-String enthält Formatierungsanweisungen wie \i, \ul und \par, die nicht als Text angezeigt werden, sondern in den genannten Beispielen dazu führen, dass Text in der Anzeige kursiv ist,

unterstrichen wird oder in einem neuen Absatz steht. Außerdem kann der RTF-Edittext den Content-String umstrukturieren, zum Beispiel indem er Angaben im RTF-Header ergänzt.

Daraus folgt, dass sich für Attribute, bei denen die Textlänge oder die Position im Text eine Rolle spielt, Besonderheiten ergeben:

- *.startsel* und *.endsel* beziehen sich Windows-konform auf die Position des Cursors bzw. der Selektion im **angezeigten** Text. Ein Rückschluss auf die Position im Content-String ist wegen der darin enthaltenen Formatierungsanweisungen nicht möglich. Deshalb kann man *.startsel* und *.endsel* nicht direkt verwenden, um den Inhalt (*.content*) des Edittextes zu bearbeiten.
- *.format*, *.formatfunc* und *.maxchars* sind zwar erlaubt, aber bei mehrzeiligen Texten nicht sinnvoll.

### 15.4.2 Attribut .textwidth

Die maximale Breite des Textes kann beim RTF-Edittext unabhängig von der Breite des Edittextes definiert werden. Das Festlegen der Textbreite ist beispielsweise sinnvoll, um die Ausrichtung von Texten (linksbündig, rechtsbündig oder zentriert) sichtbar zu machen.

Mit dem Attribut *.textwidth* kann die maximale Textbreite gesetzt und abgefragt werden. Wird *.textwidth* auf einen Wert <= *0* gesetzt, ergibt sich die Textbreite automatisch und hängt vom der Sichtbarkeit der horizontalen Scrollbar ab. Die Einzelheiten dazu sind beim Attribut .textwidth in der „Attributreferenz“ beschrieben.

### 15.4.3 Inhalt bearbeiten und formatieren

Der Inhalt eines RTF-Edittextes kann mit der Methode :replacetext() geändert werden. Mit der Methode :gettext() kann Text abgefragt und mit :findtext() kann im Inhalt nach einem Text gesucht werden. **:replacetext()** kann reinen, unformatierten Text („Plain Text“) oder Text im RTF-Format einfügen, **:gettext()** kann den abgefragten Text in beiden Formaten zurückgeben. Um Text zu formatieren oder Formatierungen abzufragen stehen die Methoden :setformat() und :getformat() zur Verfügung.

Diese Methoden sind in der „Methodenreferenz“ beschrieben.

Je nach Schriftart kann es vorkommen, dass Formatierungsanweisungen bzw. mit **:setformat()** gesetzte Formatierungsattribute vom Windows-Objekt, das der IDM für den RTF-Edittext verwendet, nicht beachtet werden. Unter Windows 7 ist dies zum Beispiel bei *text_bold* so, wenn explizit keine Schriftart gesetzt und daher implizit die Schriftart „System“ verwendet wird.

## 15.5 Anmerkungen zum IDM für Windows

- Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.
- Die Codepage "utfwin" eignet sich nicht als Anwendungscodepage, wenn Zeichenketten verarbeitet werden müssen, die <Carriage-Return>-Zeichen enthalten. Der Grund ist, dass

<Linefeed>-Zeichen (Zeilenumbruch) in die Zeichenfolge <Carriage-Return><Linefeed> umgewandelt werden. Um doppelte <Carriage-Return>-Zeichen zu vermeiden, werden diese überlesen. Statt "utfwin" sollten die Codepages "utf16" oder "utf16l" verwendet werden, die diese Spezialbehandlung nicht durchführen.

» Das Ändern der Schriftart (*.font*) an einem Edittext verschiebt den Cursor in den sichtbaren Bereich (ab IDM-Version A.05.02.i).

## 15.6 Anmerkungen zum IDM für Motif

» Der Edittext unter Motif kann im Single-Line Modus auch eine horizontale Scrollbar haben; eine vertikale Scrollbar ist in diesem Modus aber nicht möglich.

» Das Umsetzen von *.content* oder *.navigable* an einem ***Edittext*** kann Einfluss darauf haben, ob in einem anderen Edittext der Cursor (Caret) durchgezogen gezeichnet wird. Zu beachten ist, dass bei einem Edittext mit *.navigable = false* unter Motif keine Eingaben möglich sind.

» Der durchgezogene Cursor erlaubt keinen Rückschluss auf Navigationsreihenfolge oder Fokussierung eines ***Edittexts***.

## 15.7 Anmerkungen zum IDM für Qt

» Es gibt Verhaltensunterschiede zwischen einzeiligem und mehrzeiligem ***Edittext*** hinsichtlich der Darstellung einer Markierung bei Fokusverlust. Im einzeiligen Modus wird die Selektion entfernt und ebenso wie die Cursorposition nicht mehr dargestellt. Bei Wiedererlangen des Fokus stellt der IDM die Selektion wieder her. Im mehrzeiligen Modus bleibt die Selektion bei Fokusverlust erhalten.

» Je nach UI-Stil kann die Darstellung des Fokusrahmens beim einzeiligen Edittext eingeschränkt sein.

## 15.8 Beispiel

### Einzeiliges editierbares Textfeld mit Format

```
dialog Test
{
}
window Wn
{
  .height 169;
  .title "Fenster mit Eingabefeld";

  child statictext
  {
    .sensitive false;
    .xleft 1;
```

```
    .ytop 74;
    .text "Numerisch";
  }

  child edittext Et1
  {
    .active false;
    .xleft 95;
    .width 170;
    .ytop 69;
    .format "NNNNNNNN";
    .content "";
  }

  child statictext
  {
    .sensitive false;
    .xleft 10;
    .ytop 121;
    .text "Alles";
  }

  child edittext Et2
  {
    .active false;
    .xleft 95;
    .width 170;
    .ytop 118;
    .format "XXXXXXXXXXXXXX";
    .content "";
  }
}
```

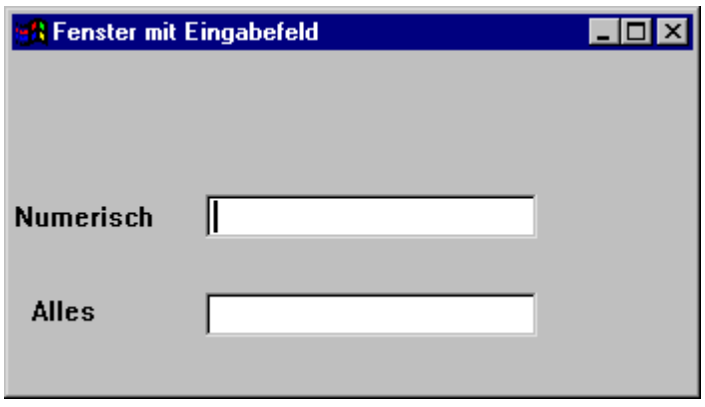



***Abbildung 6:*** *Editierbarer Text*

### Mehrzeiliger editierbarer Text

```
dialog Test
{
}
window Wn
{
  .width 289;
  .height 201;
  .title "Fenster mit mehrzeiligem Edittext";

  child edittext EtMehrzeilig
  {
    .active false;
    .xleft 62;
    .width 167;
    .ytop 51;
    .height 109;
    .content "Das ist ein \nmehrzeiliger\nText";
    .multiline true;
    .hsb_visible true;
    .vsb_visible true;
  }
}
```

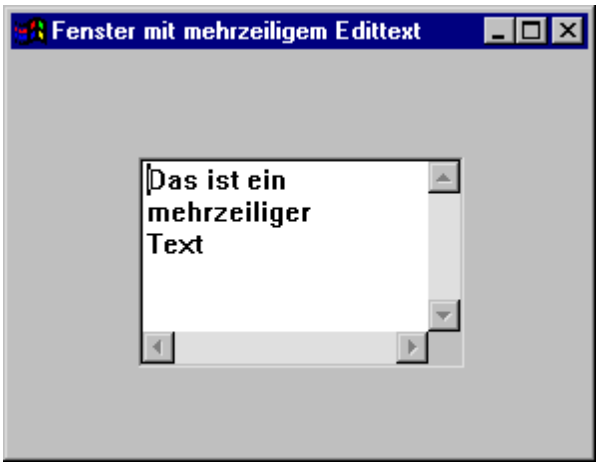



***Abbildung 7:*** *Mehrzeiliger editierbarer Text*

# 16 filereq (Filerequestor)

Das Objekt ***filereq*** (Filerequestor) ist ein modales Dateiauswahlfenster. Es unterscheidet drei Modi:

- Auswahl einer Datei zum Öffnen.
- Angabe einer Datei zum Speichern.
- Auswahl eines Verzeichnisses.

Zusätzlich können auswählbare Dateien bzw. Verzeichnisse durch ein Muster eingeschränkt werden oder die Auswahl auf existierende Dateien und Verzeichnisse beschränkt werden.

### Definition

```
{ export | reexport } { model } filereq { <Bezeichner> }
{
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

### Kinder

keine

### Vater

***dialog***

***module***

### Menü

keins

## 16.1 Attribute

bgc

changedir

class

control

cursor

cut_pending

cut_pending_changed

dialog

directory

extension

external

external[l]

fgc

firstrecord

font

groupbox

help

label

lastrecord

model

module

multisel

mustexist

navigable

notepage

options[enum]

parent

pattern

record[l]

recordcount

scope

self

startsel

statushelp

style

text[enum]

title

userdata

value

value[l]

## 16.2 Spezifische Attribute

Bitte beachten Sie auch die folgenden Kapitel, insbesondere das Kapitel „Beschreibung des filereq-Objekt“.

| Attribut | Beschreibung |
|---|---|
| changedir | Steuert die Übernahme des Verzeichnisses, in dem sich der Benutzer bei erfolgreicher Selektion in einem Dateiauswahlfenster (Filerequestor) befand, in das Attribut directory. |
| directory | Beinhaltet den Pfad des initialen Verzeichnisses, wenn das Objekt mittels der Builtin-Funktion **querybox()** geöffnet wird. |
| extension | Erlaubt die Definition eines Dateisuffixes. |
| multisel | Steuert die Auswahlmöglichkeit des Benutzers, bei Wert *true* kann er mehrere Dateien auswählen. |
| mustexist | Bei Wert *true* können nur existierende Dateien bzw. Verzeichnisse ausgewählt werden. |
| navigable | Steuert die Fokussierbarkeit (Erreichbarkeit des Objekts mittels Tastatur). |
| options[enum] | Optionen des Objekts, näheres siehe „Attributreferenz“.<br>Mögliche Indizes:<br>» *fro_createprompt*<br>» *fro_overwriteprompt*<br>» *fro_relativepath* |
| pattern | Erlaubt, die Auswahlmöglichkeit durch ein Muster einzustellen. |
| startsel | Mit dem Wert *true* wird der Pfad aus Attribut *.value* als initiale Vorgabe übernommen. Standardwert ist *false*. |
| style | Definiert den Modus bzw. Verwendungszweck des Objekts.<br>Mögliche Werte:<br>» *fr_directory*<br>» *fr_load*<br>» *fr_save* |
| text[enum] | Gibt Beschriftung von Dialogelementen eines Filereq-Objektes an (nur Motif). |

| Attribut | Beschreibung |
|---|---|
| title | Angabe des Titels des Filerequestor-Auswahldialogs. |
| value | Enthält nach erfolgreicher Selektion den vollständigen Pfad zur selektierten Datei bzw. zum selektierten Verzeichnis. |
| value[I] | Enthält nach erfolgreicher Mehrfachselektion eine Liste der vollständigen Pfade zu den selektierten Dateien bzw. zu den selektierten Verzeichnissen. |

## 16.3 Beschreibung des ***filereq***-Objekt

Das Dateiauswahlfenster wird in der Regelsprache mit Hilfe der eingebauten Funktion **querybox()** bzw. von der Applikationsseite über die Funktion **DM_QueryBox()** bzw. **DMcob_QueryBox()** geöffnet. Das Attribut *.visible* ist für dieses Objekt nicht verfügbar.

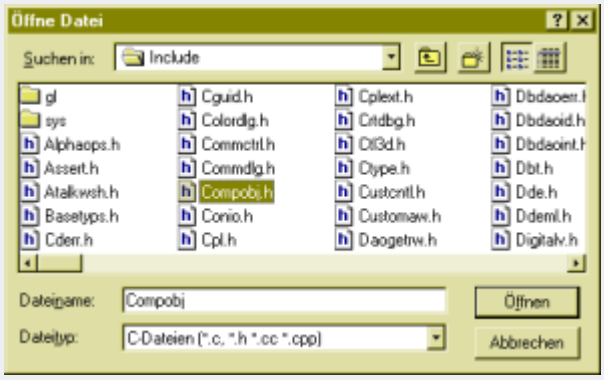
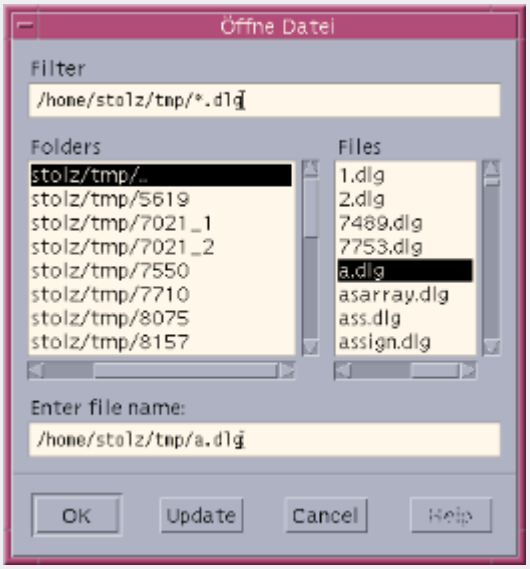
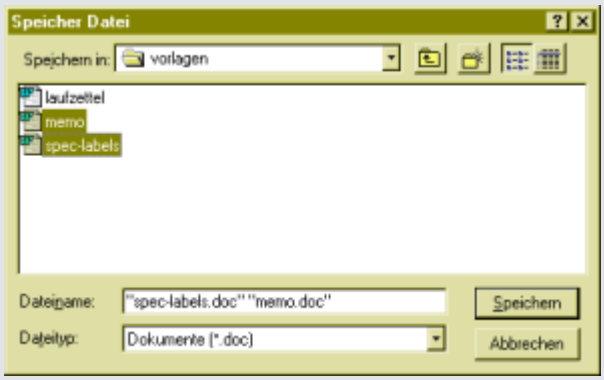
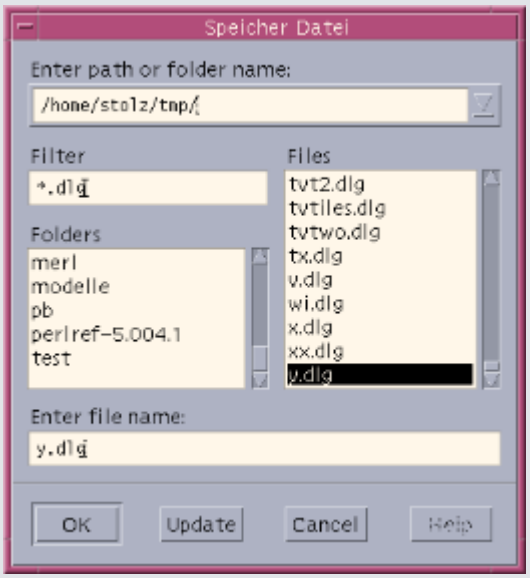
Durch Angabe des Parent-Parameters kann die Anfangsposition des Dateiauswahlfensters beeinflusst werden. Siehe dazu die Tabelle am Ende dieses Kapitels.

Während der Auswahl sind alle weiteren Eingabemöglichkeiten der Anwendung gesperrt. Der Anwender kann im Verzeichnisbaum navigieren und eine Datei mittels Maus oder Tastatur auswählen. Auf die Bestätigung der Auswahl durch den *„OK“*-Button bzw. durch Doppelklick, erhält man als Rückgabewert:

- *button_ok*
  wenn der Anwender eine Datei ausgewählt hat. Der ausgewählte Dateiname steht dann als vollständige Pfadangabe im Attribut value bzw. value[I] bei einer Mehrfachselektion. Wenn das Attribut changedir den Wert true besitzt, erhält das Attribut *.directory* den Pfad des Verzeichnisses, aus dem die Datei / das Verzeichnis ausgewählt wurde.
- *button_cancel*
  wenn der Anwender die Auswahl abgebrochen hat
- *nobutton*
  wenn ein Fehler aufgetreten ist, der das Öffnen des Auswahlfensters verhinderte.

Die Auswahlmöglichkeit kann durch Angabe eines initialen Verzeichnisses (Attribut directory) und durch ein Muster (pattern), das auf die angezeigten, auswählbaren Dateien angewandt wird, konfiguriert werden. Bei Angabe eines Dateisuffix im Attribut extension wird dieser an den ausgewählten Namen, wenn noch kein Suffix vorhanden, angehängt.

Das Auswahlfenster präsentiert sich je nach Modus (Attribut style) und Fenstersystem in folgender Form:

| Microsoft Windows | Motif |
| --- | --- |
|  |  |
| ***Abbildung 8:*** *Datei zum Laden auswählen* | ***Abbildung 9:*** *Datei zum Laden öffnen* |
|  |  |
| ***Abbildung 10:*** *Datei zum Speichern auswählen (mit Mehrfachselektion)* | ***Abbildung 11:*** *Datei zum Speichern auswählen (relative Pfade, Motif 2.1)* |

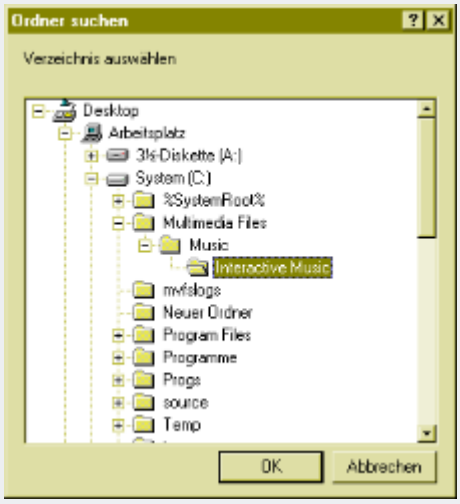
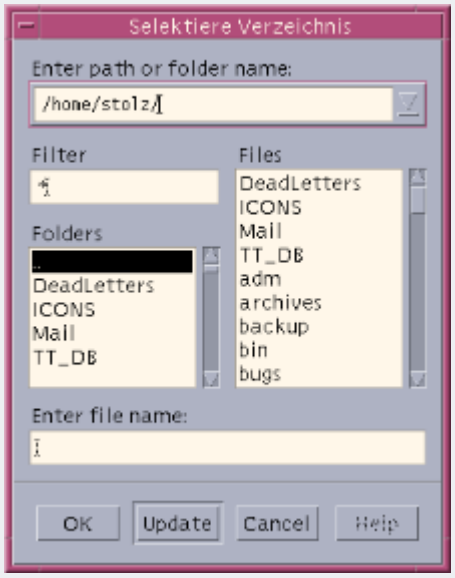
| Microsoft Windows | Motif |
|---|---|
|  |  |



***Abbildung 12:*** *Verzeichnis auswählen*



***Abbildung 13:*** *Verzeichnis selektieren*

Einige Attribute (options[enum], multisel, text[enum]) erlauben systemspezifische Funktionen zu nutzen, bieten Zusatzfunktionen oder konfigurieren die Auswahl, und sind nicht auf allen Plattformen verfügbar oder zeigen eine Wirkung.

Die Anfangsposition des Auswahlfensters ist vom Fenstersystem abhängig. Die folgende Tabelle zeigt die Wirkung des Parent-Parameters beim **querybox**-Aufruf auf die Position.

| Parent-Parameter | Anfangsposition des Auswahlfensters | | |
|---|---|---|---|
| | Motif | Windows | |
| | | Dateiauswahl | Verzeichnisauswahl |
| keine Angabe oder null | linke, obere Ecke des Desktops | linke, obere Ecke des Desktops wenn kein Anwendungsfenster sichtbar oder in der linken, oberen Ecke des letzten Fensters der Anwendung | linke obere Ecke des Desktops |
| Fenster-Objekt | zentriert im Fenster | linke, obere Ecke des Fensters | |

## 16.4 Hinweise

Das ***filereq***-Objekt wird auf die vom System angebotenen Datei-/Verzeichnis-Auswahlmöglichkeiten abgebildet. Insofern ergibt sich ein unterschiedliches Erscheinungsbild und Funktionsumfang auf den

verschiedenen Systemen. Bei Verbesserungen der Auswahlmöglichkeit profitiert man automatisch davon.

Die auf allen Systemen angebotene Möglichkeit, einen Hilfe-Button anzubieten, wird momentan nicht unterstützt.

Das momentane Arbeitsverzeichnis, in dem sich die Anwendung befindet, wird vom ***filereq***-Objekt nicht beeinflusst.

### 16.4.1 Motif

Die Darstellung (Schrift, Farben, Texte) wird normalerweise vom Window-Manager definiert. Nur Texte und Font sind vom Anwendungsprogrammierer überdefinierbar.

Die Selektion einer Datei kann durch den OK-Button oder durch einen Doppelklick geschehen.

Es handelt sich, unabhängig vom Modus des Filerequestor-Objektes, um die gleiche Art von Dateiauswahlfenster. Man muss also den Sinn über den Titel oder die Texte vermitteln.

Die Überprüfung, ob eine Datei oder ein Verzeichnis existiert, sowie das Anhängen eines Dateisuffixes wenn noch keiner existiert, werden vom IDM übernommen.

Zusätzlich ist es unter Motif möglich, die Beschriftung von Dialogelementen eines Filerequestor-Objektes mit Hilfe des Attributs text[enum] vorzunehmen.

Bitte beachten Sie in Bezug auf die Anordnung eines Filerequestors vor oder hinter anderen Fenstern und Dialogfeldern auf dem Bildschirm (Z-Ordnung) den Hinweis in Kapitel „Z-Ordnung von Fenstern und Dialogfeldern“.

### 16.4.2 Microsoft Windows

Bitte beachten Sie, dass die Darstellung (Schrift, Farben, Texte) bis auf den Titel durch die Einstellungen ihres MS-Windows-Systems definiert wird.

Auch hier ist die Selektion durch den OK-Button oder durch einen Doppelklick möglich. MS-Windows bietet eine Mehrfachselektion bei der Auswahl/Eingabe von Dateien an. In Kombination mit der `Shift`- oder `Strg`-Taste erlaubt es eine Bereichsselektion oder das Umschalten des Aktivierungszustandes.

Die Verzeichnisauswahl ist, im Gegensatz zur Motif-Implementierung, beschränkt auf die Auswahl eines vorhandenen Verzeichnisses und bietet nicht die Möglichkeit, mit einem Muster zu arbeiten. Außerdem ist, in den Dialog Manager Versionen vor A.05.01.e, bei Angabe eines initialen Verzeichnisses die Verzeichnisauswahl beschränkt auf die Verzeichnisse die darunter liegen. Ab der Version A.05.01.e entfällt diese Beschränkung. Näheres hierzu siehe in der „Attributreferenz“ zum Attribut directory.

Ein zusätzliches Feature von MS-Windows sind außerdem die Nachfragedialoge, ob eine Datei angelegt bzw. wirklich überschrieben werden soll.

### 16.4.3 Portabilitätshinweise

Um größtmögliche Konsistenz bei Funktion und Aussehen zu erhalten, sollten folgenden Punkte beherzigt werden:

- Mehrfachselektion nicht verwenden.
- Attribut .title auf jeden Fall so setzen, dass die Aktion daraus (Laden/Speichern/Selektieren) klar wird.
- Um Konsistenz in der Sprache zu erhalten, muss die im System eingestellte Sprache mit ihren Texten korrespondieren.
- Verschiedene Muster (Attribut *.pattern*) verwenden. Muster für die Verzeichnisauswahl vermeiden.
- Verzeichnisauswahl ohne initiales Startverzeichnis benutzen.
- UNIX und MS-Windows besitzen unterschiedliche Pfad-Zeichen. Dies ist bei der Verarbeitung von Pfadangaben zu beachten.
- Zur Abfrage des Systems kann das Attribut *.opsys_type* am Setup-Objekt verwendet werden.

## 16.5 Beispiel

Das Öffnen eines Dateiauswahlfensters kann wie folgt aussehen:

```
!! Beispiel fuer Microsoft Windows
filereq Fr
{
  .style fr_load;
  .title "Zeige GIF-Bild an";
  .directory "c:\";  /* Motif: "/" */
  .pattern "GIF-Datei\t*.gif\tAlle\t*.*";  /* Motif: "*.gif"; */
  .extension "gif";
}
rule void OpenFile()
{
  if button_ok = querybox(Fr) then
    ViewGif(Fr.value);
  endif
}
```

# 17 groupbox

Die ***groupbox*** ist ein Hilfsobjekt, um eine logische Struktur innerhalb der Objekte zu definieren. Alle in ihr definierten Objekte liegen auf einer logischen Ebene. Die ***groupbox*** muss auf jeden Fall verwendet werden, wenn innerhalb eines Fensters mehrere Gruppen von Radiobuttons eingesetzt werden sollen. Ansonsten kann die ***groupbox*** zur Gruppierung von beliebigen Objekten herangezogen werden.

### Definition

```
{ export | reexport } { model } groupbox { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Rasterattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Scrollbarattribute>
  <Objektspezifische Attribute>
}
```

### Anmerkung

Wenn eine Groupbox sehr viele direkte Kindobjekte besitzt kann das Scrollen in dieser Groupbox sehr lange dauern. Dieses Problem kann umgangen werden, wenn zusätzliches ein groupbox Objekt einfügt wird:

```
groupbox {
  .vsb_visible true;
  .vheight 10000;

  !! Umgehung des Scrollproblems
  groupbox {
    .xauto 0;
    .yauto 0;
    .borderwidth 0;

    child {}
    child {}
    ...
  }
}
```

### Ereignisse

*extevent*

*help*

*hscroll*

*key*

*paste*

*scroll*

*select*

*vscroll*

**Kinder**

***canvas***

***checkbox***

document

***edittext***

***groupbox***

***image***

***layoutbox***

***listbox***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***tablefield***

transformer

***treeview***

**Vater**

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 17.1 Attribute

acc_label

acc_text

accelerator

bgc

bordercolor

borderraster

borderstyle

borderwidth

child[I]

childcount

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstchild

firstrecord

focus

focus_on_click

font

function

groupbox

height

help

hsb_arrows

hsb_linemotion

hsb_optional

hsb_pagemotion

hsb_visible

index

label

lastchild

lastrecord

layoutbox

mapped

member[l]

membercount

menu

model

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_xraster

real_y

real_yraster

record[l]

recordcount

reffont

scope

sensitive

sizeraster

statushelp

tile

tilestyle

toolbar

toolhelp

userdata

vheight

visible

vsb_arrows

vsb_linemotion

vsb_optional

vsb_pagemotion

vsb_visible

vwidth

width

window

xauto

xleft

xorigin

xraster

xright

yauto

ybottom

yorigin

yraster

ytop

## 17.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| bordercolor | Farbe des Rahmens des Objekts (nur Motif). |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“.<br>Nur aktiv, wenn *.borderwidth > 0*. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a). |
| borderwidth | Rahmenbreite des Objekts, bei Wert *0* wird kein Rahmen gezeichnet. |
| focus_on_click | Legt fest, ob ein Mausklick in den Clientbereich das Objekt aktiviert, d.h.den Fokus auf das Objekt setzt. |
| hsb_arrows (nur Motif) | Definiert, ob Pfeile an den Enden der horizontalen Scrollbar vorhanden sind. |
| hsb_linemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von xorigin beim zeilenweisen horizontalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_optional | Definiert, ob die horizontale Scrollbar nur angezeigt wird, wenn sie benötigt wird.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_pagemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von xorigin beim seitenweisen horizontalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_visible | Definiert die Sichtbarkeit der horizontalen Scrollbar.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_scroll_on_focus* (nur IDM für Motif) |
| tile | Definiert das Hintergrundbild des Objekts. |
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |

| Attribut | Beschreibung |
|---|---|
| vheight | Interne („virtuelle“) Höhe des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| vsb_arrows<br>(nur Motif) | Definiert, ob Pfeile an den Enden der vertikalen Scrollbar vorhanden sind. |
| vsb_linemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von yorigin beim zeilenweisen vertikalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_optional | Definiert, ob die vertikale Scrollbar nur angezeigt wird, wenn sie benötigt wird.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_pagemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von yorigin beim seitenweisen vertikalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_visible | Definiert die Sichtbarkeit der vertikalen Scrollbar.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vwidth | Interne („virtuelle“) Breite des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| xorigin | Gibt an, um wie viele Pixel der eigentliche Inhalt horizontal verschoben ist.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| yorigin | Gibt an, um wie viele Pixel der eigentliche Inhalt vertikal verschoben ist.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |

## 17.3 Anmerkungen zum IDM für Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

## 17.4 Beispiel

Das folgende Beispiel beschreibt zwei Gruppen von Radiobuttons innerhalb eines Fensters.

```
window Radiofenster
{
  .title "Radiofenster";

  child groupbox Gruppe_1
  {
    child radiobutton First
```

```
    {
    }
    child radiobutton Second
    {
    }
    child radiobutton Third
    {
    }
  }
  child groupbox Gruppe_2
  {
    child radiobutton Four
    {
    }
    child radiobutton Five
    {
    }
    child radiobutton Six
    {
    }
  }
}
```

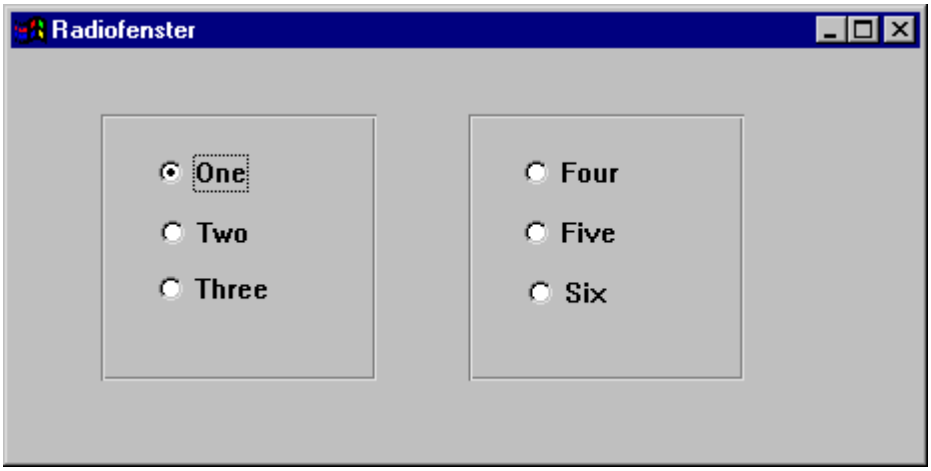



***Abbildung 14:*** *Groupbox*

# 18 image (Bild)

Dieses Objekt dient zur Darstellung von Bildern oder Grafiken. Diese Bilder können dabei entweder direkt in der Datei definiert oder von einer externen Datei eingelesen werden.

### Definition

```
{ export | reexport } { model } image { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*cut*

*dbselect*

*extevent*

*focus*

*help*

*key*

*paste*

*select*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***statusbar***

***toolbar***

***window***

**Menü**

**Popup Menü**

## 18.1 Attribute

acc_label

acc_text

accelerator

.active

.alignment

bgc

borderstyle

borderwidth

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[l]

external

external[l]

fgc

firstrecord

focus

focus_on_click

font

function

groupbox

height

help

imagebgc

imagefgc

index

label

lastrecord

layoutbox

mapped

member[I]

membercount

menu

model

module

mouseover

notepage

.options[enum]

parent

picture

.picture[enum]

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

sizeraster

spacing

statushelp

.style

text

.tilestyle

toolhelp

toolbar

userdata

visible

width

window

xauto

xleft

xmargin

xright

yauto

ybottom

ymargin

ytop

## 18.2 SpezifischeAttribute

| Attribut | Beschreibung |
|---|---|
| .active | Gibt die Zustände aktiv/inaktiv an.<br>Siehe auch Kapitel „Bilddarstellung und Mouseover Ereignisse“. |
| .alignment | Horizontale Anbindung des Textes.<br>Default: *0*.<br>(Siehe auch spacing und insbesondere die Beschreibung in der „Attributreferenz“ zu tilestyle). |
| .borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a).<br>Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |
| .focus_on_click | Legt fest, ob das Objekt durch Mausklick den Fokus bekommt. |

| Attribut | Beschreibung |
|---|---|
| .height | Höhe des Objekts.<br>Beim Wert *0* berechnet der DM automatisch die für das Objekt richtige Objektgröße. |
| .imagebgc | Hintergrundfarbe des Bildes.<br>Wird nur ausgewertet, wenn das angezeigte Bild nicht aus einer externen Datei geladen wird. |
| .imagefgc | Vordergrundfarbe des Bildes.<br>Wird nur ausgewertet, wenn das angezeigte Bild nicht aus einer externen Datei geladen wird. |
| .index | Index des Objekts im Kindvektor des Vaterobjekts. |
| .mouseover | Legt fest, ob das Objekt auf Mouseover-Ereignisse reagiert.<br>Siehe auch Kapitel „Bilddarstellung und Mouseover Ereignisse“. |
| .options[enum] | Optionen des Objekts.<br>Index: *opt_center_toolhelp* (nur IDM FÜR WINDOWS). |
| .picture | Gibt das Muster bzw. das Bild des Objekts an.<br>Siehe auch Kapitel „Bilddarstellung und Mouseover Ereignisse“. |
| .picture[enum] | Gibt das Muster bzw. das Bild des Objekts für die verschiedenen Zustände an.<br>Siehe auch Kapitel „Bilddarstellung und Mouseover Ereignisse“. |
| .spacing | Zwischenraum zwischen Bild und Text.<br>(Siehe auch alignment und insbesondere die Beschreibung in der „Attributreferenz“ zu tilestyle). |
| .style | Legt fest, ob das Objekt die Zustände aktiv/inaktiv durch unterschiedliche Bilder darstellen kann oder nicht.<br>Siehe auch Kapitel „Bilddarstellung und Mouseover Ereignisse“.<br>Zusätzlich kann unter Windows noch festgelegt werden, ob sich das Bild als Menü verhalten soll (siehe auch Kapitel „Verwendung als menuitem-Ersatz“). |
| .text | Gibt den zum Objekt gehörenden Text an.<br>Dieser wird unterhalb des Bildes dargestellt. |
| .tilestyle | Position des Bildes im Objekt.<br>Bitte beachten Sie auch die Detailbeschreibung „Attributreferenz“ zu tilestyle und die Attribute alignment und spacing. |

| Attribut | Beschreibung |
|---|---|
| .width | Breite des Objekts.<br>Beim Wert *0* berechnet der DM automatisch die für das Objekt richtige Objektgröße. |
| .xmargin | Vertikaler Abstand zwischen Inhalt und Rand.<br>Siehe auch Detailbeschreibung „Attributreferenz“ zu xmargin. |
| .ymargin | Horizontaler Abstand zwischen Inhalt und Rand.<br>Siehe auch Detailbeschreibung in der „Attributreferenz“ zu ymargin. |

### 18.2.1 Zusammenspiel der Layout-Attribute

Die Attribute .xmargin und .ymargin bestimmen die Abstände zwischen Image-Rand und Darstellungsbereich (Grenze im folgenden Bild gestrichelt dargestellt). Das Attribut .spacing dagegen definiert den Abstand zwischen Tile und Text innerhalb des Darstellungsbereich des Image.

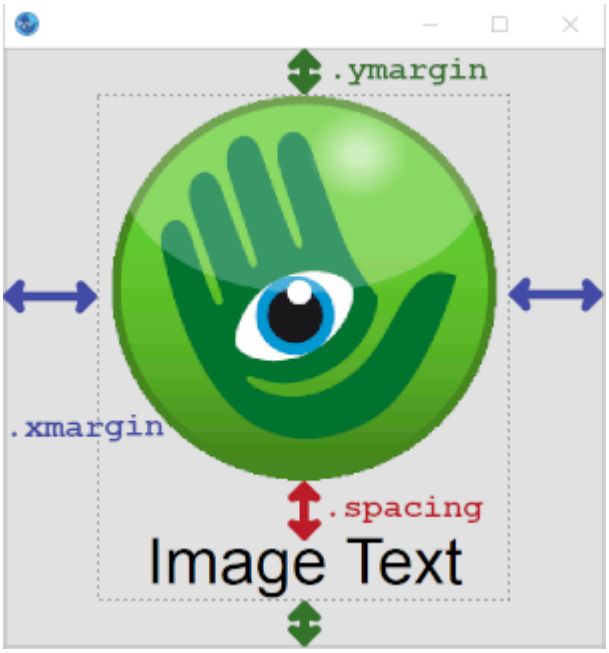



***Abbildung 15:*** *Bild mit Text*

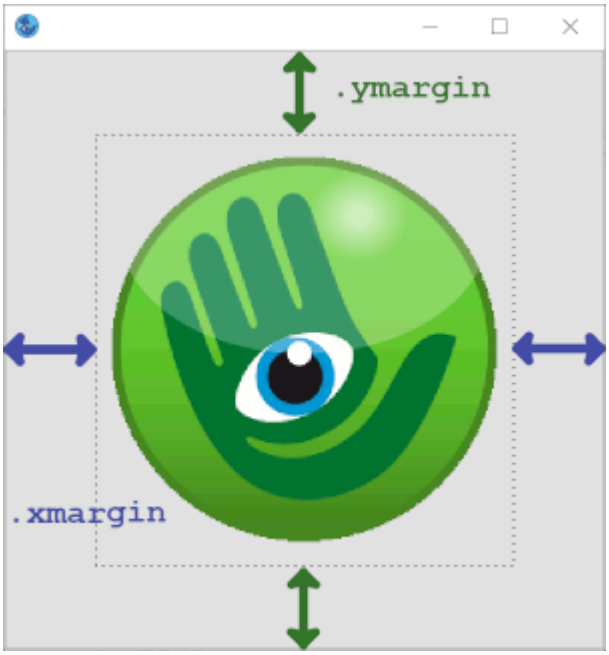



***Abbildung 16:*** *Bild ohne Text*

Im Folgenden eine beispielhafte Übersicht über die Auswirkung einiger Attribute (das Tile besitzt dabei jeweils den .scalestyle *propscale*):

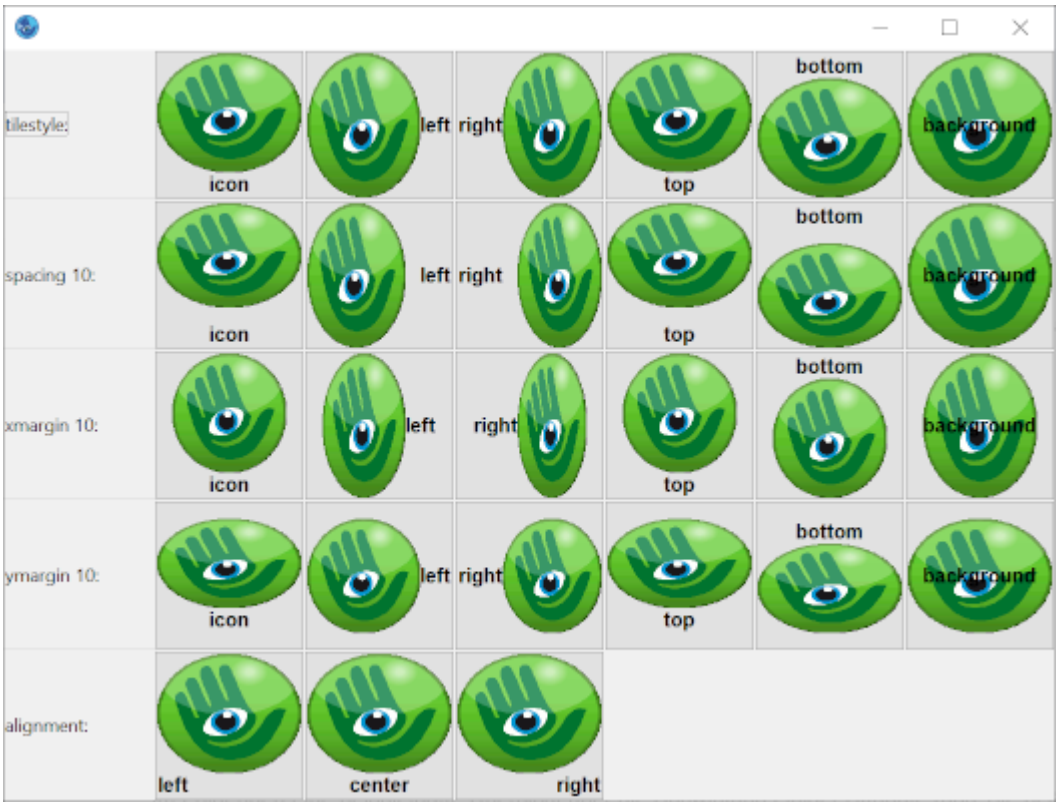



***Abbildung 17:*** *Auswirkung der Attribute*

Siehe auchKapitel„HighDPI UnterstützungSupport“ im Handbuch „Programmiertechniken“

### 18.2.2 Bilddarstellung und Mouseover Ereignisse

Das anzuzeigende Bild (z.B. eine tile-Ressource) wird normalerweise im Attribut picture definiert.

Soll das Bild-Objekt bei verschiedenen Zuständen (aktiv, inaktiv…) unterschiedliche Bilder darstellen (siehe Attribut .style und .active) oder auf das Überfahren mit dem Mauszeiger reagieren, so sind in den Attributen .picture[enum] bzw. mouseover die notwendigen Einstellungen zu treffen (siehe „Attributreferenz“).

### 18.2.3 Verwendung als ***menuitem***-Ersatz

Das ***image*** Objekt kann unter Microsoft Windows mit einem Menübox-Stil genutzt werden. Dazu muss das Attribut .style den Wert *menubox* erhalten. Es können dann Microsoft Office ähnliche Menüs entworfen werden. Sinnvollerweise sollten hier alle ***image***-Objekte, welche *.style = menubox* besitzen, als Kinder eines ***toolbar***-Objekts definiert werden. Außerdem sollte ein solches Fenster keine anderen Menüs mehr besitzen.

**Beispiel**

```
color MENU_BGC rgb(191,219,255);

tile TI_DEFAULT "xdefault.bmp" scale;
tile TI_OVER "xover.bmp" scale;
tile TI_ACTIVE "xactive.bmp" scale;

font MENU_FONT "ANSI_VAR_FONT";

model image Menu
{
.style menubox;
.borderwidth 0;
.mouseover true;
.focus_on_click false;
.font MENU_FONT;

   .picture[tile_default] TI_DEFAULT;
  .picture[tile_mouse_over] TI_OVER;
.picture[tile_active] TI_ACTIVE;
   .picture[tile_active_mouse_over] TI_ACTIVE;
   .tilestyle tilestyle_background;
}

model toolbar Menubar
{
   .docking dock_up;
.autosize true;
.sizeable[docking] false;
```

```
    .tile TI_DEFAULT;
    .tilestyle tilestyle_stretched;
.bgc MENU_BGC;
}
```

## 18.3 Beispiel

```
dialog Test
{
}
tile TestTile "IMAGE:Setup.bmp";

window WnTest
{
  .width 277;
  .height 257;
  .title "Testfenster";

  child image Im1
  {
    .xleft 34;
    .ytop 47;
    .picture TestTile;
  }
}
```


***Abbildung 18:*** *Bild*

# 19 import

Die Verwendung eines Moduls in einem anderen Modul oder Dialog wird mit dem Schlüsselwort **import** angekündigt.

Auf das Schlüsselwort folgt der logische Name des Moduls, unter dem es bekannt sein soll. Der logische Name ist im Allgemeinen nicht der Name des Moduls selbst. Unter Umständen kann sogar ein und dasselbe Modul unter verschiedenen logischen Namen in einem Modul verwendet werden, obwohl dies allgemein besser und eleganter durch Vorlagen beschrieben werden kann. Auf den logischen Namen des importierten Moduls folgt dann ein String mit dem Namen der Interface-Datei.

### Definition

```
{ export | reexport } import { <Bezeichner> } "<Dateiname>"
{
  Objektspezifische Attribute>
}
```

### Siehe auch

Weitere Informationen finden Sie beim Objekt ***module*** und im Kapitel „Modularisierung“ des Handbuchs „Programmiertechniken“.

### Ereignisse

keine

### Kinder

document

***record***

transformer

### Vater

***dialog***

***module***

### Menü

keins

## 19.1 Attribute

application

document[l]

external

external[I]

firstrecord

index

lastrecord

load

record[I]

recordcount

static

## 19.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| application | Definiert die Applikation, an welche die Funktionen gebunden werden soll. |
| load | Ladezustand des Moduls.<br>Siehe auch Kapitel „Beeinflussung des Ladevorgangs“ |
| static | Gibt an, ob ein Modul entladen werden darf. |

### 19.2.1 Beeinflussung des Ladevorgangs

Ein Modul kann in ein anderes Modul über mehrere Arten geladen werden. Unter Laden versteht man hier, dass nicht nur die Namen der Objekte, sondern auch deren Definition dem Modul bekannt sind und damit darstellbar und veränderbar sind. Dabei werden intern drei verschiedene Arten des Ladens unterschieden, die zum Teil bereits durch die im Modul definierten bzw. exportierten Objekte bestimmt werden.

- **load on use**
  Das Modul wird automatisch sofort geladen. Der Programmierer hat keinen Einfluss darauf, wann das Modul geladen wird, da Objekte aus dem Modul sofort (Defaults) und andere beim Einlesen des übergeordneten Moduls gebraucht werden. Dies ist der Fall, wenn das Modul exportierte benötigte Vorlagen oder Ressourcen enthält.
- **implicit load**
  Das Modul wird erst geladen, wenn ein exportiertes Objekt aus dem Modul wirklich benötigt wird. Dies ist zum Beispiel dann der Fall, wenn auf das Objekt in einer Regel per Zuweisung oder Abfrage zugegriffen wird.
- **explicit load**

In diesem Fall entscheidet der Dialogdesigner selbst, dass er jetzt das Modul geladen haben möchte. Er setzt das Attribut load am logischen Modulnamen (Importname) auf *true*. Daraufhin wird das Modul vom Dialog Manager geladen und verfügbar gemacht.

### Beispiel

```
module Modul
import Colors "color.if";

window W
{
  .bgc Green;           // load on use
}
on W focus
{
  this.bgc := Red;      // implicit load
}
on W select
{
  Colors.load := true;  // explicit load
}
```

**Load on use** hat Vorrang vor den beiden anderen Arten des Ladens. Ebenso hat **implicit load** Vorrang vor **explicit load**. D.h. im obigen Beispiel wird ein **load on use** durchgeführt, die beiden anderen haben keine Auswirkungen mehr, da das Modul sich bereits im Speicher befindet und die Regeln direkt ausgeführt werden können.

Ein **explicit load** kann bereits als Attribut beim Import angegeben werden.

```
module Modul
import Colors "color.if"
{
  .load true;  // explicit load
}
```

## 19.3 Beispiel

```
!! Interfacefile: module.if
module Module
export import StandardColor "color.if";
...
```

```
import Colors "PRIVLIB:mycolor.if";
  // Die Datei mycolor.if wird nur im Pfad
  // PRIVLIB gesucht
```

# 20 layoutbox

Die ***layoutbox*** ist ein Gruppierungsobjekt, in das die anzuordnenden Objekte oder Bausteine einfach hineingeworfen werden können. Diese werden automatisch, überschneidungsfrei angeordnet.

### Definition

```
{export | reexport} { model } layoutbox { <Bezeichner> }
{
  <Standardattribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*dbselect*

*extevent*

*help*

*hscroll*

*key*

*paste*

*scroll*

*select*

*vscroll*

### Kinder

***canvas***

***checkbox***

***control***

document

***edittext***

***groupbox***

***image***

***layoutbox***

***listbox***

***menubox***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***tablefield***

transformer

***treeview***

**Vater**

***control***

***dialog***

***groupbox***

***layoutbox***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

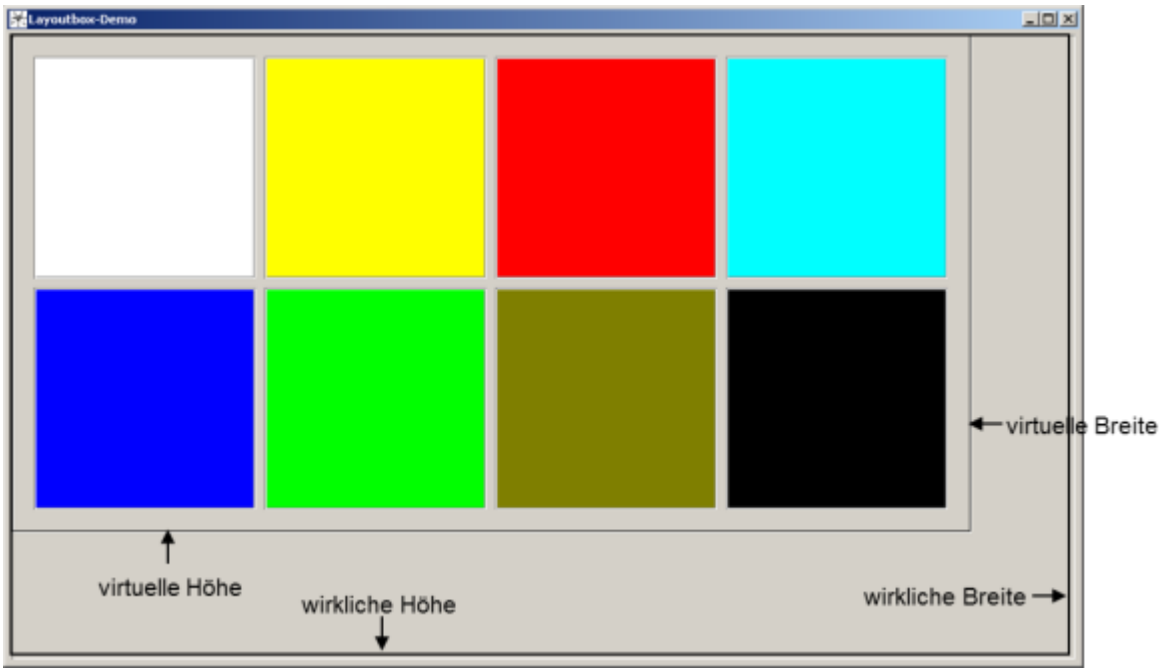
## 20.1 Verwendung der layoutbox

Sehr oft hat man das Problem, dass Objekte oder Bausteine von Objekten angeordnet werden müssen. Dies ist von Hand zwar möglich, jedoch umständlich und anfällig falls sich irgendwelche Objekte in ihrer Größe oder Position ändern.

Hier schafft die Layoutbox Abhilfe. Sie ist ein Gruppierungsobjekt, in das die anzuordnenden Objekte oder Bausteine einfach hineingeworfen werden können. Diese werden automatisch, überschneidungsfrei angeordnet.

Die Layoutbox versucht, die Objekte nacheinander anzuordnen. Passt ein Objekt nicht mehr in eine Zeile bzw. Spalte, so wird es in die nächste platziert.

Außerdem wird dafür gesorgt, dass alle Objekte erreichbar sind, d.h. wenn nicht mehr alle Objekt sichtbar dargestellt werden können (z.B. weil das Vaterfenster zu klein ist), werden Scrollbars angezeigt. Die virtuellen Größen der Layoutbox werden (solange *.wrap* = *true* ist) immer gesetzt. Sie können nicht beeinflusst werden, jedoch abgefragt werden, um die Größe des Arbeitsbereichs zu ermitteln. Die virtuellen Größen können kleiner oder größer sein als die wirklichen Größen.




Die Spaltenbreite bzw. Zeilenhöhe wird immer durch das breiteste bzw. höchste Objekt in der jeweiligen Spalte bzw. Zeile bestimmt. Alle anderen Objekte richten sich dann entsprechend aus.

Bei zeilenweiser Ausrichtung bezieht sich *.yauto* auf die Zeilenhöhe. *.xauto* hat keine Bedeutung, da die Objekte grundsätzlich im gleichen Abstand nebeneinander angeordnet werden. Analog bezieht sich *.xauto* bei spaltenweiser Ausrichtung auf die Spaltenbreite, während *.yauto* in diesem Fall keine Bedeutung hat.

## 20.2 Attribute

acc_label

acc_text

accelerator

active

activeobject

bgc

bordercolor

borderraster

borderstyle

borderwidth

child[I]

childcount

class

control

cursor

cut_pending

cut_pending_changed

dialog

direction

external

external[I]

fgc

firstchild

firstrecord

focus

focus_on_click

font

function

groupbox

height

help

label

lastchild

lastrecord

layoutbox

mapped

mincolwidth

minrowheight

model

module

navigable

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

record[I]

recordcount

reffont

scope

self

sensitive

sizeraster

source

statushelp

target

tile

tilestyle

toolhelp

userdata

vheight

visible

vwidth

width

window

wrap

xauto

xleft

xmargin

xorigin

xraster

xright

xspacing

yauto

ybottom

ymargin

yorigin

yraster

yspacing

ytop

## 20.3 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. Nur aktiv, wenn *.borderwidth > 0*. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a). |
| direction | Legt die Ausrichtungsart der Kindobjekte in der Layoutbox fest. |
| mincolwidth | Minimale Spaltenbreite der Layoutbox. |
| minrowheight | Minimale Zeilenhöhe der Layoutbox. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_center_toolhelp* (nur IDM für Windows).<br>» *opt_scroll_on_focus* (nur IDM für Motif). |
| tile | Definiert das Hintergrundbild des Objekts. |
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |
| wrap | Steuert den automatischen Umbruch in der Layoutbox. |
| xmargin | Gibt den linken und rechten Abstand zwischen der Layoutbox und ihren Kindern an. |

| Attribut | Beschreibung |
|---|---|
| xspacing | Horizontaler Abstand zwischen den Kindern der Layoutbox. |
| ymargin | Gibt den oberen und unteren Abstand zwischen der Layoutbox und ihren Kindern an. |
| yspacing | Vertikaler Abstand zwischen den Kindern der Layoutbox. |

Bei Attributänderungen der ***Layoutbox***, die Einfluss auf das Layout haben (z. B. *.direction*, *.xmargin*, *.ymargin*), werden die Kindobjekte unabhängig von *.wrap* neu angeordnet. Außerdem werden die Kinder neu angeordnet, wenn die Sichbarkeit oder Größe von Kindern geändert wird. Dies stellt sicher, dass die Kindobjekte sich nicht überlappen und dass sich keine Inkonsistenzen ergeben, wenn die Layoutbox unsichtbar und wieder sichtbar geschaltet wird.

## 20.4 Hinweise

Für die Kinder der ***layoutbox*** ändert sich die Bedeutung einiger Attribute:

| Attribut | Kurzbeschreibung |
|---|---|
| .xauto | Bei spaltenweiser Anordnung: das Objekt wird in der jeweiligen Spalte angeordnet, deren Breite sich nach dem breitesten Objekt richtet. Für –1 vom linken Rand der Spalte; für 1 vom rechten Rand der Spalte; für 0 wird es auf die Spaltenbreite aufgezogen. Falls sich nur Objekte mit *.xauto = 0* in einer Spalte befinden, bekommen sie die Breite *mincolwidth* zugewiesen.<br>Bei zeilenweiser Anordnung hat dieses Attribut keine Bedeutung. |
| .yauto | Bei zeilenweiser Anordnung: das Objekt wird in der jeweiligen Zeile angeordnet, deren Höhe sich nach dem größten Objekt richtet. Für –1 vom unteren Rand der Zeile, für 1 vom oberen Rand der Zeile; für 0 wird es auf die Zeilenhöhe aufgezogen. Falls sich nur Objekte mit *.yauto = 0* in einer Zeile befinden, bekommen sie die Höhe *minrowheight* zugewiesen.<br>Bei spaltenweiser Anordnung hat dieses Attribut keine Bedeutung. |

### 20.4.1 Posraster der Kindobjekte

Da es der Zweck der Layoutbox ist, die Kindobjekte automatisch anzuordnen, wird die Art und Weise der Anordnung von der Layoutbox und nicht von den Kindobjekten bestimmt. Aus diesem Grund wird an den Kindobjekten das Attribut *.posraster* auf *false* gesetzt. Ein gerastertes Layout innerhalb der Layoutbox wird nicht unterstützt und ist zu vermeiden.

### 20.4.2 Virtuelle Größe

Die Layoutbox berechnet ihre virtuelle Breite und Höhe selbst. Sie können zwar vom Benutzer ausgelesen und gesetzt werden, Setzungen des Benutzers werden jedoch ignoriert. Die virtuelle Größe bezieht sich immer auf den **Arbeitsbereich**, d.h. die Breite und Höhe des Rechtecks, das die Kinder der Layoutbox mit ihren Rändern (Margin) umfasst.

Bei Benutzung eines Hintergrundbildes ist Folgendes zu beachten:

Bei den Werten *tilestyle_centered* und *tilestyle_stretched* für das Attribut *.tilestyle* wird die virtuelle Größe als Berechnungsgrundlage herangezogen, das Hintergrundbild also gemäß der virtuellen Größe zentriert oder vergrößert. Ist dies nicht gewünscht, so kann der Wert *tilestyle_parent_tile* genutzt werden und das anzuzeigende Hintergrundbild am Vaterobjekt gesetzt werden (eventuell muss dafür eine Groupbox als „Zwischenobjekt“ eingefügt werden).

### 20.4.3 Scrollbarattribute

Die Layoutbox verfügt zwar über Scrollbarattribute, die Verwaltung und das Anzeigen der Scrollbars sollten aber der Layoutbox überlassen werden. Deshalb empfehlen wir, die Scrollbarattribute nicht zu verwenden:

- Setzen Sie keine Werte für die Scrollbarattribute, auch wenn dies scheinbar das gewünschte Resultat liefert.
- Verlassen Sie sich nicht auf Werte, die beim Abfragen der Scrollbarattribute zurückgeliefert werden.

## 20.5 Radmausnutzung unter Microsoft Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

## 20.6 Beispiel

Eine einfache Verwendung der ***layoutbox*** kann wie folgt aussehen:

```
dialog D

color ColBlue  "blue";
color ColWhite "white";

window Wi {
  .title  "IDM Example Layoutbox";
  .width  320;
  .height 240;

  layoutbox La {
```

```
    .bgc      ColWhite;
    .xauto    0;
    .xleft    10;
    .xmargin  20;
    .xright   10;
    .xspacing 10;
    .yauto    0;
    .ybottom  10;
    .ymargin  20;
    .yspacing 10;
    .ytop     10;
    .wrap     true;

    pushbutton Pb {
      .text "Do nothing";
    }

    statictext St {
      .text  "Resize window\nto see the\neffect of .wrap";
      .yauto -1;
    }

   groupbox Gb {
      .bgc    ColBlue;
      .height 100;
      .width  100;
    }
  }

  on close { exit(); }
}
```

Das Beispiel erzeugt folgendes Fenster:

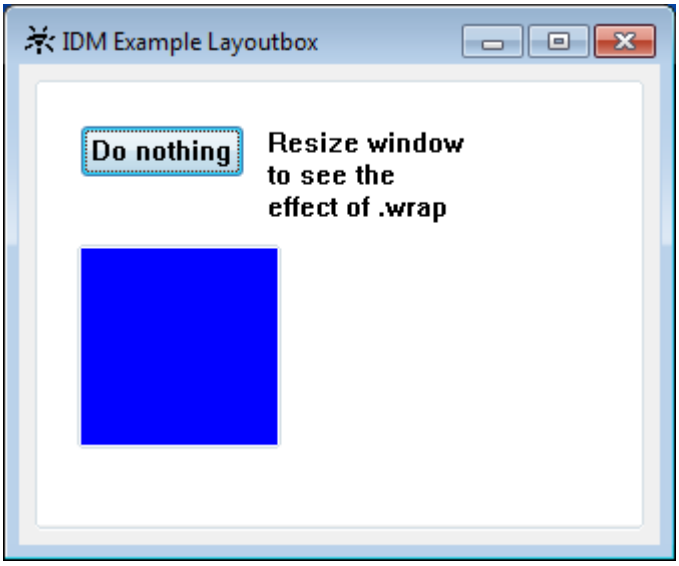



***Abbildung 19:*** *layoutbox example*

# 21 listbox

Die ***listbox*** ist ein Auswahlmechanismus, der es ermöglicht, aus einer großen Anzahl von Textelementen ein oder mehrere mit Hilfe eines Mausklicks auszuwählen.

### Definition

```
{ export | reexport } { model } listbox { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*activate*

*cut*

*dbselect*

*deactivate*

*extevent*

*focus*

*help*

*hscroll*

*key*

*paste*

*scroll*

*select*

*vscroll*

### Kinder

document

***record***

transformer

**Vater**

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

**Methoden**

:delete()

:exchange()

:find()

:insert()

## 21.1 Attribute

acc_label

acc_text

accelerator

activeitem

active[l]

bgc

bordercolor

borderraster

borderwidth

class

content[l]

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstchar

firstrecord

focus

font

function

groupbox

height

help

index

itemcount

label

lastrecord

layoutbox

mapped

member[I]

membercount

menu

model

nextactive[I]

notepage

options[enum]

parent

picheight

picture[I]

picture_hilite[I]

picwidth

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

selstyle

scope

sensitive

sizeraster

statushelp

toolbar

toolhelp

topitem

userdata

userdata[I]

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 21.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| activeitem | Aktives Element bei selstyle = single. |
| active[I] | Aktive Elemente der Listbox bei selstyle <> single. |

| Attribut | Beschreibung |
|---|---|
| content[l] | Wert des l-ten Elements der Listbox, die Werte werden **nicht** vom Modell auf die Instanzen vererbt! |
| firstchar | Nummer des ersten angezeigten Zeichens (horizontale Scroll-Position). |
| itemcount | Gesamtzahl der Einträge in der Listbox. |
| nextactive[l] | Von l ausgehend das nächste aktive Element der Listbox bei selstyle <> single. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_quick_navigate* (nur IDM für Motif)<br>» *opt_scroll_pixels* (nur IDM für Qt, ab IDM-Version A.06.01.c) |
| picheight | Angabe der Höhe eines neben den Einträgen dargestellten Bildes. |
| picture[l] | Zum jeweiligen Eintrag gehörendes Bild (IDM für Windows und IDM für Qt). |
| picture_hilite[l] | Bild, welches bei aktivem Eintrag l dargestellt wird (IDM für Windows und IDM für Qt). |
| picwidth | Angabe der Breite eines neben den Einträgen dargestellten Bildes. |
| selstyle | Steuerung der Selektionsart innerhalb der Listbox.<br>Werte:<br>» *extended*<br>» *multiple*<br>» *single* |
| sensitive | Selektierbarkeit der gesamten Listbox. |
| topitem | Erstes angezeigtes Element der Listbox bei vertikalem Scrollen. |
| userdata[l] | Userdata für beliebige Daten zu jedem Eintrag des Objekts. |

## 21.3 Hinweis zu veralteten Attributen

Die veralteten Attribute .focusitem und .focusable werden nicht mehr unterstützt und erzeugen eine ignoring-Warnung beim Setzen und Zurücksetzen (setinherit) des jeweiligen Attributs sowie ein FAIL beim Auslesen dieser Attribute.

## 21.4 Anmerkungen

### 21.4.1 Microsoft Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

Das listbox-Objekt unter Microsoft Windows unterstützt keine Texte, deren Darstellung mehr als 65.535 Pixel benötigt. Die Texte werden zwar dargestellt, aber das horizontale Scrollen und das Setzen von .firstchar funktionieren nur bis zu einem bestimmten Wert (65.535 geteilt durch mittlere Buchstabenbreite).

### 21.4.2 Motif

Die maximale Anzahl der Einträge und deren Länge sind durch die darstellbaren Pixel des Motif-Widgets beschränkt. Für das listbox-Objekt liegt die Grenze bei ca. 32.000 Pixel (16bit/2). Die Anzahl der darstellbaren Einträge und deren maximale Länge errechnen sich aus Zeilenhöhe (Schriftgröße plus Abstände) und Zeichenbreite.

**Beispiel**

Bei einer Zeilenhöhe von 10px und einer durchschnittlichen Zeichenbreite von 5px ergeben sich:

- Max. Anzahl der Einträge = 32.000/10 = 3.200
- Max. Länge der Einträge = 32.000/5 = 6.400 Zeichen

## 21.5 Beispiel

```
dialog Test
{
}
window Wn
{
  .active false;
  .title "Listboxfenster";

  child listbox Lb1
  {
    .xleft 54;
    .width 198;
    .ytop 57;
    .height 119;
    .content[1] "Erster Eintrag";
    .content[2] "Zweiter Eintrag";
    .content[3] "Dritter Eintrag";
    .content[4] "Vierter Eintrag";
```

```
    .content[5] "";
    .firstchar 1;
  }
}
```


***Abbildung 20:*** *Listbox*

# 22 listview

Das ***listview*** (Klassenname ***dmw_listview***) ist ein erweitertes Listenobjekt, das verschiedene Darstellungsarten unterstützt. Das Objekt ist prinzipiell durch den „Windows Explorer“ bekannt, wo es zur Anzeige der Verzeichnisse des Dateisystems verwendet wird.

Die folgenden Abbildungen zeigen die unterschiedlichen Darstellungsarten, des ***listview***-Objekts:

1. Die Symbolansicht stellt große Symbole zusammen mit einer Beschriftung dar. Als Beschriftung dient die erste Inhaltsspalte.


***Abbildung 21:*** *Symbolansicht mit großen Symbolen des listviews*

2. Die kleine Symbolansicht ist der Symbolansicht sehr ähnlich, einziger Unterschied ist, dass kleine Symbole verwendet werden.

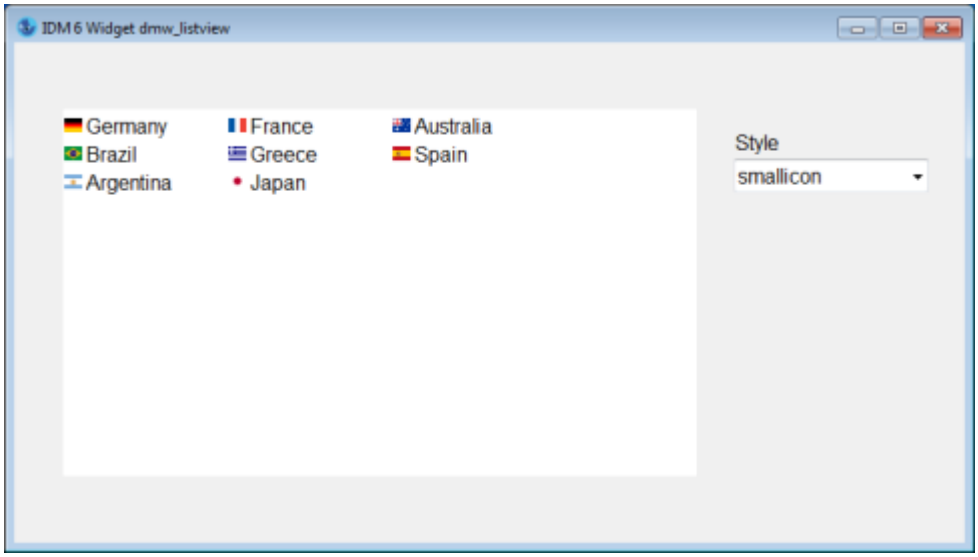



***Abbildung 22:*** *Symbolansicht mit kleinen Symbolen des listviews*

3. Die Listenansicht verwendet auch die kleinen Symbole und die Beschriftung, jedoch sind die Einträge von oben nach unten in einer Liste angeordnet. Bei Bedarf werden weitere Spalten angefügt.


***Abbildung 23:*** *Listenansicht des listviews*

4. In der Detailansicht werden alle verfügbaren Informationen in einer Tabelle angezeigt. Jetzt ähnelt das ***listview***-Objekt einer Tabelle.

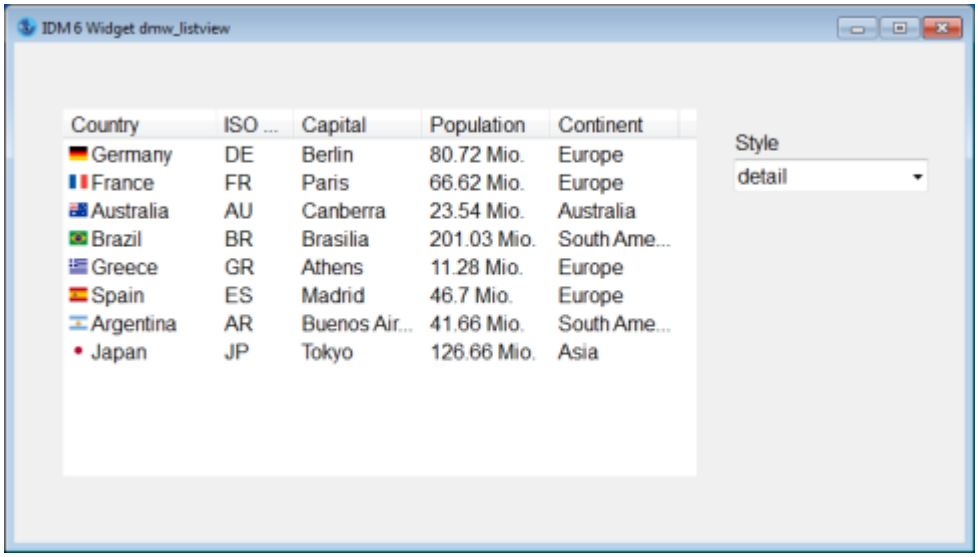



***Abbildung 24:*** *Detailansicht des listviews*

5. Die Kachelansicht entspricht der Listenansicht, zeigt aber große Symbole an. Daneben erscheint die Beschriftung.

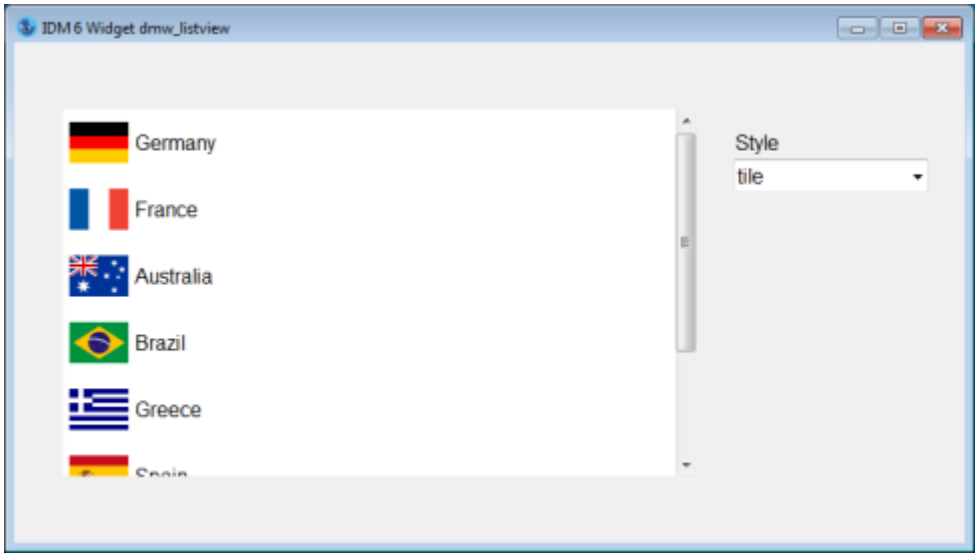



***Abbildung 25:*** *Kachelansicht des listviews*

### Verfügbarkeit

» Nur unter MICROSOFT WINDOWS.

» Kann nur mit der USW-Option des ISA Dialog Managers verwendet werden.

» Die **idmwidgets.dll** muss sich im USW-Klassenpfad befinden (Standard: *<IDM-Installationsverzeichnis>\uswclasses*).

## 22.1 Definition

```
{ export | reexport } { model } dmw_listview { <Bezeichner> }
{
  <Standardattribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

| | | | |
|---|---|---|---|
| **Ereignisse** | *activate* | *changed* | *dbselect* |
| | *deactivate* | *extevent* | *focus* |
| | *help* | *resize* | *select* |
| **Kinder** | Keine | | |
| **Vater** | ***groupbox*** | ***layoutbox*** | ***notepage*** |
| | ***splitbox*** | ***toolbar*** | ***window*** |
| **Menü** | Popup Menü | | |

## 22.2 Beschreibung der Ereignisse

### 22.2.1 activate

Das *activate*-Ereignis tritt auf, wenn ein Listeneintrag markiert wird (*.selected[I] := true*).

Das Attribut *.index* von ***thisevent*** enthält den Index des Eintrags, der markiert wurde. Der Datentyp von *thisevent.index* ist *index* und damit konsistent zum *select*-Ereignis. Der Spaltenwert von *thisevent.index* ist immer *1*, da nur diese Spalte einen Markierungsstatus besitzt.

### 22.2.2 dbselect

Das *dbselect*-Ereignis tritt auf, wenn auf dem ***listview*** ein Doppelklick ausgeführt wurde.

Das Attribut *.index* von ***thisevent*** enthält den Index des Eintrags, auf den geklickt wurde. Der Datentyp von *thisevent.index* ist *index* und damit konsistent zum *select*-Ereignis. Wurde kein Eintrag getroffen, dann ist *thisevent.index* nicht gesetzt.

Die Kopfzeile der Detailansicht (Spaltenüberschriften) besitzt kein Doppelklick-Ereignis.

### 22.2.3 deactivate

Das *deactivate*-Ereignis tritt auf, wenn ein Listeneintrag seine Markierung verliert (*.selected[l] := false*).

Das Attribut *.index* von ***thisevent*** enthält den Index des Eintrags, der seine Markierung verloren hat. Der Datentyp von *thisevent.index* ist *index* und damit konsistent zum *select*-Ereignis. Der Spaltenwert von *thisevent.index* ist immer *1*, da nur diese Spalte einen Markierungsstatus besitzt.

### 22.2.4 resize

Das *resize*-Ereignis tritt auf, wenn in der Detailansicht die Spaltenbreite geändert wurde.

Das Ereignis tritt **nicht** auf für Spalten, deren Breite auf *0* gesetzt wurde und vom ***listview*** berechnet wird.

### 22.2.5 select

Das *select*-Ereignis tritt auf, wenn auf dem ***listview*** ein Klick ausgeführt wurde.

Das Attribut *.index* von ***thisevent*** enthält den Index des Eintrags, auf den geklickt wurde. Der Datentyp von *thisevent.index* ist *index* und damit konsistent zum *activate*-Ereignis. Der Spaltenwert kann nur in der Detailansicht ungleich *1* sein.

Wurde kein Eintrag getroffen, dann ist *thisevent.index* nicht gesetzt. Bei einem Klick auf die Kopfzeile der Detailansicht (Spaltenüberschriften) ist der Zeilenwert von *thisevent.index0*.

## 22.3 Geerbte Attribute

| Attribut | Datentyp | Hinweise |
|---|---|---|
| .accelerator | object | |
| .bgc | object | |
| .borderraster | boolean | |
| .control | object | |
| .count[attribute] | anyvalue | |
| .cursor | object | |
| .cut_pending | boolean | |
| .cut_pending_changed | boolean | |
| .dialog | object | |

| Attribut | Datentyp | Hinweise |
|---|---|---|
| .document[integer] | object | |
| .export | boolean | |
| .fgc | object | |
| .firstrecord | object | |
| .focus | boolean | |
| .focus_on_click | boolean | |
| .font | object | |
| .function | object | |
| .function[integer] | object | |
| .groupbox | object | |
| .height | integer | muss > *0* sein |
| .help | string | |
| .index | index | |
| .label | string | |
| .lastrecord | object | |
| .mapped | boolean | |
| .menu[integer] | object | |
| .model | object | |
| .module | object | |
| .navigable | boolean | |
| .notepage | object | |
| .parent | object | |
| .posraster | boolean | |
| .real_height | integer | |
| .real_path[string] | string | |

| Attribut | Datentyp | Hinweise |
|---|---|---|
| .real_sensitive | boolean | |
| .real_visible | boolean | |
| .real_width | integer | |
| .record[integer] | object | |
| .recordcount | object | |
| .reexport | object | |
| .scope | object | |
| .sensitive | boolean | |
| .sizeraster | boolean | |
| .source | object | |
| .statushelp | string | |
| .target | object | |
| .toolbar | object | |
| .toolhelp | object | |
| .transformer[integer] | object | |
| .type[anyvalue] | datatype | |
| .userdata | anyvalue | |
| .visible | boolean | |
| .width | integer | muss > *0* sein |
| .window | object | |
| .xauto | integer | |
| .xleft | integer | |
| .xright | integer | |
| .yauto | integer | |
| .ybottom | integer | |
| .ytop | integer | |

## 22.4 Spezifische Attribute

| Attribut | Kurzbeschreibung | Datentyp | Standard | Zugriff | | C | V |
|---|---|---|---|---|---|---|---|
| | | | | get | set | | |
| .colcount | Anzahl der Spalten | *integer* | *1* | x | x | x | x |
| .coltitle[integer] | Spaltenüberschriften | *string* | *""* | x | x | x | x |
| .colwidth[integer] | Spaltenbreiten in der Detailansicht | *integer* | *0* | x | x | x | x |
| .content[index] | Listeneinträge, Beschriftung in Spalte 1 | *string* | *""* | x | x | x | x |
| .mincolwidth[integer] | minimale Spaltenbreite in der Detailansicht | *integer* | *0* | x | x | x | x |
| .picheight | Höhe der großen Symbole | *integer* | *0* | x | x | x | x |
| .picture[integer] | große Symbole | *object* (*tile*) | *""* | x | x | x | x |
| .picwidth | Breite der großen Symbole | *integer* | *0* | x | x | x | x |
| .rowcount | Anzahl der Zeilen | *integer* | *0* | x | x | x | x |
| .selected[integer] | Markierung der Listeneinträge | *boolean* | *false* | x | x | x | x |
| .smallpicheight | Höhe der kleinen Symbole | *integer* | *0* | x | x | x | x |
| .smallpicture[integer] | kleine Symbole | *object* (*tile*) | *""* | x | x | x | x |
| .smallpicwidth | Breite der kleinen Symbole | *integer* | *0* | x | x | x | x |
| .style | Darstellungsart | *integer* (*0…4*), *string* | *0* | x | x | x | x |

C *changed*-Ereignis bei Änderung

V Attribut wird vererbt

# 23 mapping

Das Mapping-Objekt dient dem Zweck, eine semantische Aktion zu definieren, die während einer Transformation für einen bestimmten Knoten (in einem XML-Baum oder in der IDM-Objekthierarchie) aufgerufen werden soll, wenn eine Übereinstimmung zwischen diesem Mapping-Objekt und dem Knoten gefunden wird. Die Mapping-Objekte werden als Kinder eines Transformer-Objekts definiert und beschreiben zusammen mit diesem eine Transformation von Daten.

### Definition

```
{ export | reexport } { model } mapping { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

### Ereignisse

keine

### Kinder

document

***record***

transformer

### Vater

transformer

### Menü

keins

### Methoden

:action()

## 23.1 Attribute

document[I]

external

external[I]

firstrecord

label

lastrecord

name

model

parent

record[I]

recordcount

scope

transformer[I]

userdata

## 23.2 Objektspezifische Attribute

**name**

Hier wird ein Muster (ein XPath ähnlicher Ausdruck) angegeben, das definiert, auf welche Knoten in einem XML-Baum oder in der IDM-Objekthierarchie das Mapping-Objekt passt. Dieses bestimmt, ob während einer Transformation die action Methode für den Knoten aufgerufen wird oder nicht.

## 23.3 Muster für .name

Ein Muster ist ähnlich zu einem Dialog Manager Bezeichner. Das Muster bildet einen Pfad von Elementnamen. Der Pfad beginnt bei der Wurzel des Dokuments oder IDM-Objekthierarchie (also Dialog oder Modul). Jede Hierarchiestufe wird mit dem entsprechenden Teil des Pfades verglichen. Zudem können noch bestimmte Eigenschaften wie Vorhandensein eines Attributs oder die Position innerhalb der Kinder bzw. des Vaters angegeben werden:

```
{ <Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} }
  [ .<Name> [[.<Attr>{<Op>"<Value>"}]] {[<Idx>]} ]
```

**<Name>**

**XML:** Wird mit dem name Attribut des Cursors verglichen. Der Cursor muss den nodetype *nodetype_element* besitzen. Alternativ kann hier auch * angegeben werden, dann wird das name Attribute nicht beachtet.

**IDM:** Wird mit dem Label eines IDM-Objekts verglichen. Alternativ kann hier auch * angegeben werden, dann ist jedes Label passend.

**<Attr>**

**XML:** Der XML-Knoten, auf den der Cursor zeigt, muss das angegebene Attribut besitzen.

**IDM:** Das IDM-Objekt muss das angegebene Attribut besitzen.

**<Op>**

**XML** und **IDM:** Vergleichsoperator, um das in <Attr> angegebene Attribut mit dem <Value> zu vergleichen. Es kann auf gleich **=** und ungleich **<>** verglichen werden.

**<Value>**

**XML** und **IDM:** Der Wert gegen den das <Attr> verglichen wird.

**<Idx>**

**XML:** Der XML-Knoten muss an dieser Position innerhalb der Kinder seines Vaters stehen.

**IDM:** Das IDM-Objekt muss an dieser Position innerhalb der Kinder seines Vaters stehen. Dabei werden alle Kinder aus hierarchischen Attributen zusammengefasst betrachtet.

### Besonderheiten

**.**

**XML:** Beginnt das Muster mit einem Punkt, dann ist es relativ zum aktuellen Knoten. Der Pfad beginnt also beim aktuellen Knoten.

**..**

**XML** und **IDM:** Zwei aufeinander folgende Punkte überspringen beliebig viele Hierarchiestufen.

**[<Idx>]**

**XML:** Ein Index ohne weitere Angaben selektiert den XML-Knoten an dieser Position. Der nodetype des Knotens bleibt dabei unberücksichtigt. Jeder Knoten kann somit durch einen Ausdruck der Form {**[**<Idx>**]**[**.[**<Idx>**]**]} eindeutig referenziert werden (path Attribut).

Bei allen Vergleichen wird zwischen Groß- und Kleinschreibung unterschieden.

### Beispiel

Das Muster „..CD[.Title = Yellow]“ referenziert innerhalb eines XML-Baums alle Knoten (egal wo in der Hierarchie) mit dem Tag „CD“, die ein Attribut „.Title“ mit dem Wert *Yellow* haben.

Das gleiche Muster auf IDM-Objekte angewandt wird alle Objekte mit dem Label „CD“ auswählen, die das benutzerdefinierte Attribut „.Title“ mit dem Wert *Yellow* haben.

## 23.4 Objektspezifische Methoden

**:action()**

Diese Methode wird von der **:action**-Methode des Vater-***Transformers*** aufgerufen, wenn eine Entsprechung zwischen einem Knoten in einem XML-Baum bzw. einer IDM-Objekthierarchie und dem Muster im *.name*-Attribut des ***Mapping***-Objektes gefunden wurde. Die Default-Implementierung dieser Methode tut nichts und liefert immer *true* zurück.

Da diese Methode überdefiniert werden kann, kann der IDM-Programmierer hier festlegen, was mit den Daten aus dem Knoten geschehen soll.

# 24 menubox (Menübox)

Fast jedem Objekt kann genau ein aktives Menü zugeordnet werden. Da das Objekt ***menubox*** nicht direkt als Kind an das Objekt gehängt wird, muss die Definition mit dem Schlüsselwort **menu** anstatt **child** eingeleitet werden. Nur wenn das Menü zu einem Fenster gehören soll, muss dieses über **child** an das Fenster gebunden werden.

Die eigentliche Definition eines Menüs erfolgt dann anschließend über das Keyword **menubox**, gefolgt von einem Identifikator. Zwischen den anschließenden geschweiften Klammern folgt dann die Definition der in dem Menü enthaltenen Menüeinträgen und Menüseparatoren sowie den Attributen des Menüs.

### Definition

```
{ export | reexport } { model } menubox { <Bezeichner> }
{
  <Standardattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

*open*

### Kinder

document

***menubox***

***menuitem***

***menusep***

***record***

transformer

### Vater

***canvas***

***checkbox***

***dialog***

***edittext***

***groupbox***

***image***

***layoutbox***

***listbox***

***menubox***

***module***

***notepage***

***poptext***

***pushbutton***

***radiobutton***

***rectangle***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***tablefield***

***toolbar***

***treeview***

***window***

### Menü

keins

### Anmerkung

Dem Objekt Fenster darf mehr als ein Menü zugeordnet werden; es muss daher über das Schlüsselwort **child** in die Menüleiste des Fensters eingehängt werden.

Die Menüdefinition für Objekte in Fenstern kann mit Hilfe des Attributs *.menu* gemacht werden.

Einer ***Menübox*** können Menüboxen, Menüeinträge und Menüseparatoren als Kinder in beliebiger Anzahl über das Schlüsselwort **child** zugeordnet werden.

Wenn das *open* Ereignis des ***menubox***-Objekts verwendet wird, sollten auf keinen Fall die Vaterobjekte des ***menubox***-Objekts in einer Regel zerstört werden, da es sonst zu ungewollten Effekten kommen kann. Besonders „gefährlich“ sind hier die *open*-Ereignisse von ***menubox***-Objekten, die Kind eines anderen ***menubox***-Objekts oder eines ***window***-Objekts sind.

## 24.1 Attribute

bgc

child[I]

childcount

class

control

dialog

document[I]

external

external[I]

fgc

firstchild

firstmenu

firstrecord

font

groupbox

help

helpmenu

index

label

lastchild

lastmenu

lastrecord

layoutbox

mapped

member[I]

membercount

menucount

model

notepage

options[enum]

order

parent

real_sensitive

real_visible

record[I]

recordcount

scope

sensitive

statushelp

title

userdata

visible

window

## 24.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| helpmenu | Definiert, dass die Menübox im Fenster rechts ausgerichtet wird. |
| options[enum] | Mit diesem Attribut kann die Menübox unter Motif auch als Tearoff-Menü eingestellt werden.<br>Index: *opt_enable_tearoff* |
| order | Definiert die Reihenfolge relativ zu anderen Menüboxen bzw. Menüeinträgen. |
| title | Angezeigter Text bei einem Fenstermenü. |

## 24.3 Kontextmenüs (Popup-Menüs) unter Microsoft Windows

Wenn ein Kontextmenü geöffnet wird, erhält das Objekt, an welchem das Kontextmenü (Popup-Menü) definiert ist, den Tastaturfokus. Ist dieses Objekt ein Gruppierungsobjekt (beispielsweise ein window- oder groupbox-Objekt), dann wird der Tastaturfokus auf das erste Kindobjekt, welches den Fokus erhalten kann, gesetzt. Allerdings wird der Tastaturfokus nur umgesetzt, wenn das Objekt, an dem das Kontextmenü definiert ist, oder eines der direkten oder indirekten Kindobjekte nicht schon den Tastaturfokus besitzen.

### Anmerkungen

» Popup-Menues werden unter Windows auch über die Menütaste des Systems geöffnet (Taste `Alt`). Will man direkt das Fenstermenü offnen, so sollte die Taste `F10` genutzt werden.

## 24.4 Kontextmenüs (Popup-Menüs) unter Motif

Ab Motif 2.1 ist es Standard, dass sich Popup-Menüs mit Mnemonics automatisch öffnen, wenn ein Mnemonic-Zeichen eingegeben wird. Bei Objekten mit Texteingabe kann dieses Verhalten allerdings die Texteingabe stören.

Ab IDM-Version A.05.02.f4 öffnet sich bei ***Edittexten*** mit einem **Popup-Menü**, in dem ***Mnemonics*** definiert sind, das Popup-Menü nicht mehr automatisch bei Eingabe eines Mnemonic-Zeichens.

Damit verhält sich der Edittext wie andere Objekte **mit** Texteingabe. Bei fokussierbaren Objekten **ohne** Texteingabe, wie Pushbutton, Checkbox, Radiobutton, Image, Scrollbar und Spinbox mit Statictext, wird das Popup-Menü bei Eingabe eines Mnemonic-Zeichens automatisch geöffnet.

### Anmerkungen

- Bei allen Objektklassen kann das Popup-Menü ab Motif 2.1 standardmäßig mit der Tastenkombination `Shift + F10` geöffnet werden.
- Als Alternative zu Mnemonics können Acceleratoren verwendet werden.

## 24.5 Beispiel

Fenster mit Menübar:

```
window Hauptfenster
{
  child menubox Menue_1
  {
    .title "Start Menu";
  }

  child menubox Menue_2
  {
    .title "File";
  }

  child menubox Popup
  {
    .title "3. Menue";

    child menuitem Menue_3
    {
      .text "Popup 1";
    }

    child menuitem Menue_4
    {
      .text "Popup 2";
    }
```

```
  }
}
```

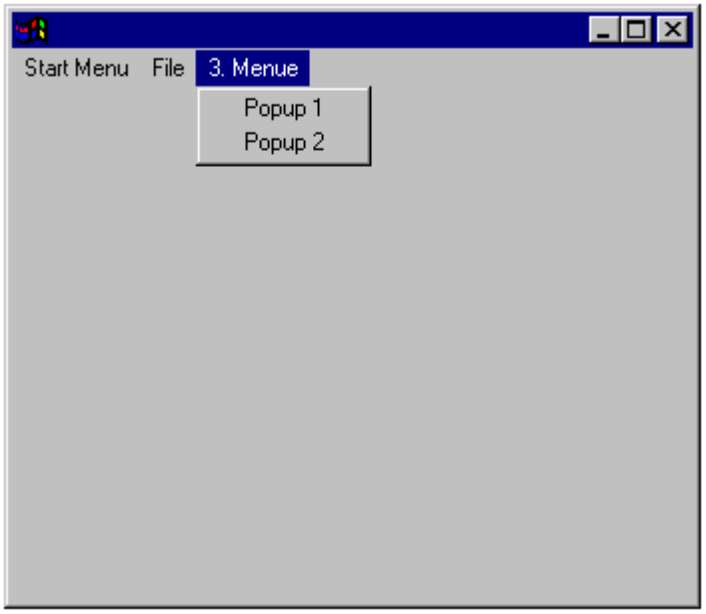



***Abbildung 26:*** *Fenster mit Menüleiste*

### Anmerkung

Soll in einem Menü ein weiteres Menü (Kaskadenmenü) enthalten sein, so muss dieses über das Schlüsselwort **menubox** als Kind des übergeordneten Menüs definiert werden.

```
child menubox Pulldown
{
  .title "Dateibearbeitung";
  .sensitive true;
  .visible true;

  child menuitem Laden
  {
    .text "Laden";
    .visible true;
  }

  child menubox Sichern /* Cascade menu */
  {
    .title "Sichern";

    child menuitem Sichern_unter
    {
      .text "Sichern unter";
    }
```

```
    child menuitem Sichern
    {
      .text "mit aktuellem Namen";
    }
  }
}
```

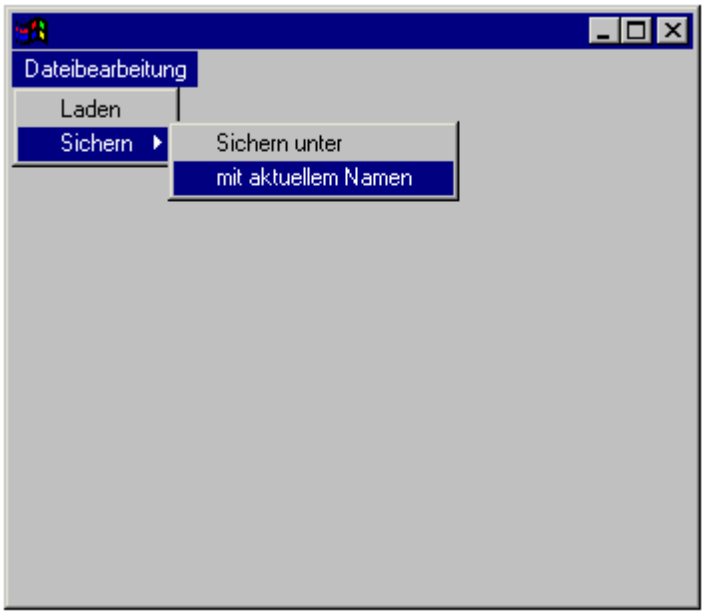



***Abbildung 27:*** *Kaskadenmenü*

# 25 menuitem (Menüeintrag)

In einem Menü (***menubox***) kann eine beliebige Anzahl von Menüeinträgen definiert werden. Das Schlüsselwort zur Definition eines Menüeintrags ist **menuitem**, gefolgt von einem Identifikator. Daran anschließend stehen in geschweiften Klammern Angaben zu den Objektattributen. Ein Menüeintrag kann nur als Kind eines Menüs (***menubox***) oder eines Fensters (***window***) definiert werden.

### Definition

```
{ export | reexport } { model } menuitem { <Bezeichner> }
{
  <Standardattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*activate*

*deactivate*

*extevent*

*help*

*key*

*select*

### Kinder

document

***record***

transformer

### Vater

***menubox***

***module***

***window***

### Menü

keins

## 25.1 Attribute

accelerator

active

bgc

class

control

dialog

document[l]

external

external[l]

fgc

firstrecord

font

function

groupbox

help

index

label

lastrecord

layoutbox

mapped

member[l]

membercount

model

notepage

options[enum]

order

parent

picture[enum]

real_sensitive

real_visible

record[l]

recordcount

scope

sensitive

statushelp

style

text

userdata

visible

window

## 25.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| active | Gibt den Aktivierungszustand bei einem Pushbutton- oder Checkbox-Menü an bzw. legt ihn fest. |
| options[enum] | Gibt an, ob ein Widget verwendet werden soll (nur Motif).<br>Index: *opt_use_widget* |
| order | Definiert die Reihenfolge relativ zu anderen Menüeinträgen in einer Menübox. |
| picture[enum] | Gibt an, welches Bild beim jeweiligen Aktivierungszustand angezeigt werden soll (bei Pushbutton-Style wird immer das mit *tile_default* angegebene Bild dargestellt).<br>Indizes:<br>*tile_default*<br>*tile_active* |
| style | Definiert, ob ein Pushbutton-, Radiobutton- oder Checkbox-Menü verwendet werden soll. |
| text | Angezeigter Text des Objekts. |

## 25.3 Beispiel

Zweistufiges Menü:

```
dialog Test
{
}
window WnTest
{
```

```
  .xleft 508;
  .ytop 126;
  .title "Testfenster";

  child menubox MbDatei
  {
    .title "Datei";

    child menuitem Mi1
    {
      .text "&\326ffnen";
    }

    child menuitem Mi2
    {
      .text "Sichern";
    }

    child menubox Mb2
    {
      .title "&Beenden";

      child menuitem Mi3
      {
        .text "&Mit Sichern";
      }

      child menuitem Mi4
      {
        .text "&Ohne Sichern";
      }

  }
}
```

Dieses Beispiel erzeugt folgendes Menü:

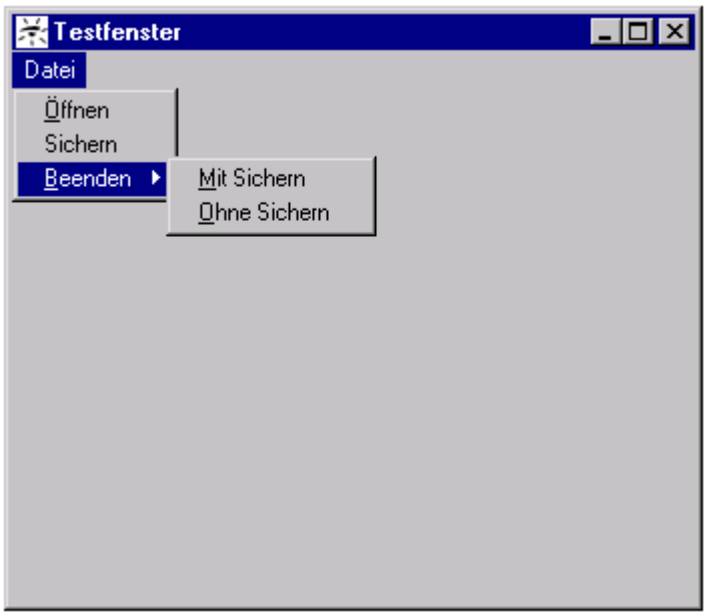



***Abbildung 28:*** *Fenster mit Menüeinträgen*

# 26 menusep (Menüseparator)

Die Menüseparatoren sind die zweite Art von Objekten, die innerhalb eines Menüs verwendet werden können. Diese werden durch das Schlüsselwort **menusep** definiert.

### Definition

```
{ export | reexport } { model } menusep { <Bezeichner> }
{
  <Standardattribute>
  <Hierarchieattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

### Kinder

document

***record***

transformer

### Vater

***menubox***

***module***

### Menü

keins

## 26.1 Attribute

bgc

class

control

dialog

document[l]

external

external[l]

fgc

firstrecord

groupbox

help

index

label

lastrecord

layoutbox

mapped

member[I]

membercount

model

notepage

order

parent

real_sensitive

real_visible

record[I]

recordcount

scope

sensitive

style

userdata

visible

window

## 26.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| order | Definiert die Reihenfolge relativ zu anderen Menüeinträgen in einer Menübox. |
| style | Definiert, ob ein ein- oder doppelzeiliger Separator verwendet werden soll. Nur unter Motif. |

## 26.3 Beispiel für das Objekt Menüseparator

Im folgenden Beispiel wird der Menüpunkt „Beenden“ durch eine Trennzeile optisch von den anderen Menüeinträgen abgesetzt.

```
child menubox DateiMitSeparator
{
  .title "Datei";
  .sensitive true;
  .visible true;

  child menuitem Mi_Oeffnen
  {
    .text "Öffnen";
    .sensitive true;
    .visible true;
  }

  child menuitem Mi_Sichern
  {
    .text "Sichern";
    .sensitive true;
    .visible true;
  }

  child menusep Separatorzeile { }

  child menuitem Mi_Beenden
  {
    .text "Beenden";
    .sensitive true;
    .visible true;
  }

  child menusep
  {
    .style 1;
  }
}
```

# 27 messagebox

Mit Hilfe dieses Objektes können die im Fenstersystem verwendeten Standard-Messageboxes benutzt werden.

Eine Messagebox wird entweder mit Hilfe der eingebauten Funktion **querybox()** oder der Funktion **DM_QueryBox()** geöffnet. Bitte beachten Sie, dass das Attribut *.visible* für dieses Objekt nicht verfügbar ist.

### Definition

```
{ export | reexport } { model } messagebox { <Bezeichner> }
{
  <Objektspezifische Attribute>
}
```

### Siehe auch

Eingebaute Funktion querybox im Handbuch „Regelsprache“

C-Funktion DM_QueryBox im Handbuch „C-Schnittstelle - Funktionen“

### Ereignisse

*extevent*

### Kinder

document

***record***

transformer

### Vater

***dialog***

***module***

### Menü

keins

## 27.1 Attribute

bgc

button[1-3]

class

cursor

defbutton

dialog

document[I]

external

external[I]

fgc

firstrecord

font

icon

index

label

lastrecord

member[I]

membercount

model

record[I]

recordcount

scope

sysmodal

text

title

## 27.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| button[1-3] | Art der erscheinenden Buttons.<br>Werte siehe „Attributreferenz“. |
| defbutton | Defaultbutton des Objekts. |
| icon | Angezeigtes Icon des Objekts.<br>Werte siehe „Attributreferenz“. |

| Attribut | Beschreibung |
|---|---|
| sysmodal | Gibt an, ob eine Messagebox vor allen anderen Fenstern auf dem Desktop erscheinen soll.<br>Vom IDM FÜR QT **nicht** unterstützt. |
| text | Angezeigter Text des Objekts. |
| title | Titel der Messagebox. |

## 27.3 Hinweise zum IDM für Motif

Die Beschriftung von Messagebox-Buttons können Sie mit Hilfe des Attributs msgboxtext[enum] am Objekt ***dialog*** definieren (Motif).

Bitte beachten Sie in Bezug auf die Anordnung einer Messagebox vor oder hinter anderen Fenstern und Dialogfeldern auf dem Bildschirm (Z-Ordnung) den Hinweis in Kapitel „Z-Ordnung von Fenstern und Dialogfeldern“.

## 27.4 Beispiel

```
dialog Mess

messagebox MsBox
{
  .text  "Wollen Sie wirklich schon die Anwendung beenden?";
  .title "Beispiel Messagebox";
  .icon  icon_warning;
}
```


***Abbildung 29:*** *Messagebox*

# 28 module (Modul)

Ein Modul ist ein Teildialog innerhalb eines übergreifenden Dialogs. Es ist eine für sich geschlossene Einheit und kann mit dem Objekt ***dialog*** verglichen werden. Es bildet die Klammer für alle in ihm enthaltenen Objekte, so dass diese in einer Datei abgespeichert werden können. In einem Modul können beliebige Ressourcen, Regeln, Defaults, Vorlagen oder Instanzen definiert werden, die die Funktionalität des Moduls darstellen.

### Definition

```
module { <Bezeichner> }
{
  <Standardattribute>
  <Hierarchieattribute>
}
```

Solch ein Dialogteil kann z.B. dienen als Bibliothek

- für Standardressourcen, die in einem Unternehmen eingesetzt werden sollen (Umsetzung firmenspezifischer Style-Guides.)
- für dialog- und projektübergreifende Vorlagen oder Modelle
- für immer wieder benötigte Funktionalitäten (z.B. Suche, Anmelden bei Datenbank…)

### Siehe auch

Weitere Informationen finden Sie beim Objekt ***import*** und im Kapitel „Modularisierung“ des Handbuchs „Programmiertechniken“.

### Ereignisse

*extevent*

*finish*

*start*

### Kinder

document

***import***

***module***

***record***

transformer

***window***

**Vater**

***dialog***

***module***

**Menü**

keins

## 28.1 Attribute

child[I]

childcount

class

document[I]

external

external[I]

firstchild

firstrecord

index

label

lastchild

lastrecord

record[I]

recordcount

# 29 notebook

Das ***notebook*** symbolisiert ein Notizbuch mit verschiedenen Seiten, die ***notepages***. Es ist in unterschiedliche Abschnitte unterteilt, die durch Reiter, sog. „Tabs“, markiert sind.

Dieses Objekt kann z.B. zur Parameterdarstellung eingesetzt werden, um bestimmte Grundeinstellungen vorzunehmen. Des weiteren können mit Hilfe des Notebooks auch Daten dargestellt werden, die in logische Gruppierungen unterteilbar sind.

### Definition

```
{ export | reexport } { model } notebook  { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

*focus*

*help*

*key*

*paste*

Da die Tabs der Notepages eigentlich zum Notebook-Objekt gehören, wird beim Fokussieren eines Notepage-Tabs ein *focus*-Ereignis am Notebook-Objekt erzeugt.

### Kinder

document

***notepage***

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 29.1 Attribute

acc_label

acc_text

activeobject

alignment

backpage

bgc

binding

bordercolor

borderraster

child[I]

childcount

class

control

cursor

cut_pending

cut_pending_changed

dialog

direction

document[I]

external

external[I]

fgc

firstchild

firstrecord

focus

font

function

groupbox

height

help

index

label

lastchild

lastrecord

layoutbox

majortabheight

majortabwidth

mapped

member[I]

membercount

menu

minortabheight

minortabwidth

model

multiline

notepage

options[enum]

parent

picheight

picwidth

posraster

real_height

real_sensitive

real_visible

real_width

record[I]

recordcount

scope

sensitive

sizeraster

statushelp

tabalignment

tabshape

tile

tilestyle

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 29.2 Spezifische Attribute

Die Eigenschaften der Tableiste (diese gehört nicht zu den Notepages sondern zum Notebook) bzw. der Notepages allgemein werden über die Attribute activeobject, alignment, backpage, binding, direction, majortabheight, majortabwidth, minortabheight, minortabwidth, multiline, tabalignment und tabshape geregelt.

| Attribut | Beschreibung |
|---|---|
| activeobject | Die aktive, d.h. die aktuell oben liegende Notepage. |
| alignment | Definiert die Ausrichtung des Statustextes, den jede Notepage besitzen kann. |
| backpage<br>(nur Motif) | Definiert die Ecke, in der die seitlich sichtbaren Seitenkanten zusammentreffen. |

| Attribut | Beschreibung |
|---|---|
| binding | Definiert die Art, wie ein Notebook gebunden werden soll. |
| direction<br>(nur Motif) | Spezifiziert wird die Richtung, in der die Bindung verlaufen soll. |
| height | Definiert die Höhe des gesamten Notebook-Objekts mit all seinen Elementen.<br>Wird die Höhe zu klein definiert, wird ein fenstersystemabhängiger Minimalwert genommen. |
| majortabheight | Definiert die Höhe aller Majortabs (Hauptindex). |
| majortabwidth | Definiert die Breite aller Majortabs (Hauptindex). |
| menu | Die Verfügbarkeit eines Popup-Menüs, das nur für das Notebook gilt, ist fenstersystemabhängig. Daher sollten - falls ein Popup-Menü vorhanden sein soll - nur zwei Varianten verfolgt werden:<br>1. für jede Notepage wird ein Popup-Menü gesetzt, aber keines für das Notebook<br>2. es wird ein Popup-Menü für das Notebook gesetzt, jedoch besitzt keine Notepage ein Popup-Menü. |
| minortabheight | Definiert die Höhe aller Minortabs (Nebenindex). |
| minortabwidth | Definiert die Breite aller Minortabs (Nebenindex). |
| multiline | Gibt an, ob die Tabs mehrzeilig dargestellt werden sollen oder nicht. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>*opt_center_toolhelp* (nur IDM für Windows). |
| picheight<br>(nur IDM für Windows) | Gibt die Höhe der in den enthaltenen ***Notepages*** dargestellten Bilder an. |
| picwidth<br>(nur IDM für Windows) | Gibt die Breite der in den enthaltenen ***Notepages*** dargestellten Bilder an. |
| tabalignment | Beschreibt die Ausrichtung des Textes in den Tabs eines Notebooks (nur IDM für Windows). |
| tabshape | Definiert die Form der Major- und Minortabs. |
| tile | Definiert das Hintergrundbild des Objekts |

| Attribut | Beschreibung |
|---|---|
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |

## 29.3 Anmerkungen für Microsoft Windows

Das ***Notebook***-Objekt zeichnet den Bereich neben den Reitern immer in der „3D-Objekt“ Hintergrundfarbe. Um zu erreichen, dass sich das ***Notebook*** nahtlos in sein Vaterobjekt einfügt, sollte die Hintergrundfarbe des Vaterobjektes auf CLR_BUTTONFACE gesetzt werden.

Die Reiter des ***Notebook***-Objekts können entweder alle gleich groß sein oder jeder Reiter besitzt eine Breite und Höhe, die dem Titel entspricht. Im letzten Fall werden die Reiter von Windows berechnet (tritt nur ein, wenn majortabheight, majortabwidth und tabalignment auf den Wert *0* gesetzt sind).

Im ersten Fall, wenn alle Reiter gleich groß sind, werden die Breite und die Höhe durch den IDM berechnet. Um die Darstellung aller Reiter in derselben Größe zu erreichen, kann entweder das Attribut *.tabalignment = 1* gesetzt werden oder mindestens eines der Attribute *.majortabwidth* und *.majortabheight* ist ungleich *0*.

Zum Erreichen der für den jeweiligen Fall besten Lösung sollten hier verschiedene Kombinationen durchgegangen werden. Beispielsweise berechnet der IDM die Breite der Reiter automatisch, wenn das Attribut *.majortabheight <> 0* ist.

Bei aktiven „Visual Styles“ wird der Titel einer ***Notepage*** nach rechts verschoben, wenn er ein **Mnemonic** enthält. Um den Effekt zu vermeiden sollten für ***Notebooks*** die unter Windows üblichen Einstellungen

```
.tabalignment := 0;
.majortabwidth := 0;
```

gewählt werden. Wenn **Mnemomics** ohne die genannten Einstellungen verwendet werden, sollten **alle** Notepages eines Notebooks Mnemonics besitzen, damit bei allen Titeln die gleiche Verschiebung auftritt.

### Siehe auch

Kapitel „Anmerkungen zum IDM für Windows“ beim Objekt notepage

## 29.4 Anmerkungen zum IDM für Motif

Ab IDM-Version A.05.01.d kommen die *activate*- und *deactivate*-Ereignisse für die ***Notepage*** im IDM FÜR MOTIF nun analog zum IDM FÜR WINDOWS.

- *activate* wird ausgelöst
  - initial bei der Sichtbarmachung eines ***Notebooks*** für die aktive ***Notepage***
  - beim Umschaltung der Aktivierung per Maus oder Tastatur

- beim Umsetzen der Aktivierung durch setzen des *.active*-Attributs auf *true* an einer anderen ***Notepage***
- beim unsichtbar Setzen oder Zerstören der gerade aktive ***Notepage*** für die nun neu aktivierte ***Notepage***

- *deactivate* wird ausgelöst
  - beim unsichtbar Machen eines ***Notebooks*** für die gerade aktive ***Notepage***
  - beim Umsetzen der aktiven ***Notepage*** per Maus oder Tastatur
  - beim Setzen von *.active* auf *true* bei einer anderen ***Notepage***

### Achtung

Beim erneuten Sichtbarmachen („Clobbering“) eines ohnehin sichtbaren ***Notebooks*** bzw. einer ohnehin sichtbaren ***Notepage***, ausgelöst durch einige spezielle Attributänderungen, kommt es ebenso zur Versendung von *activate*- und *deactivate*-Ereignissen.

Das *deactivate*-Ereignis wird nicht ausgelöst, wenn das ***Notebook*** oder die ***Notepage*** bzw. das übergeordnete Objekt mit **destroy()** zerstört wird.

## 29.5 Beispiel

```
dialog D

window W
{
  .active false;
  .width  359;
  .height 243;
  .title  "Beispiel";

  child notebook N
  {
    .xleft  45;
    .width  181;
    .ytop   19;
    .height 150;

    child notepage Np1
    {
      .active true;
      .title  "AAA";
    }

    child notepage Np2
    {
      .title "BBB";
    }
```

```
    child notepage Np3
    {
      .title "CCC";
    }
  }
}
```

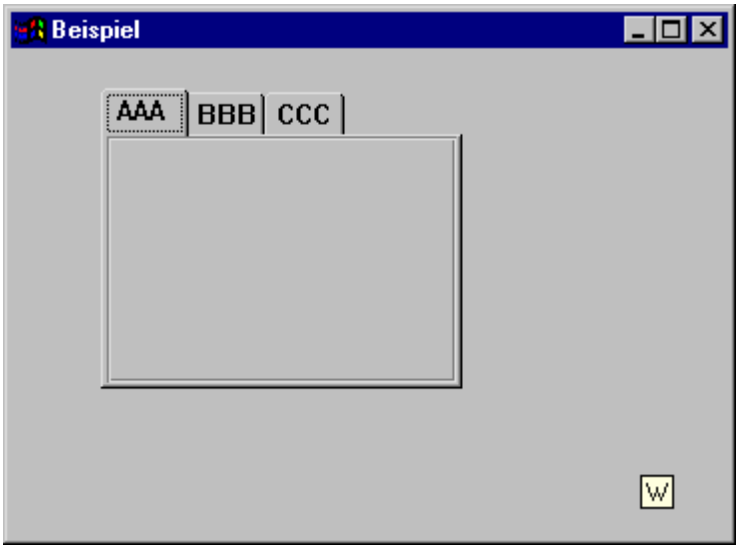



***Abbildung 30:*** *notebook*

# 30 notepage

Das Objekt ***notepage*** ist die Seite eines ***notebooks***.

Eine ***notepage*** wird wie ein Kind definiert. Das Keyword heißt **notepage** und wird optional von einem Labelnamen gefolgt.

Dabei wird die ***notepage*** immer so positioniert, dass sie die Seite des ***notebooks***, abzüglich der Tab-Reihe, insgesamt ausfüllt.

### Definition

```
{ export | reexport } { model } notepage { <Bezeichner> }
{
  <Standardattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Scrollbarattribute>
  <Objektspezifische Attribute>
}
```

### Anmerkung

Die Reihenfolge der Notepages wird über die Anlegereihenfolge in der Dialogdatei festgelegt. Die Betrachtung ist wie bei einem üblichen Ordner: Die erste Notepage ist die oben liegende. Daraus resultiert zwar eine den Kindobjekten entgegengesetzte Richtung, ergibt jedoch eine „natürliche“ Reihenfolge der Tabs.

### Ereignisse

*activate*

*cut*

*deactivate*

*extevent*

*focus*

*help*

*hscroll*

*key*

*paste*

*scroll*

*select*

*vscroll*

Da die Tabs der Notepages eigentlich zum Notebook-Objekt gehören, wird beim Fokussieren eines Notepage-Tabs ein *focus* Ereignis am Notebook-Objekt erzeugt.

### Kinder

document

***canvas***

***checkbox***

***dialog***

***edittext***

***groupbox***

***layoutbox***

***image***

***listbox***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***statictext***

***tablefield***

transformer

***treeview***

### Vater

***notebook***

### Menü

menubox (Menübox)

## 30.1 Attribute

acc_label

acc_text

accelerator

active

bgc

bordercolor

borderraster

borderwidth

child[I]

childcount

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstchild

firstrecord

focus

font

function

groupbox

help

hsb_arrows

hsb_linemotion

hsb_optional

hsb_pagemotion

hsb_visible

index

label

lastchild

lastrecord

layoutbox

mapped

member[l]

membercount

menu

model

notepage

options[enum]

parent

picture

real_height

real_sensitive

real_visible

real_width

record[l]

recordcount

reffont

scope

sensitive

tabtype

text

tile

tilestyle

title

toolbar

toolhelp

userdata

vheight

visible

vsb_arrows

vsb_linemotion

vsb_optional

vsb_pagemotion

vsb_visible

vwidth

window

xorigin

xraster

yorigin

yraster

ytop

## 30.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| active | Definiert die aktuell oberste Notepage eines Notebooks.<br>Es kann nur eine Notepage in einem Notebook den Wert *.active = true* besitzen. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“.<br>Nur aktiv, wenn *.borderwidth > 0*. |
| hsb_arrows<br>(nur Motif) | Definiert, ob Pfeile an den Enden der horizontalen Scrollbar vorhanden sind. |
| hsb_linemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von xorigin beim zeilenweisen horizontalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_optional | Definiert, ob die horizontale Scrollbar nur angezeigt wird, wenn sie benötigt wird.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_pagemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von xorigin beim seitenweisen horizontalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_visible | Definiert die Sichtbarkeit der horizontalen Scrollbar.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |

| Attribut | Beschreibung |
|---|---|
| options[enum] | Optionen des Objekts<br>Indizes:<br>» *opt_center_toolhelp* (nur MS Windows).<br>» *opt_scroll_on_focus* (nur Motif). |
| picture | Definiert das Muster, welches im zur Notepage gehörenden Tab dargestellt wird. |
| tabtype | Beschreibt den Typ des zur Notepage gehörenden Tab.<br>Werte:<br>» *tab_major*<br>» *tab_minor* |
| text | Definiert den Text, der in der Statuszeile erscheinen soll. Falls kein Text definiert ist, enthält die betreffende Notepage keine Statuszeile. Die Position, an der die Statuszeile erscheint, ist vom Fenstersystem und den Attributen *.backpage* und *.direction* am ***Notebook*** abhängig. |
| tile | Definiert das Hintergrundbild des Objekts |
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |
| title | Definiert die Beschriftung der Tabs. Defaultwert ist *null*. Ein Text ist nur vorhanden, wenn hier ein zulässiger Wert ungleich *null* angegeben ist. Die Beschriftung kann hierbei ein Text oder ein Bitmap sein. |
| vheight | Interne („virtuelle“) Höhe des Objekts<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| vsb_arrows<br>(nur Motif) | Definiert, ob Pfeile an den Enden der vertikalen Scrollbar vorhanden sind. |
| vsb_linemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von yorigin beim zeilenweisen vertikalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_optional | Definiert, ob die vertikale Scrollbar nur angezeigt wird, wenn sie benötigt wird.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_pagemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von yorigin beim seitenweisen vertikalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |

| Attribut | Beschreibung |
|---|---|
| vsb_visible | Definiert die Sichtbarkeit der vertikalen Scrollbar.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vwidth | Interne („virtuelle“) Breite des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| xorigin | Gibt an, um wie viele Pixel der eigentliche Inhalt horizontal verschoben ist.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| yorigin | Gibt an, um wie viele Pixel der eigentliche Inhalt vertikal verschoben ist.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |

## 30.3 Anmerkungen zum IDM für Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

Beim Aktivieren von ***Notepages*** (*.active ::= true*) ist Folgendes zu beachten:

- In einem ***Notebook*** ist **immer genau ein** Kindobjekt, also eine ***Notepage***, aktiv (*Np.active = true*). Solange das ***Notebook*** unsichtbar ist (*.real_visible = false*), ist dies entweder das erste ***Notepage***-Kindobjekt oder das zuletzt aktiv gesetzte ***Notepage***-Kindobjekt.
- Sobald das ***Notebook*** sichtbar ist, dürfen nur noch die ebenfalls sichtbaren ***Notepage***-Kindobjekte aktiv gesetzt werden. Wird hingegen ein nicht sichtbares ***Notepage***-Kindobjekt eines sichtbaren ***Notebooks*** aktiv gesetzt (*Np.active ::= true*) geht diese Setzung verloren. Unter Umständen wird dabei sogar ein Fehler (*.active* darf nicht auf *false* geändert werden) für ein sichtbares ***Notepage***-Kindobjekt in das Tracefile geschrieben.
- Vor dem Sichtbarmachen eines ***Notebooks*** muss also immer zuerst das aktiv gesetzte ***Notepage***-Kindobjekt sichtbar (*Np.visible ::= true*) gesetzt werden und es darf nur ein aktives ***Notepage***-Kindobjekt geben. Ist beim Sichtbarmachen des ***Notebooks*** kein sichtbares ***Notepage***-Kindobjekt vorhanden, so ist das Verhalten undefiniert und kann sich zwischen verschiedenen Versionen und Patches unterscheiden; es wird irgendein sichtbares ***Notepage***-Kindobjekt aktiviert werden.

Bei aktiven „Visual Styles“ wird der Titel einer ***Notepage*** nach rechts verschoben, wenn er ein **Mnemonic** enthält. Um den Effekt zu vermeiden sollten für ***Notebooks*** die unter Windows üblichen Einstellungen

```
.tabalignment := 0;
.majortabwidth := 0;
```

gewählt werden. Wenn **Mnemomics** ohne die genannten Einstellungen verwendet werden, sollten **alle** Notepages eines Notebooks Mnemonics besitzen, damit bei allen Titeln die gleiche Verschiebung auftritt.

Für das Layout von Notepages ist Folgendes zu beachten:

- Die Vordergrundfarbe (.fgc) des ***Notebook*** bzw. ***Notepage*** Objektes wird ignoriert. Die Texte werden in einer Systemfarbe dargestellt (wie beim***Pushbutton*** Objekt).
- Die Hintergrundfarbe (.bgc) des ***Notepage*** Objektes ändert nur die Hintergrundfarbe der eigentlichen Darstellungsfläche. Die Hintergrundfarbe des Titels bzw. Reiters ist eine Systemfarbe.
- Die Hintergrundfarbe (.bgc) des ***Notebook*** Objektes ändert nur die Farbe außerhalb der Reiter. Die Hintergrundfarbe der Reiter ist eine Systemfarbe.
- Die Schriftart (.font) des ***Notepage*** Objektes wird ignoriert. Es wird die Schriftart des ***Notebook*** Objektes für die Beschriftung der Reiter genommen.

## 30.4 Anmerkungen zum IDM für Motif

Ab IDM-Version A.05.01.d kommen die *activate*- und *deactivate*-Ereignisse für die ***Notepage*** im IDM FÜR MOTIF nun analog zum IDM FÜR WINDOWS.

- *activate* wird ausgelöst
  - initial bei der Sichtbarmachung eines ***Notebooks*** für die aktive ***Notepage***
  - beim Umschaltung der Aktivierung per Maus oder Tastatur
  - beim Umsetzen der Aktivierung durch setzen des *.active*-Attributs auf *true* an einer anderen ***Notepage***
  - beim unsichtbar Setzen oder Zerstören der gerade aktive ***Notepage*** für die nun neu aktivierte ***Notepage***
- *deactivate* wird ausgelöst
  - beim unsichtbar Machen eines ***Notebooks*** für die gerade aktive ***Notepage***
  - beim Umsetzen der aktiven ***Notepage*** per Maus oder Tastatur
  - beim Setzen von *.active* auf *true* bei einer anderen ***Notepage***

### Achtung

Beim erneuten Sichtbarmachen („Clobbering“) eines ohnehin sichtbaren ***Notebooks*** bzw. einer ohnehin sichtbaren ***Notepage***, ausgelöst durch einige spezielle Attributänderungen, kommt es ebenso zur Versendung von *activate*- und *deactivate*-Ereignissen.

Das *deactivate*-Ereignis wird nicht ausgelöst, wenn das ***Notebook*** oder die ***Notepage*** bzw. das übergeordnete Objekt mit **destroy()** zerstört wird.

## 30.5 Beispiel

```
dialog D

window W
{
```

```
.active false;
.width  359;
.height 243;
.title  "Beispiel";

child notebook N
{
  .xleft  45;
  .width  269;
  .ytop   19;
  .height 150;

  child notepage Np1
  {
    .active true;
    .title  "AAA";

    child statictext
    {
      .sensitive false;
      .xleft 18;
      .ytop  28;
      .text  "Name:";
    }

    child edittext EtName
    {
      .active  false;
      .xleft   69;
      .width   166;
      .ytop    26;
      .content "";
    }

    child statictext
    {
      .sensitive false;
      .xleft 5;
      .ytop  73;
      .text  "Vorname:";
    }

    child edittext EtVorname
    {
      .xleft 69;
      .width 165;
      .ytop  68;
```

```
      }
    }

    child notepage Np2
    {
      .title "BBB";
    }

    child notepage Np3
    {
      .title "CCC";
    }
  }
}
```

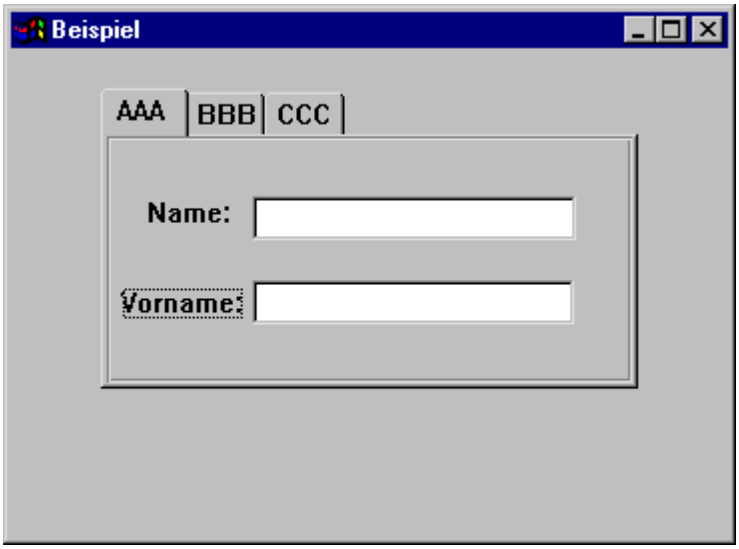



***Abbildung 31:*** *notepage*

# 31 poptext (Combobox)

Mit Hilfe dieses Objektes können textuelle Informationen platzsparend dargestellt werden, wenn der Benutzer aus verschiedenen Informationen immer nur eine benötigt.

### Definition

```
{ export | reexport } { model } poptext { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*activate*

*charinput*

*cut*

*deselect*

*deselect_enter*

*extevent*

*focus*

*help*

*key*

*modified*

*paste*

*select*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 31.1 Attribute

acc_label

acc_text

accelerator

activeitem

bgc

borderraster

class

content

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

editable

endsel

external

external[I]

fgc

firstrecord

focus

font

function

format

formatfunc

groupbox

height

help

hinttext

index

itemcount

label

lastrecord

layoutbox

mapped

maxchars

member[I]

membercount

menu

model

notepage

options[enum]

parent

picheight

picture[I]

picture_hilite[I]

picwidth

posraster

real_height

real_modified

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

showitem

sizeraster

startsel

statushelp

style

text[I]

toolbar

userdata

userdata[I]

visible

width

window

xauto

xleft

xmargin

xright

yauto

ybottom

ytop

## 31.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| activeitem | Index des aktuell angezeigten Textes.<br>Siehe auch Kapitel „Textattribute und .activeitem“. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“.<br>Nur aktiv, wenn *.style* = *listbox*. |

| Attribut | Beschreibung |
|---|---|
| content | Liefert, je nach der in .style getroffenen Einstellung, den Inhalt des aktuell angezeigten Textes bzw. den Inhalt des Eingabefeldes.<br>Siehe auch Kapitel „Textattribute und .activeitem“. |
| editable | Editierbarkeit des Objekts, auch abhängig vom jeweiligen Wert in .style. |
| endsel | Ende eines selektierten Teilstücks in einem Editierbereich. |
| format | Definiert ein Format für den vom Objekt angezeigten String. |
| formatfunc | Gibt die zum Objekt gehörende Formatfunktion an. |
| height | Höhe des Objekts.<br>Beim Wert *0* berechnet der IDM aufgrund des aktuellen Zeichensatzes automatisch die für das Objekt richtige Objektgröße. |
| hinttext | Platzhalter- oder Hinweistext des Objektes. Siehe auch Kapitel „Hinweise zum Attribut .hinttext“ |
| itemcount | Anzahl der Einträge des Objekts. |
| maxchars | Maximale Anzahl möglicher Zeichen im Eingabestring. |
| options[enum] | Optionen des Objekts.<br>Mögliche Indizes:<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_hscroll* (nur IDM FÜR WINDOWS, ab IDM-Version A.06.01.h)<br>» *opt_mnemonic*<br>» *opt_old_select* (bestimmt das Auslösen von *select* und *activate*-Ereignissen, siehe auch Kapitel „Attribut zur Steuerung des select- und activate Ereignisses“)<br>» *opt_quick_navigate* (nur IDM für Motif)<br>» *opt_sort* (nur IDM für Windows) |
| picheight | Angabe der Höhe eines neben den Einträgen dargestellten Bildes. |
| picture[I] | Zum jeweiligen Eintrag gehörendes Bild (IDM für Windows und IDM für Qt). |
| picture_hilite[I] | Bild, welches bei aktivem Eintrag I dargestellt wird IDM für Windows und IDM für Qt). |
| picwidth | Angabe der Breite eines neben den Einträgen dargestellten Bildes. |

| Attribut | Beschreibung |
|---|---|
| real_modified (nicht unter Motif) | Zeigt bei *.style = edittext* und *.style = listbox* an, ob der Inhalt seit dem Fokuserhalt tatsächlich geändert wurde. |
| showitem | Legt die in aufgeklapptem Zustand dargestellte Anzahl der Einträge fest (nur MS Windows und Motif ab Version 2.1). |
| startsel | Anfang eines selektierten Teilstücks in einem Editierbereich. |
| style | Steuert die Art der Darstellung und der Editierbarkeit der Combobox. |
| text[l] | Definiert die einzelnen anzuzeigenden Texte des Objekts.<br>Siehe auch Kapitel „Textattribute und .activeitem“. |
| width | Breite des Objekts.<br>Beim Wert *0* berechnet der IDM aufgrund des aktuellen Zeichensatzes automatisch die für das Objekt richtige Objektgröße. |
| userdata[l] | Userdata für beliebige Daten zu jedem Eintrag des Objekts. |
| xmargin | Horizontaler Abstand zwischen Umrandung und Text des Objekts in Pixeln.<br>Siehe Kapitel „Positionierung des Textes im Objekt“. |

### 31.2.1 Attribut zur Steuerung des select- und activate Ereignisses

Das Attribut .options[opt_old_select] steuert die Erzeugung der activate- und select-Ereignisse und steht standardmäßig auf *false*.

Steht das Attribut *.options[opt_old_select]* auf *false*, so werden select/activate-Ereignisse analog zu den anderen Auswahlobjekten wie Listbox/Treeview generiert.

Erste Änderung zum „alten“ Verhalten ist, das activate-Ereignisse nicht mehr das Erhalten des Fokus (edittext&listbox-Stil) anzeigen, sondern eine Änderung im Aktivierungszustand (.activeitem-Attribut). Zweitens wird ein select-Ereignis nur bei einer "abgeschlossenen" Selektion des Benutzers erzeugt, selbst wenn der gleiche Eintrag nochmals selektiert wird. Abgeschlossen heißt, dass damit die Liste zugeht (Ausnahmen: Bemerkung (1) und (2)) bzw. die Änderung überhaupt kein Sichtbarmachen der Liste erfordert.

| | |
|---|---|
| Datentyp | *boolean* |
| Wertebereich | *false*, *true* |
| Standardwert | *false* |
| Zugriff | get, set |
| Unterstützt auf | MICROSOFT WINDOWS, UNIX |

Zusammenfassung, wann ein select- oder activate-Ereignis am Poptext bei .options[opt_old_select] = false kommt:

| Aktion | Motif 1.2 | Motif 2.1 | MS Win | Ereignisse |
|---|---|---|---|---|
| Cursor-Selektion (Liste zu, auf anderen Eintrag) | * | * | * | - activate und select |
| Cursor-Selektion (Liste offen, auf anderen Eintrag) | * | * | * | - activate (und select im listbox-Stil) |
| Maus-Selektion (Liste offen, gleicher Eintrag) | * | * | * | select |
| Maus-Selektion (Liste offen, anderer Eintrag) | * | * | * | activate und select |
| Liste schließen (z.B. `Return` aber keine Maus-Selektion) | * | * | * | select |
| Liste abbrechen (2) (z.B. `Esc`) | * | * | * | activate um auf Originaleintrag zurückzusetzen |

### Bemerkungen

1. Beim Poptext im Stil Listbox ist die Liste immer offen. Insofern verhält sich hier die Tastatur-Selektion analog zur Maus-Selektion. Auch kann natürlich in diesem Stil die Liste nicht "abgebrochen" werden.
2. Nicht auf allen Fenstersystemen gibt es die Möglichkeit, eine offene Liste abzubrechen.
3. Welche Tasten zur Cursor-Selektion dienen ist systemabhängig; für gewöhnlich aber die Cursortasten `Aufwärts` und `Abwärts`.

### Beispiele zur Nutzung der einzelnen Events bei *.options[opt_old_select] = false*

Um jegliche Änderung, auch bei geöffneter Liste mitzubekommen

```
poptext Pt {
  .options[opt_old_select] false;
  on activate {...}
}
```

Um nur bei der Selektion eines anderen Eintrags zu reagieren

```
poptext Pt {
  .options[opt_old_select] false;
```

```
  on select {
    if (thisevent.value <> thisevent.index) then
      ...
    endif}
}
```

Um auf jede Selektion zu reagieren

```
poptext Pt {
  .options[opt_old_select] false;
  on select {...}
}
```

Nicht mehr zu benutzen ist `on activate`, wenn damit der Erhalt des Fokus gemeint ist. Dies ist nun wie folgt zu schreiben:

```
poptext Pt {
  .options[opt_old_select] false;
  on focus {...}
}
```

### 31.2.2 Textattribute und .activeitem

Zum Zugriff auf den Inhaltsvektor des Poptextes ist das Attribut text[l] zu benutzen.

Die Attribute .startsel und .endsel sind nur in den Styles "edittext" und "listbox" aktiv. Im Normalfall (.style poptext) werden diese Attribute ignoriert. Das Attribut .content ist im Style poptext nur lesbar und darf nicht gesetzt werden.

Beim Poptext mit .style listbox oder edittext ist .content ein dynamisches Attribut; es kann nur zur Laufzeit gesetzt werden, statisch in der Dialogdatei ist dieses Attribut nicht definierbar.

Beim Poptext mit *.style = edittext* ist .content ein eigenständiges Attribut, dessen Wert nicht unbedingt im Inhaltsvektor enthalten sein muss. Daher bleibt .content erhalten und im Eingabefeld stehen, wenn die Liste des Poptextes geleert wird, zum Beispiel indem man .itemcount = 0 setzt.

Im Gegensatz zu text[l] ist content nur als skalares Attribut (also ohne Index) verfügbar.

Das .activeitem Attribut an einem editierbaren poptext-Objekt (.style = listbox oder .style = edittext) liefert für ein sichtbar geschaltetes poptext-Objekt einen "dynamischen" Wert wenn .content geändert wurde. Der gelieferte Wert ist typisch für das Fenstersystem. In der Regel sucht das Fenstersystem in der Liste nach der ersten Zeile, die dem Wert von .content entspricht. Es wird dann die Zeilennummer dieser Zeile geliefert. Ein Wert von 0 bedeutet, dass keine passende Zeile gefunden wurde.

Reproduzierbare Werte kann es nur geben, wenn die Texte eindeutig sind und kein Text ein Anfangstext eines vorherigen Textes ist. Beispiel für eine schlechte Reihenfolge:

```
.text[1] = "kleiner";
.text[2] = "klein";  // ist Anfangstext von .text[1]
```

Wird dies nicht beachtet, ist der gelieferte Wert für .activeitem von der Vorgeschichte abhängig.

Da ein poptext-Objekt im Normalfall immer einen Text aus der Liste darstellt, kann das Attribut .activeitem nur auf einen Wert zwischen 1 und .itemcount (jeweils inklusive) gesetzt werden. Insbesondere ist eine Setzung auf 0 nicht erlaubt. Wenn ein poptext-Objekt im Ausnahmefall einen nicht in der Liste (.text[l]) enthaltenen Text anzeigen soll muss das Attribut .content dynamisch gesetzt werden (nicht bei Poptext mit .style = poptext).

### 31.2.3 Positionierung des Textes im Objekt

Unter Microsoft Windows ist das Attribut xmargin in den Styles *edittext* und *listbox* verfügbar, welches den Textabstand vom rechten und linken Rand des Objektes bestimmt. Allerdings kann bei überlangen Texten (auch abhängig vom MS Windows Patchlevel) unter Umständen der Abstand zum rechten Rand ignoriert werden.

### 31.2.4 Hinweise zum Attribut .hinttext

Dieses Attribut wird nur von WINDOWS und QT unterstützt. Am ***poptext***-Objekt muss muss dabei der Style *edittext* gesetzt sein.

Unter WINDOWS wird .hinttext darüberhinaus unterstützt, wenn das Attribut .style den Wert *listbox* besitzt.

## 31.3 Anmerkungen für Dialog Manager mit Microsoft Windows

- Ein Poptext mit *.style = edittext* vervollständigt beim Öffnen und Schließen der Liste den Inhalt des Editierfeldes (*.content* Attribut), wenn dieser dem Anfang eines Listeneintrags entspricht. Dies ist ein Standardverhalten des Windows-Objekts. Kommt es bei der Vervollständigung zu einer Änderung des aktiven Eintrags, wird ein *activate* Ereignis erzeugt. Wenn keine Vervollständigung stattfindet, wird auch kein Ereignis erzeugt.
- Wenn man die Maus über der geöffneten Liste eines Poptextes bewegt, wird der Eintrag unter dem Mauszeiger markiert. Der Eintrag wird aber nicht aktiviert, was daran zu erkennen ist, dass der Text nicht in das Editierfeld übernommen wird. Es kommt daher auch nicht zu einem *activate* Ereignis und beim Schließen der Liste entspricht der Index im *select* Ereignis dem tatsächlich aktivierten Eintrag und nicht dem Eintrag unter dem Mauszeiger. Dies ist ein Verhalten des Windows-Objekts.
- Ein editierbarer Poptext markiert den gesamten Text, wenn er den Fokus erhält. Anders als beim Edittext geschieht dies auch, wenn man mit der Maus in das Eingabefeld klickt. Um Text zu markieren, muss die Maustaste nochmals gedrückt werden.
- Wenn ein editierbarer Poptext (*.style = edittext* oder *.style = listbox*) den Fokus verliert, wird ab IDM-Version A.05.02.h der Textcursor nicht, wie unter Windows üblich, an den Anfang, sondern ans Ende des Eingabefelds gesetzt. Hierdurch wird eine konsistentere Darstellung erreicht, da sich Poptexte mit Format schon seit früheren IDM-Versionen so verhalten.

- Ein Poptext mit *.style = edittext* würde bei einer Größenänderung den gesamten Text markieren, wenn der IDM dies nicht unterbinden würde. Er tut dies, damit nicht plötzlich irgendwelche Poptexte markiert erscheinen. Er kann jedoch nicht verhindern, dass der Text minimal gescrollt wird. Der IDM stellt aber sicher, dass sich der Textcursor (d. h. *.endsel*) im sichtbaren Bereich befindet. Wenn ein Poptext bei einer Größenänderung den Fokus besitzt, wird eventuell trotzdem der gesamte Text markiert, weil der Poptext unter Umständen den Fokus kurzfristig verliert. Dies geschieht zum Beispiel beim Mausklick auf den Rahmen eines vergrößerbaren Fensters.
- Auswahl bei aktiven „Visual Styles“
  Beim Poptext war ein Eintrag mittels folgender Aktion aus der Liste wählbar:
  Linke Maustaste über dem Pfeil oder dem statischen Text drücken, gedrückt halten und zum gewünschten Eintrag fahren, dort die Maustaste loslassen.
  Sobald „Visual Styles“ aktiv sind geht dies nicht mehr; man muss vor der Auswahl die Maustaste loslassen und erneut klicken.
- Die Höhe einer Combobox kann nicht gesetzt werden; das jeweilige Fenstersystem wählt automatisch einen Wert, der vom Zeichensatz abhängt. Ab der IDM Version A.05.01.c (*.style <> listbox*) beachtet die Combobox nun die gesetzte Höhe. Dies kann zu unerwünschten Darstellungseffekten führen, wenn bisher eine beliebige Höhe gesetzt war. In einem solchen Fall muss die Höhe einfach auf 0 gesetzt werden um die alte Darstellung zu erhalten.
  Die Combobox lässt sich nicht beliebig verkleinern. Daher wird sie immer mit einer bestimmten Minimalgröße dargestellt, auch wenn im Dialogskript eine kleinere Größe angegeben ist.
  Außerdem kann die Höhe einer Combobox mit *.style = poptext* oder *.style = edittext* nicht beliebig vergrößert werden. Daher wird das Objekt mit einer bestimmten maximalen Höhe dargestellt, auch wenn im Dialogskript eine größere Höhe angegeben ist.
- Die Combobox unterstützt ab der IDM Version A.05.01.c auch Dialog Manager Drag&Drop.
  Wobei, wie bei Drag&Drop üblich, nur der aktive Eintrag aus der Liste ausgewählt (gedragged) werden kann. Es ist zu beachten, dass das Wegziehen eines Eintrags aus der Liste den Inhalt des Editierfeldes wie bei einer Auswahl ändert.
  Das index Attribut des thisevent Objekts beim *paste* bzw. *cut* Ereignis kann hierbei folgende Datentypen besitzen:

  | | |
  |---|---|
  | *void* | Position innerhalb des Poptextes kann nicht eindeutig zugeordnet werden bzw. das Editierfeld ist an Position 0 getroffen worden. |
  | *integer* | Die Liste ist an der angegebenen Position getroffen worden. |
  | *index* | Das Editierfeld ist im angegebenen Bereich getroffen worden. |

- Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.
- Folgendes Fehlverhalten besteht aufgrund von Problem mit Kundendialogen weiterhin:
  Ein editierbarer Poptext erzeugt ein *modified* Ereignis beim Verlust des Tastaturfokus nicht nur wenn der Inhalt editiert, sondern auch wenn ein anderer Eintrag in der Liste ausgewählt wurde.

» Damit das Attribut activeitem den richtigen Wert erhält, müssen sich alle enthaltenen Texte auf den ersten 511 Stellen unterscheiden.

» Der Poptext unterstützt ab der IDM Version A.06.01.h auch Dialog Manager die Option **opt_ hscroll**. Die horizontale Scrollbar erscheint innerhalb des Darstellungsbereichs und überdeckt somit die untersten Zeilen. Es wird dafür dann auch eine vertikale Scrollbar eingeblendet. Da die Bedienung von Scrollbars in einer geöffneten Liste für den Anwender nicht einfach ist, sollte diese Option nur verwendet werden, wenn es keine andere Lösung gibt.

## 31.4 Anmerkungen für Dialog Manager unter Motif

» Konform zum Standardverhalten einer Motif-Combobox mit permanenter Liste erhält ein Poptext (*.style = listbox*) beim Mausklick in die Liste nicht automatisch den Fokus. Dies geschieht erst beim Klick in den Textfeld-Bereich. Die *focus* und *deselect* Ereignisse des Poptextes kommen dementsprechend nur beim Fokuswechsel hin bzw. weg vom Textfeld-Bereich.

» Ein *getvalue* (explizit bzw. implizit via Zuweisungsoperator) auf *.activeitem* eines editierbaren Poptextes liefert ab IDM-Version A.05.01.d konsistente Werte analog zum Windows-WSI und dadurch auch *0* wenn der Inhalt von *.content* nicht einem Inhalt eines *.text[l]* entspricht.

» Es sollte **unbedingt** vermieden werden, Änderungen an einem Poptext während des Öffnens der Popup-Liste anzustoßen (z. B. in einer Deselect-Regel). Bestimmte Änderungen können nur durch ein Zerstören und Neuanlegen („clobbering“) des Poptextes ausgeführt werden. Dies wird ab IDM-Version A.05.02.h verzögert um zunächst den normalen Ablauf innerhalb des Toolkits zu ermöglichen und Abstürze zu vermeiden. Allerdings kann dadurch noch der alte Inhalt des Poptextes sichtbar sein. Außerdem verschickt der Poptext in dieser Konstellation bis zum Schließen der Liste keine Ereignisse. Bei dieser Situation wird eine Warnung in die Log-Datei geschrieben.

## 31.5 Beispiel

```
window PoptextDemo
{
  .active false;
  .title "PoptextDemo";

  child poptext Poptext
  {
    .xleft 5;
    .ytop 1;
    .text[1] "Eintrag 1";
    .text[2] "Eintrag 2";
    .text[3] "Eintrag 3";
    .text[4] "Eintrag 4";
    .activeitem 1;
    .showitem 4;
  }
```

```
  child poptext Combox
  {
    .xleft 20;
    .ytop 1;
    .text[1] "Combo 1";
    .text[2] "Combo 2";
    .text[3] "Combo 3";
    .text[4] "Combo 4";
    .activeitem 1;
    .style edittext;
    .showitem 4;
  }

  child poptext Listbox
  {
    .xleft 34;
    .ytop 1;
    .height 5;
    .text[1] "Liste 1";
    .text[2] "Liste 2";
    .text[3] "Liste 3";
    .text[4] "Liste 4";
    .text[5] "Liste 5";
    .text[6] "Liste 6";
    .text[7] "Liste 7";
    .text[8] "Liste 8";
    .activeitem 1;
    .style listbox;
  }
}
```

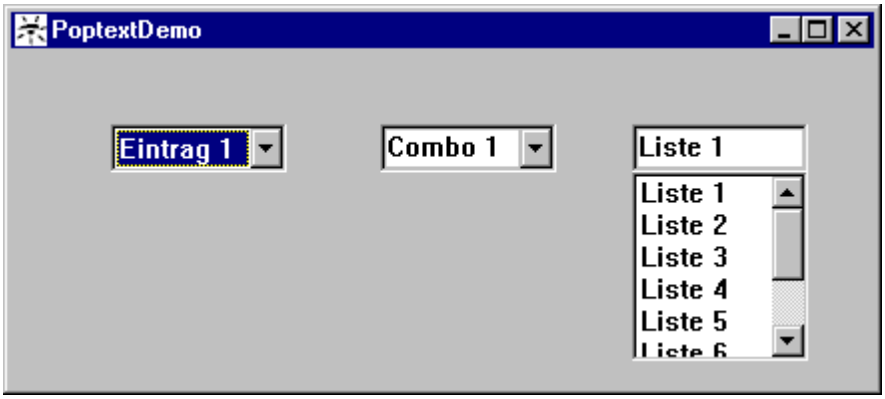
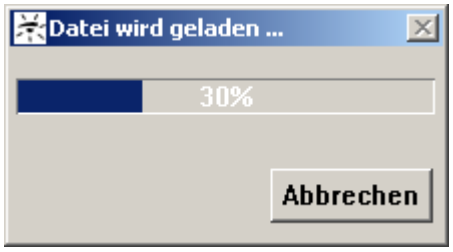



***Abbildung 32:*** *Fenster mit den verschiedenen Ausprägungen der Combobox*

# 32 progressbar

Das ***progressbar***-Objekt ermöglicht es den Verlauf einer länger andauernden Aktion visuell darzustellen. Hierzu wird eine Art Balken angezeigt, dessen Länge dem Fortschritt der Aktion angepasst wird. Dadurch kann der Anwender abschätzen, wie weit die Aktion fortgeschritten ist und wie lange sie noch dauern wird.




### Definition

```
{ export | reexport } { model } progressbar { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*cut*

*extevent*

*focus*

*help*

*paste*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***statusbar***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 32.1 Attribute

accelerator

active

bgc

borderstyle

borderwidth

control

cursor

curvalue

cut_pending

cut_pending_changed

dialog

direction

document[I]

external

external[I]

fgc

firstrecord

focus

font

function

groupbox

height

help

label

lastrecord

layoutbox

mapped

maxvalue

minvalue

model

module

navigable

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

record[I]

recordcount

scope

sensitive

sizeraster

source

statushelp

style[enum]

target

textfgc

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 32.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| curvalue | Gibt die Position des Fortschrittsbalkens an.<br>Siehe auch Kapitel „Berechnung des Fortschrittsbalkens“. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a).<br>Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |
| borderwidth | Rahmenbreite des Objekts.<br>Unter Microsoft Windows wird ein Standardrahmen gezeichnet, wenn der Wert des Attributs größer *0* ist. Es kann also lediglich bestimmt werden, ob ein Rahmen da ist oder nicht. Die Breite kann nicht beeinflusst werden. |
| direction | Ausrichtung der Progressbar.<br>*1* bedeutet vertikale Ausrichtung, der Balken läuft von unten nach oben<br>*2* bedeutet horizontale Ausrichtung, der Balken läuft von links nach rechts |
| maxvalue | Maximalwert des Progress-Bereiches.<br>Siehe auch Kapitel „Berechnung des Fortschrittsbalkens“. |
| minvalue | Minimalwert des Progress-Bereiches.<br>Siehe auch Kapitel „Berechnung des Fortschrittsbalkens“. |
| options[enum] | Optionen des Objekts.<br>Index: *opt_center_toolhelp* (nur MS Windows). |

| Attribut | Beschreibung |
|---|---|
| style[enum] | Aussehen des Fortschrittbalkens.<br>Indizes:<br>*style_continuous* – Darstellung kontinuierlich (*true*) oder in Blöcken (*false*).<br>*style_labeled* – Beschriftung (*true*) oder nicht, nur bei horizontaler Ausrichtung. |
| textfgc | Farbe der Beschriftung (soweit vom System unterstützt), bei *null* wird der Defaultwert des jeweiligen Systems genommen. |

### 32.2.1 Berechnung des Fortschrittsbalkens

Das Attribut .curvalue gibt die Position des Fortschrittsbalken an. Außerdem wird aus diesem Wert die Prozentangabe berechnet, die als optionale Beschriftung angezeigt wird. Bei einem Wert gleich .minvalue ist kein Balken sichtbar, die Prozentangabe ist 0%. Bei einem Wert gleich .maxvalue dagegen ist der gesamte Balken sichtbar und die Prozentangabe ist 100%. Die Werte dazwischen werden entsprechend umgerechnet. Der Balken läuft immer von links nach rechts bzw. von unten nach oben, egal ob *.minvalue* größer oder kleiner als *.maxvalue* ist.

## 32.3 Anmerkung zur Progressbar unter Microsoft Windows

Anmerkung für aktive „Visual Styles“:

Die Farbgebung der „Visual Styles“ lässt sich nicht einer gesetzten Farbe anpassen. Um unschöne Farbkombinationen zu vermeiden wird empfohlen, soweit als möglich auf eine Farbsetzung verzichten.

Insbesondere Abgeraten wird vor der Verwendung von *.fgc* am ***progressbar*** Objekt (es erfolgt ansonsten die Darstellung des Fortschrittbalkens mit einem breiten, von den „Visual Styles“ stammenden Rahmen).

## 32.4 Beispiel

```
dialog D {}
...
import M1  "m1.if"  { .load false; }
import M2  "m2.if"  { .load false; }
import M3  "m3.if"  { .load false; }
import M4  "m4.if"  { .load false; }
import M5  "m5.if"  { .load false; }
import M6  "m6.if"  { .load false; }
import M7  "m7.if"  { .load false; }
import M8  "m8.if"  { .load false; }
import M9  "m9.if"  { .load false; }
import M10 "m10.if" { .load false; }
```

```
...
window WnModLoad
{
  .title  "Module werden geladen ...";
  .width  300;
  .height 100;
  .dialogbox   true;
  .sizeable    false;
  .closeable   false;
  .iconifyable false;
  .borderwidth 1;

  progressbar PrModLoad
  {
    .xauto   0;
    .yauto   0;
    .xleft   5;
    .xright  5;
    .ytop    10;
    .ybottom 10;
    .style[style_continous] false;

    .minvalue 0;
    .maxvalue 10;
    .curvalue 0;
  }
}
...
on dialog start
{
  M1.load := true;
  PrModLoad.curvalue := 1;
  M2.load := true;
  PrModLoad.curvalue := 2;
  M3.load := true;
  PrModLoad.curvalue := 3;
  M4.load := true;
  PrModLoad.curvalue := 4;
  M5.load := true;
  PrModLoad.curvalue := 5;
  M6.load := true;
  PrModLoad.curvalue := 6;
  M7.load := true;
  PrModLoad.curvalue := 7;
  M8.load := true;
  PrModLoad.curvalue := 8;
  M9.load := true;
```

```
  PrModLoad.curvalue := 9;
  M10.load := true;
  PrModLoad.curvalue := 10;
  WnModLoad.visible := false;
  ...
}
```

# 33 pushbutton

Der ***pushbutton*** ist ein Knopftyp zum Treffen einer Entscheidung, wie „Ja“, „Nein“, „Beenden“ oder „Weitermachen“. Der Betätigungszustand des ***pushbuttons*** wird nicht gespeichert; er gilt nur solange der Benutzer den Knopf aktiv selektiert.

### Definition

```
{ export | reexport } { model } pushbutton { <Identifikator> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*cut*

*extevent*

*focus*

*help*

*key*

*paste*

*select*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 33.1 Attribute

acc_label

acc_text

accelerator

bgc

class

control

cursor

cut_pending

cut_pending_changed

defbutton

dialog

document[I]

external

external[I]

fgc

firstrecord

focus

font

function

groupbox

height

help

index

label

lastrecord

layoutbox

mapped

member[I]

membercount

menu

model

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

sizeraster

statushelp

text

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 33.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| defbutton | Legt den Pushbutton als Defaultbutton fest. |
| height | Höhe des Objekts, bei Angabe von *0* wird die Höhe aufgrund des aktuellen Zeichensatzes berechnet. |
| options[enum] | Optionen des Objekts<br>Indizes:<br>*opt_use_widget* (nur Motif).<br>*opt_center_toolhelp* (nur MS Windows). |
| text | Angezeigter Text des Objekts. |
| width | Breite des Objekts, bei Angabe von *0* wird die Breite aufgrund des aktuellen Zeichensatzes berechnet. |

## 33.3 Beispiel

```
dialog Test

window Wn
{
  .active false;
  .width  364;
  .height 180;
  .title  "Testfenster";

  child pushbutton PbStop
  {
    .xleft 96;
    .width 109;
    .ytop  136;
    .text  "&Stop";
  }

  child pushbutton PbWeiter
  {
    .xleft 257;
    .width 100;
    .ytop  137;
    .text  "&Weiter";
```

```
  }
}
```

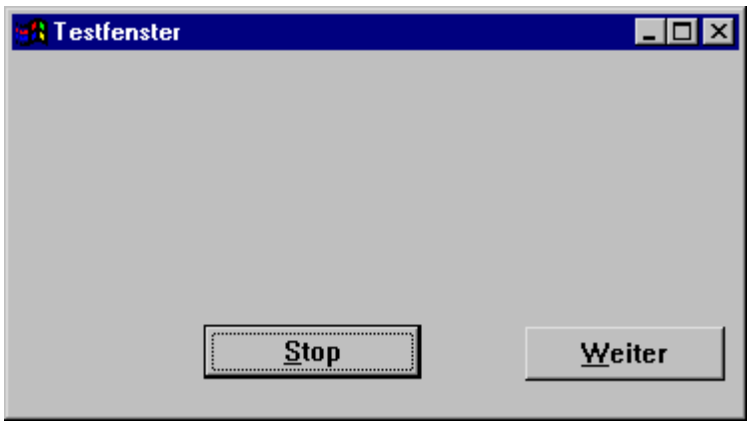



***Abbildung 33:*** *Fenster mit pushbuttons*

# 34 radiobutton

Der ***radiobutton*** (Optionsschaltfläche, Optionsfeld) ist ein Knopftyp zum Treffen einer Auswahl (entweder/oder) aus einer Gruppe von Möglichkeiten. Aus einer Gruppe von ***radiobuttons*** kann also stets genau einer angewählt sein. Die Anwahl eines neuen ***radiobuttons*** löscht den Zustand des bisher angewählten Radiobuttons.

Durch die Selektion eines ***radiobuttons*** werden alle auf derselben hierarchischen Ebene liegenden ***radiobuttons*** deselektiert. Sollen in einem Fenster mehrere ***radiobuttons*** gleichzeitig selektiert sein, so müssen ihre direkten Väter unterschiedlich sein. Dieses kann über den Einsatz von ***groupboxen***, in denen die verschiedenen Gruppen von ***radiobuttons*** liegen, erreicht werden.

### Definition

```
{ export | reexport } { model } radiobutton { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*activate*

*cut*

*deactivate*

*extevent*

*focus*

*help*

*key*

*paste*

*select*

### Kinder

document

***record***

transformer

**Vater**

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 34.1 Attribute

acc_label

acc_text

accelerator

active

bgc

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstrecord

focus

font

function

groupbox

height

help

index

label

lastrecord

layoutbox

mapped

member[l]

membercount

menu

model

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[l]

recordcount

scope

sensitive

sizeraster

statushelp

text

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 34.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| active | Definiert, ob der Radiobutton den Zustand „an“ (*true*) oder „aus“ (*false*) hat. |
| height | Höhe des Objekts.<br>Beim Wert *0* wird die Höhe aufgrund des aktuellen Zeichensatzes berechnet. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>*opt_use_widget* (nur Motif)<br>*opt_push_like* (nur MS Windows)<br>*opt_center_toolhelp* (nur MS Windows) |
| text | Angezeigter Text (Beschriftung). |
| width | Breite des Objekts.<br>Beim Wert *0* wird die Breite aufgrund des aktuellen Zeichensatzes berechnet. |

## 34.3 Hinweise zum Radiobutton unter Microsoft Windows

» Unter Microsoft Windows wird der fokussierte Radiobutton automatisch aktiviert. Ist der aktive Radiobutton insensitiv, dann wird der Fokus auf den nächsten Radiobutton gesetzt, wodurch dieser aktiviert wird. Daher sollte man vermeiden, den aktiven Radiobutton insensitiv zu setzen.

» Beachten Sie zur Hintergrundfarbe die Anmerkungen beim Attribut .bgc im Handbuch „Attributreferenz“.

## 34.4 Beispiel

Zwei Gruppen von Radiobuttons innerhalb eines Fensters:

```
dialog Test
{
```

```
}
window WnRadio
{
  .xleft 67;
  .width 455;
  .ytop 122;
  .height 204;
  .title "Radiofenster";

  child groupbox Gb1
  {
    .xleft 46;
    .width 136;
    .ytop 33;
    .height 131;

    child radiobutton Rb1
    {
      .active true;
      .xleft 26;
      .ytop 18;
      .text "One";
    }

    child radiobutton Rb2
    {
      .xleft 27;
      .ytop 48;
      .text "Two";
    }

    child radiobutton Rb3
    {
      .xleft 26;
      .ytop 76;
      .text "Three";
    }
  }

  child groupbox Gb2
  {
    .xleft 230;
    .width 136;
    .ytop 33;
    .height 131;
```

```
    child radiobutton Rb4
    {
      .xleft 25;
      .ytop 18;
      .text "Four";
    }

    child radiobutton Rb5
    {
      .xleft 25;
      .ytop 48;
      .text "Five";
    }

    child radiobutton Rb6
    {
      .xleft 27;
      .ytop 78;
      .text "Six";
    }
  }
}
```

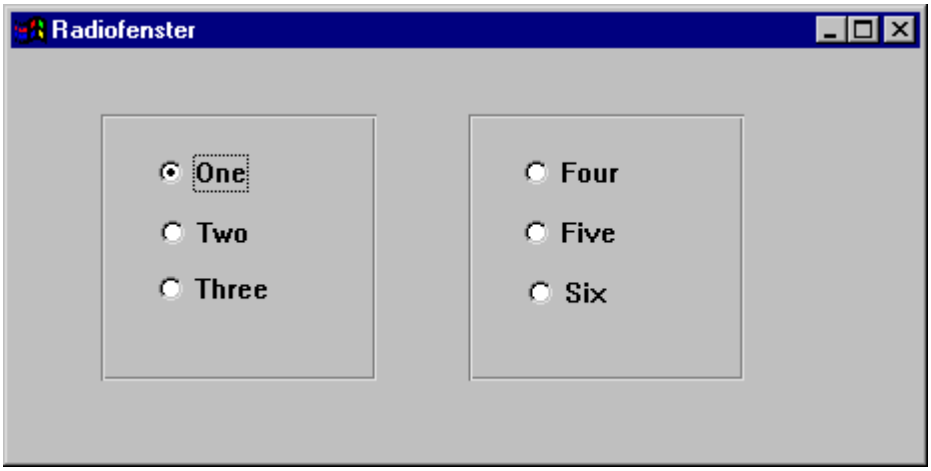



***Abbildung 34:*** *Radiobuttons in zwei Gruppen*

# 35 record

Mit Hilfe des ***records*** können beliebige Strukturen in der Regelsprache gebildet werden.

Diese ***records*** können direkt als Kinder des Dialoges (global verfügbar) oder als Kinder von beliebigen anderen Objekten definiert werden.

***record***-Objekte können auch bei Modellen definiert und an Instanzen mit vererbt werden. ***records*** selber können auch als Modell definiert und instanziiert werden.

Ein ***record*** kann also sowohl benutzerdefinierte Attribute als auch Unterrecords beinhalten.

### Definition

```
{ export | reexport } { model } record <Bezeichner>
{
  <Hierarchieattribute>
  <Objektspezifische Attribute>
  [ <Definition von benutzerdefiniertem Attribut> ; ]
  [ { export | reexport } record <Bezeichner> <record-Definition> ]
}
```

### Ereignisse

keine

### Kinder

document

***record***

transformer

### Vater

alle Objektarten

### Menü

keins

## 35.1 Attribute

class

configurable

control

dialog

document[I]

external

external[I]

firstrecord

groupbox

index

itemorder

label

lastrecord

layoutbox

member[I]

membercount

model

notepage

parent

record[I]

recordcount

scope

toolbar

window

## 35.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| configurable | Definiert, ob ein Record konfigurierbar sein soll, d.h. in der mit DM_LoadProfile geladenen Konfigurationsdatei gesetzt werden darf. |
| itemorder | Gibt die Definitionsreihenfolge für Attribute und Records innerhalb von Records an. |
| member[I] | I-tes benutzerdefiniertes Attribut des Objekts. |
| membercount | Gesamtzahl der benutzerdefinierten Attribute des Objekts. |

## 35.3 Beispiel

```
record Person
```

```
{
  string Name;
  string Vorname;
  string Wohnort;
}
function c void PutPerson (integer, record Person input);
function c void GetPerson (integer, record Person output);
```

Hiermit definieren Sie eine Struktur mit drei Elementen und zwei Funktionen, die diesen Record als Parameter erhalten.

### Record mit Kindrecord

```
record Daten
{
  string Name;
  integer Zustand;

  record Anfangswerte
  {
    integer Position;
    integer Alphawert;
  }
  record Endergebnis
  {
    boolean OK;
    integer BetaWert;
    integer GammaWert;
  }
}
```

### Abfrage der Definitionsreihenfolge in einem Record

```
record R
{
  string Name;
  record Kind1
  {
    integer Nummer;
    boolean An;
  }
  boolean Aus;
}

print R.itemorder;  // => "ARA"
```

## 35.4 Einbindung in Anwendung

Um Funktionen für Dialoge mit Records mit einer Anwendung versehen zu können, müssen vom DM generierte C-Module übersetzt und dazugebunden werden. Die Generierung dieser Module erfolgt durch Aufruf der Simulation mit der Option **+writetrampolin**:

```
idm +writetrampolin <outfile> <dialogskript>
```

Dieses Statement generiert aus einem Dialogskript die notwendigen Module zum Aufruf von Funktionen. Je nach Art der Funktionen, die solche Records verwenden, werden die entsprechenden Header-Dateien (C und/oder COBOL) erzeugt.

Werden solche Strukturen im Dialog Manager-Objekt ***application*** verwendet, wird also der verteilte Dialog Manager eingesetzt, muss noch zusätzlich die Anwendung angegeben werden, für die die Dateien generiert werden sollen.

```
idm +writetrampolin <contfile>
  +application <ApplicationName>
  <dialogskript>
```

# 36 rectangle (Rechteck)

Dieses Objekt dient lediglich zur grafischen Gestaltung. Es sollte in der Regel keinerlei Funktionalität und Bedeutung für den Dialogablauf haben.

### Definition

```
{ export | reexport } { model } rectangle { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

Da das Objekt Rechteck hauptsächlich dazu gedacht ist innerhalb einer Oberfläche Bereiche optisch voneinander zu trennen, sind für das Aussehen dieser „Trennlinie“ im wesentlichen die Attribute filled, borderwidth, bgc und fgc notwendig.

### Ereignisse

*dbselect*

*extevent*

*focus*

*help*

*key*

*select*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 36.1 Attribute

acc_label

acc_text

accelerator

bgc

borderraster

borderstyle

borderwidth

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

filled

firstrecord

focus

focus_on_click

function

groupbox

height

help

index

label

lastrecord

layoutbox

mapped

member[l]

membercount

menu

model

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[l]

recordcount

scope

sensitive

sizeraster

statushelp

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 36.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| bgc | Hintergrundfarbe des Objekts. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. Nur aktiv, wenn *.borderwidth > 0*. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a). |
| borderwidth | Randbreite des Objekts. |
| cursor | Cursorvariante des Objekts (unter Motif nicht verfügbar). |
| fgc | Vordergrundfarbe des Objekts. |
| filled | Definiert ob das Objekt mit der Hintergrundfarbe ausgefüllt wird (*true*) oder nicht. |
| menu | Zum Objekt gehörendes Menü (Popup-Menü); wird unter Motif ignoriert, da hier kein Popup-Menü möglich ist. |
| options[enum] | Optionen des Objekts.<br>Index: *opt_center_toolhelp* (nur MS Windows). |

## 36.3 Hinweis für Dialog Manager unter Motif

Unter Motif ist das Attribut toolhelp nicht verfügbar (es wird keine Toolhelp angezeigt).

## 36.4 Beispiel

```
dialog Test

window Wn
{
  .width  319;
```

```
  .height 209;
  .title  "Testfenster";

  child rectangle Re
  {
    .xleft  54;
    .width  207;
    .ytop   33;
    .height 128;
    .borderwidth 3;
    .filled true;
  }
}
```

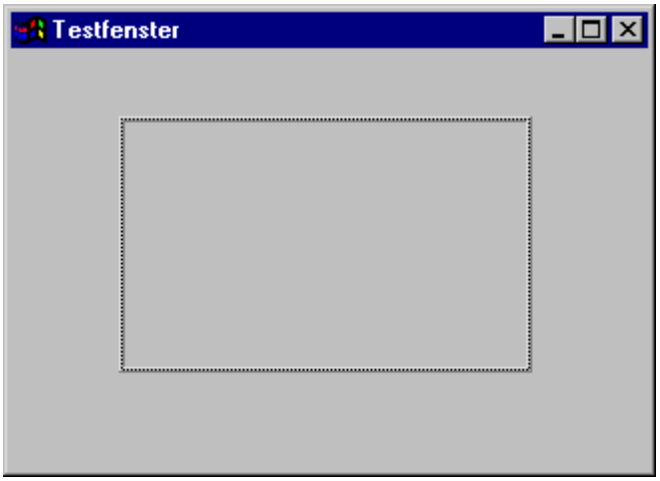



***Abbildung 35:*** *Fenster mit rectangle*

# 37 scrollbar

Eine ***scrollbar*** kann als analoges Anzeigeobjekt oder als analoger Einsteller verwendet werden. Sie kann horizontal oder vertikal ausgerichtet sein.

### Definition

```
{ export | reexport } { model } scrollbar { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*cut*

*extevent*

*focus*

*help*

*key*

*paste*

*scroll*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 37.1 Attribute

acc_label

acc_text

accelerator

arrows

bgc

class

control

cursor

curvalue

cut_pending

cut_pending_changed

dialog

direction

document[I]

external

external[I]

fgc

firstrecord

focus

function

groupbox

height

help

index

label

lastrecord

layoutbox

linemotion

mapped

maxvalue

member[I]

membercount

menu

minvalue

model

notepage

options[enum]

pagemotion

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

sizeraster

statushelp

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 37.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| arrows | Definiert, ob zwei Pfeile existieren (*true*) oder nicht (*false*). |
| curvalue | Position des Scrollbarsliders. |
| direction | Bestimmt, ob das Objekt horizontal oder vertikal ausgerichtet ist. |
| linemotion | Definiert um welchen Betrag sich curvalue beim zeilenweisen Scrollen ändert.<br>Siehe auch „Scrollbarattribute“ in der „Attributreferenz“. |
| maxvalue | Definiert den maximal möglichen Wert von curvalue. |
| minvalue | Definiert den minimal möglichen Wert von curvalue. |
| options[enum] | Optionen des Objekts.<br>Index: *opt_center_toolhelp* (nur MS Windows) |
| pagemotion | Definiert um welchen Betrag sich curvalue beim seitenweisen Scrollen ändert.<br>Siehe auch „Scrollbarattribute“ in der „Attributreferenz“. |

## 37.3 Berechnung des Scrollbarsliders

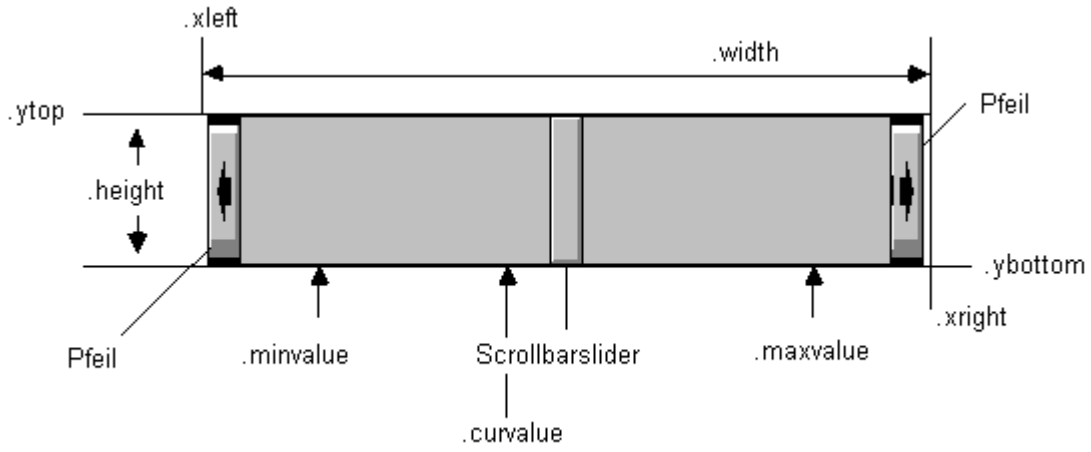



***Abbildung 36:*** *Attribute der scrollbar*

Die Größe des Scrollbar-Sliders wird aus den Werten der Attribute minvalue, maxvalue und page-motion berechnet. Wenn der Gesamtbereich über den der Slider fahren kann, eine Länge von 1 Pixel hat, so gilt:

Slidergröße = 1 * .pagemotion/(.maxvalue - .minvalue),

wobei bei Bedarf gerundet wird.

Die Größe des Sliders verhält sich also zum Darstellungsbereich wie .pagemotion zur Differenz von .maxvalue und .minvalue.

Diese Berechnung gilt sowohl für die Scrollbar als Objekt als auch für Scrollbars, die an ein Objekt gefügt sind.

Sie gilt jedoch nicht für sog. "scale widgets" (.arrows = false), wo sie eine feste Größe haben, die nicht zu beeinflussen ist.

## 37.4 Anmerkungen zur Scrollbar unter Microsoft Windows

Wenn das Attribut *..arrows = false* gesetzt wird, verwendet der Dialog Manager das Objekt "slider" anstatt des Objekts Scrollbar.

Slider werden gewöhnlich dazu verwendet, um ein allmähliches Anwachsen von Werten, wie z.B. Grade, Prozente usw., anzuzeigen (weitere Information finden Sie beim Attribut arrows in der „Attributreferenz“).

***Scrollbar***-Objekte mit *.arrows = false* verwenden die Hintergrundfarbe ihres Vaterobjekts und nicht ihre eigene.

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

## 37.5 Beispiel

```
dialog D

color Red "RED", grey(0);

window Wn
{
  .width  355;
  .height 180;
  .title  "Beispielfenster";

  child scrollbar Sb1
  {
    .fgc Red;
    .bgc Red;
    .xleft  66;
```

```
    .ytop   89;
    .arrows false;
    .direction 2;
    .curvalue  50;
  }
}
```

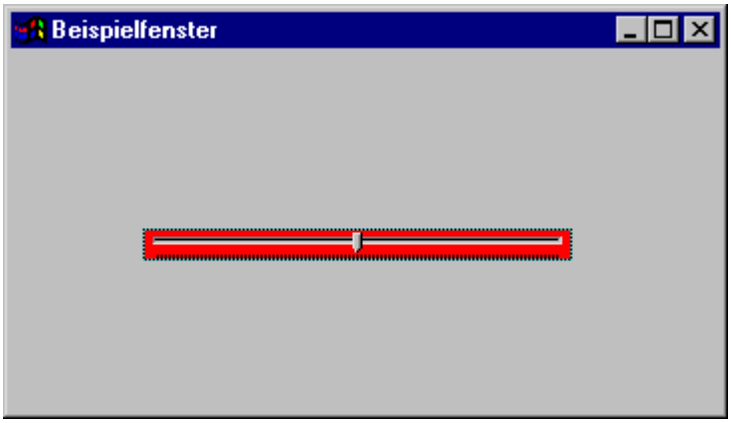



***Abbildung 37:*** *scrollbar*

# 38 setup

Mit Hilfe des ***setup***-Objektes kann aus dem Dialog heraus die Systemkonfiguration abgefragt werden. Mit dessen Attributen kann z.B. das Fenstersystem, die Bildschirmauflösung, das Betriebssystem, das Farbmodell usw. ermittelt werden.

### Definition

```
Dieses Objekt ist nicht definierbar, sondern nur in den Regeln abfragbar!
```

### Ereignisse

keine

### Kinder

keine

### Vater

keiner

### Menü

keins

### Siehe auch

C-Funktion DM_ParsePath im Handbuch „C-Schnittstelle - Funktionen“

## 38.1 Attribute

color

colorcount

colorcount[I]

color_type

color_type[I]

cursor

errfile

env[str]

envvar[str]

external

external[I]

font

.format

keyboard

language

logfile

mouse_buttons

opsys_string

opsys_type

.options[enum]

overridecursor

pointer_height

pointer_height[I]

pointer_width

pointer_width[I]

scale

.screen

screen[I]

screen_height

screen_height[I]

screen_width

screen_width[I]

screencount

searchpath

terminal

terminaltype

.tile

tiledpi

toolhelp

toolkit

toolkit_string

toolkit_version

tracefile

tracetime

tracing

version_string

winsys

winsys_string

winsys_version

xdpi

xdpi[l]

ydpi

ydpi[l]

## 38.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| color | Farbvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| colorcount | Anzahl der Farben. |
| colorcount[l] | Anzahl der Farben bei Multiscreendialogen für jeden Screen l (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |
| color_type | Abfrage des Terminaltyps (Schwarz-weiß-, Graustufen oder Farb-Terminal).<br>Mögliche Werte: *coltype_bw*, *coltype_color*, *coltype_grey*. |
| color_type[l] | Abfrage des Terminaltyps (Schwarz-weiß-, Graustufen oder Farb-Terminal) bei Multiscreendialogen für jeden Screen l (nur Motif).<br>Mögliche Werte: *coltype_bw*, *coltype_color*, *coltype_grey*.<br>Näheres zu Multiscreendialogen siehe Ressource display. |
| cursor | Cursorvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| errfile | Pfad der Fehlerdatei (Umleitung von IDM-Fehlermeldungen in eine Datei).<br>Siehe auch Kapitel „Optionen zur Tracefilesteuerung“. |
| env[str] | Umgebungsvariablen des Systems einschließlich der über die Option **-IDMenv** gesetzten Werte. |
| envvar[str] | Umgebungsvariablen des Systems. |

| Attribut | Beschreibung |
|---|---|
| font | Schriftvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| .format | Formatvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| keyboard | Acceleratorvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| language | Sprachvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| logfile | Pfad des Logfiles.<br>Siehe auch Kapitel „Optionen zur Tracefilesteuerung“. |
| mouse_buttons | Anzahl der logischen Maustasten. |
| opsys_string | Enthält die Betriebssystemkennung. |
| opsys_type | Enthält die Betriebssystemart, z.B. *os_nt*, *os_unix*. |
| .options[enum] | Optionen des Objekts.<br>Index:<br>*opt_balloon_toolhelp* – legt fest, ob die Toolhelp als Sprechblase (*true*, Default) oder als Rechteck angezeigt wird. Erst ab Version A.05.01.c unter MS Windows verfügbar. |
| overridecursor | Globale Setzung eines temporären Cursors. |
| pointer_height | Maximale Mauscursorhöhe in Pixel. |
| pointer_height[l] | Maximale Mauscursorhöhe in Pixel bei Multiscreendialogen für jeden Screen l (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |
| pointer_width | Maximale Mauscursorbreite in Pixel. |
| pointer_width[l] | Maximale Mauscursorbreite in Pixel bei Multiscreendialogen für jeden Screen l (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |
| scale | Eingestellte Systemvergrößerung. |
| .screen | Screen-Nummer des Default Screen bei Multiscreendialogen (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |
| screen[l] | Liefert die Nummer des l-ten Screens bei Multiscreendialogen (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |

| Attribut | Beschreibung |
|---|---|
| screen_height | Bildschirmhöhe in Pixel. |
| screen_height[I] | Bildschirmhöhe in Pixel bei Multiscreen- oder Multimonitordialogen für jeden Screen I (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |
| screen_width | Bildschirmbreite in Pixel. |
| screen_width[I] | Bildschirmbreite in Pixel bei Multiscreen- oder Multimonitordialogen für jeden Screen I (nur Motif und Windows). Näheres zu Multiscreendialogen siehe Ressource display. |
| screencount | Anzahl der verfügbaren Screens bei bei Multiscreen- oder Multimonitordialogen (nur Motif und Windows). Näheres zu Multiscreendialogen siehe Ressource display. |
| searchpath | Suchpfad, der für Dialog-, Modul-, Interface- und Binär-Dateien bei Imports mit **use** verwendet wird, erfragen und umsetzen. |
| terminal | Art des Terminals bei AlphaWindows. |
| terminaltype | Typ des Terminals bei AlphaWindows. |
| .tile | Bildvariante des Dialogs.<br>Siehe auch Kapitel „Zugriff auf Varianten von Ressourcen“. |
| .tiledpi | Wert DPI für die die Bilder der Anwendung entworfen wurden<br>Siehe auch Kapitel #######. |
| toolkit | Toolkitart des Systems, z.B. *toolkit_motif*, *toolkit_windows*. |
| toolkit_string | Toolkit des Systems. |
| toolkit_version | Toolkitversion des Systems. |
| tracefile | Pfad des Tracefiles.<br>Siehe auch Kapitel „Optionen zur Tracefilesteuerung“. |
| tracetime | Art der Zeitausgabe in das Tracefile.<br>Siehe auch Kapitel „Optionen zur Tracefilesteuerung“. |
| tracing | An- und Abschalten des Tracings zur Laufzeit.<br>Siehe auch Kapitel „Optionen zur Tracefilesteuerung“. |
| version_string | IDM-Versionsstring. |
| winsys | Genutzte Fenstersystemart, z.B. *winsys_windows*, *winsys_x11*. |

| Attribut | Beschreibung |
|---|---|
| winsys_string | Abfrage des Fenstersystems. |
| winsys_version | Abfrage der Fenstersystemversion. |
| xdpi | Abfrage der DPI in X-Richtung. |
| xdpi[I] | Abfrage der DPI in X-Richtung bei Multiscreendialogen für jeden Screen I (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |
| ydpi | Abfrage der DPI in Y-Richtung. |
| ydpi[I] | Abfrage der DPI in Y-Richtung bei Multiscreendialogen für jeden Screen I (nur Motif). Näheres zu Multiscreendialogen siehe Ressource display. |

### 38.2.1 Zugriff auf die Systemkonfiguration

Das jeweilige Betriebssystem oder die Fenstersystemvariante kann über die Attribute opsys_string, opsys_type, toolkit, toolkit_string, toolkit_version, winsys, winsys_string und winsys_version abgefragt werden.

Die Attribute env[str] und envvar[str] dienen zum Zugriff auf die Umgebungsvariablen des Systems bzw. des Dialogs.

Informationen über das verwendete Display können über die Attribute colorcount, color_type, .screen, screencount, screen_height, screen_width, terminal, terminaltype, xdpi und ydpi erfragt werden. Soweit diese Attribute indiziert vorhanden sind (siehe „Attribute“), dienen Sie zur Unterstützung von Multiscreendialogen unter Motif.

Informationen über den Mauscursor können zusätzlich mit den Attributen pointer_height und pointer_width abgefragt werden (indiziert für Multiscreendialoge vorhanden).

### 38.2.2 Zugriff auf Umgebungsvariablen

Über das Setup-Objekt lassen sich die Umgebungsvariablen des Programms setzen. Dazu existieren zwei Attribute: *.env[]* und *.envvar[]*.

In *.env[]* sind alle Umgebungsvariablen enthalten, wobei die über die Option **-IDMenv** gesetzten Werte die in der Umgebung gesetzten Werte überschreiben.

Die mittels der Option **-IDMenv** gesetzten Variablen sind nur für IDM-Funktionen gültig. Über das Attribut *.envvar[]* hingegen erhält man nur die Umgebungsvariablen.

Die Attribute *.env[]* und *.envvar[]* sind setzbar und abfragbar. Die Indizierung erfolgt über den Namen der Umgebungsvariablen. Beim Erfragen erhält man ein `fail` wenn die Umgebungsvariable nicht gesetzt ist.

Weiter ist es möglich sich alle mittels **-IDMenv** gesetzten Variablen anzusehen. Dazu dient das Attribute *.count[]* indiziert über *.env,* welches als Rückgabewert die Anzahl der IDM-Umgebungsvariablen

liefert. Nun kann *.env[]* über den Zahlenbereich *1 ... .count[.env]* indiziert werden, um die Namen der Umgebungsvariablen zu ermitteln.

**Beispiel**

```
!! Auslesen der PATH-Variable
print setup.envvar["PATH"];
!! Setzen einer IDM-Umgebungsvariablen
setup.env["DIALOGPATH"] := "/usr/local/idm/examples";
load("DIALOGPATH:draw.dlg");
!! Auslesen aller IDM-Variablen
for I:=1 to setup.count[.env] do
  print setup.env[I]+"="+setup.env[setup.env[I]];
endfor
!! Loeschen der Variablen
setup.env["DIALOGPATH"] := null;
!! Setzen einer ungesetzten Umgebungsvariablen
if fail(setup.env["SERVER_OPT"]="") then
  setup.env["SERVER_OPT"] := "true";
endif
```

**Hinweis**

In einem Prozess gesetzte Umgebungsvariablen bewirken keine Änderung der Umgebungsvariablen der Vaterprozesse (z.B. die Shell) oder bereits existierender Kindprozesse . Erst neu gestartete Kindprozesse können die veränderten Umgebungsvariablen bekommen.

### 38.2.3 Zugriff auf Varianten von Ressourcen

Außer über die Kommandozeilenoptionen können die Varianten verschiedener Ressourcen auch zur Laufzeit abgefragt bzw. gesetzt werden. Dies geschieht über die entsprechenden Attribute des setup Objekts.

Die Attributnamen sind dabei sprechend gewählt; so wird das Attribut color zum Zugriff auf die Farbvarianten, language für die Sprachvarianten (text-Ressource) usw. verwendet.

Folgende Attribute sind verfügbar:

color, cursor, font, .format, keyboard, language und tile.

Im folgenden Kapitel ist ein Beispiel für die Text-Ressource (Sprachvariante) dargestellt.

Der Mauscursor kann mittels des Attributs overridecursor für alle Dialogobjekte global gesetzt werden. Das Objekt Messagebox ignoriert allerdings .overridecursor und zeigt immer den für die Messagebox definierten Cursor oder den Default-Cursor an.

**Beispiel für die Anzeige des Overridecursors an einer Messagebox**

```
Msgbox1.cursor := setup.overridecursor;
querybox(Msgbox1);
```

```
Msgbox1.cursor := null;
```

#### 38.2.3.1 Beispiel anhand der Text-Ressource

Sie können die Variante einer Text-Ressource auch zur Laufzeit ändern, und zwar in der

- Regelsprache
- Programmiersprache

Bisher wurde die Sprache mit Hilfe der Kommandozeilen-Option **-lDMlanguage** beim Start einer Anwendung festgelegt.

**Regelsprache**

```
setup.language := <Variantennummer der gewünschte Sprache>
```

**Beispiel**

```
text "Language"  // English (default)
{
  1: "Sprache";  // German
}

on MENUITLANGUAGE select
{
  setup.language := 1;
}
```

#### 38.2.3.2 C- bzw. COBOL-Schnittstelle

Sie müssen dem Setup-Objekt die entsprechende Variantennummer der gewünschten Sprache mit DM_SetValue zuweisen.

### 38.2.4 Optionen zur Tracefilesteuerung

Um im ISA Dialog Manager das Tracing zu beeinflussen sind die folgenden Attribute verfügbar:

errfile, logfile, tracefile, tracetime und tracing.

**Siehe auch**

Kapitel „Tracing (Ablaufverfolgung)“ im Handbuch „Entwicklungsumgebung“

# 39 spinbox

Eine ***spinbox*** ist ein Containerobjekt, das aus zwei Pfeiltasten besteht und **genau ein** Kindobjekt enthält, bei dem es sich um einen einzeiligen ***edittext*** oder ***statictext*** handeln **muss**, der zur Anzeige des momentan eingestellten Wertes dient. Bei einer Wertänderung der ***spinbox*** – die entweder durch Betätigung einer Pfeiltaste oder durch eine Wertzuweisung erfolgt – wird der Inhalt des Anzeigefeldes automatisch aktualisiert.

### Definition

```
{ export | reexport } { model } spinbox  { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*cut*

*extevent*

*help*

*key*

*paste*

*scroll*

### Kinder

document

**1** ***edittext*** oder **1** ***statictext***

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

**Methoden**

:delete()

:exchange()

:find()

:insert()

## 39.1 Attribute

acc_label

acc_text

activeitem

borderstyle

child[I]

childcount

class

control

curvalue

cut_pending

cut_pending_changed

deltavalue

dialog

direction

document[I]

external

external[I]

focus

firstchild

firstrecord

font

height

itemcount

index

label

lastchild

lastrecord

layoutbox

mapped

maxvalue

member[I]

membercount

minvalue

model

notepage

options[enum]

parent

real_height

real_visible

real_width

record[I]

recordcount

scope

sensitive

statushelp

style

text[I]

tile

tilestyle

toolbar

toolhelp

userdata

visible

width

window

wrap

xraster

yraster

## 39.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| activeitem | Index des aktuell angezeigten Textes (nur bei *.style = string*). |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a).<br>Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |
| curvalue | Gibt den momentan angezeigten Wert an (nur bei *.style <> string*). |
| deltavalue | Bestimmt den Differenzwert, um den curvalue pro Schritt erhöht bzw. erniedrigt werden soll (nur bei *.style <> string*). |
| direction | Ausrichtung der Spinbox-Buttons.<br>*1* – Vertikale Ausrichtung (default).<br>*2* – Horizontale Ausrichtung. |
| itemcount | Anzahl der Einträge des Objekts (nur bei *.style = string*). |
| maxvalue | Maximal angezeigter Wert (nur bei *.style <> string*). |
| minvalue | Minimal angezeigter Wert (nur bei *.style <> string*). |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>*opt_center_toolhelp* (nur MS Windows). |
| style | Definiert, was in der Spinbox angezeigt werden soll.<br>*integer* – Integerwerte (default).<br>*string* – Texte.<br>*void* – anwendungsgesteuertes Spinning. |

| Attribut | Beschreibung |
|---|---|
| text[l] | Definiert die einzelnen anzuzeigenden Texte des Objekts.<br>(nur bei *.style = string*)<br>Bitte beachten Sie auch das Kapitel „Wertsetzung an der Spinbox“. |
| tile | Definiert das Hintergrundbild des Objekts |
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |
| wrap | Steuert, ob ein zirkuläres Spinning erfolgt (Default) oder beim Erreichen der Unter- bzw. der Obergrenze das Spinning abgebrochen wird. |

### 39.2.1 Wertsetzung an der Spinbox

Die Spinbox bestimmt selbst den Wert der Attribute .content bzw. .text am Kindobjekt. Aus diesem Grund sollten diese Attribute **nicht** am Kindobjekt geändert werden. Stattdessen sollten immer .activeitem und .text (.style = string) bzw. .curvalue (.style = integer) an der Spinbox gesetzt werden.

Insbesondere ist es zu vermeiden, .content bzw. .text des Kindobjekts auf einen inkonsistenten Wert zu setzen, da hier der angezeigte Text zufällig sein kann.

#### Achtung

Das Ändern von .activeitem und .text bzw. .curvalue an der Spinbox ändert immer auch .content bzw. .text am Kindobjekt! Deshalb ist es unter allen Umständen zu vermeiden diese Attribute zu ändern, während der Anwender den Text bearbeitet (Kindobjekt besitzt den Focus). Während dieser Textänderung durch den Anwender ist auch die einzige Situation gegeben, in der eine direkte Setzung von .content/.text am Kindobjekt sinnvoll sein kann.

## 39.3 Anmerkungen für Dialog Manager mit Microsoft Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

## 39.4 Beispiel

```
dialog Spinbox
{
}
color White "WHITE", grey(0);
window Wn
{
  .xleft 73;
  .width 260;
```

```
.ytop 29;
.height 222;
.title "Testfenster";

child spinbox Sp1
{
  .xleft 62;
  .ytop 64;
  .curvalue 1;

  child edittext
  {
    .xauto 0;
    .xleft 0;
    .xright 0;
    .yauto 0;
    .ytop 0;
    .ybottom 0;
  }
}

child spinbox Sp2
{
  .xleft 62;
  .width 100;
  .ytop 113;
  .style string;
  .maxvalue 7;
  .text[1] "Montag";
  .text[2] "Dienstag";
  .text[3] "Mittwoch";
  .text[4] "Donnerstag";
  .text[5] "Freitag";
  .text[6] "Samstag";
  .text[7] "Sonntag";
  .activeitem 1;

  child edittext Et2
  {
    .xauto 0;
    .xleft 0;
    .xright 0;
    .yauto 0;
    .ytop 0;
    .ybottom 0;
  }
```

```
    }
}
```

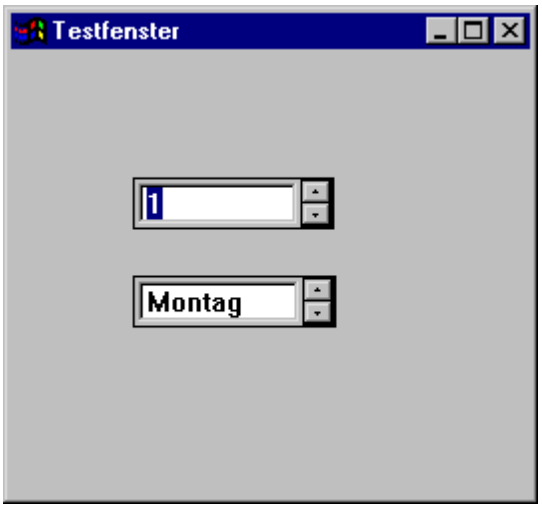



***Abbildung 38:*** *Spinbox*

# 40 splitbox

Das Objekt ***splitbox*** ist ein Hilfsobjekt (ähnlich einer ***groupbox***), um einen Bildschirmausschnitt in Bereiche zu unterteilen. Alle Bereiche werden durch Splitbars voneinander getrennt, die interaktiv vom Benutzer mit der Maus verschoben werden können, um so eine neue Aufteilung der Bereiche zu erzielen. Jedem sichtbaren Kind einer ***splitbox*** wird ein Bereich zugewiesen, in dem sich nur dieses Kind befinden kann, und dessen Größe und/oder Sichtbarkeit sich nach der Größe des ihm zugewiesenen Splitbereichs richtet. Möchte man in einem Splitbereich mehrere Kinder platzieren, so muss dort eine ***groupbox*** oder ***layoutbox*** als Kind benutzt werden, die dann die gewünschte Anzahl von Kindern aufnimmt. Mit einer ***groupbox*** als Kind können Splitbereiche auch mit virtuellen Größen ausgestattet werden.

### Definition

```
{ export | reexport } { model } splitbox { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Rasterattribute>
  <Hierarchieattribute>
  <Hierarchieattribut>
  <Objektspezifische Attribute>
}
```

Für jeden Splitbereich kann der Anwender seine gewünschte Größe festlegen. Außerdem können Minimal- und Maximalwerte pro Splitbereich angegeben werden, die interaktiv (verschieben einer Splitbar mit der Maus) nicht unter- bzw. überschritten werden können.

Die Ausrichtung der Splitbars kann entweder horizontal oder vertikal erfolgen.

### Ereignisse

*cut*

*extevent*

*help*

*key*

*paste*

*resize*

*select*

### Kinder

***canvas***

***checkbox***

***control***

document

***edittext***

***groupbox***

***image***

***layoutbox***

***listbox***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***rectangle***

***record***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***tablefield***

transformer

***treeview***

**Vater**

***dialog***

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 40.1 Attribute

acc_label

acc_text

accelerator

active

barwidth

bgc

bordercolor

borderraster

borderstyle

borderwidth

childcount

control

cursor

cut_pending

cut_pending_changed

dialog

direction

external

external[I]

fgc

firstchild

firstrecord

focus

focus_on_click

font

function

groupbox

height

help

label

lastchild

lastrecord

layoutbox

mapped

maxsize[I]

minsize[I]

model

module

navigable

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_size[I]

real_visible

real_width

record[I]

recordcount

reffont

scope

self

sensitive

size[I]

sizeraster

source

statushelp

target

tile

tilestyle

toolhelp

userdata

visible

width

window

xauto

xleft

xraster

xright

yauto

ybottom

yraster

ytop

## 40.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| barwidth | Breite der Splitbars. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“.<br>Nur aktiv, wenn *.borderwidth > 0*. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a). |
| direction | Ausrichtung der Splitbars.<br>*1* – vertikal.<br>*2* – horizontal. |
| maxsize[l] | Maximale Größe des Splitbereichs l.<br>Siehe auch Kapitel „Größenbestimmung der Splitbereiche“ und „Besonderheiten“. |
| minsize[l] | Minimale Größe des Splitbereichs l.<br>Siehe auch Kapitel „Größenbestimmung der Splitbereiche“ und „Besonderheiten“. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>*opt_center_toolhelp* (nur MS Windows). |

| Attribut | Beschreibung |
|---|---|
| real_size[I] | Momentan wirklich vorhandene Größe des Splitbereichs I.<br>Siehe auch Kapitel „Größenbestimmung der Splitbereiche“ und „Besonderheiten“. |
| size[I] | Größe der einzelnen Splitbereiche I.<br>Siehe auch Kapitel „Größenbestimmung der Splitbereiche“ und „Besonderheiten“. |
| tile | Definiert das Hintergrundbild des Objekts |
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |

### 40.2.1 Größenbestimmung der Splitbereiche

Die Größe der einzelnen Splitbereiche wird über das Attribut size[I] angegeben.

Soll ein Splitbereich durch den Anwender durch Verschieben der Splitbar eine minimale bzw. maximale Größe nicht unter-/überschreiten können, ist dies in den Attributen minsize[I] bzw. maxsize[I] anzugeben. Dabei wird I = 0 als Defaultwert für alle Splitbereiche, die Werte I = 1 bis .childcount für den jeweiligen Splitbereich benutzt. Zum genauen Zusammenspiel dieser Attribute, insbesondere bei Größenänderungen durch den Anwender, siehe die folgenden Kapitel.

Die momentan vorhandene Größe eines Splitbereichs kann über real_size[I] abgefragt werden.

## 40.3 Besonderheiten

Das besondere Augenmerk im Zusammenhang mit der Splitbox ist in erster Linie auf das Layout und Geometriemanagement und der Verwendung von Randbedingungen wie eingeschaltetem Sizerastering oder der Benutzung der Minimal-/Maximalwerte zu legen.

### 40.3.1 Sizerastering

Im Gegensatz zu anderen Objekten, die die Benutzung von Attributen wie .xraster, .yraster und .reffont erlauben, wird bei der Splitbox das Rastergitter nicht auf das gesamte Objekt angewandt, sondern jeweils auf die einzelnen Splitbereiche. Damit sollen Rundungsprobleme vermieden werden, die unvermeidlich wären, wenn man einzelne Splitbereiche am globalen Rastergitter ausrichten müsste, und daneben noch Splitbars (deren Breite ja in Pixelwerten angegeben wird) in diesem Muster unterbringen müsste. Die Breiten der Splitbars ebenfalls in Rastergrößen anzupassen, erscheint als wenig attraktiv (es sieht einfach hässlich aus). Die Verwendung des Rastergitters nur lokal für das jeweilige Splitbereich löst dieses Problem. Darüber hinaus kann die bisher übliche Berechnungsvorschrift zur Pixelwertbestimmung vereinfacht werden. Üblich ist: Pixelwert = (Rasterwert – 1)*Raster. Für die Größenberechnung der Splitbereiche gilt Pixelwert = Rasterwert*Raster.

## 40.3.2 Größenberechung der Splitbereiche

Zur Berechnung der tatsächlichen Größen der Splitbereiche wird folgender Algorithmus verwendet.

1. Zunächst wird versucht, jedem einzelnen Bereich die Größe zuzuweisen, die im Attribut .size[I] angegeben ist.
2. Wird nach Schritt 1 festgestellt, dass dabei mehr Platz (Breiten der Splitbereiche plus Breiten der Splitbars) verbraucht wurde, als das Objekt durch die Definition der Geometrieattribute wie .width, .height, .xauto, .yauto, usw. zur Verfügung stellt, werden Splitbereiche unter Berücksichtigung der Minimalwerte anfangend von rechts bzw. unten in den Größen beschnitten, bis die Platzeinschränkung erfüllt wird. Wird dieses nicht erreicht, weil z.B. die Summe aller Minimalwerte (minsize[I]) bereits jeglichen Rahmen sprengt, werden Splitbereiche nochmals von rechts bzw. unten durchgegangen. Diesmal werden diese gekürzt ohne Rücksicht auf Minimalwerte. Auf der rechten bzw. unteren Seite können dabei eine Reihe von Splitbereichen mit der Größe 0 entstehen. Man sieht also nur noch übrig gebliebene Splitbars. Am Ende dieses Prozesses kann es passieren, dass obwohl alle Splitbereiche auf Null reduziert wurden, es immer nicht genug Platz zur Verfügung gibt (Summe der Breiten der Splitbars ist größer als die Ausdehnung der Splitbox). In diesem Fall werden die Splitbereiche samt deren Splitbars, für die der Platz nicht ausreicht, nach rechts bzw. unten außerhalb des sichtbaren Bereichs verschoben.
3. Wird nach Schritt 1 festgestellt, dass dabei zu wenig Platz verbraucht wurde, als das was von der Splitbox durch die Definition der Geometrieattribute zur Verfügung gestellt wurde, werden Splitbereiche unter Berücksichtigung der Maximalwerte von rechts bzw. unten soweit vergrößert, bis die gesamte Fläche der Splitbox ausgefüllt ist. Wird dieses Ziel auf Anhieb nicht erreicht, weil z.B. die Summe aller Maximalwerte und der Breiten der Splitbars bereits kleiner ist als die Ausdehnung der Splitbox, wird der letzte Splitbereich unter Missachtung des ihm zugewiesenen Maximalwertes, soweit notwendig, vergrößert.

Man beachte, dass während dieses Prozesses die Attribute .size[I], .minsize[I], maxsize[I] nicht verändert werden. Sprich, dort stehen immer noch die Wunschwerte des IDM-Programmierers, die lediglich aufgrund der widrigen Bedingungen nur ansatzweise berücksichtigt werden konnten. Diese Eigenschaft ermöglicht eine Neuberechnung der tatsächlichen Splitbereichbreiten nach jedem Verändern der äußeren Maße der Splitbox. Wird z.B. die Splitbox mit .xauto=0 und .yauto=0 an einem Fenster angebunden, so kann beim interaktiven Verkleinern des Fensters und damit auch der Splitbox beobachtet werden, wie die Splitbereiche nacheinander mit und dann auch ohne Berücksichtigung der Minimalwerte verkleinert werden. Wird das Fenster wieder vergrößert, so kehren die Splitbars wieder in ihre Ausgangspositionen zurück.

Die Werte im .size[I] werden lediglich verändert, wenn diese im Widerspruch zu den Minimal-/Maximalwerten in minsize[I]/maxsize[I] stehen, wobei die Werte in minsize[I] für den IDM höhere Priorität haben als die Werte in maxsize[I].

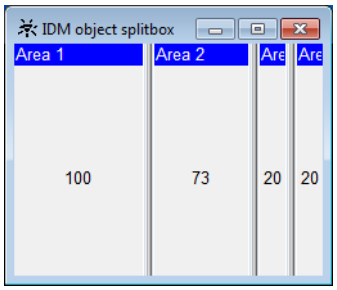
Sollte es zu den in Schritten 2 und 3 beschriebenen Übertretungen der Minimal-/Maximalwerte kommen, können die betroffenen Splitbars interaktiv nicht mehr verschoben werden, da diese ohnehin schon an eigentlich nicht erlaubten Positionen stehen. Durch das Anfassen und dann Loslassen der Splitbars wird jedoch die Größe, und zwar die in .size[I]-Vektor, der betroffenen Splitbereiche auf neue Werte gesetzt. Dadurch werden auch EM_resize-Events ausgelöst, obwohl rein optisch sich

nichts verändert hat; aber eben die Attributwerte in .size[I]. In jedem Fall wird der IDM-Programmierer dadurch informiert, dass seine Wunschwerte im size[I]-Vektor durch den Endanwender verändert worden sind. Wird versucht, dieselbe Splitbar nochmals zu verschieben, wird rein optisch sich wieder nichts ändern, da die von den Minimal-/Maximalwerten herrührenden Einschränkungen immer noch gelten. Diesmal werden jedoch keine weiteren EM_resize-Events verschickt, da die Werte in size[I] nach dem ersten Versuch bereits passend gesetzt wurden.

Eine Besonderheit ist beim Sizerastering zu beachten. Da es in der Praxis so gut wie unmöglich ist, dass alle Splitbereiche bei dem oben beschriebenen Anpassungsprozeß genau auf die Rastergröße gebracht werden können, existiert in der Regel (höchstens) ein Splitbereich, dessen Größe nicht auf das Vielfache von Rasterwerten passt. Es handelt sich dabei um den Splitbereich, der als letzter von dem oben beschriebenen Algorithmus angepasst wird.

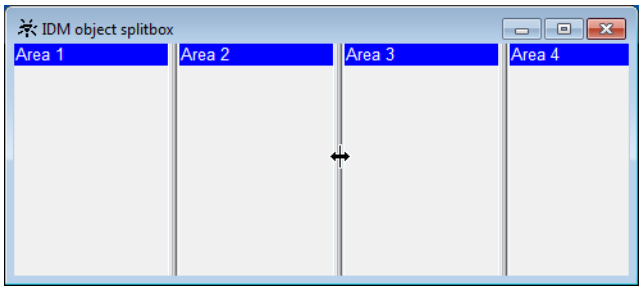
Als Beispiel betrachten wir folgende Situation:

Es sei eine Splitbox mit .direction=1, .size[0]=10, minsize[0]=2 gegeben. Sizerastering ist eingeschaltet und am Vaterobjekt ist .xraster=10 gesetzt. Die Splitbox ist mit .xauto=0 und .yauto=0 an dem Vaterfenster angebunden. In der folgenden Abbildung ist diese Splitbox dargestellt. Man sieht, dass durch die äußeren Einflüsse nicht alle Werte aus dem .size[I]-Vektor erfüllt werden können. Die Zwei letzten Splitbereiche haben bereits ihre Minimalgrößen erreicht und zwar 20 Pixel. Der erste Splitbereich hat seine gewünschte Größe von 100 Pixeln. Der Splitbereich mit der Nummer 2 ist derjenige, der von dem Algorithmus als letzter verändert wurde, und nun nicht mehr in das Rastergitter passt.



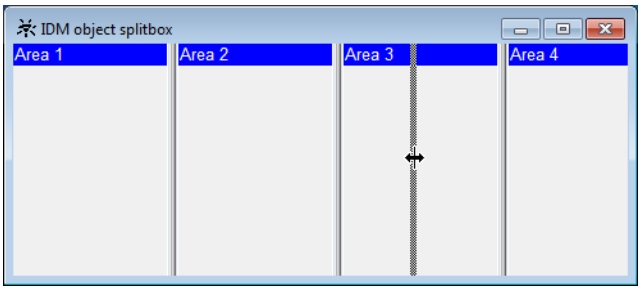

### 40.3.3 Größenänderung (resize) durch den Benutzer

Ein typischer Anwendungsfall einer Splitbox besteht darin, die Größe der Splitbereiche interaktiv zu verändern. Hierzu muss der Mauszeiger über einer der Splitbars positioniert werden. Wie in der nachfolgenden Abbildung zu sehen ist ändert sich dabei das Aussehen des Mauszeigers.

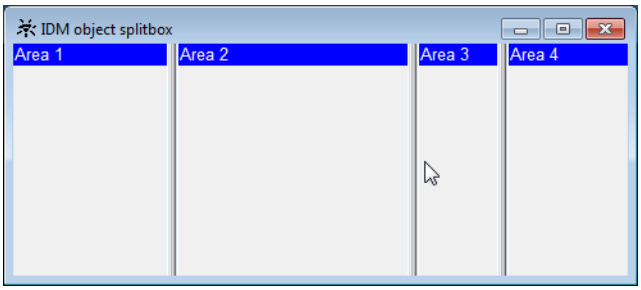


Nun wird die linke Maustaste gedrückt und die Splitbar (in diesem Beispiel) nach rechts/links verschoben. Wie in der nachfolgenden Abbildung gezeigt, wird dabei als optisches Feedback ein teils durchsichtiger Balken zusammen mit dem Mauszeiger mit bewegt. Wird beim Verschieben irgendeine der Minimal-/Maximalbedingungen verletzt (etwa die Maximalgröße des zweiten Splitbereichs überschritten oder die Minimalgröße des dritten Splitbereichs unterschritten) bleibt der gezogene Balken an der zuletzt gültigen Position stehen.

Unter MICROSOFT WINDOWS kann die Verschiebeaktion durch das Betätigen der `Escape`-Taste abgebrochen werden.




Wenn die gewünschte Position erreicht wurde, kann die linke Maustaste losgelassen werden, wodurch die Splitbox dazu veranlasst wird, die betroffenen Splitbereiche auf neue Größen zu bringen. Die folgende Abbildung dokumentiert das Endergebnis der Aktion.




Nun werden auch maximal zwei resize-Events verschickt. Eins, um anzuzeigen, dass sich die Größe des zweiten Splitbereichs verändert hat (dabei ist in erster Linie die Änderung des Wertes im size[I]-Vektor relevant; ist dieser Wert unverändert geblieben, wird kein Event ausgelöst). Im thisevent.index ist der Index (einsbasiert) des Kindes aus dem betroffenen Splitbereich eingetragen. Der zweite resize-Event ist den Änderungen im dritten Bereich gewidmet. Entsprechend ist im thisevent.index der Index des dritten Kindes eingetragen.

### Hinweis

Wenn an dem Objekt .sizeraster auf true gesetzt ist, so wirkt sich dieses während der Verschiebeaktion zunächst gar nicht aus. Die Splitbar wird kontinuierlich mit dem Mauszeiger mit bewegt. Erst wenn die Splitbar losgelassen wird, schnappt sie auf die links/oben naheliegende Rasterposition ein, so dass die betroffenen Nachbarbereiche wieder auf die Rastergrößen gebracht werden.

## 40.4 Beispiel

Eine einfache Verwendung des splitbox-Objektes kann wie folgt aussehen:

```
dialog Dialog

window W
{
  .title  "splitbox-Demo";
  .width  400;
  .height 400;

  child splitbox Splbox
  {
    .xauto   0;
    .yauto   0;
    .xleft   5;
    .xright  5;
    .ytop    5;
    .ybottom 50;
    .size[0] 50;
    .minsize[0] 20;
    .maxsize[0] 100;

    child pushbutton Pb1
    {
      .text "in Bereich 1";
    }

    child statictext St1
    {
      .text  "in Bereich 2";
      .yauto -1;
      .xauto -1;
    }

    child groupbox Gb1
    {
      .xauto   0;
      .yauto   0;
      .xleft   4;
      .xright  4;
      .ytop    4;
      .ybottom 4;
    }

    child treeview Tv
    {
```

```
      .xauto   0;
      .yauto   0;
      .xleft   4;
      .xright  4;
      .ytop    4;
      .ybottom 4;
      .content[1]  "Eins";
      .content[2]  "a";
      .content[3]  "b";
      .content[4]  "c";
      .content[5]  "Zwei";
      .content[6]  "d";
      .content[7]  "Drei";
      .content[8]  "Vier";
      .content[9]  "e";
      .content[10] "f";
      .content[11] "g";
      .level[2]  2;
      .level[3]  2;
      .level[4]  2;
      .level[6]  2;
      .level[9]  2;
      .level[10] 2;
      .level[11] 2;
      .firstchar 1;
      .editable  true;
      .style[style_lines]   true;
      .style[style_buttons] true;
      .style[style_root]    true;
      .selstyle  multiple;
    }
  }

  on close { exit(); }
}
```

# 41 statictext (statischer Text)

Zur Definition des vom Benutzer nicht interaktiv veränderbaren Textes dient das Schlüsselwort **statictext**. Es können einzeilige Texte innerhalb eines Fensters positioniert und angezeigt werden.

### Definition

```
{ export | reexport } { model } statictext { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*cut*

*extevent*

*focus*

*help*

*key*

*paste*

*select*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***spinbox***

***splitbox***

***toolbar***

***window***

**Menü**

**Popup-Menü**

## 41.1 Attribute

acc_label

acc_text

accelerator

alignment

bgc

class

control

cursor

cut_pending

cut_pending_changed

depth

dialog

document[I]

external

external[I]

fgc

firstrecord

focus

focus_on_click

font

function

groupbox

height

help

index

label

lastrecord

layoutbox

mapped

member[I]

membercount

menu

model

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

sizeraster

statushelp

text

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 41.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| alignment | Ausrichtung des Textes im Objekt. |
| depth | Art der Darstellung des Objekts in einer Statusbar (erhöht oder vertieft). |
| height | Höhe des Objekts, bei Angabe von *0* wird die Höhe aufgrund des aktuellen Zeichensatzes berechnet. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>*opt_use_widget* (nur Motif).<br>*opt_center_toolhelp* (nur MS Windows). |
| text | Inhalt (Text) des Objekts, siehe auch Kapitel „Statictext als Kind einer Spinbox“. |
| width | Breite des Objekts, bei Angabe von *0* wird die Breite aufgrund des aktuellen Zeichensatzes berechnet. |

### 41.2.1 Statictext als Kind einer Spinbox

Ist ein Statictext ein Kindobjekt eines ***spinbox***-Objekts, darf das Attribut text nicht genutzt werden. Die Wertsetzung erfolgt ausschließlich über die entsprechenden Attribute des Spinbox-Objekts.

## 41.3 Beispiel

Text innerhalb eines Fensters:

```
window HAUPT
{
  .title "Testfenster";

  child statictext FixText
  {
    .xleft 10;
    .ytop 10;
    .text "Firma Meier";
  }
```

```
}
```

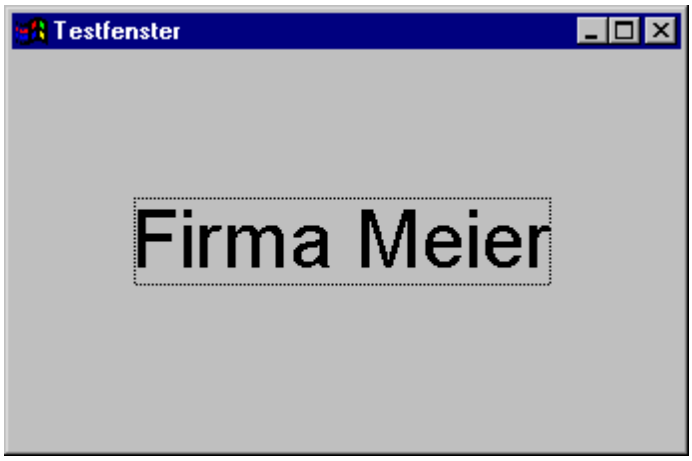



***Abbildung 39:*** *Statischer Text*

# 42 statusbar

Die ***statusbar*** dient zum Anzeigen von Statusinformationen und Hilfetexten. Sie wird an den unteren Rand des Fensters angehängt und verändert damit nicht die Größe des Nutzbereichs. Das Rahmenfenster wird durch die ***statusbar*** größer.

### Definition

```
{ export | reexport } { model } statusbar { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*extevent*

### Kinder

document

***image***

***progressbar***

***record***

***statictext***

transformer

### Vater

***module***

***window***

### Menü

keins

## 42.1 Attribute

acc_label

acc_text

borderstyle

child[I]

childcount

class

control

dialog

document[I]

external

external[I]

fgc

firstchild

firstrecord

font

helppos

index

label

lastchild

lastrecord

layoutbox

mapped

member[I]

membercount

model

module

parent

real_height

real_visible

real_width

record[I]

recordcount

sensitive

scope

sizeraster

tile

tilestyle

toolhelp

userdata

visible

vsb_visible

window

## 42.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a).<br>Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |
| helppos | Gibt an in welcher Position der Hilfetext statushelp darzustellen ist.<br>*0* - (Defaultwert) Hilfeinformation wird ganzzeilig dargestellt.<br>*n* - Der Hilfetext wird über alle Kinder bis zu dem n-ten Kind dargestellt.<br>Ist der Wert < *0* wird keine Hilfeinformation dargestellt. Dieses Attribut kann von Fenstersystem zu Fenstersystem variieren. Die Werte *0* und *-1* sind aber immer setzbar und werden entsprechend obiger Beschreibung angezeigt. |
| sizeraster | Wird zur Berechnung der Feldbreiten innerhalb der Statusbar verwendet.<br>*false* – (Default) Feldbreiten werden in Pixel angegeben. Die Breite der Trennung zwischen zwei Textfeldern besitzt eine systemabhängige Größe.<br>*true* - Feldbreiten werden in Rasterkoordinaten angegeben. Die Breite der Trennung zwischen zwei Textfeldern ist ein Raster. |
| visible | Steuert das Aus- und Einblenden der Statusbar. |

### 42.2.1 Hinweis zu den Geometrieattributen

Die eigentlichen Geometrie-Attribute (Breite, Höhe, Position) werden alle ignoriert, da die Position der Statusbar fest ist. Die Statusbar befindet sich am unteren Fensterrand unterhalb von der horizontalen Scrollbar, falls eine solche vorhanden ist. Sie erstreckt sich über die gesamte Fensterbreite und ihre Höhe wird aus dem angegebenen Zeichensatz bestimmt.

Die Statusbar wird nicht mit dem Fensterinhalt mitgescrollt, deshalb muss der Anwendungsprogrammierer darauf achten, dass die Information der Statusbar komplett sichtbar sein kann. Für die Geometrieberechnung des Fensters gehört die Statusbar nicht zum Fensterinhalt wie die Scrollbar, sondern zum Fensterrahmen.

Ausgewertet werden lediglich die Attribute .xraster und .yraster, da sie für die Berechnung der Position der Kinder dienen.

## 42.3 Verwendung der Statusbar

Statusbars können in Top-Level- und Kind-Fenstern verwendet werden. Besitzt ein Kind-Fenster keine sichtbare Statusbar, werden Statushilfetexte in der Statusbar des Vater-Fensters gezeigt (wenn vorhanden).

Die Hilfetexte werden in den jeweiligen Objekten mit dem Attribut statushelp gesetzt. Diese werden bei Auswahl eines Menüs oder bei anderen Objekten bei Überquerung mit der Maus gezeigt.

Als Kindobjekte sind nur statische Texte zugelassen. Das Attribut .depth des statischen Textes bestimmt, ob flach, vertieft oder erhöht dargestellt wird. Die Texte werden in der Kinder-Reihenfolge von links nach rechts mit der Breite .width gezeigt. Das letzte Kind wird bis ganz nach rechts verbreitert. Ist dies nicht erwünscht, kann ein ‘dummy’ angefügt werden.

Eine Besonderheit für die statischen Texte in der Statusbar ist die Möglichkeit, Texte mit "\t" zu positionieren.

Die folgende Graphik beschreibt das Aussehen einer Statusbar schematisch:

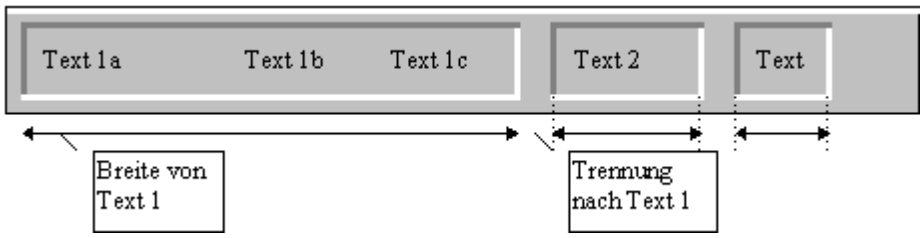



***Abbildung 40:*** *Schematische Darstellung einer Statusbar*

Eine Statusbar besteht aus bis zu 256 Textfeldern, die entweder direkt aneinander anschließen oder voneinander getrennt sein können. Vor dem ersten und hinter dem letzten Textfeld befindet sich eine "Trennung". Innerhalb eines Textfeldes ist der Text linksbündig dargestellt. Die "Trennung" zwischen zwei Textfeldern hat die Breite von einem Rasterpunkt, falls das Raster eingeschaltet ist; ansonsten hat sie eine systemabhängige Breite.

Die Statusbar kann auch einen Hilfetext darstellen, der alle oder mehrere Kinder überlappen kann, so dass wichtige Informationen weiterhin sichtbar bleiben können.


***Abbildung 41:*** *Hilfetextdarstellung in Statusbar*

## 42.4 Hinweis zu Unicode Zeichensätzen unter Windows

Ein ***Statictext***-Objekt innerhalb einer ***Statusbar*** stellt bei manchen ***Font***-Ressourcen an Stelle von Unicode-Zeichen das Ersatzzeichen dar. Dies liegt am ***Statusbar***-Objekt von Windows und kann vom IDM nicht erkannt werden, sodass in einem solche Fall eine falsche Breite berechnet wird. Das Problem tritt nicht unter Windows auf, wenn „Visual Styles“ aktiv sind.

## 42.5 Beispiel

Fenster mit ***statusbar***, die durch Einstellungen im Fenster verändert werden kann.

```
window W
{
  .width   400;
  .height  200;
  .vwidth  400;
  .vheight 200;
  .hsb_visible true;
  .vsb_visible true;
  .title   "statbartest";

  child statictext StHelppos
  {
    .xleft 200;
    .ytop  80;
    .text  ".helppos";
  }

  child statictext Sw
  {
    .yauto   -1;
    .ybottom 0;
    .text    "Text unten";
    .depth   1;
  }
  ...
  child statusbar S
  {
```

```
    child statictext St0
    {
      .width 0;
      .text  "text 0";
      .depth 0;
    }

    child statictext St1
    {
      .width 0;
      .text  "Text 1";
      .depth -1;
    }

    child statictext St2
    {
      .width 120;
      .text  "links\tmitte\trechts";
      .depth 1;
    }

    child statictext St3
    {
      .width 100;
      .text  "";
      .depth 0;
    }
  }
}
```

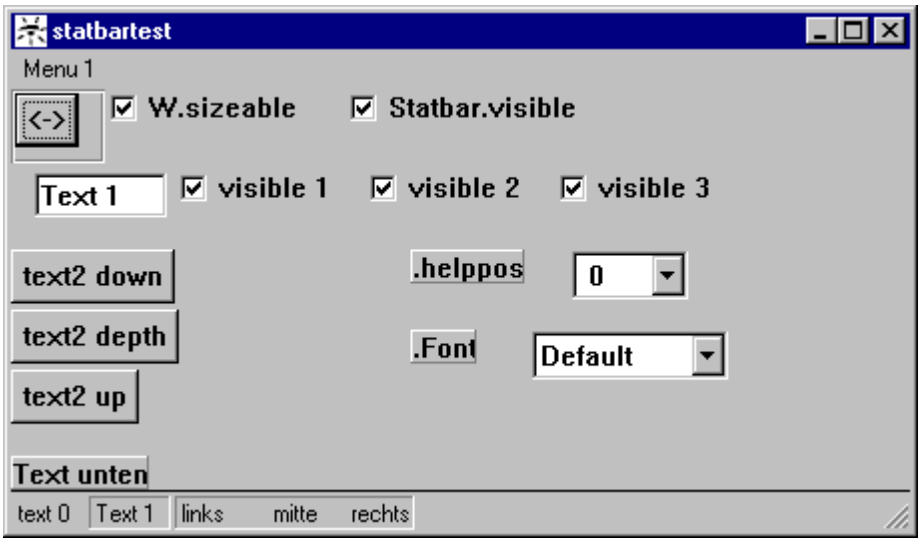



***Abbildung 42:*** *Fenster mit statusbar-*
***Abbildung 43:*** *window with statusbar*

# 43 subcontrol

OLE-Objekte, die als Server im IDM mit Hilfe des IDM-Objektes ***control*** genutzt werden können, haben in der Regel eine recht komplexe Struktur und können aus einer Reihe weiterer Kindobjekte bestehen. Zum Beispiel hat das OLE-Control „Word.Application“ eine Kollektion „Documents“, wo die Dokumente verwaltet werden. Diese Dokumente wiederum haben „Words“, „Sentences“, „Ranges“ usw. als eigene Kinder. Um auf diese Unterobjekte aus der Regelsprache zugreifen zu können, sprich Methoden aufrufen oder Attribute abfragen, wurde das Objekt ***subcontrol*** eingeführt. Das ***subcontrol*** repräsentiert somit ein OLE-Kindobjekt, das direkt oder indirekt von einem OLE-Server verwaltet wird. Der Ausgangspunkt für den Zugriff auf ein solches Objekt ist deswegen immer ein ***control***.

### Definition

```
{ export | reexport } subcontrol { <Bezeichner> }
{
  <Hierarchieattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

Keine

### Kinder

subcontrol

### Vater

control

subcontrol

### Menü

Keins

## 43.1 Attribute

connect

dialog

external

external[l]

firstrecord

firstsubcontrol

groupbox

label

lastrecord

lastsubcontrol

layoutbox

model

module

notepage

parent

record[I]

recordcount

scope

subcontrol[I]

subcontrolcount

toolbar

userdata

window

## 43.2 Implizite Erzeugung

Die herausragende Besonderheit der Subcontrols ist, dass diese Objekte implizit erzeugt werden können, ohne dass man vorher das Objekt statisch in einem Dialog definiert oder mit **create()** dynamisch erzeugt hat. Allerdings sollten bei Abfrage von dynamischen OLE Eigenschaften, welche ein Subcontrol erzeugen, unbedingt die Hinweise in Kapitel „Dynamische OLE-Eigenschaften und Subcontrols“ beachtet werden.

Vermutlich ist die implizite Verwendung dieser Objekte die in der Praxis am häufigsten eingesetzte. So kann man beispielsweise bei der Verwendung von Word als OLE-Server wie folgt auf die einzelnen Dokumente zugreifen:

### Beispiel

Sei ***Control***„Co“ irgendwo im Dialog definiert.

```
child control Co
{
  .mode mode_client;
  .name "Word.Application";
  .visible true;
```

```
  .connect true;
}
```

Dann könnte in einer Regel folgendes stehen:

```
rule OLETest ()
{
  variable object Doc;
  Doc := Co.Documents:Add();
  // Word besitzt eine Collection mit dem Namen Documents. Mit
  // Co.Documents wird auf dieses OLE-Objekt zugegriffen. Um das
  // Object im IDM ansprechen zu können, wird an dieser Stelle
  // ein Subcontrol mit dem Bezeichner "Documents" implizit
  // erzeugt, das als Kind von Control Co eingetragen wird.
  // Die :Add()-Methode ist ein Mitglied der Collection
  // "Documents". Deswegen kann sie nun auf das gerade eben
  // erzeugte Subcontrol angewandt werden. Dabei wird in Word
  // ein neues Document erzeugt, und als Rückgabewert ein Zeiger
  // auf das IDispatch-Interface dieses Dokuments
  // zurückgeliefert. Um ein entsprechendes Pendant auf der Seite
  // des IDM zu haben, wird auch hier ein Subcontrol als Kind
  // des ersten Subcontrols erzeugt, das selber mit dem
  // Word-Document verbunden ist. Dieses Objekt wird schliesslich
  // der Variablen "Doc" zugewiesen.

  // Nun kann das Subcontrol in Doc dazu benutzt werden, um
  // Methoden des OLE-Documents aufzurufen oder seine Properties
  // zu setzen/abzufragen.
  print Doc.FullName;
}
```

Generell gilt folgendes: Wird auf der Seite von OLE als Rückgabewert einer Methode oder Property ein Zeiger auf ein IDispatch-Interface zurückgeliefert oder wird dieser in einem Parameter an den IDM übergeben, so erzeugt der IDM an dieser Stelle ein Subcontrol, das mit dem IDispatch-Interface verbunden ist. D.h. das *.connect*-Attribut eines solchen Subcontrols ist bereits gleich *true*.

In der Regel wird der Bezeichner (Label) eines so erzeugten Objektes nicht gesetzt, so dass im Tracefile so etwas wie *„subcontrol Co.SUBCONTROL[2]“* erscheint. Eine Ausnahme bildet dabei der Zugriff auf die Properties, die OLE-Objekte zurückliefern. Da diese in der Regel Collections darstellen und dem IDM somit der Name bekannt ist, bekommt ein implizit erzeugtes Subcontrol in diesem Fall als Bezeichner den Namen der Property.

### Beispiel

```
Doc1 := Co.Documents:Add();
Doc2 := Co.Documents:Item(1);  // Annahme: In Word gab es zuvor
                               // keine geoeffneten Dokumente.
```

In diesem Beispiel werden in der ersten Zeile zwei Subcontrols erzeugt: Eines mit dem Bezeichner „Documents“, ein anderes mit dem Bezeichner „SUBCONTROL“. Der Bezeichner des zweiten Subcontrols erklärt sich dadurch, dass der IDM beim Verarbeiten der Add()-Methode keine weiteren Informationen zur Verfügung hat. In der zweiten Zeile wird nur noch ein Subcontrol mit dem Bezeichner „SUBCONTROL[2]“ erzeugt. Das Erzeugen des Subcontrols für Documents entfällt, da der IDM an dieser Stelle erkennt, dass das Objekt bereits existiert. Es entsteht deswegen folgende Situation. Auf der Seite von OLE haben wir zwei Objekte: zum einem die Collection „Documents“ zum anderen das leere Dokument. Auf der Seite vom IDM haben wir drei Objekte: ein Subcontrol für die Collection „Documents“ und zwei Subcontrols in den Variablen Doc1 und Doc2, die mit dem leeren Dokument auf der OLE-Seite verbunden sind. Es bleibt also dem IDM-Programmierer überlassen, durch guten Programmierstil eine Flut von überflüssigen Subcontrols zu vermeiden.

Ein besorgter Leser mag sich jetzt vielleicht fragen, was denn nun mit all den Subcontrols geschehen muss, wenn diese nicht länger gebraucht werden. Sehen Sie dazu die Erläuterungen in Kapitel „Garbage Collection“.

## 43.3 Dynamische OLE-Eigenschaften und Subcontrols

Beim Aufruf einer dynamischen Eigenschaft eines OLE-Objekts kann es zum Anlegen eines neuen Subcontrols kommen, wenn diese Eigenschaft (z.B. .„ActiveSheet“ bei MICROSOFT EXCEL) ein Objekt zurückgibt. Dies ist erst einmal unproblematisch. Bei einem erneuten Aufruf dieser dynamischen Eigenschaft referenziert der IDM nun aber nicht das neue, jetzt gelieferte Objekt, sondern das bereits vorhanden und angelegte Subcontrol, welches der IDM quasi „cached“.

Sollte dieses Caching unerwünscht bzw. problematisch sein, kann es wie folgt umgangen werden:

- Nach dem Zugriff auf eine OLE-Eigenschaft, welche ein Subcontrol erzeugt, kann dieses Subcontrol explizit mit **:destroy()** zerstört werden, wenn es nicht mehr benötigt wird (vor dem nächsten Zugriff auf diese OLE-Eigenschaft!)
- Nach dem Zugriff auf eine OLE-Eigenschaft, welche ein Subcontrol erzeugt, kann das Attribut *.label* dieses Subcontrols auf "" (Leerstring) gesetzt werden. Hierdurch bleibt das Subcontrol-Objekt weiterhin verbunden und verfügbar. Es wird im Zuge der normalen Garbage Collection (siehe Kapitel „Garbage Collection“) gelöscht. Bei einem erneuten Zugriff auf die OLE-Eigenschaft wird ein neues Subcontrol-Objekt anlegen, da das alte nicht mehr gefunden wird (Die interne Suche erfolgt über *.label*).

Ein Setzen von *.connect* des betroffenen Subcontrols auf *false* ist hingegen keine Lösung, da das Subcontrol erhalten bleibt und lediglich alle weiteren Zugriffe fehl schlagen.

## 43.4 Garbage Collection

Alle implizit erzeugten Subcontrols müssen auch wieder gelöscht werden. Zu diesem Zweck merkt sich der IDM, von wie vielen Objekten ein Subcontrol gerade referenziert wird. Wird dabei festgestellt, dass ein Subcontrol von keiner Variablen (lokal, global, statisch) oder benutzerdefiniertem Attribut mehr verwendet wird, so wird das Subcontrol zerstört. Es gibt dabei zwei Strategien, die im

Folgenden vorgestellt werden. Wichtig: Folgende Erklärungen sind auf jeden Fall mit einem Änderungsvorbehalt versehen, da es nicht abzusehen ist, ob in Zukunft andere Strategien eingesetzt werden.

1. Wird ein implizit erzeugtes Subcontrol einer Variablen (oder Ähnlichem) zugewiesen, und wird diese Variable zu einem späteren Zeitpunkt wieder mit einem anderen Wert überschrieben, so kann das Subcontrol wieder freigegeben werden, vorausgesetzt, es hat keine Kinder.

   **Beispiel**

   ```
   Doc := Co.Documents:Add();  // 2 Subcontrols (A für Documents und B für
   Dokument, das
                               // von Add() erzeugt worden ist) werden implizit
   erzeugt.
   Doc := Co;                  // Das Subcontrol B wird nicht länger von
   Variable Doc
                               // benutzt und kann deswegen zerstoert werden.
   Nun kann
                               // auch Subcontrol A feststellen, dass es keine
   Kinder
                               // mehr besitzt und kann sich ebenfalls
   zerstoeren.
   ```

2. Bei ungünstigen Konstellationen kann es passieren, dass ein Subcontrol es nicht merkt, dass es eigentlich zerstört werden kann. Dies passiert z.B., wenn ein gerade erzeugtes Subcontrol nirgendwo zugewiesen wird. In diesem Fall fehlt das auslösende Moment, um das Objekt wegzuwerfen. Deswegen startet der IDM von Zeit zu Zeit Säuberungsaktionen. Dabei werden implizit erzeugte Subcontrols daraufhin getestet, ob diese nicht mehr gebraucht werden und im positiven Fall weggeworfen.

### Wichtig

Da die Garbage-Collection vom IDM nur die Referenzen aus der Regelsprache berücksichtigen kann, ist Vorsicht geboten, wenn die ObjectIDs von Subcontrols aus der C-Schnittstelle (für COBOL und C++ gilt das gleiche) verwaltet werden. Wenn eine solche ID in C irgendwo zwischengespeichert wird, bekommt der IDM dieses nicht mit. Nach dem Aufruf der C-Funktion könnte der IDM feststellen, dass das Subcontrol nicht mehr gebraucht wird, und dieses entsorgen. Wird die Kontrolle wieder auf die C-Seite übergeben, kann dort nicht mehr davon ausgegangen werden, dass die zuvor gespeicherte ObjektID noch gültig ist. Es sollte deswegen vermieden werden, die Subcontrols in C zu speichern. Möchte man das trotzdem tun, muss sicher gestellt werden, dass das Objekt in der Regelsprache noch referenziert wird (z.B. in einer globalen oder statischen Variablen oder in einem benutzerdefinierten Attribut). Nur so kann garantiert werden, dass das Subcontrol nicht weggeworfen wird.

## 43.5 Beispiel

In dem folgenden kleinen Beispiel wird gezeigt, wie man Subcontrols verwenden kann:

```
dialog Word
```

```
window WnFenster
{
  .active false;
  .width 400;
  .height 100;
  .title "Word.Document.8";

  on close
  {
      if Co.connect then
        Co:Quit();
      endif
      exit();
  }

  child control Co
  {
    .visible true;
    .active false;
    .xauto 0;
    .xleft 20;
    .xright 20;
    .yauto 0;
    .ytop 20;
    .ybottom 40;
    .mode mode_client;
    .name "Word.Application";
    .connect true;
  }
  child pushbutton PbOpen
  {
    .xauto 1;
    .xleft 100;
    .width 76;
    .yauto -1;
    .height 27;
    .ybottom 10;
    .text "Open Doc";
    on select
    {
        variable object Doc:=null;

        // Setzt voraus, dass die Datei tatsaechlich existiert
        Doc := Co.Documents:Open("d:/tmp/altesdokument.doc");
```

```
      if Doc <> null then
          print "DocName: " + Doc.FullName;
          print "DocSaveStatus: " + Doc.Saved;
      endif
  }
}
child pushbutton PbAdd
{
  .xauto 1;
  .xleft 20;
  .width 76;
  .yauto -1;
  .height 27;
  .ybottom 10;
  .text "New Doc";
  on select
  {
      Co.Documents:Add("d:/tmp/neuesdokument.doc");
      // Setzt voraus, dass die Datei tatsaechlich existiert
  }
}
child pushbutton PbClose
{
  .xleft 180;
  .width 92;
  .yauto -1;
  .height 27;
  .ybottom 10;
  .text "Close Doc";
  on select
  {
      // Erstes (bzgl. der Documents-Liste) Document schliessen
      Co.Documents:Item(1):Close();
  }
}
child pushbutton PbQuit
{
  .xauto -1;
  .width 85;
  .xright 20;
  .yauto -1;
  .height 27;
  .ybottom 10;
  .text "Quit";
  on select
  {
```

```
        // Word beenden
        Co:Quit();
    }
  }
}
on dialog start
{
    Co.Visible := true;
}
```

# 44 tablefield

Zur Definition dieses Objektes dient das Schlüsselwort **tablefield**. Damit können beliebig formatierte Listen ausgegeben und bearbeitet werden.

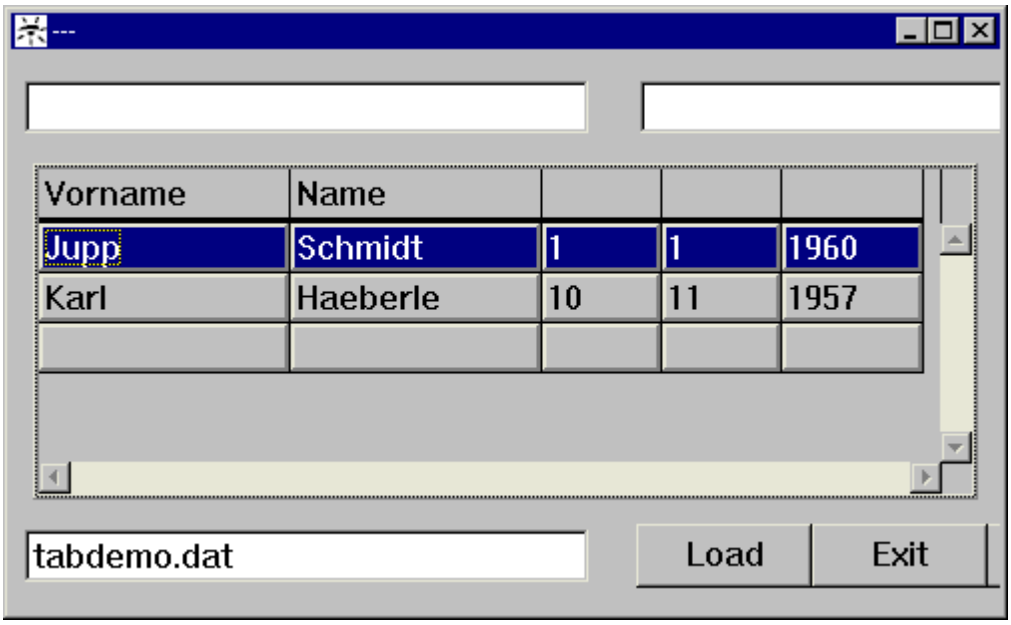



***Abbildung 44:*** *tablefield*

### Definition

```
{ export | reexport } { model } tablefield { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Rasterattribute>
  <Objektspezifische Attribute>
}
```

Über die beiden Scrollbars können die Inhalte des Tablefields sowohl horizontal als auch vertikal gescrollt werden. Der Inhalt des Tablefields besteht aus statischen Texten, die selektiert werden können.

### Ereignisse

*activate*

*cut*

*charinput*

*deactivate*

*dbselect*

*extevent*

*focus*

*help*

*hscroll*

*key*

*modified*

*paste*

*scroll*

*select*

*vscroll*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

### Menü

**Popup Menü**

### Methoden

:clear()

:delete()

:exchange()

:find()

:insert()

### Anmerkung

Damit der Dialogprogrammierer genauer unterscheiden kann, was der Benutzer wirklich gemacht hat, kann das zusätzliche Abfragen von Attributen notwendig sein. Der zum Tablefield gehörende editierbare Text verschickt keinerlei Ereignisse. Diese werden alle an das Tablefield verschickt und können dort abgefragt werden.

### Anmerkungen zu wichtigen Ereignissen beim Tablefield

Das select-Ereignis wird immer dann ausgelöst, wenn der Benutzer mit der Maus klickt (Ausnahme siehe unten „Mauscapture freigeben“ oder wenn der Benutzer die `Leer`-Taste drückt.

Das *select*-Ereignis gibt es in zwei Ausprägungen:

- Mit Index
  Es ist ein Feld selektiert worden, das sensitiv ist.
- Ohne Index
  Es hat ein *select*-Ereignis stattgefunden, das nicht einem einzelnen Feld zuzuordnen ist. D.h. der Benutzer hat entweder in ein insensitives Feld geklickt oder in einen Bereich, in dem sich keine Felder befinden.

Das dbselect-Ereignis wird immer dann ausgelöst, wenn der Benutzer mit der Maus doppelt klickt (Ausnahme siehe unten „Mauscapture freigeben“) oder wenn der Benutzer gleichzeitig die `Strg`- und die `Return`-Taste drückt.

Das *dbselect*-Ereignis gibt es in zwei Ausprägungen:

- Mit Index
  Ein *dbselect*-Ereignis hat in einem Feld stattgefunden, das sensitiv ist.
- Ohne Index
  Es hat ein *dbselect*-Ereignis stattgefunden, das nicht einem einzelnen Feld zuzuordnen ist. D.h. der Benutzer hat entweder in ein insensitives Feld geklickt oder in einen Bereich, in dem sich keine Felder befinden.

Für DM mit MOTIF gilt jedoch, dass das Ereignis *dbselect* immer mit Index verschickt wird. *dbselect* für insensitive Felder gibt es nicht.

Das focus-Ereignis wird immer dann erzeugt, wenn sich das Fokusobjekt bzw. das Fokusfeld ändert. Das *focus*-Ereignis gibt es in zwei Ausprägungen:

- Mit Index
  Ein neues Feld, das fokussierbar ist, hat den Fokus erhalten.
- Ohne Index
  Das Tablefield hat den Fokus erhalten. Es gibt kein Feld, das den Fokus erhalten kann.

Das modified-Ereignis enthält den Index des Felds, das geändert wurde.

Die Ereignisse *activate* bzw. *deactivate* werden ohne Index erzeugt, da sich in der Regel bei mehr als einem Feld der Aktivierungszustand ändert.

*Behandlung von Ereignissen mit Index*

Es ist immer darauf zu achten, dass - bevor man auf einen Index zugreift - überprüft wird, ob überhaupt ein Index gesetzt ist. Eine Überprüfung kann mit folgenden Code-Fragment erfolgen:

```
if (typeof (thisevent.index) = index) then
  /*
   * hier kann mit dem Index gearbeitet werden.
   * "thisevent.index.first" entspricht der Zeile und
   * "thisevent.index.second" entspricht der Spalte.
   */
else

endif
```

### Implizites Zurücksetzen von Tablefieldzuständen beim Ändern von Tablefieldattributen

Bei MOTIF wird der Inhalt bei keinem der unten aufgeführten Attribute verworfen.

Die folgende Aufstellung gilt für MICROSOFT WINDOWS.

Die Aufstellung erhebt keinen Anspruch auf Vollständigkeit. Die Indexattribute für das Innere des Tablefields sind in der Liste nicht enthalten; es gilt jedoch das Entsprechende.

*Deaktivieren des Edittextes ohne Abspeichern des Inhalts*

Unabhängig von edittype wird der Edittext deaktiviert und der eventuell geänderte Inhalt verworfen bei folgenden Attributänderungen:

colcount, colvisible[I], edittext, rowcount, rowvisible[I]

Ab IDM-Version A.05.02.h führt die Verwendung einer Methode, die den Inhalt ändert (z.B. **:insert**, **:exchange**), zum Deaktivieren des angebundenen Edittextes ohne Abspeichern seines Inhalts. In früheren IDM-Versionen blieb der Edittext aktiviert, allerdings wurden nicht alle Änderungen sofort dargestellt.

*Deaktivieren des Edittextes mit Abspeichern des Inhaltes*

Bei den folgenden Attributänderungen wird der Edittext deaktiviert und der eventuell geänderte Inhalt abgespeichert bzw. verworfen. Bei *.edittype = locking* wird der Edittext deaktiviert und der Inhalt verworfen.

Wenn das Attribut active ohne Index umgesetzt wird, wird der Edittext aktiviert oder deaktiviert, wobei der Inhalt nur abgespeichert wird, wenn *.edittype* nicht *locking* ist.

### Mauscapture freigeben

Beim Drücken der linken Maustaste wird ein Mauscapture (exklusiver Mauszugriff) geholt. Beim Loslassen der Maustaste wird dieser Mauscapture wieder freigegeben und ein *select*-Ereignis erzeugt. Es wird **kein** *select*-Ereignis erzeugt, wenn beim Loslassen der linken Maustaste kein Mauscapture aktiv ist. Der Mauscapture wird außerdem bei den unten aufgeführten Attributänderungen

freigegeben. Damit ist es möglich, dass *select*-Ereignisse verhindert werden. Dieses Risiko besteht vor allem beim Abarbeiten von *focus*-Ereignissen, da der Fokus auf das Drücken der Maustasten hin ausgelöst wird. Diese komplizierte Verarbeitung ist gerechtfertigt, um Mausklicks richtig zuordnen zu können.

*Anmerkung*

Dieses Verhalten gilt nicht für DM mit MOTIF.

## 44.1 Attribute

acc_label

acc_text

accelerator

active

active[I,J]

activeitem

bgc

bgc[I,J]

borderraster

borderwidth

class

colalignment[I]

colcount

colfirst

colheader

colheadfgc

colheadfont

colheadshadow

colheadvisible

collinewidth[I]

colsizeable[I]

colwidth[I]

colvisible[I]

content[I,J]

contentfunc

control

cursor

cut_pending

cut_pending_changed

direction

document[I]

editpos

editable

editable[I,J]

edittext

edittype

external

external[I]

fgc

fgc[I,J]

field[I,J]

fieldactive[I,J]

fieldfocus

fieldfocus[I,J]

fieldfocusable

fieldshadow

firstrecord

focus

focus[I,J]

font

font[I,J]

format[I,J]

formatfunc

function

height

help

hsb_optional

index

label

lastrecord

layoutbox

mapped

maxchars[I,J]

member[I]

membercount

menu

model

navigable

nextactive[I,J]

notepage

options[enum]

posraster

preeditsel

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

reffont

rowalignment[I]

rowcount

rowfirst

rowheader

rowheadfgc

rowheadfont

rowheadshadow

rowheadvisible

rowheight[I]

rowlinewidth[I]

rowsizeable[I]

rowvisible[I]

scope

selection[enum]

selstyle

sensitive

sensitive[I,J]

sizeraster

statushelp

toolbar

toolhelp

userdata

userdata[I,J]

visible

vsb_optional

width

xalignment[I,J]

xauto

xleft

xraster

xright

yalignment[I,J]

yauto

ybottom

yraster

ytop

## 44.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| accelerator | Accelerator des Objekts.<br>Wird einem Tablefield ein Accelerator zugeordnet und dieser gedrückt, wird der Fokus auf das Feld im Tablefield gesetzt, das zuletzt den Fokus hatte. Hatte noch kein Element den Fokus, so wird der Fokus auf das linke obere Feld im Inneren des Tablefields gesetzt. |
| active[I,J] | Aktivitätszustand der einzelnen Zelle eines Tablefields. |
| activeitem | Aktives Element des Objekts. |
| bgc[I,J] | Hintergrundfarbe einer Zelle des Tablefields. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. |
| colalignment[I] | Textausrichtung in einer Spalte.<br>Siehe auch Kapitel „Textausrichtung im Tablefield“. |
| colcount | Anzahl der Spalten. |
| colfirst | Erste, neben den Spaltenköpfen angezeigte Spalte des Tablefields. |
| colheader | Anzahl der Spaltenköpfe. |
| colheadfgc | Vordergrundfarbe der Spaltenköpfe. |
| colheadfont | In den Spaltenköpfen verwendeter Zeichensatz. |
| colheadshadow | Schattendarstellung (ähnlich wie bei einem Button) der Spaltenköpfe. |
| colheadvisible | Sichtbarkeit der Spaltenköpfe. |
| collinewidth[I] | Breite der abschließenden Linie einer Spalte.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“.<br>See also chapter „Größenberechnung im Tablefield“. |
| colsizeable[I] | Steuert die interaktive Vergrößerbarkeit der Spalte. |
| colwidth[I] | Angabe der Spaltenbreite. |
| colvisible[I] | Sichtbarkeit der Spalte. |
| content[I,J] | Inhalt einer Tablefieldzelle. |
| contentfunc | Definiert die Funktion, welche für das dynamische Nachladen des Tablefields aufgerufen werden soll. |

| Attribut | Beschreibung |
|---|---|
| direction | Steuert die Ausrichtung des Tablefields.<br>**Siehe auch**<br>Kapitel „Ausrichtung des Tablefields“ |
| editpos | Definiert, wie und wo das Tablefields editiert wird. |
| editable | Bestimmt die generelle Editierbarkeit des gesamten Tablefields. |
| editable[I,J] | Bestimmt die Editierbarkeit einer Zelle des Tablefields. |
| edittext | Definiert den Identifikator des zu einem Tablefield zugehörigen editierbaren Textes. Dieser editierbare Text muss denselben Vater wie das Tablefield haben. |
| edittype | Beschreibt, wie der zu dem Tablefield gehörende editierbare Text und der Inhalt des Tablefieldes zusammenarbeiten.<br>Werte: *edit_locking*, *edit_offline*, *edit_online*. |
| fgc | Vordergrundfarbe des Objekts. |
| fgc[I,J] | Vordergrundfarbe einer Zelle des Tablefields. |
| field[I,J] | Setzt jedes einzelne Textfeld im Tablefield, allerdings keine Spalten- oder Zeilenköpfe. |
| fieldactive[I,J] | Aktivierungszustand jedes einzelnen Textfeldes im Tablefield, allerdings ohne Spalten- oder Zeilenköpfe. |
| fieldfocus | Abfrage oder Setzen des Feldes im Tabelleninneren, welches den Focus hat bzw. haben soll. |
| fieldfocus[I,J] | Setzt den Focus in ein einzelnes Textfeld im Tablefield, allerdings ohne Spalten- oder Zeilenköpfe. |
| fieldfocusable | Definiert, ob die Cursorsteuerung auch nicht sensitive Elemente im Tablefield berücksichtigen soll oder nicht. Dieses Attribut wird nur dann ausgewertet, wenn .sensitive von diesem Feld false ist. |
| fieldshadow | Darstellung der sensitiven Felder des Tablefields mit oder ohne Schatten. |
| focus | Das Tablefield bekommt den Focus. |
| focus[I,J] | Eine bestimmte Zelle des Tablefields bekommt bzw. hat den Focus. |
| font | Festlegung des zum Objekt gehörenden Zeichensatzes. |

| Attribut | Beschreibung |
|---|---|
| font[I,J] | Festlegung des zu einer Zelle des Objekts gehörenden Zeichensatzes. |
| format[I,J] | Festlegung des zu einer Zelle des Tablefields gehörenden Formats. |
| formatfunc | Gibt die zum Objekt gehörende Formatfunktion an. |
| function | Definiert die zum Objekt gehörende Funktion. |
| height | Höhe des Objekts.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |
| hsb_optional | Bestimmt, ob die horizontale Scrollbar des Objekts immer angezeigt werden soll oder nur dann, wenn es notwendig ist. |
| maxchars[I,J] | Maximale Anzahl möglicher Zeichen im Eingabestring einer Zelle des Tablefields. |
| navigable | Steuert die Fokussierbarkeit (Erreichbarkeit des Objekts mittels Tastatur). |
| nextactive[I,J] | Nächste aktive Zelle beim Tablefield und selstyle <> *single*. |

| Attribut | Beschreibung |
|---|---|
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_center_toolhelp* (nur MS Windows).<br>» *true*: Wenn der Platz dafür ausreicht, wird die Toolhelp zentriert unter dem Objekt angezeigt, zu dem sie gehört.<br>» *false* (Standard): Die Toolhelp wird am Mauszeiger angezeigt.<br>» *opt_new_align* (Motif und MS Windows).<br>» *true*: Für die Textausrichtung sind die Attribute .xalignment[I,J] und .yalignment[I,J] maßgeblich.<br>» *false* (Standard): Für die Textausrichtung sind die Attribute .colalignment[I] und .rowalignment[I] maßgeblich.<br>Siehe auch Kapitel „Textausrichtung im Tablefield“.<br>» *opt_new_colwidth* (nur Motif).<br>In älteren Versionen des IDM (vor A.03.10.f) wurde unter Motif bei Tabellen mit einem *reffont* die Spaltenbreite falsch berechnet. Dazu gab es in Version A.03.10.f eine **zweite** Korrektur, bei der die Option *opt_new_colwidth* eingeführt wurde.<br>» *true* (Standard): Korrigierte Spaltenbreite (Berechnung auf Basis von Rastergrößen, wie beim Layout anderer Objekte).<br>» *false*: Berechnung der Spaltenbreite entsprechend der Option *opt_old_colwidth*.<br>» *opt_old_colwidth* (nur Motif).<br>In älteren Versionen des IDM (vor A.03.04.a) waren unter Motif bei Tabellen mit einem *reffont* die Spalten zu breit. Dazu gab es in Version A.03.04.a eine **erste** Korrektur, bei der die Option *opt_old_colwidth* eingeführt wurde.<br>» *true*: Spaltenbreite wie vor der Korrektur.<br>» *false* (Standard): Korrigierte, kleinere Spaltenbreite.<br>**Hinweis**<br>Ab Version A.03.10.f wurde die Spaltenbreite noch einmal korrigiert und dabei die Option *opt_new_colwidth* (siehe oben) eingeführt. Wenn *opt_new_colwidth* den Wert *true* hat, dann hat diese Einstellung Vorrang vor *opt_old_colwidth*.<br>» *opt_old_select* (nur Motif, ab IDM-Version A.05.02.h).<br>Mit dieser Option kann eine auswahl- oder eine aktionsorientierte Verschickung von *Select*-Ereignissen eingestellt werden. Dadurch wird ein konsistentes Verhalten der Motif- und Windows-Version des IDM ermöglicht. |

| Attribut | Beschreibung |
|---|---|
| | » *true*: Select-Ereignisse werden bei Änderungen des Aktivierungszustands verschickt (auswahlorientiert). Dies entspricht dem Verhalten bis einschließlich IDM version A.05.02.g.<br>» *false* (Standard): Select-Ereignisse werden bei Mausklick bzw. beim Betätigen der Selektionstaste verschickt (aktionsorientiert). Dies entspricht dem Verhalten unter Microsoft Windows. Damit zeigt das Select-Ereignis keine Änderung des Aktivierungszustands mehr an. Um auch bei *.options[opt_old_select] = false* auf Änderungen der Auswahl zu reagieren, können die Ereignisse „activate“ und „deactivate“ genutzt werden. |
| preeditsel | Steuert die initiale Markierung des Textes, wenn der Edittext eines Tablefields explizit oder implizit aktiviert wird.<br>Werte: *presel_default*, *presel_all*, *presel_begin*, *presel_end*. |
| reffont | Referenzfont zur Größenberechnung des Objekts.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |
| rowalignment[l] | Textausrichtung in einer Zeile.<br>Siehe auch Kapitel „Textausrichtung im Tablefield“. |
| rowcount | Anzahl der Zeilen. |
| rowfirst | Erste, unter den Kopfzeilen angezeigte Zeile des Tablefields. |
| rowheader | Anzahl der Kopfzeilen des Tablefields. |
| rowheadfgc | Vordergrundfarbe der Kopfzeilen. |
| rowheadfont | In den Kopfzeilen verwendeter Zeichensatz. |
| rowheadshadow | Schattendarstellung (ähnlich wie bei einem Button) der Kopfzeilen. |
| rowheadvisible | Sichtbarkeit der Kopfzeilen. |
| rowheight[l] | Angabe der Zeilenhöhe.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |
| rowlinewidth[l] | Breite der abschließenden Linie einer Zeile. |
| rowsizeable[l] | Steuert die interaktive Vergrößerbarkeit der Zeile. |
| rowvisible[l] | Sichtbarkeit der Zeile. |

| Attribut | Beschreibung |
|---|---|
| selection[enum] | Zulässige Art der Selektion im Tablefield.<br>Indizes: *sel_column*, *sel_header*, *sel_row*, *sel_single*. |
| selstyle | Steuerung der Selektionsart innerhalb des Tablefields.<br>Werte: *single*, *multiple*, *extended*. |
| sensitive | Definiert, ob das Tablefield selektierbar sein soll.<br>Damit kann das Objekt mit einem Schritt insensitiv gemacht werden. |
| sensitive[I,J] | Sensitivitätssteuerung jeder Zelle des Objekts. |
| sizeraster | Aktivierung des Größenrasters.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |
| userdata[I,J] | Userdata für beliebige Daten zu jeder Zelle des Objekts. |
| vsb_optional | Bestimmt, ob die vertikale Scrollbar des Objekts immer angezeigt werden soll oder nur dann, wenn es notwendig ist. |
| width | Breite des Objekts.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |
| xalignment[I,J] | Beschreibt die Textausrichtung innerhalb einer Zelle in horizontaler Richtung.<br>Siehe auch Kapitel „Textausrichtung im Tablefield“. |
| xraster | Horizontales Raster des Objekts<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |
| yalignment[I,J] | Beschreibt die Textausrichtung innerhalb einer Zelle in vertikaler Richtung.<br>Siehe auch Kapitel „Textausrichtung im Tablefield“. |
| yraster | Vertikales Raster des Objekts.<br>Siehe auch Kapitel „Größenberechnung im Tablefield“. |

### 44.2.1 Größenberechnung im Tablefield

Die Attribute height und width können auch den Wert 0 haben. Dies bedeutet, dass der Dialog Manager die entsprechende Größe so berechnen soll, dass alle Elemente in der entsprechenden Richtung dargestellt werden können, ohne dass eine Scrollbar notwendig ist. Wird z.B. die Breite des Tablefields auf 0 gesetzt, wird es so breit, dass alle Spalten vollständig in dem Objekt dargestellt werden können.

Wenn das Tablefield sizeraster gesetzt hat, und sowohl xraster/yraster als auch reffont = 0 sind, wird die Rasterdefinition des direkten Vaterobjektes übernommen. Das Tablefield berechnet seine eigene Größe also wie ein normales Objekt aus den Rasterfaktoren seines Parents. Lediglich die Grö-

ßendefinitionen im Inneren des Tablefields, collinewidth[I] und rowheight[I], beziehen sich auf die beim Tablefield definierten Rasterattribute.

Die Berechnung der Pixelwerte erfolgt aber nach der beim Tablefield definierten Methode.

## 44.2.2 Textausrichtung im Tablefield

Die horizontale und vertikale Ausrichtung von Text in einer Tablefield-Zelle wird durch die Attribute colalignment[I] und rowalignment[I] gesetzt, jedoch die horizontale Ausrichtung nur spaltenweise und die vertikale Ausrichtung nur zeilenweise.

Mittels der Option options[opt_new_align] mit dem Wert *true* werden statt *.colalignment[I]* und *.rowalignment[I]* die Attribute xalignment[I,J] und yalignment[I,J] berücksichtigt. Diese haben den gleichen Wertebereich/die gleiche Bedeutung wie *.colalignment[I]* und *.rowalignment[I]*; allerdings sind *.xalignment[I,J]* und *.yalignment[I,J]* doppelt indiziert und die Vererbung richtet sich nach der Direction wie bei allen anderen doppelt indizierten Attributen auch.

Der Defaultwert von *.options[opt_new_align]* ist *false*.

Unterstützt unter Motif und MS Windows.

## 44.2.3 Attributdefaultwerte

Bei einigen Attributen ist *.attribut[0]* als Default definiert. Dieses wird immer dann genommen, wenn zu dem wirklich benötigten Wert kein Wert mehr vorhanden ist. Das erste richtig nutzbare Element ist immer *.attribut[1]*!

Bei einigen Attributen ist *.attribut[0,0]* bzw. *.attribut[y,0]* und *.attribut[0,x]* als Default definiert. Diese werden immer dann genommen, wenn zu dem wirklich benötigten Wert kein Wert mehr vorhanden ist. Das erste richtig nutzbare Element ist immer *.attribut[1,1]*! Bei der Verwendung dieser Defaults wird zweistufig vorgegangen:

- Ist bei dem benötigten Index kein Wert definiert, so wird der Index auf 0 gesetzt, der durch das Attribut *.direction* definiert ist. Ist hier ein Wert definiert, so wird dieser genommen.
- Ist hier aber auch kein Wert definiert, so wird der für das Attribut globale Default bei *.attribut[0,0]* genommen.

Aufgrund dieser Vorgehensweise sollten Sie bei benutzten Attributen immer ein Default *.attribut[0,0]* definieren.

## 44.2.4 Indizierung der Tabelle

Alle mit zwei Indizes versehenen Attribute werden immer mit [I] = Row/Zeile und [J] = Column/Spalte indiziert, also *.attribut[I, J]*.

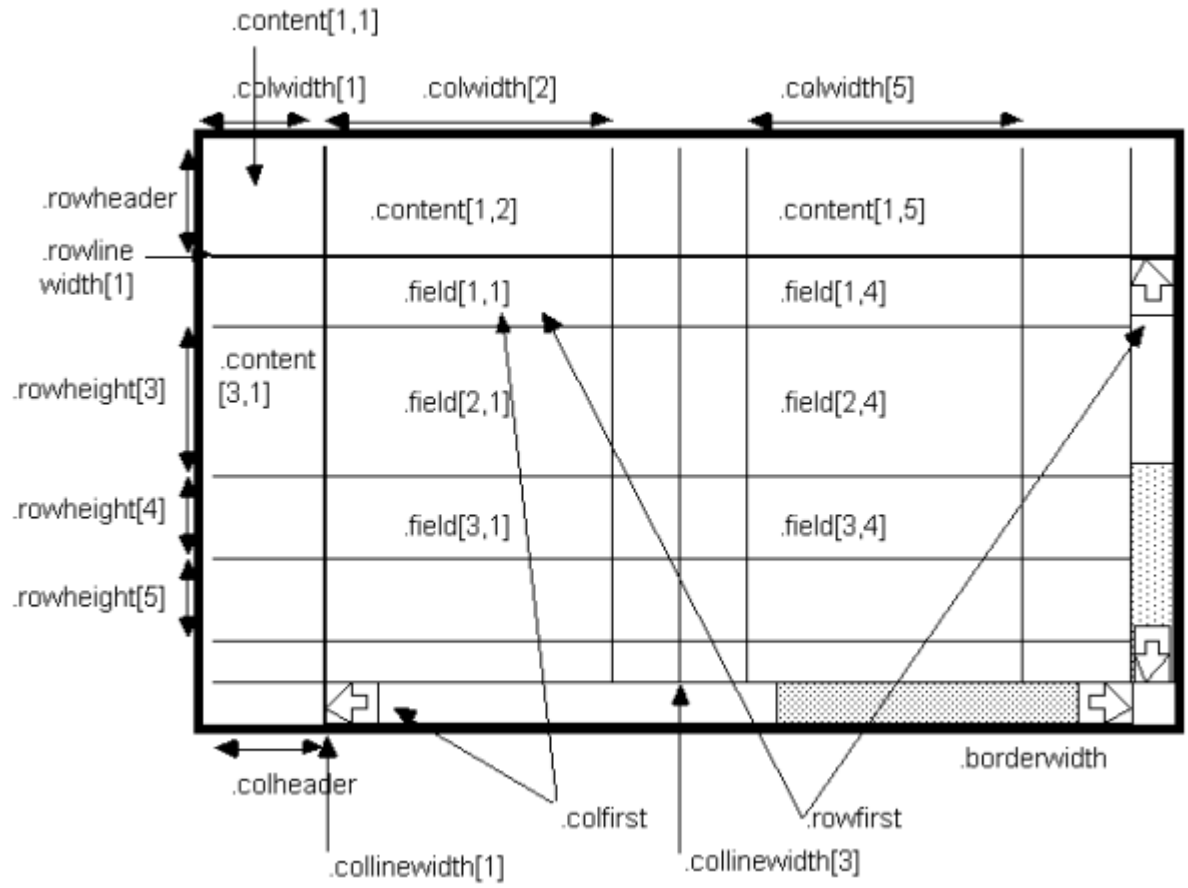



***Abbildung 45:*** *Schematische Darstellung des Tablefields*

Im obigen Bild sind alle Indizes so angegeben, wie dies definiert werden muss.

## 44.3 Hinweis zu veralteten Attributen

Das veraltete Attribute *.focusable* wird nicht mehr unterstützt und erzeugt eine ignoring-Warnung beim Setzen und Zurücksetzen (setinherit) des Attributs sowie ein FAIL beim Auslesen des Attributs.

## 44.4 Ausrichtung des Tablefields

Das *.direction*-Attribut definiert die Ausrichtung des ***tablefields*** und steuert ob Zeilen oder Spalten Vorrang haben. Das Attribut beeinflusst die Übernahme von Zeilen- und Spalten-Standardwerten, die Selektion und Navigation sowie bestimmte Methoden.

### Wertebereich

***1*** (Standard)
Vertikale Ausrichtung mit Spaltenpriorität.

*2*
Horizontale Ausrichtung mit Zeilenpriorität.

### Auswirkungen

#### Standardwerte

Bei *.direction = 1* werden für nicht gesetzte Zellwerte (*.content[index]*, *.bgc[index]*, *.fgc[index]*…) die Spalten-Standardwerte übernommen, d.h. die Attributwerte mit den Indizes *[0,<C>]*. Bei *.direction = 2* werden die Zeilen-Standardwerte, d.h. die Attributwerte mit den Indizes *[<R>,0]* übernommen.

#### Selektion

Wenn sowohl *.selection[sel_column]* als auch *.selection[sel_row]* auf *true* gesetzt sind und die Zelle *[<R>,<C>]* ausgewählt wird, dann wird bei *.direction = 1* die Spalte <*C*> und bei *.direction = 2* die Zeile <*R*> selektiert.

In einem ***tablefield*** mit Mehrfachselektion sucht .nextactive[index] bei *.direction = 1* zeilenweise und bei *.direction = 2* spaltenweise nach der nächsten selektierten Zelle. Sind beispielsweise die Zellen *[4,2]* und *[3,5]* selektiert, liefert *.nextactive[1,1]* bei *.direction = 1* als Ergebnis *[3,5]* und bei *.direction = 2* als Ergebnis *[4,2]*.

#### Navigation

Bei *.direction = 1* wird der Eingabefokus beim Drücken von `Return` in die nächste selektierbare Zelle rechts von der aktuellen Zelle bewegt. `Pos 1` und `Ende` springen an den Anfang bzw. das Ende der Zeile mit der aktuell fokussierten Zelle.

Bei *.direction = 2* wird der Eingabefokus beim Drücken von `Return` in die nächste selektierbare Zelle unter der aktuellen Zelle bewegt. `Pos 1` und `Ende` springen an den Anfang bzw. das Ende der Spalte mit der aktuell fokussierten Zelle.

#### :find()-Methode

Das *.direction*-Attribut beeinflusst die Suchrichtung der **:find()**-Methode. Bei *.direction = 1* wird das ***tablefield*** zeilenweise durchsucht, bei *.direction = 2* spaltenweise. Befindet sich der gesuchte Wert beispielsweise in den Zellen *[4,2]* und *[3,5]*, dann liefert **:find()** ohne Angabe eines Suchbereichs bei *.direction = 1* die Fundstelle *[3,5]* und bei *.direction = 2* die Fundstelle *[4,2]*.

#### Methoden :clear(), :delete(), :exchange(), :insert() und :move()

Das Attribut *.direction* steuert, zusammen mit dem optionalen *Direction*-Parameter der Methoden, ob die Methoden auf Zeilen oder Spalten angewendet werden. Wenn der *Direction*-Parameter nicht angegeben ist, werden die Methoden bei *.direction = 1* auf Zeilen und bei *.direction = 2* auf Spalten angewendet.

## 44.5 Interaktionen mit der Maus

Soweit vorhanden, sind mit der Maus folgende Interaktionen möglich:

- Selektion
- Scrollen

### 44.5.1 Selektion im Tablefield

Die Selektion eines Feldes ist nur dann möglich, wenn es sensitiv ist. Dies wird über das Attribut *.sensitive* für das gesamte Tablefield und über das feldspezifische Attribut *.sensitive[I,J]* definiert. Neben diesen Attributen hat auch das Attribut *.selstyle* Einfluss auf die Selektierbarkeit eines Feldes. Im Normalfall wird das Feld selektiert, auf das der Benutzer geklickt hat. Auf dieses Feld wird der Eingabefokus gesetzt und als selektiert und mit Fokus versehen, gekennzeichnet. Der Inhalt des Feldes wird sofort in den assoziierten, editierbaren Text übernommen und dort zur Änderung freigegeben.

Damit die Einzelfeldselektion von einer Zeilen-/Spaltenselektion unterschieden werden kann, müssen zwei Mausselektionsarten definiert werden. Die **Einzelfeldselektion** erfolgt dabei so, wie der Benutzer normalerweise ein Objekt selektiert, d.h. durch Klick auf das Objekt mit der linken Maustaste. Die **Zeilen-/Spaltenselektion** sollte durch Drücken der linken Maustaste und gleichzeitigem Drücken der `Shift`-Taste erfolgen. Bei der Bestimmung der auszulösenden Aktion muss die Gewichtigkeit der einzelnen Selektionsarten berücksichtigt werden.

Dabei gilt folgende Reihenfolge ( > = "wichtiger als", "geht vor"):

- Für die Mauseinzelselektion
  single > row/column
- Für die Zeilen-/Spaltenselektion
  row/column > single

Damit ergibt sich folgendes Verfahren für die Auswertung:

- Einzelfeldselektion
  Zunächst wird überprüft, ob der Mausklick als "single"-Selektion bzw. als "header"-Selektion interpretiert werden kann. Ist dies nicht möglich, wird er als "row/column"-Selektion interpretiert. Dabei wird zuerst die Richtung untersucht, die im Attribut *.direction* angegeben ist.
- Zeilen-/Spaltenselektion
  Zunächst wird überprüft, ob der Mausklick als "row/column"-Selektion interpretiert werden kann. Dabei wird zuerst die Richtung untersucht, die im Attribut *.direction* angegeben ist. Ist dies nicht möglich wird versucht, ihn als "single"-Selektion bzw. als "header"-Selektion zu interpretieren.

Aktionen bei den einzelnen Tablefield-Selektionsarten:

| .selection[I] true | Reaktion |
|---|---|
| sel_single | Das Feld, das gerade den Fokus hat, verliert den Fokus und gibt ihn an das Element unter der Maus weiter, falls diese beiden Elemente verschieden sind. Sind die beiden Objekte identisch, passiert nichts. |
| sel_header | Das Feld, das gerade den Fokus hat, verliert den Fokus und gibt ihn an das Element unter der Maus weiter, falls diese beiden Elemente verschieden sind. Sind die beiden Objekte identisch, passiert nichts. |

| .selection[l] true | Reaktion |
|---|---|
| sel_column | Die Spalte/Zeile unter der Maus wird selektiert. Das Feld auf das geklickt worden ist, erhält den Fokus. Ist die Spalte/Zeile schon selektiert, passiert nichts, außer dass der Fokus auf das Feld unter der Maus gesetzt wird. |
| sel_row | |

### 44.5.2 Scrollen im Tablefield

Das Scrollen in alle Richtungen erfolgt immer in Sinneinheiten. Es wird immer so gescrollt, dass ganze Zeilen oder Spalten aus dem Anzeigebereich gelöscht und durch neue ersetzt werden. Dieses gilt sowohl für das zeilen-/spaltenweise als auch für das seitenweise Scrollen. Dabei kann immer so weit gescrollt werden, dass die letzte Spalte oder Zeile gerade vollständig sichtbar ist und rechts und unten möglicherweise eine leere Fläche entsteht. Das kommt daher, dass der Dialog Manager garantiert, dass in der linken oberen Ecke ein Element sichtbar ist.

Im Einzelnen sieht das wie folgt aus:

| Scrollaktion | Reaktion |
|---|---|
| Scroll Right | Die linke Spalte des Tablefieldinneren wird aus dem sichtbaren Bereich hinausgescrollt. Die bisher zweite Spalte wird neben die Zeilenköpfe gesetzt. Der Rest des Anzeigebereiches wird mit den nachfolgenden Spalten aufgefüllt. Wenn nur eine Spalte dargestellt werden kann, wird die nachfolgende Spalte in den Anzeigebereich gebracht. |
| Scroll Left | Die bisher nicht sichtbare Spalte vor der ersten sichtbaren Spalte wird in den Anzeigebereich geholt und die restlichen sichtbaren Spalten entsprechend verschoben. Wenn nur eine Spalte dargestellt werden kann, wird die nachfolgende Spalte in den Anzeigebereich gebracht. |
| Scroll Down | Die erste Zeile des Tablefieldinneren wird aus dem sichtbaren Bereich hinausgescrollt. Die bisher zweite Zeile wird unterhalb der Überschriften gesetzt. Der Rest des Anzeigebereiches wird mit den nachfolgenden Zeilen aufgefüllt. Wenn nur eine Spalte dargestellt werden kann, wird die nachfolgende Spalte in den Aneigebereich gebracht. |
| Scroll Top | Die bisher nicht sichtbare Zeile des Tablefieldinneren wird vor der ersten Zeile im Anzeigebereich geschoben und die restlichen sichtbaren Zeilen entsprechend verschoben. Wenn nur eine Spalte dargestellt werden kann, wird die nachfolgende Spalte in den Anzeigebereich gebracht. |

| Scrollaktion | Reaktion |
|---|---|
| Page Right | Die bisher letzte sichtbare Spalte wird zur ersten sichtbaren Spalte und der Rest des Anzeigebereiches wird mit den nachfolgenden Spalten gefüllt. Wenn nur eine Spalte dargestellt werden kann, wird die nachfolgende Spalte in den Anzeigebereich gebracht. |
| Page Left | Die bisher erste sichtbare Spalte wird zur letzten sichtbaren Spalte und der Rest des Anzeigebereiches wird mit den vorangehenden Spalten gefüllt. Wenn nur eine Spalte dargestellt werden kann, wird die vorhergehende Spalte in den Anzeigebereich gebracht. |
| Page Down | Die bisher letzte sichtbare Zeile wird zur ersten sichtbaren Zeile und der Rest des Anzeigebereiches wird mit den nachfolgenden Zeilen gefüllt. Wenn nur eine Zeile dargestellt werden kann, wird die nachfolgende Zeile in den Anzeigebereich gebracht. |
| Page Top | Die bisher erste sichtbare Zeile wird zur letzten sichtbaren Zeile und der Rest des Anzeigebereiches wird mit den vorangehenden Zeilen gefüllt. Wenn nur eine Zeile dargestellt werden kann, wird die vorhergehende Zeile in den Anzeigebereich gebracht. |
| Slider | Die Scrollaktion des Benutzers wird immer so korrigiert, dass die erste Zeile/Spalte im Tablefieldinneren korrekt sichtbar ist. Dabei kann es natürlich vorkomen, dass die untere Zeile und die rechte Spalte nur teilweise sichtbar sind. |

Durch dieses Scrollen wird vom Dialog Manager garantiert, dass die erste Zeile und die erste Spalte im Tablefieldinneren immer korrekt und vollständig dargestellt werden, wenn das Tablefield nur groß genug ist. Die letzte sichtbare Zeile und die letzte sichtbare Spalte können bei ungünstiger Dimension des Tablefields nur teilweise dargestellt werden. Ist die Scrollaktion in der beschriebenen Form nicht durchführbar, wird das Scrollen so weit wie möglich durchgeführt.

### Beispiel

Der Benutzer möchte seitenweise Scrollen, es wird aber nur noch eine Spalte nicht angezeigt. Es wird dann so gescrollt, dass diese Spalte gerade vollständig in den Anzeigebereich kommt. Dabei kann es vorkommen, dass rechts und unten ein leerer Platz bleibt, da die linke obere Ecke immer korrekt dargestellt wird.

Die Berechnung der Slidergröße und des Maximalwertes des Sliders basiert dabei auf Pixel, nicht auf Sinneinheiten des Tablefields. Bei der Berechnung der Slidergröße wird der zur Verfügung stehende Platz im Tablefieldinneren zu dem eigentlich benötigten Bereich ins Verhältnis gesetzt.

Das geschieht nach folgender Formel:

» Für die horizontale Scrollbar
Slidersize = Scrollbarsize * (nutzbare Breite /Summe der Breite aller Spalten (ohne Zeilenköpfe))

» Für die vertikale Scrollbar
Slidersize = Scrollbarsize * (nutzbare Höhe /Summe der Höhen aller Zeilen (ohne Überschriften))

Beim Scrollen wird der Fokus im Tablefield nicht geändert. Dadurch ist es möglich, dass der Fokus auf einem gerade nicht sichtbaren Objekt im Tablefield sitzt. Um dies zu visualisieren, wird dann der Fokusrahmen um das gesamte Objekt gemalt. Wird das Objekt mit dem Fokus wieder in den sichtbaren Bereich des Tablefields gescrollt, wird der Fokus nur um das Feld gemalt und nicht mehr um das gesamte Objekt.

#### Anmerkungen für Dialog Manager mit Microsoft Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

## 44.6 Interaktionen über die Tastatur

Das Tablefield ist vollständig ohne Maus bedienbar. Bei der Tablefield-Cursorsteuerung müssen vom Tablefield automatisch alle notwendigen Scrolloperationen ausgeführt werden, damit das Feld, das durch die Cursorsteuerung erreicht wurde, im sichtbaren Bereich des Tablefields liegt. Die Cursorsteuerung ist dabei für keine Richtung zyklisch; d.h. liegt in der eigentlichen Suchrichtung kein für die Cursorsteuerung zugelassenes Feld mehr, bleibt der Cursor stehen. Ein für die Cursorsteuerung zugelassenes Feld ist dabei entweder *.sensitive* = *true* oder bei dem Tablefield wurde das Attribut *.fieldfocusable* auf *true* gesetzt (d.h. damit können auch nicht-sensitive Objekte mit der Cursorsteuerung erreicht werden!).

### 44.6.1 Tastaturbelegung für die Navigation

Die Tastaturbelegung für die Navigation sieht wie folgt aus:

| Taste | Aktion |
|---|---|
| Cursor `Abwärts` | Mit Hilfe dieser Taste soll der Eingabefokus auf das nächste in steigender y-Richtung liegende Feld gesetzt werden, das für die Cursorsteuerung zugelassen ist. Damit geht der Eingabefokus optisch nach unten. |
| Cursor `Aufwärts` | Mit Hilfe dieser Taste soll der Eingabefokus auf das nächste in fallender y-Richtung liegende Feld gesetzt werden, das für die Cursorsteuerung zugelassen ist. Damit geht der Eingabefokus optisch nach oben. |
| Cursor `Links` | Mit Hilfe dieser Taste soll der Eingabefokus auf das nächste in fallender x-Richtung liegende Feld gesetzt werden, das für die Cursorsteuerung zugelassen ist. Damit geht der Eingabefokus optisch nach links. |
| Cursor `Rechts` | Mit Hilfe dieser Taste soll der Eingabefokus auf das nächste in steigender x-Richtung liegende Feld gesetzt werden, das für die Cursorsteuerung zugelassen ist. Damit geht der Eingabefokus optisch nach rechts. |

| Taste | Aktion |
|---|---|
| `Return` | Mit Hilfe dieser Taste soll der Eingabefokus auf das nächste in der durch *.direction* definierten Richtung liegende Feld gesetzt werden. Wird dabei das Ende einer Spalte/Zeile erreicht, bleibt der Fokus auf diesem Element. |
| `Tab`<br>`Shift` + `Tab` | Diese beide Tasten dienen dem Verlassen des Tablefields. Dabei soll zum nächsten/vorhergehenden Objekt, entsprechend dem üblichen Verhalten des Fenstersystems gegangen werden. |
| `Pos 1` (Home) | Mit Hilfe dieser Taste wird der Fokus auf das erste Element einer Zeile (.direction=2) oder Spalte (.direction =1) im Tablefieldinneren gesetzt (rowindex > rowheader bzw. colindex>colheader). Es wird beginnend mit dem Element [rowheader + 1] [?] in der Zeile nach rechts (.direction = 2) bzw. beginnend mit dem Element [?] [colheader + 1] in der Spalte nach unten (.direction = 1) gesucht, bis ein Element gefunden wird, das den Fokus erhalten kann (.sensitive und .fieldfocusable). Dieses Element wird dann in den sichtbaren Bereich des Tablefields gescrollt. |
| `Strg` + `Pos 1` | Mit Hilfe dieser Taste wird der Fokus auf das erste Element (oben links) im Tablefieldinneren gesetzt (rowindex > rowheader bzw. colindex > colheader). Es wird beginnend mit dem Element [rowheader + 1] [colheader + 1] in der Zeile nach rechts (.direction = 2) bzw. in der Spalte nach unten (.direction = 1) gesucht, bis ein Element gefunden wird, das den Fokus erhalten kann (.sensitive und .fieldfocusable). Wird in der Zeile bzw. Spalte kein Element gefunden, wird die Suche in der darunterliegenden Zeile bzw. rechtsliegenden Spalte fortgesetzt. Das gefundene Element wird dann in den sichtbaren Bereich des Tablefields gescrollt. |
| `Ende` | Mit Hilfe dieser Taste wird der Fokus auf das letzte Element einer Zeile (.direction =2) oder Spalte (.direction = 1) im Tablefieldinneren gesetzt (rowindex > rowheader bzw. colindex > colheader). Es wird beginnend mit dem Element [rowcount] [?] in der Zeile nach links (.direction = 2) bzw. beginnend mit dem Element [?] [colcount] in der Spalte nach oben (.direction = 1) gesucht, bis ein Element gefunden wird, das den Fokus erhalten kann (.sensitive und .fieldfocusable). Dieses Element wird dann in den sichtbaren Bereich des Tablefields gescrollt. |

| Taste | Aktion |
|---|---|
| `Strg` + `Ende` | Mit Hilfe dieser Taste wird der Fokus auf das letzte Element (unten rechts) im Tablefieldinneren gesetzt (rowindex > rowheader bzw. colindex > colheader). Es wird beginnend mit dem Element [rowcount] [colcount] in der Zeile nach links (.direction = 2) bzw. in der Spalte nach oben (.direction = 1) gesucht, bis ein Element gefunden wird, das den Fokus erhalten kann (.sensitive und .field-focusable). Wird in der Zeile bzw. Spalte kein Element gefunden, wird die Suche in der darüberliegenden Zeile bzw. linksliegenden Spalte fortgesetzt. Das gefundene Element wird dann in den sichtbaren Bereich des Tablefields gescrollt. |
| logisch: Show | Mit Hilfe dieser Taste kann das Element im Tablefield in den Anzeigebereich geholt werden, das gerade den Fokus hat. Ist das Objekt bereits im sichtbaren Bereich, passiert nichts. |

#### Anmerkung zu Cursor Aufwärts, Abwärts, Rechts, Links

Wird durch die Cursorsteuerung vom Headerbereich in den Bereich des Tablefieldinneren gewechselt, wird der Cursor auf das erste unterhalb/neben den Headern sichtbare Element gesetzt, ohne dass ein Scrollen ausgelöst wird.

Wird im ersten Element unterhalb/neben den Headern die Cursorsteuerung in Richtung Header (Cursor `Aufwärts`, Cursor `Links`) ausgelöst, wird erst dann auf den Header positioniert, wenn die entsprechende Scrollbar auf ihrem Initialwert steht.

(this->tablefield.?->first = this->tablefield.?->header + 1)

Solange dies nicht der Fall ist, wird entsprechend gescrollt.

Damit müssen die Aktionen Cursor `Links` und anschließend Cursor `Rechts`, sowie Cursor `Abwärts` und anschließend Cursor `Aufwärts` **nicht** inverse Operationen sein.

### 44.6.2 Tastaturbelegung für das Scrollen

Die Tastaturbelegung für das Scrollen sieht wie folgt aus:

| Taste | Aktion |
|---|---|
| Page Up | Über diese Taste soll der Inhalt des Tablefields um maximal eine Seite nach oben verschoben werden. |
| Page Down | Über diese Taste soll der Inhalt des Tablefields um maximal eine Seite nach unten verschoben werden. |
| Page Left | Über diese Taste soll der Inhalt des Tablefields um maximal eine Seite nach links verschoben werden. |

| Taste | Aktion |
|---|---|
| Page Right | Über diese Taste soll der Inhalt des Tablefields um maximal eine Seite nach rechts verschoben werden. |
| Line Up | Über diese Taste soll der Inhalt des Tablefields um eine Zeile nach oben verschoben werden. |
| Line Down | Über diese Taste soll der Inhalt des Tablefields um eine Zeile nach unten verschoben werden. |
| Row Left | Über diese Taste soll der Inhalt des Tablefields um eine Spalte nach links verschoben werden. |
| Row Right | Über diese Taste soll der Inhalt des Tablefields um eine Spalte nach rechts verschoben werden. |

Für alle diese Scrollaktionen gilt das im Kapitel „Interaktionen mit der Maus“ beschriebene Verhalten beim Scrollen.

#### Anmerkung zum Scrollen über Tastatur

Wird über die Tastatur das Scrollen ausgelöst, ändert der Fokus seine Postion auf dem Bildschirm nicht. Dadurch kann sich aber seine interne Postion bzw. das Element, das gerade den Fokus hat, verändern. Steht der Cursor z.B in einer Überschrift und es wird vertikal gescrollt, bleibt der Cursor sowohl auf dem Bildschirm als auch intern auf derselben Position. Steht der Cursor aber im Tablefieldinneren, ändert er auf jeden Fall seine interne Position; die auf dem Bildschirm sichtbare bleibt dagegen weitgehend erhalten.

### 44.6.3 Tastaturbelegung für die Selektion

| Taste | Aktion |
|---|---|
| Space, log. Selektion | Über diese Taste wird das Feld mit dem Fokus selektiert, falls es noch nicht selektiert ist. |
| log.Zeilen-/Spaltenselektion | Mit dieser Taste wird die Zeilen-/Spaltenselektion ausgelöst. |

Die beiden Tasten "logische Selektion" und "logische Zeilen-/Spaltenselektion" sind vom Fenstersystem abhängig.

## 44.7 Interaktionen mit dem editierbaren Text

Der zu dem Tablefield gehörende editierbarer Text wird automatisch mit dem Inhalt des Feldes versehen, das gerade den Fokus hat. Zusätzlich werden dabei die dem aktuellen Fokusobjekt entsprechenden Attribute *.maxchars[I,J]* und *.format[I,J]* übernommen.

Je nach Attribut *.edittype* werden die Aktionen im editierbaren Text in das Tablefield übernommen.

» Bei *.edittype online* werden alle Veränderungen im editierbaren Text sofort in das Tablefield übernommen.

» Bei *.edittype offline* werden die Änderungen erst bei der Deselektion des editierbaren Textes in das Tablefield übernommen.

» Bei *.edittype locking* wird das Tablefield nach der ersten Eingabe in den editierbaren Text gesperrt und erst nach Eingabe von Return wieder freigegeben und der Inhalt übernommen.
Wird der editierbare Text dabei ohne Return und nur durch eine normale Deselektion oder der Eingabe von `ESC` verlassen, werden die Änderungen nicht übernommen, das Tablefield wird jedoch wieder freigegeben. Der editierbare Text selbst verschickt keine Ereignisse.

### Anmerkungen

» Ein editierbarer Text wird nicht durch die Cursornavigation aktiviert, sondern erst durch das erste, für den editierbaren Text bestimmte, Zeichen.

» Wenn ein editierbarer Text aktiviert ist, gilt folgende Navigation:

| Taste | einzeiliger ET | mehrzeiliger ET |
|---|---|---|
| Cursor `Links`, `Rechts` | ET erhält Taste | ET erhält Taste |
| Cursor `Aufwärts`, `Abwärts` | ET erhält Taste (kann auch ignoriert werden) | ET erhält Taste |
| `Return` | Nächster (vorheriger) Eintrag | ET erhält Taste |
| `Tab` (`Shift` + `Tab`) | Nächster (vorheriger) Eintrag | Nächster (vorheriger) Eintrag |

» Ein mehrzeiliger editierbarer Text hat immer Scrollbars, unabhängig von *.editpos*.

» Die Höhe eines Eintrages hat keinen Einfluss auf die Anzahl der Textzeilen, die vom Benutzer eingegeben werden können.

» Da das Attribut *.format* vom Tablefield kontrolliert wird, sollte das Attribut des angebundenen Edittextes **nicht** verwendet werden.

## 44.8 Interaktive Spalten- und Zeilengrößenänderung

Wird die Maus über den Rand eines manipulierbaren Feldes bewegt, so ändert sich der Mauszeiger und zeigt ein Größenänderungssymbol. Das Aussehen des Cursors ist vom Anwendungsprogrammierer nicht spezifizierbar. Bei den Spalten kann die Größenänderung jeweils am rechten Rand einer Spalte durchgeführt werden. Bei den Zeilen ist die untere Begrenzung relevant.

Ber Benutzer kann mit der linken Maustaste die Größenänderung einleiten. Es erscheint eine graue schraffierte Linie, die die Position der Maus anzeigt (Die Breite der schraffierten Linie ist abhängig von den Attributen *.collinewidth[l]* und *.rowlinewidth[l]*). Wird die Maus (mit gedrückter linker Maustaste)

bewegt, wandert die schraffierte Linie mit. Die Größenänderung wird abgeschlossen, indem die Maustaste losgelassen wird.

Die neuen Attributwerte der vergrößerten oder verkleinerten Zeilen/Spalten werden automatisch übernommen und das Tablefield entsprechend neu gezeichnet (Es werden die Attributwerte von *.rowheight[I]* und *.colwidth[I]* verändert). Dies ist konsistent zu Größenänderungen am Fenster.

An das Tablefield wird das indizierte Ereignis *resize[I,J]* geschickt, auf das reagiert werden kann. Jeweils eine Komponente von thisevent.index ist Null. Wurde eine Zeile verändert, so ist second(thisevent.index) Null und mit first(thisevent.index) kann die veränderte Zeile abgefragt werden. Wurde eine Spalte verändert, so kann die veränderte Spalte mit second(thisevent.index) erfragt werden.

Die Belegung im Überblick:

| Verändert wurde | first(thisevent.index) | second(thisevent.index) |
|---|---|---|
| Zeile | Index der veränderten Zeile | 0 |
| Spalte | 0 | Index der veränderten Spalte |

## 44.9 Hinweis zumTablefield unter Microsoft Windows

### 44.9.1 Auswahl der Anwendungscodepage

Die Codepage „utfwin“ eignet sich nicht als Anwendungscodepage, wenn Zeichenketten verarbeitet werden müssen, die <Carriage-Return>-Zeichen enthalten. Der Grund ist, dass <Linefeed>-Zeichen (Zeilenumbruch) in die Zeichenfolge <Carriage-Return><Linefeed> umgewandelt werden. Um doppelte <Carriage-Return>-Zeichen zu vermeiden, werden diese überlesen. Statt „utfwin“ sollten die Codepages „utf16“ oder „utf16l“ verwendet werden, die diese Spezialbehandlung nicht durchführen.

### 44.9.2 Verwendung von Farben und „Visual Styles“

Wenn beim ***Tablefield*** keine Farben gesetzt sind, verwendet der IDM die Systemfarben des jeweiligen Zustands. Bei *.fieldshadow* = *true* werden die „Visual Styles“ des Windows-Objekt „pushbutton“ verwendet. Er definiert die zugeordneten Farben. Wenn *.rowheadshadow* bzw. *.colheadshadow* angeschaltet sind, werden die Farben von den „Visual Styles“ des Windows-Objekts „headeritem“ übernommen.

Sind Farben gesetzt, werden diese verwendet. Bei einem aktiven Feld bestimmt *.fgc* die Hintergrundfarbe und *.bgc* die Vordergrundfarbe. Ist **nur eine** der beiden Farben gesetzt, wird für die **andere** Farbe die entsprechende Systemfarbe verwendet. Bei nur einer gesetzten Farbe werden also nicht Vorder- und Hintergrundfarbe vertauscht, sondern die gesetzte Farbe anders zugeordnet. Für ein aktives **und** insensitives Feld gibt es keine eigene Farbzuordnung. Es wird genauso wie ein aktives Feld dargestellt.

### Empfehlung

Setzen Sie entweder **beide** Farben – *.fgc* und *.bgc* – oder **keine** der beiden Farben.

### Anmerkung

Auch im IDM für Windows 2000 wurden die Farben bereits wie beschrieben zugeordnet. Allerdings wurden für aktive Objekte praktisch nur Vorder- und Hintergrundfarbe vertauscht, was auch bei nur einer gesetzten Farbe meistens ein gutes Ergebnis brachte. Erst die „Visual Styles“ (speziell des Windows-Pushbuttons) vertauschen zum Teil nicht mehr die Farben, sondern verwenden nur einen anderen Hintergrund.

## 44.10 Hinweis zumTablefield unter Motif

Das Motif-Tablefield ignoriert in den Tabellenzellen führende Zeilenumbrüche (ein oder mehrere \n am Anfang der Texte). Um dennoch Leerzeilen vor einem Text anzuzeigen, kann vor den Zeilenumbrüchen ein Leerzeichen in den Text eingefügt werden.

### Beispiel

" \nXYZ" statt "\nXYZ"

## 44.11 Hinweis zumTablefield unter Qt

In einem extrem kleinen Tablefield, bei dem die Kopfzeilen und -spalten nicht komplett dargestellt werden können, sind Scrollbars vorhanden. Diese haben allerdings keine Wirkung.

Das Tablefield besitzt unten und rechts eine Leerzeile bzw. Leerspalte, die den freien Bereich ausfüllen, wenn ganz nach unten bzw. rechts gescrollt wird. Wenn das Tablefield so dimensioniert ist, dass die letzte Inhaltszeile bzw. Inhaltsspalte nicht komplett dargestellt werden können, dann kann der Anwender auf diese Leerzeile bzw. Leerspalte scrollen. Diese Position kann programmatisch nicht abgefragt oder eingestellt werden.

Es wird das Selektionsmodell von Qt verwendet, daher kann es leichte Unterschiede zu den anderen Fenstersystemen geben. Das Verhalten ist dadurch aber Qt-typisch. Wenn beim Attribut .selection [enum] von den Elementen *sel_column*, *sel_row* und *sel_single* mehrere den Wert true haben, dann wird *sel_single* verwendet. Wenn keines der Element *sel_column*, *sel_row* und *sel_single* beim .selection[enum] Attribut den Wert true besitzt wird *sel_single* verwendet.

Es werden aktivierte Zellen deaktiviert, wenn auf eine Header-Zelle bei *.selstyle[sel_header] = false* geklickt wird.

Bei *.fieldfocusable = true* erfolgt auch bei Klick auf eine insensitive Zelle eine Umaktivierung. Es werden die entsprechenden *activate* und *deactivate* Ereignisse erzeugt, es kommt aber kein *select* Ereignis mit Index.

Folgende Attribute sind nicht implementiert:

- .colheadshadow
- .fieldshadow
- .rowheadshadow
- .xmargin
- .ymargin

## 44.12 Beispiel

Das nachfolgende Beispiel zeigt die Verwendung der Erweiterungen:

Die Überschriften des Tablefields sind interaktiv manipulierbar.

```
window
{
  .title "Title";

  child tablefield Tf1
  {
    .xauto 0;
    .xleft 2;
    .width 30;
    .xright 2;
    .ytop 2;
    .height 30;
    .colcount 5;
    .colsizeable[1] true;
    .rowsizeable[1] true;
    .rowheight[0] 5;
    .colheader 1;
    .rowheader 1;
  }
}

!!
!! Output of changed items into Tracefile
!!
on Tf1 resize
{
  print "row: " + itoa(first(thisevent.index)) + " has been changed";
  print "column: " + itoa(second(thisevent.index)) + " has been changed";
}
```

# 45 timer

Mit Hilfe dieses Objektes können Sie Aktionen zu festdefinierten Zeitpunkten mit festdefinierten Abständen durchführen. Z.B. kann damit an einem bestimmten Wochentag ein spezieller Prozess gestartet werden, oder an Termine (Tage, Stunden…) erinnert werden.

Sie können mit dem ***timer*** Zeiten in Monaten, Wochen, Tagen, Stunden, Minuten und Sekunden definieren. Ebenso sind Zeitabstände in diesen Einheiten definierbar.

### Definition

```
{ export | reexport } { model } timer { <Bezeichner> }
{
  <Objektspezifische Attribute>
}
```

Da Timer über das Message-Handling des Fenstersystems arbeiten, gibt es Situationen, in denen kein Timer-Ereignis auftreten kann. Vor allem auf den PC Fenstersystemen (Microsoft Windows) funktioniert der Timer in folgenden Situationen nicht:

- Ein Menü ist offen
- Eine zur Anwendung gehörende Messagebox ist offen

### Ereignisse

*select*

### Kinder

document

***record***

transformer

### Vater

***dialog***

***module***

### Menü

keins

## 45.1 Attribute

active

class

count

dialog

external

external[I]

firstrecord

incrtime

index

label

lastrecord

model

record[I]

recordcount

scope

starttime

## 45.2 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| active | Aktivität des Timers:<br>» *true*<br>Timer wird gestartet bzw. läuft gerade.<br>» *false*<br>Timer wird angehalten und zurückgesetzt bzw. ist gerade nicht aktiv. |
| count | Anzahl der Wiederholungen. Bei *0* wird der Timer unendlich wiederholt. |
| incrtime | Abstände zwischen den Aktivierungen des Timers (nur bei Zeitabstand). |
| starttime | Startzeitpunkt des Timers (bei absoluter Zeitdefinition oder beim Zeitabstand). |

### 45.2.1 Definition der Zeitabstände

Für die Zeitdefinition gibt es zwei Möglichkeiten:

» absolute Zeitdefinition

» relative Zeitdefinition

### Absolute Zeitdefinition

```
{<hour>}:{<minute>}{:{<second>}}
{<day>}.{<month>}.{<year>} {<hour>}:{<minute>}{:{<second>}}
{<month>}/{<day>}/{<year>} {<hour>}:{<minute>}{:{<second>}}
{<DoW>} {<hour>}:{<minute>}{:{<second>}}

DoW:  Wochentag (1 = Montag, 2 = Dienstag, .... , 7 = Sonntag)
```

### Zeitabstand

```
+ {<hour>}:{<minute>}{:{<second> {' <millisecond>}}
+ n d
+ n D
+ n m
+ n M
+ n y
+ n Y
```

| | |
|---|---|
| d, D: | Tage |
| m, M: | Monate |
| y, Y: | Jahre |

Wird bei der **absoluten Zeitdefinition** ein Wert nicht mitangegeben, wird als Default der entsprechende Wert aus der aktuellen Zeit, dem "Jetzt", übernommen.

Werden die **Sekunden** nicht angegeben, werden sie bei fehlendem : auf 0, bei angegebenem : auf die Sekunden der aktuellen Zeit gesetzt.

Für jeden **Monat** sind Tageswerte bis 31 erlaubt. Hat der Monat weniger Tage, so wird der letzte Tag als letzter Tag des Monats angenommen.

Als **Jahre** können Werte von 1901 bis 2099 angegeben werden. Alternativ sind Werte von 0 bis 99 erlaubt, die auf den Zeitraum 1970 bis 2069 abgebildet werden.

Die Kennzeichnung der Millisekunden wird mit dem Zeichen ' eingeleitet.

### Beispiel

```
.incrtime "+0:0:0'5";
```

Der Timer schickt jede halbe Sekunde einen Select-Event.

### Hinweise

Aus Systemgründen können die Windows-Systeme keine bessere Auflösung als 55 Millisekunden. Der Dialog Manager wählt die nächst mögliche höhere Zeiteinheit.

Die Angabe von Millisekunden ist nur bei relativen Zeitangaben unter einer Minute möglich. Bei absoluten Zeitangaben liefern Millisekunden eine Fehlermeldung. Bei relativen Zeitangaben über eine Minute werden Millisekunden ignoriert - es erfolgt keine Fehlermeldung.

**Beispiele**

» Jeden Freitag um 12.00 Uhr

```
.starttime  "5 12:00";
.incrtime   "+7D";
```

» Jeden Tag um 16.00 Uhr

```
.starttime  "16:00";
.incrtime   "+1D";
```

» Weihnachten 19.00 Uhr

```
.starttime  "24.12 19:00";
.incrtime   "+1Y";
```

» 4 Stunden nach Programmstart

```
.starttime  "+04:00";
```

## 45.3 Beispiel

Verwendung eines ***timers*** für eine Autosave-Funktion:

```
timer TiAutoSave { }

window WnAutoSave
{
  child checkbox CbAutoSave { }
  !! Timer is activated and deactivated with this checkbox.

  child edittext EtMin { }
  !! Includes the number of minutes after which the
  !! select event for the timer is to be sent.

  child edittext EtFileName { }
  !! Name of the file in which is to be stored.
}

on CbAutoSave select
{
  if CbAutoSave.active
  then
    CbAutoSave.text     := "yes";
    TiAutoSave.incrtime := (("+:" + EtMin.content) + ":");
    TiAutoSave.active   := true;
  else
    CbAutoSave.text   := "no";
    TiAutoSave.active := false;
  endif
}
```

```
on TiAutoSave select
{
  !! Rule R_SaveFile(EtFileName.content) is called at each
  !! select event.
}
```

# 46 toolbar

Das ***toolbar***-Objekt dient als Gruppierungsobjekt um eine Toolbar (Symbolleiste, Funktionsleiste) zu definieren. Nur Fenster können eine oder mehrere Toolbars als Kinder beinhalten.

Eine Toolbar kann entweder ans Vater-Fenster angedockt sein oder als eigenständiges Tool-Window erscheinen.

Die Hintergrundfarbe der Toolbar entspricht der Menüfarbe, die Kindobjekte besitzen keinen Fokusrahmen, lassen sich also nur über Maus oder einen Accelerator bedienen. Ansonsten zeigt ein Rahmen bei angedockten Toolbars ihren Geltungsbereich an.

### Definition

```
{ export | reexport } { model } toolbar { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Objektspezifische Attribute>
}
```

### Ereignisse

*activate*

(nur Toolbar als Fenster)

*close*

*cut*

*deactivate*

(nur Toolbar als Fenster)

*extevent*

*move*

*paste*

*resize*

### Kinder

***canvas***

***checkbox***

document

***edittext***

***groupbox***

***image***

***layoutbox***

***listbox***

***menubox***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***statictext***

***tablefield***

transformer

***treeview***

**Vater**

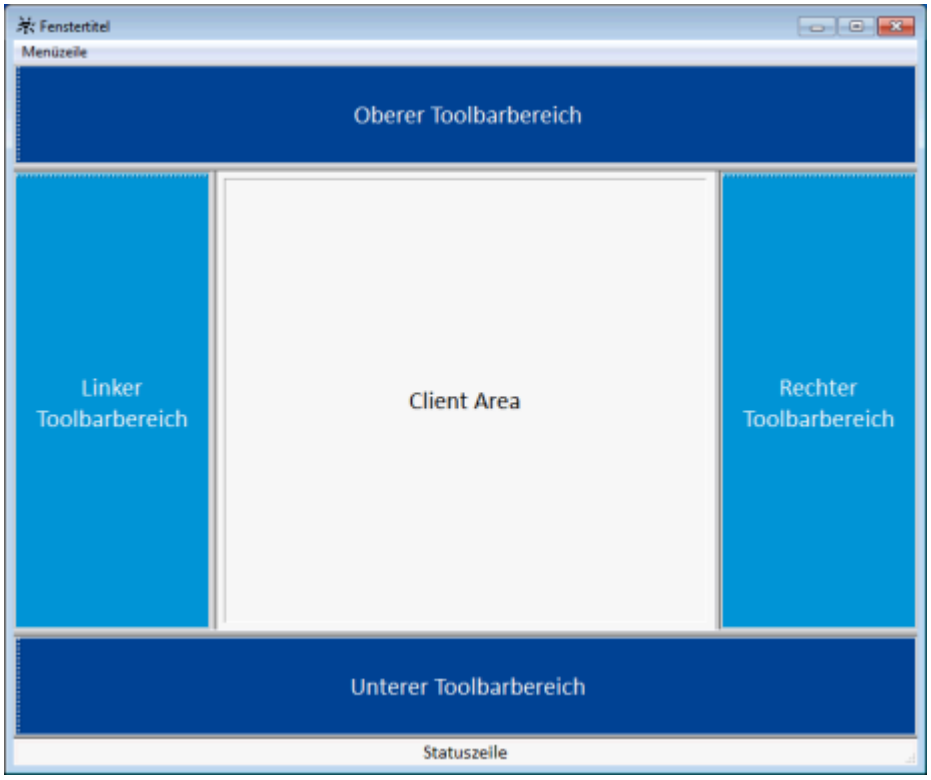
***module***

***window***

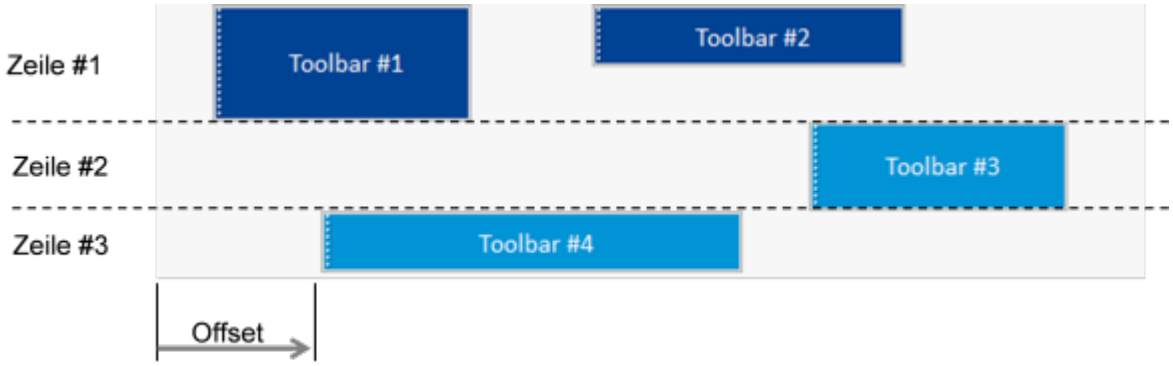
**Menü**

***menubox***

Das nachfolgende schematisierte Bild eines Fensters zeigt die Anordnung der einzelnen Bereiche. Das Hinzufügen und Entfernen von Toolbars verändert allerdings nicht die Größe des Fensters, sondern beeinflusst die Größe des Client-Bereichs, in dem der eigentliche Fensterinhalt dargestellt wird.




Innerhalb eines Toolbarbereichs können eine oder mehrere Toolbars auf mehrere Zeilen oder (exklusiv) Spalten verteilt liegen. Die Größe dieses Bereichs ist dabei abhängig von der Anzahl, Größe und Verteilung der Toolbars, eine Zeile ohne Toolbars gibt es aber nicht.




Der Offset wird entweder vom linken oder vom oberen Rand des Toolbarbereichs genommen, je nachdem ob es eine horizontale (dock_up, dock_down) oder vertikale (dock_left, dock_right) Toolbar ist.

Bei eingedockten Toolbars werden Move-Ereignisse nur erzeugt, wenn sich die Attribute dock_offset oder dock_line ändern. Es gibt z.B. kein Move-Ereignis, wenn eine unten eingedockte Toolbar nach unten verschoben wird, aber in derselben Toolbar-Zeile bleibt. Bei einer indirekten Änderung von .dock_offset oder .dock_line – zum Beispiel infolge der Vergrößerung einer anderen Toolbar – wird die Änderung nur virtuell ausgeführt, um die definierte Position nicht permanent zu ändern. Dadurch nehmen Toolbars wieder ihre definierte Position ein, sobald der Platz dafür vorhanden ist.

## 46.1 Attribute

accelerator

active

autoalign

autosize

bgc

bordercolor

borderraster

borderstyle

borderwidth

child[l]

childcount

class

cursor

cut_pending

cut_pending_changed

dialog

dockable[enum]

docking

dock_line

dock_offset

external

external[l]

fgc

firstchild

firstrecord

focus

focus_on_click

font

function

height

height[class]

help

label

lastchild

lastrecord

mapped

maxheight

maxwidth

minheight

minwidth

model

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

record[l]

recordcount

scope

sensitive

sizeable

sizeable[class]

sizeraster

tile

tilestyle

title

titlebar

titlebgc

titlefgc

toolhelp

userdata

visible

width

width[class]

window

xauto

xleft

xraster

xright

yauto

ybottom

yraster

ytop

## 46.2 Spezifische Attribute

Eine Besonderheit der toolbarspezifischen Attribute ist, dass sie sich gegenseitig stark beeinflussen. Die meisten geerbten Attribute verhalten sich, wie von anderen Objekten, zum Beispiel dem Fenster, gewohnt.

Toolbarspezifische Besonderheiten zu den einzelnen Attributen entnehmen Sie bitte der Tabelle und den folgenden Unterkapiteln.

| Attribut | Beschreibung |
|---|---|
| autoalign | Schaltet das automatische Anordnen des Objekts hinter der letzten Toolbar bzw. am Anfang der Zeile oder Spalte. |
| autosize | Automatisches Vergrößern der Toolbar in der Ausrichtung um die Toolbar-Zeile oder Toolbar-Spalte komplett auszufüllen. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“.<br>Nur aktiv, wenn *.docking* = *dock_window* **oder** *.borderwidth* > *0*. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a).<br>Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |

| Attribut | Beschreibung |
|---|---|
| dockable[enum] | Steuert, an welchen Stellen sich eine Toolbar befinden kann.<br>Mögliche Indizes:<br>» *dock_window* (Standard)<br>» *dock_up*<br>» *dock_down*<br>» *dock_left*<br>» *dock_right* |
| docking | Bestimmt den momentanen Dock-Zustand des Objekts.<br>Mögliche Werte:<br>» *dock_window* (Standard)<br>» *dock_up*<br>» *dock_down*<br>» *ock_left*<br>» *dock_right*<br>Allerdings darf bei einer Instanz nur ein Wert gesetzt werden, für den das Attribut dockable[enum] *true* ist. |
| dock_line | Bestimmt die Reihenfolge mehrerer Toolbars in einem Fenster. |
| dock_offset | Gibt den Abstand des Objekts zum rechten bzw. oberen Rand des Vaterfensters an. |
| focus_on_click | Legt fest, ob ein Mausklick in den Clientbereich das Objekt aktiviert, d.h.den Fokus auf das Objekt setzt.<br>Siehe auch Kapitel „.focus_on_click“. |
| height | Höhe des Objekts.<br>Besonderheiten bei der Vererbung siehe Kapitel „.height, .width“. |
| height[class] | Höhe des Objekts für einen bestimmten Zustand.<br>Indizes:<br>» *toolbar*<br>» *window*<br>Siehe auch Kapitel „.height[class], .width[class]“. |

| Attribut | Beschreibung |
|---|---|
| maxheight | Maximale Größe des Objekts.<br>Siehe auch Kapitel „.minwidth, .maxwidth, .minheight, .maxheight“ und das Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| maxwidth | Maximale Breite des Objekts.<br>Siehe auch Kapitel „.minwidth, .maxwidth, .minheight, .maxheight“ und das Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| minheight | Minimale Höhe des Objekts.<br>Siehe auch Kapitel „.minwidth, .maxwidth, .minheight, .maxheight“ und das Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| minwidth | Minimale Breite des Objekts.<br>Siehe auch Kapitel „.minwidth, .maxwidth, .minheight, .maxheight“ und das Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>*opt_center_toolhelp* (nur MS Windows) |
| sizeable | Gibt an, ob das Objekt interaktiv in der Größe geändert werden kann (Default-wert).<br>Siehe auch Kapitel „.sizeable und .sizeable[class]“. |
| sizeable[class] | Gibt für den jeweiligen Index an, ob das Objekt interaktiv in der Größe geändert werden kann.<br>Indizes:<br>» *toolbar*<br>» *window*<br>Siehe auch Kapitel „.sizeable und .sizeable[class]“. |
| tile | Definiert das Hintergrundbild des Objekts. |
| tilestyle | Gibt die Art und Weise an, wie das in tile definierte Hintergrundbild dargestellt wird. |
| title | Angabe des Fenstertitels (bei ausgedocktem Zustand). |
| titlebar | Definiert, ob im ausgedockten Zustand eine Titelleiste dargestellt wird. |
| width | Breite des Objekts.<br>Besonderheiten bei der Vererbung siehe Kapitel „.height, .width“. |

| Attribut | Beschreibung |
|---|---|
| width[class] | Breite des Objekts für einen bestimmten Zustand.<br>Indizes:<br>» *toolbar*<br>» *window*<br>Siehe auch Kapitel „.height[class], .width[class]“. |
| xauto | Gibt die vom IDM bei der horizontalen Ausrichtung automatisch ermittelte Größe an.<br>Nur bei Toolwindows, bei angedockten Toolbars wird das Attribut ignoriert. |
| xleft | Abstand von links.<br>Nur bei Toolwindows, bei angedockten Toolbars wird das Attribut ignoriert. |
| xright | Abstand von rechts.<br>Nur bei Toolwindows, bei angedockten Toolbars wird das Attribut ignoriert. |
| yauto | Gibt die vom IDM bei der vertikalen Ausrichtung automatisch ermittelte Größe an.<br>Nur bei Toolwindows, bei angedockten Toolbars wird das Attribut ignoriert. |
| ybottom | Abstand von unten.<br>Nur bei Toolwindows, bei angedockten Toolbars wird das Attribut ignoriert. |
| ytop | Abstand von oben.<br>Nur bei Toolwindows, bei angedockten Toolbars wird das Attribut ignoriert. |

### 46.2.1 .dockable, .docking[enum]

Bei den Attributen .dockable und .docking[enum] wird eine Vererbung auf eine Instanz, die zu Inkonsistenzen führen würde, ignoriert.

Dies geschieht auch, wenn sich durch die Vererbung des Attributs .docking die Größenattribute des Vaterfensters der Toolbar-Instanz ändern würden und das Vaterfenster sichtbar aber nicht sizeable ist.

Die Vererbungsabhängigkeit bleibt als solche trotzdem bestehen, d.h. es ist kein Aufruf von setinherit () erforderlich, damit zukünftige Wertänderungen am Modell wieder vererbt werden.

### 46.2.2 .focus_on_click

Mit Hilfe des Attributs .focus_on_click kann eine Toolbar definiert werden, welche beim Anklicken per Maus den Fokus nicht aus dem zugehörigen Fenster herausnimmt. Da im ISA Dialog Manager jedoch auch alle anderen, innerhalb der Toolbar befindlichen sensitiven Kindobjekte den Fokus bekommen

können, ist dieses Attribut auch bei den Objekten Groupbox, Image, Rechteck und Statictext vorhanden. Allerdings ist dabei zu beachten, dass eine Groupbox mit *.focus_on_cklick = false* keine Elemente enthält, welche einen Focus bekommen können oder sogar zu Eingabezwecken benötigen (z.B. editierbarer Text).

Das Attribut kann nur unter MS Windows genutzt werden.

### 46.2.3 .height, .width

Diese Attribute sind die Default-Werte für width[class] bzw. height[class]. Das .width/.height-Attribut verhält sich damit analog zu normalen (benutzerdefinierten) Attributen mit Default-Wert. Man kann somit die Größe für den eingedockten wie auch ausgedockten Fall gleichzeitig setzen.

Dabei werden dann auch für diese Attribute changed-Ereignisse erzeugt.

Ist die Toolbar angedockt und das Vaterfenster sichtbar und nicht sizeable, kann bei einer horizontal angedockten Toolbar die Höhe und bei einer vertikal angedockten Toolbar die Breite nicht verändert werden, da sich sonst das .width- bzw. .height-Attribut des Vaterfensters ändern würde.

### 46.2.4 .height[class], .width[class]

Das Attribut height[class] gibt die Höhe der Toolbar im angedockten (class = toolbar) oder ausgedockten Zustand an (class = window). Wird im angedockten Zustand .height[toolbar] oder im ausgedockten Zustand .height[window] geändert so gibt es auch in diesem Fall für .height ein changed-Ereignis, allerdings mit dem entsprechenden Index.
Für width[class] gilt dasselbe.

### 46.2.5 .minwidth, .maxwidth, .minheight, .maxheight

Die Attribute haben dieselbe Bedeutung wie beim Fenster und sind sowohl im an- als auch im ausgedockten Zustand gültig. Der Programmierer muss deswegen beim Umdocken (z.B. von oben nach rechts) diese Attribute gegebenenfalls anpassen.

### 46.2.6 .sizeable und .sizeable[class]

Das Attribut sizeable[class] verhindert interaktive Größenveränderungen durch den Anwender. Es ist ein Feldattribut wie *.height[class]* mit Defaultwert bei sizeable. Man kann die interaktive Größenveränderung getrennt für die unterschiedlichen Docking-Zustände bestimmen und *.sizeable* wirkt sich im eingedockten wie auch im ausgedockten Zustand aus.

### 46.2.7 .xleft, .xright, .xauto, .ytop, .ybottom, .yauto

Diese Attribute beziehen sich nur auf Toolwindows, bei angedockten Toolbars werden sie ignoriert.

## 46.3 Interaktive Bedienung

Die Kindobjekte einer Toolbar verhalten sich wie in einem Fenster.

Die interaktive Bedienung der Toolbar beschränkt sich gegenwärtig bei einem Toolwindow auf Schließen, Verschieben und Vergrößern/Verkleinern mit der Maus oder dem Systemmenü und bei jeder Toolbar auf das an- und ausdocken mit einem Doppelklick (linke Maustaste) in die freie Toolbarfläche oder den Nonclient-Bereich.

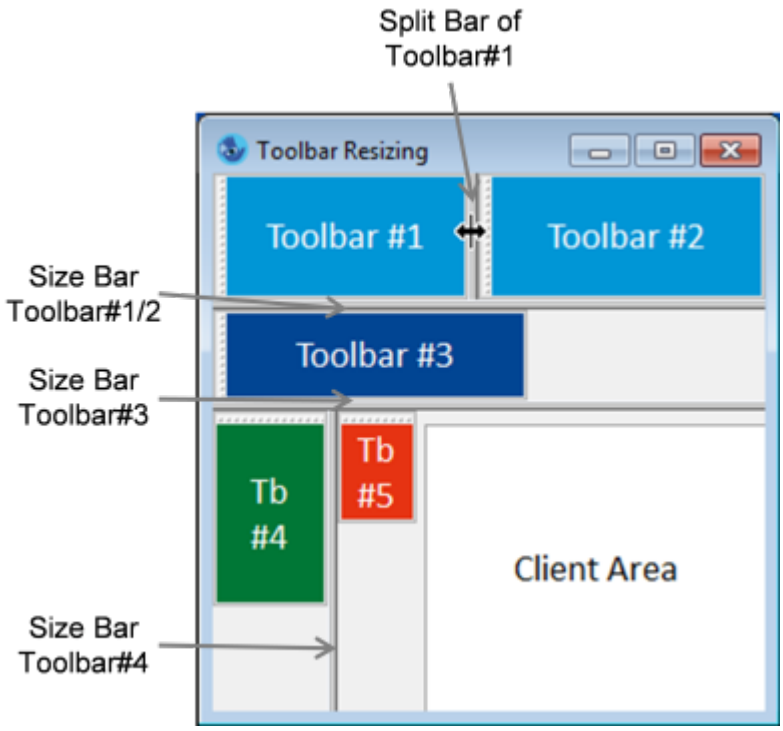
Wird eine Toolbar per Doppelklick an- oder ausgedockt, springt sie dorthin, wo sie sich zuletzt im aus- bzw. angedockten Zustand befunden hat. Beim Andocken einer Toolbar, die noch nie zuvor angedockt war, wird defaultmäßig oben angedockt.

Das interaktive An- und Ausdocken kann durch das Attribut .dockable blockiert werden. Ist das Vaterfenster sichtbar und nicht sizeable, kann ebenfalls nicht an- oder ausgedockt werden.

Der Griff angedockter Toolbars hat noch keine Funktion.

## 46.4 Interaktives Toolbar-Resizing

Für das Resizing von Objekten existiert das .sizeable-Attribut. Dies ist an der Toolbar ebenso vorhanden, wird aber nur im ausgedockten Fall (analog zum Fenster) verwendet. Im Folgenden geht es um den Fall des Resizings von eingedockten Toolbars.




Jede eingedockte, in ihrer Größe veränderbare Toolbar erhält Bars um ihre Größe zu verändern. Bars die innerhalb der Toolbar-Zeile/Spalte hintereinander liegende Toolbars trennen werden im Folgenden als Splitter-Bars bezeichnet. Zur Unterscheidung werden Bars, welche die Höhe einer

Toolbarzeile bzw Breite einer Toolbarspalte beeinflussen, Sizer-Bars genannt. Veränderungen an Sizer-Bars beeinflussen gleich mehrere Toolbars sowie die Größe des Client-Bereichs. Splitter-Bars wirken sich erst mal nur auf die Größe einer Toolbar aus und als Folge davon auf die anderen Toolbars der Zeile/Spalte und u.U. auch den Clientbereich (Stichwort: Umbruch).

Im obigen Bild sind alle Toolbars bis auf die kleinste Toolbar#5 resizable. Splitter-Bars und Sizer-Bars sind hier wie folgt vorhanden:

- Sizer-Bars rechts bzw. unterhalb (entsprechend der Ausrichtung) der Toolbar, außer für die letzte Toolbar der Zeile/Spalte.
- Splitter-Bars wenn eine Toolbar innerhalb einer Zeile/Spalte resizable ist. Diese befinden sich an der Toolbar-Seite die dem Client-Bereich am nächsten liegt.

Die Bars werden wie die Splitbars des splitbox-Objektes betätigt. Das heißt:

- Visuelles Feedback wenn Maus über Bar-Bereich durch einen veränderten Cursor (horizontaler bzw. vertikaler Doppelpfeil).
- Wird die Maus über einem Bar-Bereich gedrückt (linke Maustaste) erfolgt ein visuelles Feedback durch eine „gerasterte“ Barlinie die sich beim Bewegen der Maus mit gedrückter Maustaste in der Toolbar-Zeile bzw. Spalte mitbewegt.
- Wird die Maus losgelassen erfolgt schließlich das Resizing.

Konsequenterweise erfolgt die Versendung eines resize-Events für Toolbars deren Größe sich geändert hat (width/height-Attribut ist geändert).

Die min/max-Werte werden nur für sizable-Toolbars berücksichtigt. Dies ist wie folgt definiert:

- Interaktives Resizing: Toolbar-Zeile/Spalte kann nicht kleiner werden als die größte Min-Größe und nicht größer werden als die kleinste Max-Größe. Außnahmen gibt es hier natürlich bei unvereinbaren Min/Max-Größen bzw. bei Fenstergrößen die dies nicht zulassen.
- Natürlich bei der Arrangierung/Layout der Toolbars

### 46.4.1 Automatisches Sizing/Positionieren von Toolbars

Zusätzlich gibt es zwei sinnvolle und notwendige Features:

- .autoalign = true
  Automatisches Positionieren von eingedockten Toolbars hinter der letzten bzw. ganz an den Anfang.
- .autosize = true
  Automatisches Vergrößern von Toolbars sodass diese die ganze Zeile/Spalte füllen. Ausgehend von der letzten Toolbar der Zeile/Spalte werden Toolbars die noch Spielraum haben vergrößert. Dies ist unabhängig von sizable. Mit Ausfüllen ist **nur** die Größenänderung entsprechend der Docking-Ausrichtung gemeint; Breite bei horizontaler Ausrichtung und Höhe bei vertikaler Ausrichtung.

## 46.5 Koordinaten und Größe

Die Toolbars und zwischen ihnen liegende freie Bereiche machen den Toolbarbereich des Vaterfensters aus. Dieser Bereich liegt außerhalb des Clientbereichs vom Fenster und wird durch einen Rahmen von diesem optisch getrennt. Menü und Statusbar liegen, soweit vorhanden, weiter außen, als der Toolbarbereich, Scrollbars weiter innen. Bei überlappenden Toolbars geht die Priorität von oben nach unten und dann von links nach rechts.

Toolbars unterstützen sowohl Größen- als auch Positionsraster. Ist beim Vaterfenster ein Größenraster eingeschaltet, so werden die Toolbarbereiche auf Rasterweite ausgedehnt. Dadurch entstehen im Toolbarbereich Lücken zwischen den Toolbars und dem Rahmen des Bereichs.

Ist ein Fenster sichtbar, so wird beim An-, Aus- oder Umdocken einer Toolbar, beim Erzeugen oder Zerstören, Sichtbar- oder Unsichtbarmachen angedockter Toolbars oder bei Größenänderungen einer Toolbar der Clientbereich des Vaterfensters in der Größe angepasst, d.h. die Werte der Attribute .width und .height am Vaterfenster ändern sich. Dadurch wird ein "Springen" des Fensters beim interaktiven An- und Ausdocken von Toolbars vermieden. Deswegen wird an das Fenster-Objekt ein resize Ereignis verschickt (natürlich wird dabei auch ein move Ereignis an die Toolbar selbst geschickt).

Ist das Fenster dagegen nicht sichtbar, so bleibt die Größe des Clientbereichs erhalten und es wird stattdessen die äußere Fenstergröße angepasst.

Zu diesem Standardverhalten gibt es Ausnahmen:

- Ist das Vaterfenster sichtbar aber nicht sizeable, können keine Toolbars an- oder ausgedockt werden. Aber Achtung, sonst gilt nämlich allgemein: Wenn Toolbars aus der Regelsprache heraus manipuliert werden, kann sich der Clientbereich des Vaterfensters auch dann ändern, wenn das Fenster nicht sizeable ist!
- Sind an einem sichtbaren Vaterfenster Maxi- und Minimalgrößen für den Clientbereich angegeben, und würden diese Werte durch das Ausdocken/Zerstören bzw. durch das Andocken/Erzeugen einer Toolbar über- bzw. unterschritten, so wird entgegen dem Standardverhalten die Außengröße des Fensters so weit wie nötig angepasst.
- Sind am Vaterfenster keine Max-Attribute definiert, dafür aber eine virtuelle Größe, so wird auch hier beim Ausdocken/Zerstören einer Toolbar die Außengröße des Fensters ggf. so verringert, dass der Clientbereich nicht über die virtuelle Größe hinaus anwachsen kann.

## 46.6 Besonderheiten beim Focusing

Wichtig: Anders als bei Toolbars üblich, bekommen Objekte in IDM-Toolbars einen Fokus. Diese Eigenschaft lässt sich unter Microsoft Windows über das .focus_on_click für die Toolbar selbst und die Objekte Groupbox, Image, Rechteck und Statictext beeinflussen. Unter Motif ist dieses Attribut nicht vorhanden.

Das Focusing-Verhalten der Toolbar unterscheidet sich nur unwesentlich von dem des Fenster-Objekts.

Falls eine Toolbar ausgedockt ist (Toolwindow), kann sie mit einem Mausklick aktiviert werden. Dann wechselt auch der Fokus auf das Kind der Toolbar, das zuletzt den Fokus hatte. Hatte bis dahin keines der Kinder je einen Fokus, bekommt das erste Kind den Fokus. Sollte die Toolbar kinderlos sein, bekommt niemand den Fokus, was sich darin äußert, dass kein Objekt auf dem Bildschirm mehr einen Fokusrahmen aufweist.

Falls eine Toolbar angedockt ist, kann sie nicht aktiviert werden, da in diesem Fall das entsprechende Vaterfenster aktiv ist. Trotzdem ist es möglich, mit einem Mausklick auf die Toolbar den Fokus in diese zu ziehen bzw., um genauer zu sein, auf eines der Kinder zu setzen. Welches Kind den Fokus bekommt, ist genauso geregelt, wie schon oben beschrieben wurde. Sollte die Toolbar keine Kinder besitzen, wechselt der Fokus zum (bzw. bleibt auf dem) einen Kind des Vaterfensters. Die Toolbar selbst kann in diesem Fall keinen Fokus bekommen.

Soll das Fokusobjekt einer eingedockten Toolbar bei einer Fensteraktivierung erhalten bleiben, so muss an der Toolbar das Attribut *.focus_on_click = true* gesetzt sein. Ein Klick in die Arbeitsfläche des Vaterfensters setzt den Fokus allerdings in jedem Fall auf ein Kind des Vaterfensters.

Innerhalb einer Toolbar kann der Fokus wie üblich mit der `Tab`-Taste von einem Kind zum anderen weitergereicht werden und zwar in der Reihenfolge, in der die Kinder im Dialog erzeugt wurden. Wird dabei das letzte Kind erreicht, springt der Fokus wieder zurück zum ersten Kind.

Befindet sich der Fokus auf einem Kindobjekt einer Toolbar und wird die Toolbar insensitiv geschaltet, so wird das zuletzt aktive Fenster aktiviert und der Fokus wechselt auf ein Kind des Fensters, bzw. zum Fenster selbst, wenn dieses keine Kinder hat.

## 46.7 Anmerkung Windows 10 und 11 mit mehreren Monitoren:

Unter WINDOWS kann es nach dem Hinzufügen/Entfernen eines Monitors oder dem Ändern der Anzeigeeinstellungen zu folgenden Problemen kommen:

- Es erscheint kein Verschieberahmen beim Verschieben eines **toolbar** Objektes. Es kann auch vorkommen, dass ein Rechteck in der Größe des **toolbar** Verschieberahmens mit einem falschen Hintergrund auf einem anderen Monitor dargestellt wird.
- Ausgedockte **toolbar** Objekte passen sich nicht an Änderungen der DPI Wertes an. Sowohl beim Verschieben auf einen anderen Monitor oder auch beim Ändern der Vergrößerung.

Das Problem liegt an WINDOWS, das zum einen die Koordinaten beim Zeichnen auf den Desktop falsch umsetzt und zum anderen wichtige Nachrichten (*WM_DPICHANGED*) nicht an sogenannte Tool-Fenster versendet und den Window DPI Wert nicht ändert.

Nachdem der Bildschirmschoner aktiv war oder man sich neu angemeldet hat (und nach einem Neustart), tritt das Problem nicht mehr auf.

## 46.8 Besonderheiten der Toolbar unter Motif

Die interaktive Repositionierung der ausgedockten Toolbar erlaubt kein Eindocken durch Verschieben der Titelzeile.

## 46.9 Besonderheiten der Toolbar unter Qt

Die ***Toolbar*** kann unter Qt zwei unterschiedliche Ausprägungen haben, die über das Attribut *.style* eingestellt werden. Standardmäßig ist der Stil *toolbar* aktiv, der optisch den bekannten Toolbars entspricht.

***Tabelle 1:*** *Attribut .style der Toolbar*

| Wert | Beschreibung |
|---|---|
| *.style = toolbar* (Standardwert) | Herkömmliche Toolbar, entsprechend der bekannten IDM-Toolbar. |
| *.style = notepage* | Toolbars verwenden einen Dock-Bereich, der zwischen Toolbars und innerem Bereich liegt. „Tabbed Toolbars“ und „Nested Toolbars“ sind möglich (siehe Kapitel „Besonderheiten des Fensters unter Qt“ beim Objekt ***window***). |

„Abbildung 46“ zeigt die Unterteilung eines Fensters mit verschiedenen Toolbars im eingedockten und ausgedockten Zustand. „Tb16“ und „Tb18“ sowie „Tb19“ und „Tb20“ sind „Tabbed Toolbars“, die sich ihren Platz teilen. „Tb17“ ist eine „Nested Toolbar“, die in die zweite Reihe des rechten Bereichs eingebettet ist.

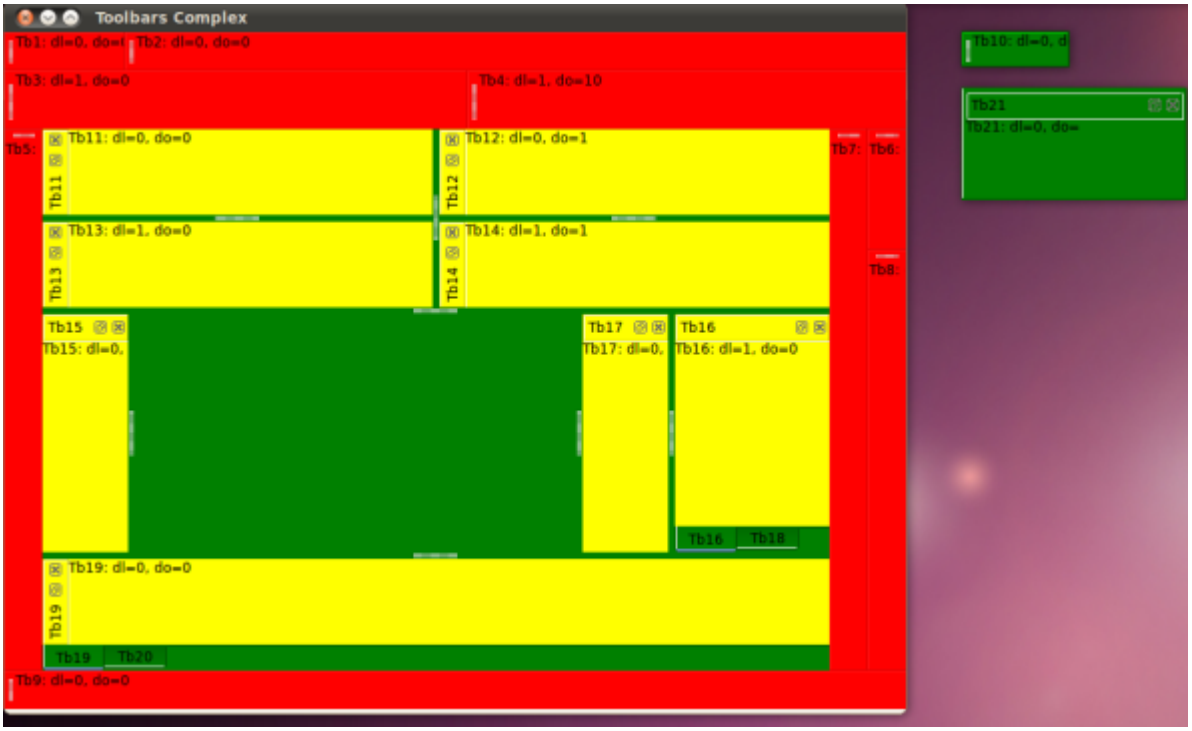



***Abbildung 46:*** *Toolbar-Arten im IDM für Qt*

Der Stil *notepage* erlaubt ein Verschachteln mehrerer ***Toolbars*** in einem Dockbereich und ihre Anordnung in Form von Registerkarten (siehe Kapitel „Besonderheiten des Fensters unter Qt“ für die entsprechenden Steuerungsoptionen). Die Toolbars haben dann eine Titelzeile und können über ihre Titelschaltflächen ausgedockt und geschlossen werden.

***Toolbars*** unterschiedlichen Stils, die im selben Dockbereich definiert werden, können nicht gemischt werden und werden entsprechend ihrem Stil gruppiert angeordnet. Dabei werden Toolbars mit dem Stil *toolbar* immer am äußeren Fensterrand positioniert und Toolbars mit dem Stil *notepage* immer zwischen dem Clientbereich des Fensters und den Toolbars mit dem Stil *toolbar* (siehe „Abbildung 47“).

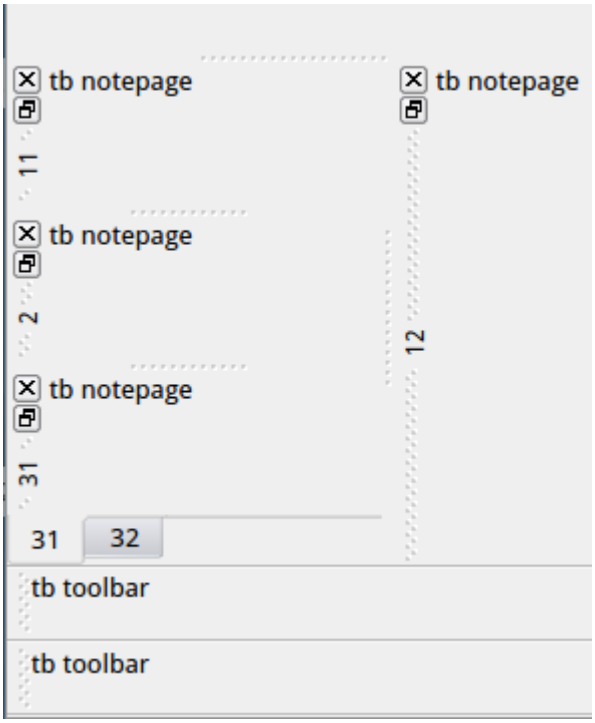



***Abbildung 47:*** *Toolbars verschiedenen Stils im unteren Dockbereich (dock_down)*

Toolbars können nicht mit Abständen zwischen ihnen positioniert werden. Das Attribut *.autoalign* hat keine Funktion.

Das Attribut *.sizeable* verhält sich bei *true* wie das Attribut *.autosize*, das heißt die gesamte verfügbare Breite oder Höhe wird ausgefüllt. Bei *false* wird die gesetzte Größe unter allen Umständen beibehalten.

Ausgedockte Toolbars mit dem Stil *toolbar* können nicht per Maus in der Größe verändert werden und erzeugen beim Verschieben keine *move*-Ereignisse.

Eingedockte Toolbars mit dem Stil *toolbar* und *.sizeable* = *true* können je nach ihrer Ausrichtung nur in einer Richtung manuell vergrößert werden: Horizontale Toolbars nur in der Breite, vertikale Toolbars nur in der Höhe. Die Größe in der anderen Richtung wird jeweils von der größten Toolbar in der selben Zeile bzw. Spalte bestimmt. Eine kleinere Toolbar wird von Qt automatisch an die Höhe bzw. Breite der höchsten bzw. breitesten Toolbar angepasst. Daher wird empfohlen, bei *.sizeable* = *true* keine unterschiedlichen Höhen bzw. Breiten innerhalb einer Toolbarzeile bzw. -spalte zu definieren.

Außerdem ist ein manuelles Vergrößern in einer Richtung nur an den Toolbar-Übergängen möglich.

Toolbars mit demStil *notepage* und *.sizeable* = *true* ignorieren je nach ihrer Ausrichtung die gesetzte Breite (*.width* bei vertikaler Ausrichtung) bzw. Höhe (*.height* bei horizontaler Ausrichtung). Sie werden

immer in ihrer Minimalbreite bzw. -höhe angezeigt. Daher wird empfohlen, immer auch das Attribut *.minwidth* bzw. *.minheight* zu setzen, andernfalls werden von Qt bestimmte Standardwerte verwendet.

Toolbars werden **nicht** automatisch in die nächste Zeile oder Spalte umgebrochen, wenn sie nicht alle in die vorgegebene Fensterbreite oder -höhe passen. Abhängig von *.sizeable* kann dabei folgendes Verhalten auftreten:

» *.sizeable = true*
Breiten bzw. Höhen der Toolbars werden von Qt ignoriert und so verkleinert, dass alle Toolbars in der vorgegebenen Fensterbreite bzw. -höhe Platz finden.

» *.sizeable = false*
Qt vergrößert wenn notwendig automatisch die Breite bzw. Höhe des Fensters um allen Toolbars einer Zeile Platz zu bieten. Die gesetzte Fensterbreite bzw. -höhe wird von Qt also außer Kraft gesetzt.

Eine neue Zeile oder Spalte kann nur über das Attribut *.dock_line* erzwungen werden.

Bei ***Toolbars*** mit dem Stil *notepage* hängt die Positionierung von den ***Window***-Optionen *opt_nested_docks* und *opt_tabbed_docks* ab. Sind beide Optionen deaktiviert (Standard), dann wird *.dock_line* ignoriert und alle Toolbars werden nebeneinander (horizontal) oder übereinander (vertikal) angeordnet.

Ist die Option *opt_nested_docks* aktiv, dann wird *.dock_line* beachtet. Die Schachtelung erfolgt immer relativ zur vorherigen Toolbar desselben Dockbereichs. In „Abbildung 48“ haben die Toolbars „eins“, „zwei“ und „drei“ jeweils *.dock_line = 1*, „vier“ hat *.dock_line = 2* und nistet sich bei „drei“ ein.

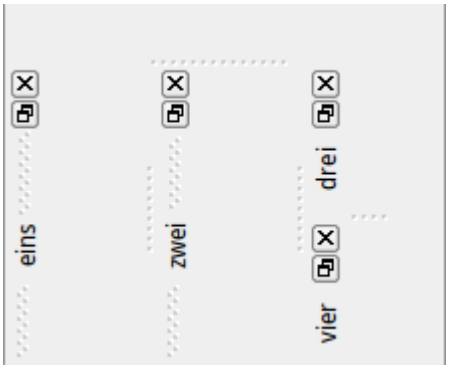



***Abbildung 48:*** *Toolbars mit dem Stil notepage bei aktivem opt_nested_docks*

Ist die Option *opt_tabbed_docks* aktiv, dann werden alle Toolbars mit derselben *.dock_line* als hintereinander liegende Registerkarten mit Reitern dargestellt („Abbildung 49“).


***Abbildung 49:*** *Toolbars mit dem Stil notepage bei aktivem opt_tabbed_docks*

Je nach UI-Stil wird bei *.style = notepage* unter Umständen keine Abgrenzung zum Clientbereich dargestellt.

Beim Wechsel zwischen horizontalem und vertikalem Eindocken werden Breite und Höhe nicht vertauscht, da je nach Art der Toolbar Ausrichtungswechsel bereits während des Verschiebens auftreten können.

Die Attribute *.height[]* und *.width[]* werden nur im Stil *notepage* unterstützt und angewendet.

### 46.9.1 Verhalten

Bei „Tabbed Toolbars“ wird beim Aktivieren eines Tabs bzw. dem damit verbundenen Sichtbarmachen ein *activate*-Ereignis verschickt (analog zu Notebook und Notepage).

Das Verschicken von Ereignissen wie *activate*, *deactive*, *close*, *move* und *resize* hängt stark vom verwendeten Qt-Toolbar-Objekt ab. Interne Zustände wie „Benutzer ist gerade noch am Verschieben“ können nicht berücksichtigt werden.

„Tabelle 2“ fasst Unterschiede und Einschränkungen der ***Toolbar*** des IDM FÜR QT zusammen.

***Tabelle 2:*** *Unterschiede und Einschränkungen der Toolbar*

| .style | Bedingungen | Einschränkungen und Unterschiede |
|---|---|---|
| alle | | *.autoalign* wird nicht unterstützt. |
| alle | *.sizeraster = true* und *.docking <> dock_window* | Toolbar-Bereich wird nicht an Rastergröße angepasst. |
| alle | *.dock_offset > 0* und *.docking <> dock_window* | Toolbars lassen sich nicht mit Abstand (Leerraum) positionieren. |
| *notepage* | | *.max_height*, *.max_width*, *.min_height* und *.min_width* haben keine Auswirkung. |

| .style | Bedingungen | Einschränkungen und Unterschiede |
|---|---|---|
| *toolbar* | *.sizeable = true* und *.docking <> dock_window* | Keine Sizebar, Toolbar-Größe kann nur in einer Richtung (abhängig von der Dock-Position) beeinflusst werden. Die Größe in der anderen Richtung wird durch die maximale Breite bzw. Höhe bestimmt.<br>Verhalten entspricht *.autosize = true*. |
| *toolbar* | *.sizeable = true* und *.docking = dock_window* | Toolbar-Größe kann nicht verändert werden. |
| *toolbar* | *.autosize = true* und *.docking <> dock_window* | Maximal mögliche Breite oder Höhe. |
| *toolbar* | *.docking <> dock_window* und gesetzte minimale bzw. maximale Breite oder Höhe | Minimal- und Maximal-Größen wirken nicht auf die Gesamtgröße der Toolbar, sondern schneiden Inhalte ab oder erzeugen leere Bereiche. |
| *toolbar* | | Keine *activate*- und *deactivate*-Ereignisse. |
| *toolbar* | Verschieben mit Beibehaltung des Dock-Zustands bei *.docking = dock_window* | Keine *move*-Ereignisse. |

## 46.10 Sonstiges

Die Kindobjekte von Toolbars sind - auch beim Umdocken - vom IDM-Anwender selbst anzuordnen. Wird aus der Regelsprache heraus umgedockt, sollte die Toolbar zuvor unsichtbar gemacht werden. Nach dem Umdocken können dann die Kindobjekte neu angeordnet werden, bevor man die Toolbar wieder sichtbar schaltet.

Sind für Kindobjekte von Toolbars Acceleratoren definiert, so sind diese auch im Vaterfenster wirksam. Sind dort dieselben Acceleratoren definiert, so haben diese Vorrang vor denen der Toolbarkinder.

## 46.11 Beispiel

```
dialog D
font Fn "12.Arial",0,bold;
color ColWhite "White", grey(100), white;
!! color ColBlue "Blue", grey(100), white;
source SrcCut
{
  0: .action action_cut;
     .type type_object;
```

```
}
source SrcCopy
{
  0: .action action_copy;
     .type type_object;
}
target Tar
{
  0: .action action_paste;
     .type type_object;
}
tile Ti_Other 20,20,
"####################",
"#            #     #",
"#            # ### #",
"#            #  #  #",
"#            #     #",
"####################",
"                    ",
"#######             ",
"##   ##   ###  #### ",
"# # # #  #   # #   #",
"#  #  #  ##### #### ",
"# # # #  #   # #   #",
"##   ##  #   # #### ",
"#######             ",
"                    ",
"####################",
"#                  #",
"# ### ####  ####   #",
"# ### ####   ##    #",
"####################";
tile Ti_Person 20,20,
"            ### ##  ",
"     ##     ########",
"    #  #    ########",
"    #  #    ########",
"    #  #    ########",
"    #  #    ########",
"     ##     ###   ##",
"     ##     #       ",
"    #  #    #       ",
"   ##  ######       ",
"  # #  #    #       ",
" #  #  #    #       ",
"    #  #    #       ",
```

```
"    ####    #       ",
"    #  #    #       ",
"    #  #    #       ",
"    #  #    #       ",
"  ###  ###  #       ",
"           #       ",
"           #       ";
tile TiUp 24,16,
"########################",
"########################",
"# ## ## ## ## ## ## ## #",
"########################",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"########################";
tile TiLeft 24,16,
"########################",
"########################",
"# #                    #",
"###                    #",
"###                    #",
"# #                    #",
"###                    #",
"###                    #",
"# #                    #",
"###                    #",
"###                    #",
"# #                    #",
"###                    #",
"###                    #",
"# #                    #",
"########################";
tile TiRight 24,16,
"########################",
"########################",
"#                    # #",
"#                    ###",
```

```
"#                   ###",
"#                   # #",
"#                   ###",
"#                   ###",
"#                   # #",
"#                   ###",
"#                   ###",
"#                   # #",
"#                   ###",
"#                   ###",
"#                   # #",
"########################";
tile TiDown 24,16,
"########################",
"########################",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"#                      #",
"########################",
"# ## ## ## ## ## ## ## #",
"########################";
tile TiWin 24,16,
"##################      ",
"##################      ",
"##################      ",
"#                #      ",
"#                #      ",
"#                #      ",
"#                #      ",
"#                #      ",
"#                #      ",
"#                #      ",
"#    ###################",
"#    ###################",
"######                 #",
"     # ### ### ### ### #",
"     #                 #",
"     ###################";
```

```
default image
{
  .width 32;
  .height 32;
  on cut
  {
    variable object Parent := this.parent;
    if (Parent.class = toolbar) then
      destroy (this);
      Pos(Parent, Parent.docking);
    endif
  }
}
model image MImUp
{
  .picture TiUp;
  on select
  {
    Pos(this.parent,dock_up);
  }
}
model image MImDown
{
  .picture TiDown;
  on select
  {
    Pos(this.parent,dock_down);
  }
}
model image MImLeft
{
  .picture TiLeft;
  on select
  {
    Pos(this.parent,dock_left);
  }
}
model image MImRight
{
  .picture TiRight;
  on select
  {
    Pos(this.parent,dock_right);
  }
}
model image MImWindow
```

```
{
  .picture TiWin;
  on select
  {
    Pos(this.parent,dock_window);
  }
}
!! ------------------------------------------------------
!! --------------- Model Listbox ------------------------
!! ------------------------------------------------------
model listbox MLbTitle
{
  .ytop 2;
  .xleft 5;
  .height 100;
  .width 150;
  .font Fn;
  .content[1] "Title";
  .content[2] "Toolbar";
  .content[3] "Überschrift";
  on cut
  {
    variable object Parent := this.parent;
    if (Parent.class = toolbar) then
      destroy (this);
      Pos(Parent, Parent.docking);
    endif
  }
  on select
  {
    variable object Parent := this.parent;
    if (Parent.class = toolbar) then
      Parent.title := this.content[this.activeitem];
    endif
  }
  rule void CopyContent(object This)
  {
    variable integer I;
    for I:=1 to this.itemcount do
      This.content[I] := this.content[I];
    endfor
  }
}
!! ------------------------------------------------------
!! --------------- Model Poptext ------------------------
!! ------------------------------------------------------
```

```
model poptext MPt
{
  .xleft 156;
  .width 192;
  .ytop 31;
  .text[1] "Auswahl 1";
  .text[2] "Auswahl 2";
  .text[3] "Auswahl 3";
  .text[4] "Auswahl 4";
  on cut
  {
    variable object Parent := this.parent;
    if (Parent.class = toolbar) then
      destroy (this);
      Pos(Parent, Parent.docking);
    endif
  }
  rule void CopyContent(object This)
  {
    variable integer I;
    for I:=1 to this.itemcount do
      This.content[I] := this.content[I];
    endfor
  }
}
!! -----------------------------------------------------
!! --------------- Model Pushbutton --------------------
!! -----------------------------------------------------
model pushbutton MPb
{
  .xleft 100;
  .width 110;
  .height 30;
  .ytop 57;
  .font Fn;
  .text "Toolbar";
  on cut
  {
    variable object Parent := this.parent;
    if (Parent.class = toolbar) then
      destroy (this);
      Pos(Parent, Parent.docking);
    endif
  }
}
!! -----------------------------------------------------
```

```
!! --------------- Model Edittext ----------------------
!! -----------------------------------------------------
model edittext MEt
{
  .xleft 159;
  .width 228;
  .ytop 98;
  on cut
  {
    variable object Parent := this.parent;
    if (Parent.class = toolbar) then
      destroy (this);
      Pos(Parent, Parent.docking);
    endif
  }
}
!! -----------------------------------------------------
!! --------------- Model Toolbar ----------------------
!! -----------------------------------------------------
model toolbar MTb
{
  .docking dock_up;
  .target Tar;
  on paste
  {
    variable object O;
    O := this:AddChild (thisevent.value.model);
    if O.model = MLbTitle then
      MLbTitle:CopyContent(O);
    endif
  }
  rule object AddChild (object Model)
  {
    variable object O;
    O := create(Model, this);
    O.source := SrcCut;
    Pos(this, this.docking);
    return O;
  }
}
!! -----------------------------------------------------
!! ----------------- Toolbar Fenster -------------------
!! -----------------------------------------------------
window W {
  .xauto 0;
  .xleft 0;
```

```
  .xright 0;
  .yauto 0;
  .ytop 0;
  .ybottom 0;
  .vwidth 1000;
  .vheight 1000;
  .bgc ColWhite;
  .vsb_optional false;
  .vsb_visible true;
  .title "Toolbar Demo";
  .font Fn;
  child menubox
  {
    .title "Toolbar";
    .font Fn;
    child menuitem
    {
      .text "Einrichten";
      .font Fn;
      on select
      {
        Nb.visible := true;
      }
    }
    child menuitem
    {
      .text "Neue Toolbar";
      .font Fn;
      on select
      {
        create(MTb,W);
      }
    }
  }
  child MTb Tb1
  {
    .title "Toolbar 1";
    .dock_line 1;
    .font Fn;
child MPt Pt
    {
      .font Fn;
    }
  }
  !! ----------------------------------------------
  !! -------------------- Notebook ----------------
```

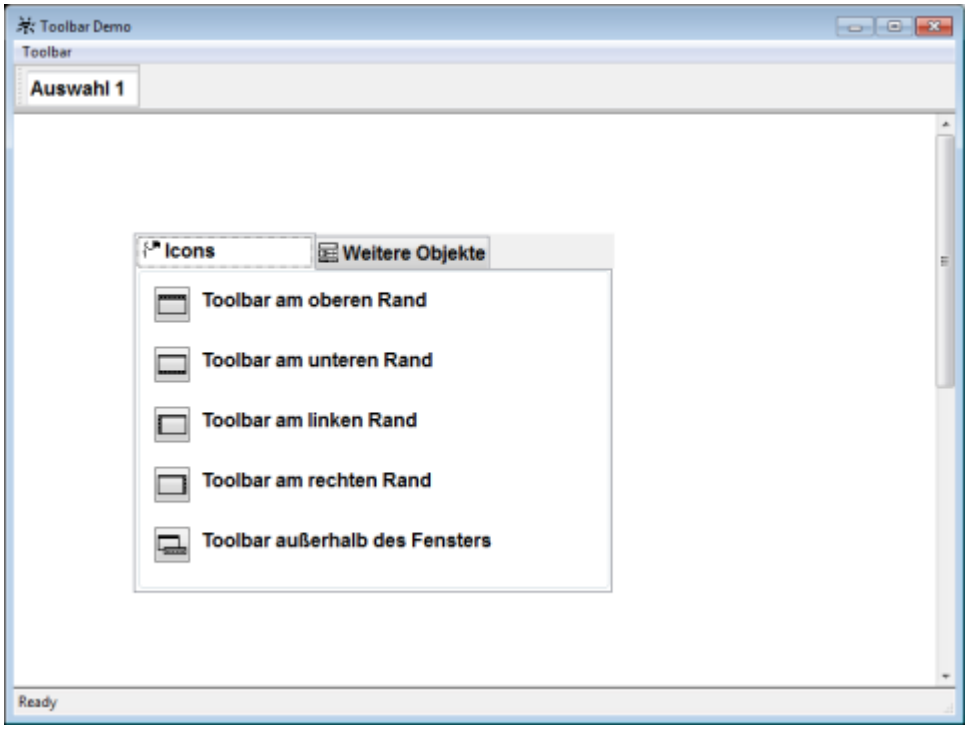
```
!! ------------------------------------------
child notebook Nb
{
  .xleft 100;
  .ytop 100;
  .height 300;
  .width 400;
  .font Fn;
  .visible false;
  !! ------------------------------------------
  !! ---------------- Notepage 1 ----------------
  !! ------------------------------------------
  child notepage Np1
  {
    .target Tar;
    .title "Icons";
    .picture Ti_Person;
    .font Fn;
    child statictext
    {
      .xleft 50;
      .ytop 13;
      .text "Toolbar am oberen Rand";
      .font Fn;
      .sensitive false;
    }
    child MImUp ImUp
    {
      .ytop 10;
      .xleft 10;
      .source SrcCopy;
    }
    child statictext
    {
      .xleft 50;
      .ytop 63;
      .text "Toolbar am unteren Rand";
      .font Fn;
      .sensitive false;
    }
    child MImDown
    {
      .ytop 60;
      .xleft 10;
      .source SrcCopy;
    }
```

```
  child statictext
  {
    .xleft 50;
    .ytop 113;
    .text "Toolbar am linken Rand";
    .font Fn;
    .sensitive false;
  }
  child MImLeft
  {
    .ytop 110;
    .xleft 10;
    .source SrcCopy;
  }
  child statictext
  {
    .xleft 50;
    .ytop 163;
    .text "Toolbar am rechten Rand";
    .font Fn;
    .sensitive false;
  }
  child MImRight
  {
    .ytop 160;
    .xleft 10;
    .source SrcCopy;
  }
  child statictext
  {
    .xleft 50;
    .ytop 213;
    .text "Toolbar außerhalb des Fensters";
    .font Fn;
    .sensitive false;
  }
  child MImWindow
  {
    .ytop 210;
    .xleft 10;
    .source SrcCopy;
  }
}
!! ------------------------------------------------------
!! ------------------- Notepage 'Objekte' ---------------
!! ------------------------------------------------------
```

```
  child notepage Np2
  {
    .target Tar;
    .title "Weitere Objekte";
    .picture Ti_Other;
    child statictext
    {
      .xleft 10;
      .ytop 10;
      .text "Listbox";
      .font Fn;
      .sensitive false;
    }
    child MLbTitle LBT
    {
      .ytop 30;
      .xleft 10;
      .width 150;
      .height 100;
      .source SrcCopy;
      .font Fn;
    }
    child statictext
    {
      .xleft 200;
      .ytop 10;
      .text "Pushbutton";
      .font Fn;
      .sensitive false;
    }
    child MPb Pb
    {
      .xleft 200;
      .ytop 40;
      .source SrcCopy;
    }
  }
}
statusbar S
{
  statictext
  {
    .text "Ready";
  }
}
on close { exit; }
```

```
}
on dialog start
{
  Tb1:AddChild (MImUp);
  Tb1:AddChild (MImDown);
  MLbTitle:CopyContent(LBT);
}
rule void Pos (object This, enum P)
{
  variable integer I;
  variable integer X;
  variable integer Y;
  if (This.class = toolbar) then
    case P
      in dock_right, dock_left:
        Y := 3;
        for I:=1 to This.childcount do
          if (This.child[I].class <> image ) then
            This.child[I].visible := false;
          else
            This.child[I].xleft := 3;
            This.child[I].ytop := Y;
            Y := Y + This.child[I].height + 4;
          endif;
        endfor
      in dock_up, dock_down, dock_window:
        X := 3;
        for I:=1 to This.childcount do
          This.child[I].visible := true;
          This.child[I].ytop := 3;
          This.child[I].xleft := X;
          X := X + This.child[I].width + 4;
        endfor
    endcase
    This.docking := P;
    if (P = dock_window) then
      This.height := 100;
      This.width := 400;
    endif
  endif
}
```

Das ergibt folgendes Fenster:


Toolbar Demo
Toolbar
Auswahl 1
Icons
Weitere Objekte
Toolbar am oberen Rand
Toolbar am unteren Rand
Toolbar am linken Rand
Toolbar am rechten Rand
Toolbar außerhalb des Fensters
Ready

# 47 transformer

Das Transformer-Objekt ermöglicht den Durchlauf eines XML-Dokuments oder einer IDM-Objekthierarchie, wobei während dieses Durchlaufs an einzelnen Knoten semantische Aktionen ausgeführt werden können. Auf diese Weise kann leicht eine Transformation von Daten implementiert werden, bei der z.B. entweder XML-Daten in IDM-Objekte übertragen werden oder umgekehrt mit Daten, die in IDM-Objekten stehen, ein XML-Dokument generiert wird. Da die semantischen Aktionen durch benutzerdefinierte Codes beschrieben werden, ist die Art der Transformation prinzipiell beliebig.

### Definition

```
{ export | reexport } { model } transformer { <Bezeichner> }
{
  [ <Attributdefinition> ]
  [ <Methodendefinition> ]
}
```

Zur Definition einer Transformation stehen dem IDM-Programmierer folgende Mittel zur Verfügung.

- Eine Transformation wird durch den Aufruf der :apply() Methode eines Transformer-Objekts gestartet. Als Parameter werden hier die Quelle und das Ziel der Transformation übergeben (z.B.: ein Doccursor-Objekt als Quelle und ein IDM-Objekt als Ziel, das dann die Daten aufnehmen soll oder diese weiter delegiert).
- Während jeder Transformation möchte man durch eine festgelegte Menge von Knoten, ob es IDM-Objekte oder Knoten eines XML-Baums sind, in einer bestimmten Reihenfolge durchlaufen. Die apply Methode realisiert einen solchen Durchlauf, indem sie ausgehend vom Startknoten (Quelle) in einer Schleife die select_next Methode des Transformers aufruft, die den Nachfolger des aktuellen Knotens bestimmt. Der Durchlauf wird beendet, sobald die select_next Methode *null* zurückliefert. Die Default-Implementierung der select_next Methode definiert eine Pre-Order-Reihenfolge. Der IDM-Programmierer kann in diesem Prozess an zwei Stellen eingreifen. Zum einen kann die apply Methode überdefiniert werden, um so den Startknoten festzulegen oder die Abbruchbedingung zu verändern. Zum anderen kann die select_next Methode überdefiniert werden und somit die gesamte Reihenfolge, in der die Knoten besucht werden.
- Nachdem sichergestellt ist, dass jeder Knoten irgendwann besucht wird, muss es eine Möglichkeit geben um festzulegen, an welchen Knoten welche semantische(n) Aktion(en) ausgeführt wird (werden). Zu diesem Zweck werden am Transformer-Objekt Mapping-Objekte als Kinder definiert. Jedes dieser Objekte definiert in seinem Attribut name ein Muster, das zur Entscheidung herangezogen wird, ob der gerade besuchte Knoten eine Aktion auslösen soll. Eine solche Aktion wird durch die action Methode des Mapping-Objekts festgelegt, die vom IDM-Programmierer überdefiniert wird. Diese Methode bekommt als ersten Parameter den gerade besuchten Knoten und als zweiten Parameter das Ziel-Objekt, das von der apply Methode des Transformers herrührt.

» Um das Bild zu vervollständigen: Das Transformer-Objekt hat ebenfalls eine action Methode. Diese Methode wird in der oben erwähnten Schleife der apply Methode für jeden besuchten Knoten aufgerufen und untersucht dabei, ob ein Muster eines der Mapping-Kinder auf den Knoten passt. Die Reihenfolge, in der das passiert, entspricht der Definitionsreihenfolge der Mappings beim Vater-Transformer. Die geerbten Mapping-Objekte werden als letzte untersucht. Wird bei der Untersuchung ein erstes passendes Mapping-Objekt gefunden, so ruft die action Methode des Transformers die action Methode dieses Mapping-Objekts auf. Diese kann dann die Daten vom aktuellen Knoten zum Ziel-Objekt übertragen. Als Rückgabewert kann diese Methode ein *true* zurückliefen. In diesem Fall wird angenommen, dass der Knoten komplett abgearbeitet worden ist und keine weiteren Mapping-Objekte untersucht werden sollen. Anderenfalls werden die übrigen Mappings untersucht und gegebenenfalls deren action Methoden aufgerufen, bis entweder alle Mappings abgearbeitet worden sind oder die action Methode eines von diesen *true* zurückgeliefert hat.

### Ereignisse

keine

### Kinder

document

mapping

***record***

transformer

### Vater

***application***

***canvas***

***checkbox***

***dialog***

doccursor

document

***edittext***

***groupbox***

***image***

***import***

***layoutbox***

***listbox***

mapping

***menubox***

***menuitem***

***menusep***

***messagebox***

***module***

***notebook***

***notepage***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***statusbar***

***tablefield***

***timer***

***toolbar***

transformer

***treeview***

***window***

**Menü**

keins

**Methoden**

:action()

:apply()

:select_next()

## 47.1 Attribute

document[I]

external

external[I]

firstrecord

label

lastrecord

mapping[I]

model

module

parent

record[I]

recordcount

root

scope

userdata

## 47.2 Objektspezifische Attribute

### mapping[I]

Über das .mapping Attribut kann auf die Mapping-Kinder zugegriffen werden. Das Attribut wird mit dem Objektindex indiziert (ähnlich zu child). Die Reihenfolge der Mappings in diesem Vektor bestimmt die Reihenfolge, in der während einer Transformation ein Knoten mit einzelnen Mappings verglichen wird. Die vererbten Mappings werden in diesen Vektor nicht übernommen. Während einer Transformation wird mit solchen Mappings erst zum Schluss verglichen, die direkten Instanzen haben also Vorrang. Das ist anders, als bei anderen vererbten Kindobjekten im IDM, die im Kindvektor der Vaterinstanz vorne eingefügt werden.

### root

Nach dem Aufruf der apply Methode wird in diesem Attribut der Ausgangspunkt der Transformation gespeichert. Damit kann während einer Transformation entschieden werden, ob der Ausgangspunkt wieder erreicht worden ist und die Transformation beendet werden kann.

Dabei sind folgende Fallunterscheidungen zu beachten.

- Ist der Src-Parameter der apply Methode ein Document- oder Doccursor-Objekt so wird in .root ein String abgespeichert, der die entsprechende Position im XML-Baum beschreibt. Die Syntax dieses Strings entspricht der Konvention, die im .path Attribut des Doccursor-Objekts verwendet wird (siehe Objektbeschreibung zu doccursor). Vergleiche mit dem .path Attribut von Doccursor sind deswegen besonders einfach.
- Ist der Src-Parameter der apply Methode ein IDM-Objekt, so wird im .root dieses Objekt gespeichert. Folglich ist in diesem Fall der Typ dieses Attributs object.

Anhand des Wertes im .root Attribut entscheiden die action und select_next Methoden des Transformers, was diese tun müssen (siehe hierzu die Beschreibung dieser Methoden).

Am Ende der apply Methode wird *.root* wieder auf *void* gesetzt.

Default-Wert ist *void*.

## 47.3 Objektspezifische Methoden

### :apply()

Mit dieser Methode wird die Transformation angestoßen. Die Default-Implementierung des Algorithmus sieht folgendermaßen aus:

» Ist der Src-Parameter ein Document- oder Doccursor-Objekt, so wird angenommen, dass Daten aus einem XML-Baum zum IDM übertragen werden sollen. Im Falle eines Document-Objekts wird ein temporäres Doccursor-Objekt erzeugt, das zur Navigation im XML-Baum benutzt wird. Im folgenden Pseudocode wurde der Einfachheit halber angenommen, dass in Src immer ein Doccursor übergeben wird.

```
:apply(anyvalue Src, anyvalue Dest)
{
  variable object NextObj;

  this.root ::= Src.path;
  NextObj := Src;
  while NextObj <> null do
    this:action(Src, Dest);
    NextObj := this:select_next(NextObj);
  endwhile
  this.root ::= null;
  return true;
}
```

» Ist der Src-Parameter ein IDM-Objekt (außer Document- oder Doccursor-Objekt), so wird angenommen, dass Daten vom IDM nach XML bzw. sonst wohin übertragen werden sollen. Der zugrundeliegende Code ist mit dem oben aufgeführten Codefragment identisch, außer das hier in .root nicht Src.path sondern Src selbst gespeichert wird. Also:

```
this.root ::= Src;
```

Die apply Methode kann überdefiniert werden (ähnlich zu init).

### :action()

Diese Methode wird von der apply Methode des Transformers benutzt (siehe oben). Damit wird festgestellt, ob der aktuelle Knoten auf eines der Mapping-Kinder passt und deswegen die action Methode dieses Mapping-Objekts aufgerufen werden muss.

Die action Methode kann überdefiniert werden (ähnlich zu init).

**:select_next()**

Diese Methode wird von der apply Methode des Transformers zum Durchlaufen aller Knoten eines XML-Baums oder der IDM-Objekthierarchie benutzt. (siehe oben). Die Default-Implementierung dieser Methode sieht so aus, dass das wiederholte Aufrufen der Methode einen Pre-Order-Durchlauf realisiert.

Die select_next Methode kann überdefiniert werden (ähnlich zu init).

## 47.4 Beispiel

Dieses Beispiel ist im Verzeichnis *examples/xml* zu finden.

Als XML-Dokument dient folgende Datei mit dem Namen „CD-Katalog.xml"":

```
<?xml version="1.0" ?>
<CATALOG>
  <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
  </CD>
  <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tyler</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
  </CD>
  <CD>
    <TITLE>Maggie May</TITLE>
    <ARTIST>Rod Stewart</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>Pickwick</COMPANY>
    <PRICE>8.50</PRICE>
    <YEAR>1990</YEAR>
  </CD>
</CATALOG>
```

Dann können die Daten aus dieser Datei beispielsweise wie folgt ausgelesen werden.

```
dialog D {}

window Wi
{
```

```
  .title "XML-CD-CATALOG";

  on close { exit(); }

  child treeview Tv
  {
    .xauto 0;
    .yauto 0;
    .style[style_lines]   true;
    .style[style_buttons] true;
    .style[style_root]    true;
    integer CdIdx := 0;

    rule NewCatalog() {
      if this.itemcount = 0 then
        this.itemcount ::= this.itemcount + 1;
      endif
      this.content[this.itemcount] := "CD-Catalog";
      this.open[this.itemcount] := true;
    }

    rule AddCD() {
      this:insert(this.itemcount+1, 4);
      this.CdIdx := this.CdIdx + 1;
      this.content[this.itemcount-3] := ""+this.CdIdx+". CD";
      this.level[this.itemcount-3] := 2;
    }

    rule AddTitle(string S input) {
      this.content[this.itemcount-2] := "Title: " + S;
      this.level[this.itemcount-2] := 3;
    }

    rule AddArtist(string S input) {
      this.content[this.itemcount-1] := "Artist: " + S;
      this.level[this.itemcount-1] := 3;
    }

    rule AddPrice(string S input) {
      this.content[this.itemcount] := "Price: " + S;
      this.level[this.itemcount] := 3;
    }
  }
}

transformer Tr
{
  !! transformer ist zur Übernahme der Daten aus einem XML-Dokument gedacht
```

```
  !! in Src ist deswegen immer ein doccursor zu erwarten

  child mapping MCatalog {
    .name "..CATALOG";

    :action() {
      Dest:NewCatalog();
      return true;
    }
  }

  child mapping MCD {
    .name "..CD";

    :action() {
      Dest:AddCD();
      return true;
    }
  }

  child mapping MTitle {
    .name "..CD.TITLE";

    :action() {
      Dest:AddTitle(Src.text);
      return true;
    }
  }

  child mapping MArtist {
    .name "..CD.ARTIST";

    :action() {
      Dest:AddArtist(Src.text);
      return true;
    }
  }

  child mapping MPrice {
    .name "..CD.PRICE";

    :action() {
      Dest:AddPrice(Src.text);
      return true;
    }
  }
}
```

```
document Doc {}

on dialog start
{
  Doc:load("CD-Katalog.xml");

  Tv.visible := false;
  Tr:apply(Doc, Tv);
  Tv.visible := true;
}
```

# 48 treeview

Das ***treeview***-Objekt bietet die Möglichkeit hierarchische Informationen in einer Baumstruktur darzustellen. Dabei können Äste des Baumes (man spricht auch von Teilbäumen) zugeklappt werden, um die Übersichtlichkeit zu erhöhen. Die Programmierung entspricht weitgehend der einer ***listbox***, allerdings ist nur eine Einfachselektion erlaubt.

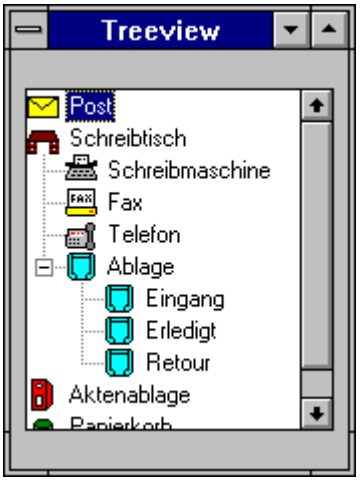



***Abbildung 50:*** *Beispiel eines treeview-Objekts*

### Definition

```
{ export | reexport } { model } treeview { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Textattribute>
  <Objektspezifische Attribute>
}
```

Der Benutzer kann interaktiv zwei unterschiedliche Aktionen durchführen:

» **Öffnen/Schließen eines Teilbaumes**
Dies kann entweder durch Umschalten des +/- Knopfes oder durch ein Doppelklick auf den Text geschehen.

» **Selektion**
Einfach- oder Doppelklick auf einen Eintrag oder Wechsel des Eintrages über die Tastatursteuerung. Zur Tastatursteuerung gehören die Cursortasten Links, Rechts, Hoch, Runter, Bild Hoch, Bild Runter, Pos 1 und Ende, wobei Links/Rechts bei der ersten Betätigung den Effekt hat,

den Teilbaum zu Schließen/Öffnen und erst wenn dieser Zustand erfüllt ist, nach Oben/Unten zu gehen.

Das Treeview ist auf den Fenstersystemen Motif und MS-Windows verfügbar, allerdings erlaubt nur die MS-Windows-Plattform die Darstellung mit Bildern und allen Darstellungsstilen (Detaillierte Unterschiede sind in Ref: Attribute .style[] notiert). Bei allen anderen Plattformen erfolgt eine rein textuelle Darstellung.

Diese textuelle "Emulation" hat neben den sichtbaren Unterschieden noch andere Bedienungsunterschiede:

» Der +/- Knopf kann nicht getrennt angesteuert werden, das Öffnen/Schließen eines Teilbaumes geschieht nur über einen Doppelklick.

» Bei der Tastatursteuerung mittels Links oder Rechts wird der Öffnungszustand des Teilbaumes nicht verändert.

Für das vollständige Setzen des Treeview-Inhaltes inklusive .open[], .level[] und .picture[] von C- oder COBOL-Seite aus wurden keine neuen Schnittstellenfunktionen eingeführt, sondern die vorhandenen Funktionen DM_SetVectorValues() und DM_GetVectorValues() sind zu benutzen.

Folgende Ereignisse werden aufgrund von interaktiven Benutzeraktionen erzeugt:

| Ereigniss (e) | Belegte Attribute am thi-sevent | Auslösung |
|---|---|---|
| open, close | .index | Benutzeraktion Öffnen bzw. Schließen eines Teilbaumes |
| activate | .index | Selektion wechselt auf einen anderen Eintrag |
| select | .index | Einfachklick auf einen Eintrag (Selektion) |
| dbselect | .index | Doppelklick auf einen Eintrag |

Daneben gibt es noch weitere Standard-Events die in der nachfolgenden Tabelle aufgelistet sind und deren Bedeutung im Handbuch "Regelsprache" beschrieben ist.

### Ereignisse

*activate*

*close*

*cut*

*dbselect*

*extevent*

*focus*

*help*

*hscroll*

*key*

*open*

*paste*

*scroll*

*select*

*vscroll*

### Kinder

document

***record***

transformer

### Vater

***groupbox***

***layoutbox***

***module***

***notepage***

***splitbox***

***toolbar***

***window***

### Menü

**Popup-Menü**

### Methoden

:childcount()

:childindex()

:delete()

:exchange()

:find()

:insert()

:parent()

:reparent()

## 48.1 Attribute

acc_label

acc_text

accelerator

activeitem

bgc

bordercolor

borderraster

borderwidth

class

content[I]

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

fgc

firstchar

firstrecord

focus

focusitem

font

function

groupbox

height

help

index

itemcount

label

lastrecord

layoutbox

level[I]

mapped

member[I]

membercount

menu

model

notepage

open[I]

options[enum]

parent

picture[I]

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sizeraster

statushelp

style[enum]

toolbar

toolhelp

topitem

userdata

userdata[I]

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

## 48.2 Spezifische Attribute

Das Aussehen des Treeviews wird durch das Zusammenspiel der Attribute content[I], level[I], picture [I], open[I] und style[enum] bestimmt.

| Attribut | Beschreibung |
|---|---|
| activeitem | Aktives Element des Objekts. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. |
| content[I] | Wert des I-ten Elements des Treeview, die Werte werden **nicht** vom Modell auf die Instanzen vererbt!<br>Bitte beachten Sie auch das Kapitel „.content und contentspezifische Attribute“. |
| firstchar | Nummer des ersten angezeigten Zeichens (horizontale Scroll-Position). |
| itemcount | Gesamtzahl der Einträge des Treeview. |
| level[I] | Legt die Einrückung des jeweiligen Eintrags fest.<br>Bitte beachten Sie auch das Kapitel „.content und contentspezifische Attribute“. |
| open[I] | Legt fest, ob ein Knoten geöffnet ist oder nicht.<br>Bitte beachten Sie auch das Kapitel „.content und contentspezifische Attribute“. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_rightclick_selects* (nur IDM für Windows)<br>» *opt_scroll_on_focus* (nur IDM für Motif) |
| picture[I] | Angezeigtes Bild des jeweiligen Eintrags.<br>Bitte beachten Sie auch das Kapitel „.content und contentspezifische Attribute“. |

| Attribut | Beschreibung |
|---|---|
| style[enum] | Legt das Aussehen der Linien, Knöpfe und des obersten Eintrags (root Eintrag) fest. |
| topitem | Erstes angezeigtes Element des Treeview bei vertikalem Scrollen. |
| userdata[I] | Userdata für beliebige Daten zu jedem Eintrag des Objekts. |

Weitere Informationen erhalten Sie in den folgenden Kapiteln bzw. in der „Attributreferenz“.

### 48.2.1 .content und contentspezifische Attribute

Der Inhalt des ***Treeviews*** wird im Attribut content[I] gesetzt. Dabei müssen die einzelnen Einträge (und damit der Index I) aufeinanderfolgend (1, 2, 3, …) sein.

Das Attribut level[I] legt danach die Einrückung des jeweiligen Eintrags fest. Soll ein Bild angezeigt werden, so wird dieses im Attribut picture[I] definiert.

Das Attribut open[I] legt fest, ob der jeweilige Knoten geöffnet ist oder nicht.

#### Anmerkung

Es ist zu beachten, dass erst durch Setzen des *.content[I]* der weitere Zugriff auf die contentspezifischen Attribute dieses Eintrags erlaubt ist. Daher muss zuerst der Content und danach die contentspezifischen Attribute gesetzt werden.

### 48.2.2 Hinweis zu veralteten Attributen

Die veralteten Attribute *.focusitem* und *.focusable* werden nicht mehr unterstützt und erzeugen eine „ignoring“ Warnung beim Setzen und Zurücksetzen (**setinherit**) sowie ein `fail` beim Auslesen dieser Attribute.

## 48.3 Informationsinhalt

Die hierarchische Information wird in Attribut-Feldern gespeichert. Pro Eintrag (= Knoten des hierarchischen Baums) wird in den Attributen .content[] und .picture[] ein statischer Text und ein Muster gespeichert, im .level[]-Attribut wird durch einen Zahlenwert die hierarchische Beziehung definiert. Zum Sichtbar/Unsichtbar-machen von Teilbäumen dient das boolsche .open[]-Attribut.

Diese Attribute dienen zum Aufnehmen von dynamischer Information und werden deshalb nicht vererbt! Der Indizierungsbereich geht für das .content[]-Attribut von 1 ... n, wobei n der Wert von .itemcount repräsentiert. Für die Attribute .level[], .picture[] und .open[] ist der Index-Bereich von 0 ... n definiert, das 0-te Element dient als Default-Wert für ungesetzte Werte im Indexbereich 1 ... n. Genau wie bei der Listbox kann die Feldgröße über das .itemcount-Attribut definiert oder dynamisch, durch das Sezten des n+1-ten .content[]-Feldes, um einen Eintrag vergrößert werden.

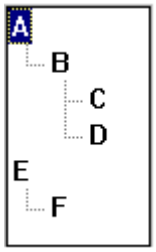
## 48.4 Baumkonzept

Die Abbildung der Einträge 1 ... n in eine hierarchischen Struktur, ein Baum mit Vater-Kind-Beziehungen, wird über das .level[]-Attribut definiert. Dessen Integer-Wert stellt quasi eine „Einrückungstiefe“ dar, allerdings ist nicht so sehr der absolute Wert entscheidend, sondern der Unterschied zum Vorgänger. Ist der Wert im Vergleich zum Vorgänger größer, so ist der Eintrag ein Kind des Vorgängers, ist er gleich, so handelt es sich um Geschwister, ist er aber kleiner, so handelt es sich um ein Kind eines weiter oben liegenden Vorgängers, dessen „Einrückungstiefe“ noch kleiner ist. Wenn kein Vater im Indexbereich 1 ... n zu finden ist, so spricht man auch von einer Wurzel.

### Beispiel

| Index I | .content[I] | .level[I] |
|---|---|---|
| 1 | A | 1 |
| 2 | B | 2 |
| 3 | C | 3 |
| 4 | D | 3 |
| 5 | E | 1 |
| 6 | F | 2 |

ergibt die Baumstruktur:




Für eine detailliertere Definition siehe auch Attribut level[I] in der „Attributreferenz“.

Alle von der Listbox bekannten Methoden lassen sich auch beim Treeview anwenden, allerdings wirken sie in gleicher Weise auch auf die Attribute .level[], .open[] und .picture[]. Zusätzlich gibt es spezielle Methoden für die Manipulation und Abfrage der Baumstruktur.

Äquivalent zur Listbox erlaubt das Attribut .activeitem, einen Eintrag im Bereich 1 ... n des Baumes als selektiert zu setzen, bzw. den Selektionsstatus abzufragen. Die Selektierbarkeit hängt dabei mit der Sichtbarkeit des Eintrages im Baum zusammen, somit auch mit den Attributen .level[] und .open[]. Dabei gilt für die Regelsprache genauso wie für die interaktive Bedienung: Beim Setzen von .activeitem werden alle Väter auf .open[] := true gesetzt, wird umgekehrt ein Teilbaum mittels .open[I] := false unsichtbar gesetzt, obwohl ein Kind darin selektiert ist, erhält der Eintrag I nun die Selektion und .activeitem wird dementsprechend umgesetzt.

## 48.5 Darstellung

Die Darstellung unterscheidet sich im Vergleich zur Listbox durch folgende Punkte:

1. Repräsentation der hierarchischen Beziehung von Einträgen über die Einrückung und optional zusätzlich durch Linien (Ref. Attribute .style[] und Attribute .level[]). Von Teilbäumen, deren Wurzel unsichtbar gesetzt ist (.open[] ist false), wird nur die Wurzel dargestellt.
2. Optionales Bildchen zu einem Texteintrag. (Ref.: Attribute .picture[])
3. Optionaler +/- Knopf, der bei Einträgen mit Kindern sichtbar wird und zum Öffnen(+)/Schließen(-) eines Teilbaumes dient. (Ref. Attribut .open[])

Ansonsten gelten alle für die Listbox relevanten Geometrie- und Layoutattribute.

## 48.6 Anmerkungen zum IDM für Windows

Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

Das Treeview-Objekt unter Microsoft Windows lässt zwar Texte mit beliebiger Länge zu, es werden aber nur die ersten 260 Zeichen dargestellt. Da sich dieses Verhalten in neueren Versionen der "comctl32.dll" (> 6.0) ändern kann, erzeugt der IDM keine Meldung, wenn der Text zu lang ist.

## 48.7 Anmerkung zum IDM für Motif

### Klarstellung einiger Benutzerereignisse für das Motif-WSI

| | |
|---|---|
| *select* | Kommt beim Mausklick auf einen Eintrag oder der Betätigung der `Space`-Taste auf den aktiven Eintrag.<br>Ebenso wird das *select*-Ereignis ausgelöst, wenn mittels Cursortasten (`Aufwärts`, `Abwärts`, `Pos 1`, `Ende`) im Baum navigiert wird. |
| *dbselect* | Wird durch einen Doppelklick bzw. `Return` analog zum *select*-Ereignis ausgelöst.<br>Zusätzlich wird der Öffnungszustand des Teilbaums (einschließlich der dementsprechenden *open*- bzw. *close*-Ereignisse) geändert. |
| *open*, *close* | Wird ausgelöst durch Doppelklick, `Return` bzw. Klick auf "+-"-Button und zeigt an, ob der Teilbaum geöffnet bzw. geschlossen wurde. |
| *activate* | Zeigt an, dass sich die Aktivierung geändert hat und wird durch Mausklick als auch durch einen Wechsel des aktiven Eintrags mittels Tastatur ausgelöst. |

Ein Treeview kann bei *.options[opt_scroll_on_focus] = false* per Tastaturnavigation unerreichbar sein, falls sich der aktive Eintrag außerhalb des sichtbaren Bereichs befindet. Er bleibt aber per Mausklick fokussierbar.

## 48.8 Anmerkung zum IDM für Qt

In den **Qt-Versionen 4.x** gibt es beim ***Treeview***-Objekt einen Darstellungsfehler (**QTBUG-9352**), der unter Umständen bewirkt, dass Einträge entweder abgeschnitten oder überhaupt nicht dargestellt werden. Dies liegt daran, dass Qt die Spaltenbreite für die Anzeige der ***Treeview***-Einträge nur anhand der Einträge im sichtbaren Bereich (+/- einem Offset) berechnet. Das heißt, dass dieser Effekt in einem ***Treeview*** mit besonders vielen Einträgen, deren Länge sich teilweise deutlich unterscheidet, zu tragen kommen kann. Ebenso betroffen sind ***Treeviews***, die besonders viele Einträge mit vielen Hierarchiestufen enthalten.

### Workaround

Als Workaround sollte man in diesem Fall den längsten Eintrag (im Sinne von benötigtem Darstellungsbereich, also Einrückung und Text) vor dem initialen Sichtbarmachen des ***Treeviews*** in den angezeigten Bereich bringen (z.B. über *.topitem*).

## 48.9 Beispiel

```
dialog Dialog

window Wn
{
  .active false;
  .xleft  229;
  .width  446;
  .ytop   131;
  .height 180;
  .title  "Beispielfenster";

  child treeview Tv
  {
    .xleft  38;
    .width  163;
    .ytop   15;
    .height 151;
    .content[1] "Firma A";
    .content[2] "Mueller";
    .content[3] "Meier";
    .content[4] "Firma B";
    .content[5] "Schulz";
    .content[6] "Schmidt";
    .content[7] "Fischer";
    .activeitem 1;
    .firstchar  1;
    .style[style_lines]   true;
    .style[style_buttons] true;
    .style[style_root]    true;
```

```
  .level[2] 2;
  .level[3] 2;
  .level[5] 2;
  .level[6] 2;
  .level[7] 2;
  integer Index;
  boolean Select := false;

  on select
  {
    if Tv.Select then
      Tv:reparent(Tv.Index, 1, Tv.activeitem, Tv.activeitem);
      Tv.Select := false;
      St.text   := "";
    endif
  }
}

child pushbutton Pb_rep
{
  .xleft 240;
  .width 194;
  .ytop  26;
  .text  "Firma angliedern";

  on select
  {
    variable integer Index;

    Tv.Index  := Tv.activeitem;
    St.text   := "Waehlen sie eine neue Firma";
    Tv.Select := true;
  }
}

child statictext St
{
  .xleft 241;
  .ytop  72;
  .text  "";
}

child pushbutton Pb_Exit
{
  .xleft 355;
  .width 78;
  .ytop  132;
```

```
    .text  "Exit";

    on select { exit(); }
  }
}
```

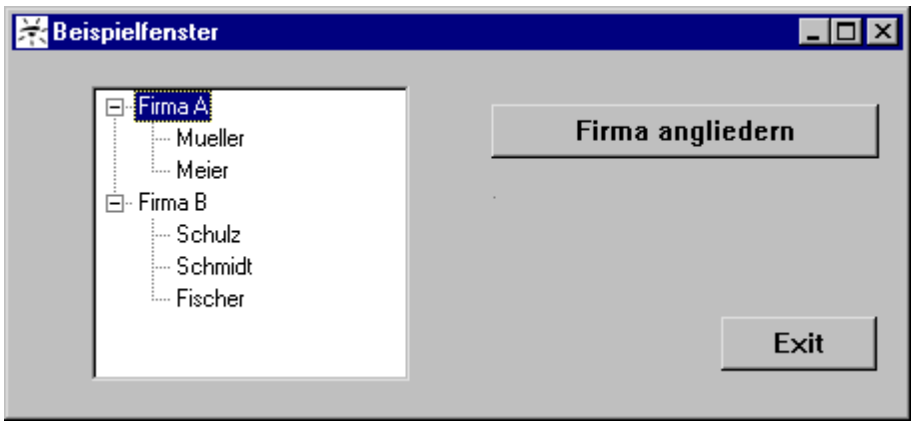



***Abbildung 51:*** *Fenster mit treeview-Objekt*

# 49 window (Fenster)

Das zugrundeliegende Objekt eines Fenstersystems ist – wie der Name bereits sagt – das Fenster. Alle weiteren Objekte müssen innerhalb eines Fensters erzeugt werden.

Die Deklaration eines Fensters beginnt mit dem Schlüsselwort **window**; anschließend steht der Identifikator des Fensters, gefolgt von der Fensterdefinition.

Mit der Fensterdefinition werden das Aussehen, die Größe und die verschiedenen weiteren Eigenschaften eines Fensters festgelegt.

### Definition

```
{ export | reexport } { model } window { <Bezeichner> }
{
  <Standardattribute>
  <Allgemeine Attribute>
  <Geometrieattribute>
  <Rasterattribute>
  <Hierarchieattribute>
  <Layoutattribute>
  <Scrollbarattribute>
  <Objektspezifische Attribute>
}
```

## 49.1 Definition von Kindobjekten

Alle anderen vom DM unterstützten Objekte liegen innerhalb eines Fensters. Aus diesem Grund müssen sie entweder direkt oder indirekt als Kind eines Fensters deklariert werden. Das Schlüsselwort hierfür ist **child**. Es folgt ein Schlüsselwort, das den Typ des Objektes beschreibt und ein optionaler Identifikator. Dieser Identifikator muss jedoch nur angegeben werden, wenn das Objekt später aus den Regeln oder aus der Anwendung heraus angesprochen werden soll. Eingeschlossen in geschweifte Klammern folgen die Objektdefinitionen (siehe Kapitel „Vererbung von Attributen“ im Handbuch „Programmiertechniken“).

### Definition

```
{export | reexport} child <Objektklasse>  {<Identifikator>}
{
  Objektdefinition
}
```

### Anmerkung

Das Einhängen eines Menüs in das Fenster muss über das Schlüsselwort **child** erfolgen. Die über **child** eingehängten Menüs werden dabei in der Menübar des Fensters platziert. Ihre Anzahl ist dabei

beliebig.

### Ereignisse

*activate*

*close*

*deactivate*

*deiconify*

*extevent*

*help*

*hscroll*

*iconify*

*key*

*move*

*paste*

*resize*

*scroll*

*select*

*vscroll*

### Kinder

***canvas***

***checkbox***

document

***edittext***

***groupbox***

***image***

***listbox***

***layoutbox***

***menubox***

***menuitem***

***menusep***

***notebook***

***poptext***

***pushbutton***

***radiobutton***

***record***

***rectangle***

***scrollbar***

***spinbox***

***splitbox***

***statictext***

***tablefield***

***toolbar***

transformer

***treeview***

***window***

### Vater

***dialog***

***module***

***window***

### Menü

**Popup-Menü**

***menuitem***

## 49.2 Attribute

acc_label

acc_text

accelerator

active

barwidth

bgc

bordercolor

borderstyle

borderraster

borderwidth

child[I]

childcount

class

closeable

cursor

cut_pending

cut_pending_changed

dialog

dialogbox

display

document[I]

external

external[I]

fgc

firstchild

firstmenu

firstrecord

focus

font

function

height

help

hsb_arrows

hsb_linemotion

hsb_optional

hsb_pagemotion

hsb_visible

icon

iconic

iconifyable

ignorecursor

index

label

lastchild

lastrecord

lastmenu

mapped

maxheight

maximized

maxwidth

member[l]

membercount

menu

menu[l]

menubgc

menucount

menufgc

minheight

minwidth

model

moveable

options[enum]

parent

posraster

.preedit

real_height

real_sensitive

real_visible

real_width

real_x

real_xraster

real_y

real_yraster

record[l]

recordcount

reffont

scope

sensitive

sizeable

sizeraster

statushelp

sysmodal

tile

tilestyle

title

titlebar

titlebgc

titlefgc

toolbar[I]

toolbarcount

toolhelp

top_most

userdata

userplaced

vheight

visible

vsb_arrows

vsb_linemotion

vsb_optional

vsb_pagemotion

vsb_visible

vwidth

width

xauto

xleft

xorigin

xraster

xright

yauto

ybottom

yorigin

yraster

ytop

## 49.3 Spezifische Attribute

| Attribut | Beschreibung |
|---|---|
| barwidth | Breite der Bars für die in der Größe änderbaren toolbar-Objekte in Pixeln. |
| borderraster | Legt die Art der Geometrieberechnung bei aktivem Raster fest. Die detaillierte Beschreibung finden sie in der Attribut-Beschreibung in der „Attributreferenz“. |
| borderstyle | Definiert den Stil, also Darstellung und Ausprägung, des Rahmens (ab IDM-Version A.06.01.a).<br>Attribut wird unterstützt, jedoch nur *border_none* und *border_toolkit* zulässig. *border_plain*, *border_raised* und *border_sunken* werden als *border_toolkit* umgesetzt. |
| closeable | Definiert, ob das Fenster geschlossen werden kann. |
| dialogbox | Gibt an, ob das Fenster eine Dialogbox ist. |
| display | Definiert den Screen, in welchem das Fenster darzustellen ist, siehe auch das Kapitel zur display-Ressource in der „Ressourcenreferenz“. |
| hsb_arrows (nur IDM FÜR MOTIF) | Definiert, ob Pfeile an den Enden der horizontalen Scrollbar vorhanden sind. |
| hsb_linemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von xorigin beim zeilenweisen horizontalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_optional | Definiert, ob die horizontale Scrollbar nur angezeigt wird, wenn sie benötigt wird.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_pagemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von xorigin beim seitenweisen horizontalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| hsb_visible | Definiert die Sichtbarkeit der horizontalen Scrollbar.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |

| Attribut | Beschreibung |
|---|---|
| icon | Um ein eigenes Icon in der linken oberen Fensterecke anzuzeigen wird in diesem Attribut die entsprechende tile-Ressource gesetzt. |
| iconic | Definiert, ob ein Fenster iconisch (minimiert in der Taskbar) ist. |
| iconifyable | Definiert, ob ein Fenster iconifizierbar sein soll oder nicht. |
| ignorecursor | Definiert, ob ein temporär gesetzter Override-Cursor ignoriert wird. |
| maxheight | Maximale Größe des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| maximized | Maximierungszustand des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| maxwidth | Maximale Breite des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| minheight | Minimale Höhe des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| minwidth | Minimale Breite des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| moveable | Legt fest, ob das Fenster vom Benutzer verschoben werden kann. |
| options[enum] | Optionen des Objekts.<br>Indizes:<br>» *opt_animated* (nur IDM für Qt)<br>» *opt_center_toolhelp* (nur IDM für Windows)<br>» *opt_nested_docks* (nur IDM für Qt)<br>» *opt_scroll_on_focus* (nur IDM für Motif)<br>» *opt_tabbed_docks* (nur IDM für Qt)<br>» *opt_window_size* (nur IDM für Qt) |
| .preedit<br>(nur IDM FÜR MOTIF) | Steuert Anzeige und Auswahl des Eingabemodus für die ***edittexte*** innerhalb des Fensters. |
| sizeable | Definiert, ob das Fenster vergrößerbar bzw. verkleinerbar ist. |
| sysmodal | Gibt an, ob ein als „Dialogbox“ definiertes Fenster (*.dialogbox* = *true*) vor allen anderen Fenstern auf dem Desktop erscheinen soll. |

| Attribut | Beschreibung |
|---|---|
| tile | Definiert das Hintergrundbild des Objekts. |
| tilestyle | Gibt die Art und Weise an, wie das in *.tile* definierte Hintergrundbild dargestellt wird. |
| title | Angabe des Fenstertitels.<br>Siehe auch Kapitel „Hintergrundbild“. |
| titlebar | Definiert, ob das Fenster eine Titelleiste besitzt. |
| top_most | Mit Hilfe dieses Attributes können Fenster als ganz vorne liegend definiert werden. Der Default für dieses Attribut ist *false* (nur Microsoft Windows). |
| vheight | Interne („virtuelle“) Höhe des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| visible | Definiert, ob das Fenster sofort beim Dialogstart sichtbar sein soll. |
| vsb_arrows (nur IDM FÜR MOTIF) | Definiert, ob Pfeile an den Enden der vertikalen Scrollbar vorhanden sind. |
| vsb_linemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von yorigin beim zeilenweisen vertikalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_optional | Definiert, ob die vertikale Scrollbar nur angezeigt wird, wenn sie benötigt wird.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_pagemotion | Anzahl der Pixel bzw. Rastereinheiten, um die sich der Wert von yorigin beim seitenweisen vertikalen Scrollen ändert.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vsb_visible | Definiert die Sichtbarkeit der vertikalen Scrollbar.<br>Siehe auch Kapitel „Scrollbarattribute“ in der „Attributreferenz“. |
| vwidth | Interne („virtuelle“) Breite des Objekts.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| xorigin | Gibt an, um wie viele Pixel der eigentliche Inhalt horizontal verschoben ist.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |
| yorigin | Gibt an, um wie viele Pixel der eigentliche Inhalt vertikal verschoben ist.<br>Siehe auch Kapitel „Geometrieattribute“ in der „Attributreferenz“. |

**Hintergrundbild**

Ein im Attribut tile definiertes Hintergrundbild wirkt sich nur auf die eigentliche Fensterfläche aus, nicht jedoch auf

- Titelleiste
- Menüleiste
- Statusbar
- Eingedockte Toolbar

## 49.4 Hinweise zum IDM FÜR WINDOWS

### 49.4.1 Automatische Größenanpassung bei unzulässigen Größenwerten

Werden bei einem Fenster unzulässige Größenwerte angegeben, dann wird die Größe automatisch angepasst und ein *resize*-Ereignis generiert. Ein unzulässiger Wert ist hierbei ein Wert, der von Microsoft Windows abgelehnt wird. Eine Verletzung von .minwidth, .maxwidth, … ist ein Fehler des Dialogskriptes, weswegen das Verhalten hier undefiniert ist.

### 49.4.2 Maximalgrößen von Fenstern

Unter Microsoft Windows kann ein Fenster normalerweise maximal die Größe des Bildschirms plus die Größe des Fensterrahmens annehmen. Dies gilt entsprechend für MDI-Kindfenster, hier ist es maximal die Maximalgröße des Vaterfensters (abhängig vom Wert in maxheight und maxwidth) plus der Größe des Fensterrahmens.

Soll ein Fenster größer als diese Maximalgrößen gezeichnet werden, so ist muss entweder maxheight bzw. maxwidth gesetzt werden oder, falls maxheight bzw. maxwidth nicht gesetzt sind, ergibt sich die Maximalgröße implizit aus vheight und vwidth (hierbei müssen aber hsb_visible bzw. vsb_visible auf *true* gesetzt sein). Bei Vorgabe der Maximalgröße eines MDI-Kindfensters durch das Vaterfenster erfolgt die Priorisierung entgegengesetzt, d.h. zuerst über vheight und vwidth und danach über maxheight bzw. maxwidth.

### 49.4.3 Fensterzustand bei ungewöhnlichen Attributwerten

In Version A.05.02.g wurde die Behandlung der Attribute *.iconic* und *.maximized* überarbeitet, da der Zustand eines Fensters nicht immer den Einstellungen dieser Attribute entsprach. In den vorherigen Versionen konnten Probleme auftreten, wenn andere Attribute, wie *.titlebar*, *.sizeable* und *.borderwidth*, ungewöhnliche – in der Praxis selten verwendete – Werte oder Wertkombinationen aufwiesen.

Ab Version A.05.02.g gibt es folgende Änderungen und Korrekturen:

» **Änderung**
Bei

```
.maximized true;
.iconic true;
```

wird ein Fenster minimiert dargestellt und zu maximiert restauriert. In den vorherigen Versionen wurde es manchmal nicht minimiert dargestellt und oft zur normalen Größe restauriert.

» **Korrektur**
Bei

```
.titlebar false;
.sizeable true;
.borderwidth 0;
```

wird ein Fenster mit der korrekten Breite – und nicht wie in vorherigen Versionen um *2* Pixel zu breit – angelegt.

» **Korrektur**
Bei

```
.titlebar false;
.sizeable true;
.borderwidth 0;
.iconic true;
```

wird ein Fenster korrekt minimiert dargestellt.

### 49.4.4 Weitere Hinweise

» Zur Benutzung einer Radmaus zum Scrollen bitte Kapitel „Radmaus bzw. Radmausunterstützung unter Microsoft Windows“ beachten.

» Ist am Fenster keine maximale Breite bzw. Höhe gesetzt (*.maxwidth = 0* bzw. *.maxheight = 0*), dann wird die virtuelle Breite bzw. Höhe als maximale Breite bzw. Höhe verwendet. Voraussetzung ist, dass die virtuelle Breite bzw. Höhe gültig ist (*.vwidth <> 0* und *.hsb_visible = true* bzw. *.vheight <> 0* und *.vsb_visible = true*).
Ein Fenster ist also nur dann beliebig vergrößerbar, wenn weder eine maximale noch eine virtuelle Größe gesetzt ist.
Außerdem ist zu beachten, dass die Schieberegler (scrollbars) zur Darstellungsfläche gehören. Hierdurch kann die Darstellungsfläche noch verschoben (gescrollt) werden, obwohl das Fenster die virtuelle Größe besitzt.

» Die minimale Größe eines Fensters muss, um Probleme zu vermeiden, so gesetzt werden, dass die Höhe und Breite des Arbeitsbereichs nicht kleiner als 0 werden kann. Insbesondere muss die minimale Höhe so gesetzt werden, dass alle Menüs der Menüzeile Platz haben, wenn das Fenster die minimale Breite besitzt. Aus diesem Grund kann es sein, dass ein Fenster höher dargestellt wird als gewünscht, bzw. dass ein Fenster sich vergrößert, wenn zusätzliche Menüs sichtbar geschaltet werden. Um solche Effekte zu vermeiden reicht es die minimale Breite (*.minwidth*) so groß zu wählen, dass die Menüzeile nicht mehrzeilig werden kann (alle Menüs passen nebeneinander).

» Bei Änderung der Taskbar-Einstellungen werden unter Microsoft Windows Fenster unter Umständen so verschoben, dass sie komplett zu sehen sind. Das selbe Verhalten kann auch beobachtet werden, wenn der Rechner vom Benutzer gesperrt und wieder entsperrt wird. Dies ist ein generelles Verhalten von Microsoft Windows und kann auch mit anderen Applikationen (Internet Explorer, Eingabeaufforderung (DOS-Box), Outlook, Notepad…) beobachtet werden. Auch ein maximiertes Fenster, dessen Größe eingeschränkt ist, wird auf diese Weise behandelt und damit verschoben.

» Unter Microsoft Windows lässt es sich nicht umgehen, dass ein in der Taskleiste dargestelltes Fenster aktiviert werden kann. Ist zu diesem Zeitpunkt ein ***Window***-Objekt mit *.dialogbox = true* (Dialogbox) oder ein ***Messagebox***- bzw. ***Filerequestor***-Objekt geöffnet, wird daher zunächst ein falsches Objekt aktiviert. Kurz danach aktiviert der Dialog Manager das richtige Objekt, also die Dialogbox bzw. das ***Messagebox***- oder ***Filerequestor***-Objekt.
Hierbei gibt es folgende Einschränkung:
Unter bestimmten Umständen muss der Dialog Manager heuristisch nach dem richtigen Objekt suchen. Um zu verhindern, dass hierbei ein falsches Objekt aktiviert wird sollte folgendes vermieden werden:

  » Mehrere ***Messagebox***- oder ***Filerequestor***-Objekte sind gleichzeitig geöffnet. Toplevel-Objekte, die nicht durch den Dialog Manager erzeugt wurden, sind ebenso problematisch.

  » Es wird beim Schließen dieser Objekte die Reihenfolge (zuletzt geöffnet wurde zuerst geschlossen) nicht beachtet.

  » Nach dem Öffnen eines modalen Objekts werden am Toplevel-Fenster der Anwendung die Attribute *.sensitive*, *.mapped*, *.iconic* oder *.visible* geändert.

» Der IDM unterstützt das **Multi Document Interface** (MDI) von MICROSOFT WINDOWS.
Um ein „MDI-frame-window“ zu erzeugen, muss das Fenster als Toplevel-Fenster definiert sein, eine Titelleiste haben und darf keine anderen Kinder als Fenster haben. Andere Kinder als Fenster oder das Fehlen einer Titelleiste (*.titlebar = false*) erzeugen Fehler.
Um ein „MDI-child-window“ zu erzeugen, muss es als direktes Kind eines „MDI-frame-window“ definiert werden. Ein „MDI-child-window“ hat immer eine Titelleiste und kann keine Menüleiste haben.
Alle anderen Fenster sind normale Fenster, die Toplevel-Fenster oder Kind-Fenster sein können. Ein normales Kind-Fenster kann nicht den Rahmenstil eines „aktiven“ Fensters sowie keine Menüleiste haben und besitzt immer eine Titelleiste sowie denselben System-Accelerator wie das Toplevel-Fenster.
Kind-Fenster, die nicht das MDI benutzen, werden nicht unterstützt, weshalb sich viele Attribute und Ereignisse nicht wie erwartet verhalten.

**Anmerkung**
MICROSOFT pflegt das „Multi Document Interface“ seit WINDOWS 8 nicht mehr (und rät schon seit WINDOWS 95 von seiner Verwendung ab). Dies betrifft zum Beispiel die „Visual Styles“, sodass MDI-Kindfenster anders aussehen als normale Toplevel-Fenster.

## 49.5 Hinweis zum IDM für Motif

Bitte beachten Sie in Bezug auf die Anordnung von Fenstern vor oder hinter anderen Fenstern und Dialogfeldern auf dem Bildschirm (Z-Ordnung) den Hinweis in Kapitel „Z-Ordnung von Fenstern und Dialogfeldern“.

## 49.6 Besonderheiten des Fensters unter Qt

Das ***Window***-Objekt verfügt über vier weitere Optionen für die Unterstützung von Toolbars und „Tabbed Docks“ sowie um Größe und Position eines Fensters Qt-konform zu setzen.

***Tabelle 3:*** *Optionen des Window-Objekts*

| .options[…] Wert | Standardwert | Bedeutung |
|---|---|---|
| *opt_window_size* | ***false*** | *true*: Größenangaben (*.width*, *.height*, Minimal- und Maximalwerte) beziehen sich auf das gesamte Fenster, inklusive Menüleiste, Toolbars, Tabbed Widgets und Statusleiste, aber ohne Dekoration (Titelzeile, Ränder).<br>***false***: Größenangaben beziehen sich auf den Innenbereich („client area“) des Fensters. |
| *opt_animated* | ***false*** | *true*: Animiertes, interaktives Verschieben von ***Toolbars***. Dies führt allerdings zu deutlich mehr *resize*- und *move*-Ereignissen.<br>***false***: Keine Animationen beim Verschieben von ***Toolbars***. |
| *opt_nested_docks* | ***false*** | Die Option wirkt sich nur auf ***Toolbars*** mit dem Stil *notepage* aus.<br>*true*: Nebeneinander liegende, mehrreihige Docks sind möglich. Dies führt allerdings zu einer nicht mehr so klaren Bedienung bei Interaktionen und Verschiebungen.<br>***false***: Nur einreihige Docks sind möglich. |
| *opt_tabbed_docks* | ***false*** | Die Option wirkt sich nur auf ***Toolbars*** mit dem Stil *notepage* aus.<br>*true*: „Tabbed Docks“, die sich den Platz teilen, sind möglich. „Tabbed Docks“ lassen sich ähnlich wie Notebooks und Notepages bedienen (siehe Kapitel „Besonderheiten der Toolbar unter Qt“ beim Objekt ***toolbar***).<br>***false***: „Tabbed Docks“ sind nicht möglich. |

Wie unter MOTIF (und im Unterschied zu MICROSOFT WINDOWS) wirkt das Attribut .sensitive unter Qt nicht auf die Titelzeile des ***Fensters***. Auch wenn *.sensitive = false* ist (und sich der Fensterinhalt entsprechend nicht mehr bedienen lässt), können dennoch die Funktionen der Titelleiste (Schließen, Verschieben, Minimieren, Maximieren…) genutzt werden.

Das Attribut .moveable hat unter Qt keine Auswirkung. Fenster lassen sich immer verschieben.

Generell werden beim IDM FÜR QT deutlich mehr *move*- und *resize*-Ereignisse erzeugt als bei den anderen IDM-Versionen, da für jedes Pixel, um das vergrößert, verkleinert oder verschoben wird, ein entsprechendes Ereignis ausgelöst wird. Es existiert keine zuverlässige Möglichkeit, Zustand und Ende der Aktionen zu bestimmen.

Manche Fensterzustände lassen sich nur durch internes Neuanlegen („Clobbering“) ändern, wodurch allerdings *activate*- und *deactivate*-Ereignisse ausgelöst werden.

Es kann noch vorkommen, dass Größenangaben in bestimmten Konstellationen nicht korrekt umgesetzt werden (z. B. bei ihrer Änderung im minimierten Zustand). Momentan kann das Verhalten je nach Window Manager unterschiedlich sein (z. B. beim Minimieren).

## 49.7 Beispiel

### Erzeugen eines Fensters

```
dialog Test
{
}
window WnTest
{
  .width 506;
  .height 238;
  .title "Urlaubsplanung";
}
```


***Abbildung 52:*** *Fenster*

### Erzeugung eines Fensters mit Statusbar und Scrollbar rechts und unten

```
dialog Test
{
}
window WnTest
{
  .width 506;
  .height 238;
  .vwidth 600;
  .vheight 500;
  .hsb_visible true;
  .hsb_optional false;
  .vsb_visible true;
  .vsb_optional false;
  .title "Urlaubsplanung";

  child statusbar StBar
  {
    .xleft 38;
    .ytop 227;

    child statictext St1
    {
      .xleft 38;
      .ytop 227;
      .text "Hier steht der erste Text";
    }

    child statictext
    {
```

```
      .xleft 38;
      .ytop 227;
      .text "Und hier der zweite";
      .depth -2;
    }
  }
}
```

Resultat des vorstehenden Beispiels zur Erzeugung eines Fensters mit Statusbar und Scrollbar rechts und unten:


***Abbildung 53:*** *Fenster mit Statusbar und Scrollbars*

# Index

## E

## F



## G



## H



## I



## K

## L



## M

**N**



**O**

**P**

## Q



## R



## S

## T