ISA Dialog Manager

RESSOURCENREFERENZ

A.06.03.b

In diesem Handbuch sind alle Ressourcen des ISA Dialog Managers beschrieben. Ressourcen sind Objekte wie Cursors, Farben, Texte und Schriftarten über die bestimmte Merkmale des Aussehens oder Verhaltens von IDM Objekten festgelegt werden können.

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

- Layoutressourcen
- Programmierressourcen

im Dialog Manager.

Jedes Objekt, das im Dialog Manager wieder angesprochen werden soll, muss mit einem erlaubten Identifikator versehen sein. Im Gegensatz zu Objekten können Ressourcen (mit wenigen Ausnahmen) nur dann angesprochen werden, wenn sie einen Namen tragen. Dieser Name muss im Modul oder Dialog eindeutig sein.

Ressourcen sind Objekte, die als Attributwerte anderen Objekten zugewiesen werden oder die zur Programmierung eines Ablaufs notwendig sind. Diese Ressourcen werden mit einem logischen Namen definiert und können teilweise verschiedene physikalische Ausprägungen haben.

Dem Dialogdesigner stehen folgende Ressourcen zur Verfügung:

- Layoutressourcen
  - Accelerator
    Tastaturzugriff für Objekte.
  - Cursor
    Um dem Benutzer bestimmte Dialogzustände, z.B. Wartesituationen, anzuzeigen, können unterschiedliche Cursor eingesetzt werden.
  - Display
    Mit der Display-Ressource kann ein Fenster einem bestimmten Bildschirm zugeordnet werden. Systeme mit mehreren unabhängigen Bildschirmen werden nur vom ISA Dialog Manager für Motif unterstützt.
  - Farbe (color)
    Mit Hilfe von Farben kann das Aussehen der Objekte definiert werden.
  - Format
    Um Strings in editierbaren Texten und Tablefields formatieren zu können, können Format-Ressourcen definiert werden.
  - Muster (tile)
    Mit Hilfe dieser Ressource können Muster definiert werden.
  - Text
    Mit Hilfe der Texte können unterschiedliche Sprachen definiert werden.
  - Zeichensatz (font)

Mit Hilfe dieser Ressource kann einzelnen Objekten ein beliebiger Zeichensatz zugeordnet werden.

- Programmierressourcen
  - Message
    Mit dieser Ressource werden Nachrichten definiert, die als externe Ereignisse an Objekte geschickt werden können.
  - Source
    Mit Hilfe dieser Ressource wird ein Objekt als per Drag&Drop verschiebbar gekennzeichnet. Es wird mit dieser Ressource definiert, welche Aktion (cut, copy) das Verschieben auslöst.
  - Target
    Mit Hilfe dieser Ressource wird ein Objekt so gekennzeichnet, dass es das Ziel einer Drag&Drop-Operation sein kann.

2 Layoutressourcen

Die Layoutressourcen dienen dazu, Details des Betriebssystems und des Grafiksystems vor dem Benutzer zu verbergen. Das heißt, jede Ressource erhält vom Dialog-Designer einen DM-internen Identifikator. Mit diesem Identifikator kann dann der Dialog-Designer arbeiten.

Zu den Layoutressourcen zählen:

- Accelerator
- Cursor
- Farbe
- Format
- Muster
- Text
- Zeichensatz

Der generelle Formalismus zur Ressourcendefinition lautet wie folgt:

Ressourcendefinition

```
{export | reexport}
<Ressourcenklasse>  <Identifikator>  <Ressourcenbeschreibung>;
```

Anmerkung

Die Keywords bzw. 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 oder die Ressource innerhalb eines Moduls definiert ist.

Siehe auch

Kapitel „Modularisierung“ im Handbuch „Programmiertechniken“

In der oben aufgeführten Definition wird einer Ressource **ein** Wert zugewiesen.

Um portable Anwendungen für unterschiedliche Plattformen (was z.B. verschiedene Tastaturen, Bildschirme, Sprachen und Fenstersysteme anbelangt) zu definieren, gibt es eine zweite Methode, mehrere Varianten für den Wert einer stark plattformspezifischen Ressource (z.B. Farbe, Cursor, Zeichensatz und Muster) zu definieren.

Da beide Möglichkeiten zur Verfügung stehen, erreicht man einen hohen Grad an Flexibilität: einfache Fälle kann man mit der oben beschriebenen Methode abdecken, komplizierte Fälle mit dem im Folgenden beschriebenen Verfahren.

Anmerkung

Die Programmierressourcen Funktion und Variable sind nicht von der Plattform abhängig, auf der der Dialog läuft. Darum werden variante Notationen für diese beiden Ressourcen nicht unterstützt.

Keyword:

<resource class>

Zusätzlich zu dem oben beschriebenen Spezifikationsformat (z.B. Farbspezifikation unter Verwendung von RGB-Werten oder Zeichensatzspezifikation unter Verwendung des logischen Zeichensatzformates) gibt es ein weiteres Spezifikationsformat, das erlaubt, Varianten für die Ressourcen Accelerator, Farbe, Cursor, Zeichensatz, Text oder Muster.

Die nicht-variante Syntax der Ressourcendefinition kann in der Varianten-Notation unter Verwendung der gleichen syntaktischen Elemente geschrieben werden, wie in der folgenden Definition gezeigt wird.

Nicht-variante Definition

```
<Ressourcenklasse>  <Identifikator>  <Ressourcenwert>;
```

Variantendefinition

```
<Ressourcenklasse>  <Identifikator>
{
  0:          <Ressourcenwert>;
  <Ziffer>:   <Ressourcenwert>;
  [<Ziffer>:  <Ressourcenwert>;]
}
```

Eine variante Notation mit nur einer Variante 0 wird vom Dialog Manager der Übersichtlichkeit halber als nicht-variante Ressource abgespeichert und muss folglich auch als nicht-variante Notation definiert werden.

Die Variante 0 sollte für alle varianten Ressourcen definiert werden, außer für die Ressource Text, bei der sie nicht benötigt wird.

Im Folgenden finden Sie für jede Ressource zuerst die nicht-variante Beschreibung, gefolgt von der varianten Definition.

2.1 accelerator und Mnemonic

Accelerators und Mnemonics ermöglichen geübten Benutzern schnelles und effektives Arbeiten mit der Anwendung. Mnemonics und Accelerators werden primär im Menüsystem verwendet. Der Dialog Manager selbst unterstützt für alle selektierbaren Objekte Mnemonics und Accelerators. Ihre Einsatzfähigkeit ist jedoch stark vom Fenstersystem abhängig.

Beim Einsatz von Mnemonics und Accelerators sind die Randbedingungen Multilingualität und Verwendung unterschiedlicher Tastaturen zu beachten, denn Alpha-Zeichen sind sprachabhängig, und Funktionstasten und Sondertasten sind tastaturabhängig. Es gibt bei Tastaturen keine verlässliche Untergruppe von Funktions- und Sondertasten.

Anmerkungen

- Im Dialog Manager wird bei Acceleratoren und Mnemonics nicht zwischen Groß- und Kleinschreibung bei Buchstaben unterschieden. So ist z.B. für die Auslösung von Objekten unbedeutend, ob der Benutzer „a“ oder „A“ drückt, wenn dieser Buchstabe als Accelerator oder Mnemonic definiert wurde.
- Mnemonics lösen identisch zu den Acceleratoren das entsprechende Objekt sofort aus.
- Wird ein Mnemonic oder Accelerator an einem nicht selektierbaren statischen Text definiert, bewirkt die Selektion des entsprechenden Mnemonics oder Accelerators, dass das dem statischen Text nachfolgende Objekt fokussiert wird, falls dieses Objekt sichtbar und selektierbar ist.

2.1.1 accelerator

Ein Accelerator ist eine Taste oder eine Tastenkombination, die mit einem Dialogobjekt verbunden ist. Durch Betätigung einer Accelerator-Taste oder

- alphanumerischer Tasten, die sprachabhängig sind, aber tastaturabhängig sein können,
- alphanumerischer Tasten, die nicht sprachabhängig sind, aber tastaturabhängig sein können.

Um einem Objekt einen Accelerator zu geben, muss zuerst eine Ressource ***accelerator*** definiert werden. Die Ressource ***accelerator*** besteht aus dem Schlüsselwort **accelerator**, einem Identifikator, einer öffnenden geschweiften Klammer, einer oder mehrerer Tastenspezifikationen und der schließenden geschweiften Klammer. Vor der Tastenspezifikation steht eine Zahl, gefolgt von einem Doppelpunkt. Diese Zahl steht für den Tastaturtyp. Durch Setzen des Tastaturtyps wird die jeweilige Tastenspezifikation ausgewählt.

Durch Drücken der Tastenkombination, wird das mit diesem Accelerator verbundene Dialogobjekt selektiert und die mit diesem Objekt verbundene Aktion ausgeführt.

Definition

```
{ export | reexport } accelerator <Bezeichner> <acceleratorSpec> |
<variantDef>
```

```
acceleratorSpec ::=
  <Tastendefinition> { , "<angezeigter Name>" | <text-Ressource> } ;

Tastendefinition ::=
  { <Modifikator> + } { <Modifikator> + } { <Modifikator> + }
  & | '<alphanumerische Taste>' | <Funktionstaste> | <Sondertaste>

Modifikator ::= alt | cntrl | shift

Funktionstaste ::= F1 | F2 | … | F98 | F99

variantDef ::=
{
           0 : <acceleratorSpec>
  [ <Nummer> : <acceleratorSpec> ]
}
```

Die Tastenkombination für einen Accelerator wird bei einigen Dialogobjekten rechts neben dem Text angezeigt (fenstersystemabhängig).

Über den optionalen String oder die optionale Textressource nach der eigentlichen Definition der Taste kann der Text angegeben werden, der als Acceleratortext im Objekt erscheinen soll. Dieser Text wird dann in Menüeinträgen entsprechend angezeigt.

Accelerators besitzen im Gegensatz zu Mnemonics kein visuelles Feedback. Das Attribut *.real_visible* muss *true* sein, das mit dem Accelerator verbundene Objekt muss nicht im sichtbaren Bereich liegen. Selektiert werden können nur Dialogobjekte mit Accelerators im aktiven Fenster. Die Accelerators aller Dialogobjekte in einem Fenster müssen eindeutig sein. Wird ein Accelerator in einem Fenster mehrfach verwendet, ist nicht vorhersagbar, welcher Accelerator wirksam werden wird.

Accelerators sind sprach- und tastaturabhängig und werden deshalb wie andere Ressourcen (Farben und Zeichensätze) behandelt, die ebenfalls hardwareabhängig sind.

Der DM unterscheidet zwischen drei Arten von Tasten:

- Funktionstasten, die tastaturabhängig sind,
- alphanumerische Tasten, die sprachabhängig sind, aber tastaturabhängig sein können,
- alphanumerische Tasten, die nicht sprachabhängig sind, aber tastaturabhängig sein können.

Die Acceleratorvariante kann beim Aufruf des Dialog Managers mit der Option

```
-IDMkeyboard [<Zahl>]
```

eingestellt werden.

0 ist der Default-Tastaturtyp, der benutzt wird, wenn kein Tastaturtyp gesetzt worden ist oder der Accelerator für den eingestellten Tastaturtyp keine Variante besitzt. Daher ist es nützlich, die 0-Variante in der Accelerator-Definition allgemein zu verwenden.

Eine Accelerator-Tastenkombination besteht aus einer Taste, der eine beliebige Kombination von Modifiertasten durch das +-Symbol vorangestellt sein kann. Modifiertasten sind *alt*, *cntrl* und *shift*. Alphanumerische Tasten werden mit 'Hochkomma-Zeichen-Hochkomma' spezifiziert.

Beispiel

```
accelerator A
{
  0:  alt+'a';
}
```

In der vorliegenden Motif-Version des Dialog Managers werden zurzeit Acceleratoren nur im Objekt menuitem beschriftet.

Beispiel

```
accelerator A_QUIT
{
  0:  cntrl+F2;
  1:  cntrl+F5;
}
```

Soll ein Accelerator sprachabhängig sein und sich durch Änderung der Sprache automatisch mit ändern, so spezifiziert man an Stelle der Taste ein „&“. Dies bewirkt, dass das mnemonische Zeichen für das Dialogobjekt auch für den Accelerator benutzt wird. Voraussetzung ist, dass das jeweilige Objekt ein Mnemonic besitzt.

Beispiel

```
accelerator A_QUIT
{
  0:  cntrl+&;
  1:  cntrl+alt+&;
}
text  "E&xit"
{
  1:  "&Beenden";
  2:  "&Finito";
}
child menuitem M1
{
  .text         E&xit;
  .accelerator  A_QUIT;
}
```

Diese Ressource mit dem Namen A_QUIT kann jetzt einem Dialogobjekt zugewiesen werden. Das Dialogobjekt wird automatisch mit dem Text des Accelerators beschriftet, unter der Voraussetzung, dass das verwendete Fenstersystem eine Beschriftung für dieses Objekt zulässt.

Anmerkung

Die Tasten `Strg`, `Alt` und `Entf` sollten nicht alle drei gleichzeitig verwendet werden.

Modifiertasten können nicht als Accelerator verwendet werden, alleinstehende alpha-numerische Zeichen und Cursor-Tasten allein sollten nicht als Accelerator verwendet werden. Das heißt sie sollten in Kombination mit anderen Tasten verwendet werden.

Tasten oder Tastenkombinationen, die bereits eine vordefinierte Funktion im Fenstersystem haben, sollten nicht verwendet werden.

Bitte beachten Sie außerdem den Abschnitt „Anmerkung zu problematischen Situationen“.

Alpha-numerische Zeichen können nicht in Verbindung mit dem *shift*-Modifier spezifiziert werden. **Funktions-** und **Sondertasten** werden mit den Namen angegeben, die in der folgenden Tabelle aufgelistet sind:

BackSpace	/* back space, back char */
Begin	/* BOL */
Break	
Cancel	/* Cancel, stop, abort, exit */
Clear	
Delete	/* Delete, rubout */
Down	/* Move down, down arrow */
End	/* EOL */
Escape	
Execute	/* Execute, run, do */
Find	/* Find, search */
Home	
Insert	/* Insert, insert here */
Left	/* Move left, left arrow */
Linefeed	/* Linefeed, LF */
Next	/* Next, Page Down */
Pause	/* Pause, hold, scroll lock */
Print	
Prior	/* Prior, previous, PageUp */

Redo	/* redo, again */
Return	/* Return, enter */
Right	/* Move right, right arrow */
Select	/* Select, mark */
Tab	
Undo	/* Undo previous*/
Up	/* Move up, up arrow */

Funktionstasten:

F1 - F99

Modifiertasten:

- alt
- cntrl
- shift

Anmerkung zur Caps-Lock-Taste

Die `Shift`-Taste wird bei den Funktionstasten in der üblichen Art und Weise interpretiert, die `Caps Lock`-Taste jedoch ignoriert. Wenn der Benutzer also `Shift` + `F3` auslösen möchte, muss er die `Shift`-Taste und die `F3`-Taste drücken, die Kombination `Caps Lock`-Taste und `F3`-Taste hingegen führt nicht zum Auslösen des entsprechenden Accelerators.

Anmerkung zu problematischen Situationen

Es sollten keine Systemacceleratoren verwendet werden, da es ansonsten zu einer Mehrfachauslösung bzw. einem Abfangen des Ereignisses (*on key* bzw. *select*) kommen kann; dies kann außerdem je nach Accelerator und Fenstersystem unterschiedlich sein.

Beispiele

- Unter Microsoft Windows stellen sowohl `F1` als auch `Strg` + `F1` Systemacceleratoren dar und sollten deshalb nicht verwendet werden. In beiden Fällen wird hier ein *on help* Event an das jeweiligen Objekt geschickt, die Erzeugung des Events geschieht allerdings in Microsoft Windows selbst.
- Der Accelerator `F10` ruft unter Windows das Systemmenü des aktiven Fensters auf, allerdings wird das Menü in den meisten Fällen nicht geöffnet (es öffnet sich, wenn man nach dem Accelerator die Pfeiltaste `Curosr Abwärts` drückt). Zu beachten ist, dass die Ereignisverarbeitung der Anwendung angehalten wird, sobald das Systemmenü des Fensters aufgerufen wird.

Außerdem sollte die Verwendung eines Accelerators

» an mehreren Objekten in einer *on key* Regel

» an einem oder mehreren Objekt in einer *on key* Regel und zugewiesenem Attribut *.accelerator* (Auslösung des *select* Ereignisses)

genau überlegt sein. Je nach verwendetem Fenstersystem, augenblicklichem Dialogzustand, Art und Weise bzw. Reihenfolge der Sichtbarmachung der Fenster könnte es zu Doppelauslösungen des betroffenen Accelerators an verschiedenen Objekten oder am selben Objekt kommen.

2.1.1.1 Menüs und Acceleratoren unter Microsoft Windows

Unter Microsoft Windows dienen Acceleratoren dazu, Menüfunktionen auszuwählen ohne dazu ein Menü zu öffnen. Daraus ergibt sich, dass bei einem geöffneten Menü die Acceleratoren nicht mehr ausgelöst werden. Stattdessen kann man jetzt die jeweiligen Mnemonics zur Auswahl eines Menüeintrags nutzen.

2.1.2 Mnemonic

Mnemonics zählen nicht zu den Ressourcen als solche; sie müssen als Attribute definiert werden. Aber aufgrund ihrer engen Verwandtschaft zum Accelerator wird ihre Definition an dieser Stelle aufgeführt.

Ein Mnemonic ist genau ein im Text eines Dialogobjektes hervorgehobenes, alphanumerisches Zeichen, mit dem das entsprechende Objekt angewählt werden kann.

Beispiel

In einer Menübox:	**A**blage
In einem Pushbutton:	**E**xit

Das entsprechende Menü wird durch Drücken einer speziellen Taste, der "Mnemonictaste", zusammen mit dem in diesem Menü markierten Zeichen geöffnet. Ein Menüeintrag kann nun durch das Drücken der Taste aktiviert werden, welche dem im Menü markierten Zeichen entspricht. Nun kann keine Spezial-Taste mehr aktiviert werden.

Mnemonics haben üblicherweise ein visuelles Feedback, d.h. das angewählte Dialogobjekt erscheint auf dem Bildschirm selektiert und die mit dem Objekt verbundene Aktion wird ausgeführt, wenn die zugeordnete Taste gedrückt wird. Mit Mnemonics können nur die Dialogobjekte angewählt werden, die sich im aktiven Fenster befinden.

Das mnemonische Zeichen stammt immer aus dem gegebenenfalls sprachabhängigen Text und wird mit diesem Text zusammen spezifiziert. Der auf das *&*-Symbol folgende Buchstabe ist das mnemonische Zeichen und wird hervorgehoben auf dem Bildschirm dargestellt. Das *&*-Symbol wird nicht angezeigt. Soll ein *&*-Zeichen in einem Dialogobjekt auf dem Bildschirm erscheinen, müssen zwei aufeinanderfolgende *&*-Zeichen spezifiziert werden.

Die Sprache wird beim Aufruf des Dialog Managers mit

```
-IDMlanguage [<Zahl>]
```

eingestellt.

Beispiel

```
text  "E&xit"
{
  1:  "&Beenden";
  2:  "&Finito";
}
child menuitem M1
{
  .text  E&xit;
}
```

Die Verfügbarkeit der Mnemonics ist abhängig vom jeweiligen Fenstersystem und funktionieren nur, wenn das Objekt vom Benutzer selektiert werden könnte.

Microsoft Windows

Mnemonics werden für folgende Objekte unterstützt:

- Bild
- Checkbox
- Menübox
- Menüeintrag
- Pushbutton
- Poptext
- Radiobutton
- Statischer Text

Mnemonics können ausgelöst werden durch Drücken von

- `Alt` zusammen mit dem Mnemonic-Zeichen oder
- zuerst `Alt` und dann dem Mnemonic-Zeichen.

Alternativ zu `Alt` kann auch das Menü-Zeichen `F10` verwendet werden.

Mnemonic-Zeichen können ohne `Alt`-Taste verwendet werden, wenn ein Menü markiert ist.

Motif

Mnemonics werden für folgende Objekte unterstützt:

- Menübox
- Menüeintrag
- Poptext-Eintrag
- Pushbutton

Mnemonics können ausgelöst werden durch gleichzeitiges Drücken von

» `Alt` zusammen mit dem Mnemonic-Zeichen.

Ist schon ein Menü aktiviert, muss nur noch das mnemonische Zeichen gedrückt werden.

Sie können nur Objekte mit Mnemonics anwählen, die Sie gerade sehen.

Qt

Mnemonics werden für folgende Objekte unterstützt:

» ***Checkbox***
» ***Menubox***
» ***Menuitem***
» ***Pushbutton***
» ***Radiobutton***
» ***Statictext***

2.2 color (Farbe)

Alle Farben, die verwendet werden sollen, müssen als Ressource deklariert werden. Die Anzahl der darstellbaren Farben ist von dem von Ihnen verwendeten Grafiksystem abhängig.

Die Deklaration einer Farbe beginnt mit dem Schlüsselwort **color**. Anschließend steht der Identifikator der Farbe, gefolgt von der Farbdefinition. Die Deklaration wird durch ein ; abgeschlossen.

Definition

```
{ export | reexport } color <Bezeichner> <colorSpec> | <variantDef>

colorSpec ::=
  { rgb(<R>, <G>, <B>) | hls(<H>, <L>, <S>) | gradient("<kind> [ , <arg> ]")
|
    "<vordefinierte Farbe>" }
  { , grey(<N>) } { , black | white } ;

variantDef ::=
{
          0 : <colorSpec>
  [ <Nummer> : <colorSpec> ]
}
```

Beispiele

Nicht variant

```
color RED         rgb(255, 0, 0), grey(50), white;
color AQUA        hls(90, 127, 255), grey(127), white;
color YELLOW      rgb(255, 255, 0);
color BLUE        "blue", black;
color ClMagenta   "magenta";
color BEAUTYCOLOR "orchid", grey(10), white;
```

Variant

```
color BEAUTYCOLOR
{
  0: rgb(100,200,200);
  1: "orchid", grey(10), white;
  2: rgb(150,100,190);
  3: "pink";
}
```

Auf Farbbildschirmen wird ausschließlich der Farbeintrag verwendet, auf Graustufen-Bildschirmen nur die Graustufe und auf Schwarzweiß-Bildschirmen der Schwarzweiß-Wert.

rgb(<R>, <G>, <B>)

R, G, B sind die Farbintensitäten der Anteile der Farben **R**ot, **G**rün und **B**lau. Der Wertebereich für alle drei Werte ist 0 bis 255. Klammern und Kommata sind Bestandteile der Farbdeklaration.

Je nach Wahl der RGB-Werte verwendet MS Windows gerasterte Mischfarben. Diese können durch geeignete Wahl der RGB-Werte weitestgehend vermieden werden, z.B. 0, 63, 127, 191, 255.

```
color  MyGrey  rgb(127,127,127);
color  Azure  rgb(63,127,191);
```

hls(<H>, <L>, <S>)

Im HLS-Farbmodell sind Farben durch Farbton **H** (hue), Helligkeit **L** (lightness, luminance) und Farbsättigung **S** (saturation) definiert. Der Wertebereich für alle drei Werte ist *0* bis *255*.

Die Farben sind in einem Zylinder angeordnet, bei dem die obere Fläche Weiß und die untere Schwarz ist. Die Helligkeit L steigt von *0* (dunkel) in der unteren bis *255* (hell) in der oberen Fläche. Bei den Farben in der Kreisfläche, die den Mittelpunkt des Zylinders schneidet, ist *L = 127*. Diese Farben sind weder aufgehellt (durch Mischung mit Weiß) noch abgedunkelt (durch Mischung mit Schwarz).

Der Farbton H wird durch den Winkel auf der Kreisfläche festgelegt. Der Wertebereich für H von *0* bis *255* und der Winkel W im Farbkreis (eigentlich von *0* bis *359* Grad) hängen wie folgt zusammen:

- H = W / 2.
- W = (H modulo 180) * 2; das heißt, Werte von *180* bis *255* entsprechen den Farbtönen von *0* bis *75*.

Rot hat den Wert *H = 0* (oder *180*), *H = 60* (oder *240*) ergibt Grün und *H = 120* Blau.

Auf der senkrechten Geraden im Zentrum des Zylinders, liegen die Grautöne mit der Farbsättigung *S = 0*. Die Farben auf dem Zylindermantel haben die höchste Farbsättigung *S = 255*. Auf dem Umfang der mittleren Kreisfläche, liegen die sogenannten reinen Farben mit *L = 127* und *S = 255*.

grey(<N>)

Ist der Graustufenwert, der auf einem Graustufen-Bildschirm für die Farbe verwendet werden soll. Der Wertebereich von N ist 0 bis 255.

black | white

Sind die Farbwerte für Schwarzweiß-Bildschirme.

gradient("<kind> [, <arg>]")

Verfügbarkeit

Nur IDM FÜR QT

color-Ressourcen unterstützen unter Qt Farbverläufe (Gradienten).

Der Parameter <*kind*> definiert die Verlaufsart. Es darf nur eine Verlaufsart angegeben werden.

Die weiteren Parameter können Folgendes enthalten:

- Ergänzungsparameter zur Verlaufsart
- Stopp-Punkt-Definitionen
- Farbdefinitionen

Diese **Verlaufsarten** stehen zur Verfügung:

Tabelle 1: *Arten von Farbverläufen*

Verlauf	Definition	Erklärung
linear	`gradient("Linear, ...")` `gradient("LinearV, ...")`	vertikaler Verlauf
	`gradient("LinearH, ...")`	horizontaler Verlauf
radial	`gradient("Radial, ...")` `gradient("Radial, <R>, ...")`	radialer Verlauf mit Standardradius 50% radialer Verlauf mit **Ergänzungsparameter** Radius R% Radius wird in Prozent des verfügbaren Platzes angegeben
konisch	`gradient("Conical, ...")` `gradient("Conical, <S>, ...")`	konischer Verlauf mit Start bei 90° konischer Verlauf mit **Ergänzungsparameter** S°, der den Startwinkel angibt

Die **Farbdefinitionen** stehen immer nach der Verlaufsart und eventuellen Ergänzungsparametern. Es können Farbnamen, HTML-Notation sowie die für Farbressourcen bekannten Notationen *rgb(…)*, *hls(…)* und *grey(…)* verwendet werden.

Zusätzlich kann zu jeder Farbdefinition ein **Stopp-Punkt** angegeben werden, mit dem die Gewichtung der Farbe festgelegt wird. Ein Stopp-Punkt ist ein Prozentwert mit optionalem Prozentzeichen, der immer vor der jeweiligen Farbe steht und beeinflusst, wie viel Raum diese Farbe im Farbverlauf einnimmt. Ein Farbverlauf startet bei *0%* und endet bei *100%*, bezogen auf den Bereich, den er ausfüllt. Die Angabe *… , 20%, green, ...* bedeutet zum Beispiel, dass nach *20%* des zu füllenden Bereichs die Farbe Grün gesetzt wird. Wenn für den Bereich *0–20%* keine andere Farbe gesetzt ist, werden die ersten *20%* des Bereichs grün eingefärbt. Falls vor *20%* bereits eine andere Farbe gesetzt ist, dann wird ein Übergang zwischen dieser Farbe und Grün dargestellt.

Stopp-Punkte sollten immer in aufsteigender Reihenfolge gesetzt werden. Sind mehrere Farben mit demselben Stopp-Punkt definiert, gilt die Farbe, die in der Parameterliste am weitesten hinten steht. Die Definition *... , 20%, green, 40%, blue, 20%, red, ...* erzeugt beispielsweise einen Farb-

verlauf mit einem Rotton bei *20%*, der in einen Blauton übergeht, welcher ab *40%* satt dargestellt wird.

Beispiele

```
gradient("Linear, green, yellow, red");  benannte Farben, gleichmäßig
verteilt
gradient("Linear, #00FFFF, #00FF00, #FF0000");  HTML-Notation,
                                                Farben gleichmäßig
verteilt
gradient("Linear,red, #00FF00, rgb(0,0,255)");  gemischte Notationen,
                                                Farben gleichmäßig
verteilt
gradient("Linear, 20%, green, 60%, yellow, 80%, red");  benannte Farben
mit pro-
                                                        zentualen Stopp-
Punkten
gradient("Linear, 20, green, 60, yellow, 80 ,red");  das %-Zeichen bei den
                                                     Stopp-Punkten ist
optional
```

Hinweise

Qt lässt das Setzen von Farbverläufen an den meisten Stellen zu, wendet sie aber nicht unbedingt an. Ob ein Farbverlauf angezeigt wird, hängt sehr stark vom Objekt und UI-Stil ab. Dabei sind Flächen (z. B. Hintergründe) meist unproblematisch, bei filigranen Strukturen (z. B. Texten) wird der Verlauf oft durch eine einzelne Farbe ersetzt. Bei Gruppierungsobjekten werden Gradienten als Hintergrund in der Regel dargestellt.

<vordefinierte Farbe>

Ein durch das Fenstersystem definierter Farbidentifikator, der im ISA Dialog Manager verwendet werden kann.

2.2.1 Vordefinierte Farben

Eine Liste der auf dem jeweiligen System zur Verfügung stehenden vordefinieren Farben kann über das Attribut .colorname[integer] am setup Objekt erfragt werden.

2.2.1.1 Unabhängige UI-Farben

Die UI-Farbressourcen sind WSI-unabhängige vordefinierte Farben, die auf allen Systemen in ähnlicher Ausprägung bzw. Bedeutung verfügbar sind. Herangezogen werden hierzu die Default-Farben, die vom aktuellen Desktop/Window Manager oder Theme verwendet werden. Die einheitlichen COLOR-Ressourcen sind wie folgt definiert:

» UI_BG_COLOR
(allgemeine Hintergrundfarbe)

- UI_FG_COLOR
  (allgemeine Vordergrund-/Textfarbe)
- UI_BUTTONBG_COLOR
  (Hintergrundfarbe von Button-artigen Widgets)
- UI_BUTTONFG_COLOR
  (Vordergrund-/ Textfarbe von Button-artigen Widgets)
- UI_INPUTBG_COLOR
  (Hintergrundfarbe von Eingabe/ Auswahl-Widgets)
- UI_INPUTFG_COLOR
  (Vordergrund-/ Textfarbe von Eingabe/ Auswahl-Widgets)
- UI_ACTIVEBG_COLOR
  (Hintergrundfarbe von aktiven Elementen von Eingabe/ Auswahl-Widgets)
- UI_ACTIVEFG_COLOR
  (Vordergrund-/ Textfarbe von aktiven Elementen von Eingabe/ Auswahl-Widgets)
- UI_MENUBG_COLOR
  (Hintergrundfarbe von Menüs)
- UI_MENUFG_COLOR
  (Vordergrund-/ Textfarbe von Menüs)
- UI_TITLEBG_COLOR
  (Hintergrundfarbe von Titelleisten, Dialograhmen)
- UI_TITLEFG_COLOR
  (Vordergrund-/ Textfarbe von Titelleisten, Dialograhmen)
- UI_HEADERBG_COLOR
  (Hintergrundfarbe von Tabellenheadern)
- UI_HEADERFG_COLOR
  (Vordergrund-/ Textfarbe von Tabellenheadern)
- UI_BORDER_COLOR
  (Rahmenfarbe)
- UI_NULL_COLOR
  (Keine Farbe)

Hinweis für Windows

Die UI_*_COLOR Farben greifen auf die Windows Systemfarben zu. Hierbei ist zum einen zu beachten, dass mit Einführung der Visual Styles von den einzelnen Objekten nicht mehr die Systemfarben sondern durch das Theme definierte Bitmaps verwendet werden. Je nach ausgewähltem Theme kann es somit starke Diskrepanzen geben. Deshalb sollten unter Windows mach Möglichkeit keine Farben (oder UI_NULL_COLOR) gesetzt werden. Zum anderen ist zu beachten, daß mit Windows 10 die Farbvielfalt verringert wurde. Die Farben UI_HEADERBG_COLOR, UI_HEADERFG_COLOR, UI_MENUBG_COLOR, UI_MENUFG_COLOR, UI_TITLEBG_COLOR und UI_TITLEFG_COLOR

werden zum Beispiel nicht mehr unterstützt. Das heißt, dass diese Farben zwar noch verwendet werden können, sie aber von keinem Objekt verwendet werden.

2.2.1.2 Motif (X-Windows)

Bei der Verwendung von Systemnamen für Farb-Ressourcen ist es möglich die Default-Farben von Motif anzusprechen. Folgende Systemnamen stehen zur Verfügung:

- BACKGROUND
- FOREGROUND
- TOP_SHADOW
- BOTTOM_SHADOW
- SELECT
- ACTIVE_BACKGROUND
- ACTIVE_FOREGROUND
- ACTIVE_TOP_SHADOW
- ACTIVE_BOTTOM_SHADOW
- ACTIVE_SELECT
- TEXT_BACKGROUND
- TEXT_FOREGROUND
- TEXT_TOP_SHADOW
- TEXT_BOTTOM_SHADOW
- TEXT_SELECT
- INACTIVE_BACKGROUND
- INACTIVE_FOREGROUND
- INACTIVE_TOP_SHADOW
- INACTIVE_BOTTOM_SHADOW
- INACTIVE_SELECT
- SECONDARY_BACKGROUND
- SECONDARY_FOREGROUND
- SECONDARY_TOP_SHADOW
- SECONDARY_BOTTOM_SHADOW
- SECONDARY_SELECT

Darüberhinaus sind unter Motif (X-Windows) alle in der Farbdatei **rgb.dat** definierten Farben verfügbar.

- black / Black
- blue / Blue

- light blue / LightBlue
- medium blue / MediumBlue
- midnight blue / MidnightBlue
- navy blue / NavyBlue
- navy / Navy
- sky blue / SkyBlue
- coral / Coral
- cyan / Cyan
- gold / Gold
- yellow / Yellow
- green / Green
- dark green / DarkGreen
- forest green / ForestGreen
- lime green / LimeGreen
- pale green / PaleGreen
- spring green / SpringGreen
- dark slate grey / DarkSlateGrey
- dim grey / DimGrey
- light grey /LightGrey
- magenta/ Magenta
- maroon / Maroon
- orange / Orange
- pink / Pink
- red / Red
- indian red / IndianRed
- orange red /OrangeRed
- violet red / VioletRed
- salmon / Salmon
- sienna / Sienna
- tan / Tan
- turquoise / Turquoise
- violet / Violet
- blue violet / BlueViolet
- wheat / Wheat

- white / White

2.2.1.3 Microsoft Windows

- BLACK
- BLUE
- CYAN
- DARKBLUE
- DARKCYAN
- DARKGREEN
- DARKGREY
- DARKPINK
- DARKRED
- GREEN
- PINK
- RED
- WHITE
- YELLOW

Folgende von aktuellen Einstellungen abhängige Systemfarben stehen im IDM FÜR WINDOWS zur Verfügung:

- CLR_ACTIVEBORDER
- CLR_ACTIVETITLE
- CLR_ACTIVETITLETEXT
- CLR_APPWORKSPACE
- CLR_BACKGROUND
- CLR_BUTTONFACE
- CLR_BUTTONHIGHLIGHT
- CLR_BUTTONSHADOW
- CLR_BUTTONTEXT
- CLR_GRAYTEXT
- CLR_HIGHLIGHT
- CLR_HIGHLIGHTTEXT
- CLR_INACTIVEBORDER
- CLR_INACTIVETITLE
- CLR_INACTIVETITLETEXT

- CLR_MENU
- CLR_MENUTEXT
- CLR_SCROLLBAR
- CLR_WINDOW
- CLR_WINDOWFRAME
- CLR_WINDOWTEXT

2.2.1.4 Qt

Als Farbbezeichner können alle SVG-Farbnamen verwendet werden (**http://www.w3.org/TR/SVG/types.html#ColorKeywords**), wobei Groß- und Kleinschreibung nicht beachtet wird. Bei der Verwendung dieser Farbnamen (zum Beispiel „green“) werden die Farben entsprechend der SVG-Farbnamen des W3C umgesetzt. Deshalb unterscheiden sich manche Farben in Ton und Intensität von den X11-Farben.

Außerdem stehen folgende, von den aktuellen Einstellungen abhängige Qt-Standardfarben zur Verfügung:

- BASE Hintergrund von Eingabefeldern und bestimmten Container-Objekten (Listen)
- BASETEXT Vordergrund von Eingabefeldern und bestimmten Container-Objekten (Listen)
- BUTTON Schaltflächenhintergrund
- BUTTONTEXT Schaltflächenvordergrund
- HIGHLIGHT Hintergrund für Selektion und aktiven Eintrag
- HIGHLIGHTTEXT Vordergrund für Selektion und aktiven Eintrag
- SHADOW Schattenfarbe, meist sehr dunkel bis schwarz
- WINDOW Fensterhintergrund
- WINDOWTEXT Fenstervordergrund
- BRIGHTTEXT Textfarbe bei schwachem Kontrast
- ALTERNATEBASE alternativ Hintergrundfarbe (meist bei Tabellen)
- TOOLTIPBASE Tooltiphintergrund
- TOOLTIPTEXT Tooltipvordergrund
- LINK Link-Vordergrund
- LINKVISITED besuchter Link-Vordergrund
- PLACEHOLDERTEXT Platzhalter Vordergrund für verschiedene Eingabe-Windgets, ab Qt 5.12
- ACTIVE_BASE
- ACTIVE_BASETEXT
- ACTIVE_BUTTON
- ACTIVE_BUTTONTEXT

- ACTIVE_HIGHLIGHT
- ACTIVE_HIGHLIGHTTEXT
- ACTIVE_SHADOW
- ACTIVE_WINDOW
- ACTIVE_WINDOWTEXT
- ACTIVE_BRIGHTTEXT
- ACTIVE_ALTERNATEBASE
- ACTIVE_TOOLTIPBASE
- ACTIVE_TOOLTIPTEXT
- ACTIVE_LINK
- ACTIVE_LINKVISITED
- ACTIVE_PLACEHOLDERTEXT ab Qt 5.12
- DISABLED_BASE
- DISABLED_BASETEXT
- DISABLED_BUTTON
- DISABLED_BUTTONTEXT
- DISABLED_HIGHLIGHT
- DISABLED_HIGHLIGHTTEXT
- DISABLED_SHADOW
- DISABLED_WINDOW
- DISABLED_WINDOWTEXT
- DISABLED_BRIGHTTEXT
- DISABLED_ALTERNATEBASE
- DISABLED_TOOLTIPBASE
- DISABLED_TOOLTIPTEXT
- DISABLED_LINK
- DISABLED_LINKVISITED
- DISABLED_PLACEHOLDERTEXT ab Qt 5.12
- INACTIVE_BASE
- INACTIVE_BASETEXT
- INACTIVE_BUTTON
- INACTIVE_BUTTONTEXT
- INACTIVE_HIGHLIGHT
- INACTIVE_HIGHLIGHTTEXT

- INACTIVE_SHADOW
- INACTIVE_WINDOW
- INACTIVE_WINDOWTEXT
- INACTIVE_BRIGHTTEXT
- INACTIVE_ALTERNATEBASE
- INACTIVE_TOOLTIPBASE
- INACTIVE_TOOLTIPTEXT
- INACTIVE_LINK
- INACTIVE_LINKVISITED
- INACTIVE_PLACEHOLDERTEXT ab Qt 5.12

Normal- und Active-Zustand sind dabei identisch.

Der Bezeichner *DEFAULT* dient als Null-Farbe. Setzt man diese Farbe, wird die jeweilige Farbe auf die entsprechende Standardfarbe (abhängig von den aktuellen Systemeinstellungen) zurückgesetzt.

2.2.2 Dynamisch änderbare Attribute

In der nachfolgenden Tabelle werden die Attribute dargestellt, die dynamisch zur Laufzeit verändert werden können. Dabei ist zu beachten, dass sich *.rgb[enum]*, *.hls[enum]* und *.name* gegenseitig ausschließen, beim „get“ also ein *„can't get value“* kommt.

***Tabelle 2:** Änderbare Attribute der color-Ressource*

Attribut	Datentyp	Indexbereich	Beschreibung
.rgb[enum]	*integer* *0 … 255*	*color_red*, *color_green*, *color_blue*	Rot, Grün, Blau
.hls[enum]	*integer* *0 … 255*	*color_hue*, *color_light*, *color_sat*	Farbton, Helligkeit, Sättigung
.gradient	*string*	–	Art und Parameter eines Farbverlaufs
.name	*string*	–	Systemname der Farbe
.grey	*integer* *0 … 255*	–	Graustufe
.bw	*enum* *color_black*, *color_white*	–	Schwarz-Weiß-Wert

Bei den Attributen *.rgb[enum]* und *.hls[enum]* wird die Farbe erst nach Umsetzen des letzten Wertes (*color_blue* bzw. *color_sat*) aktualisiert, dadurch hat man die Möglichkeit die Farbe als Gesamtes zu setzen.

2.3 cursor

Grafikcursor, die verwendet werden sollen, müssen zuvor deklariert werden. Das Schlüsselwort hierzu ist **cursor**, gefolgt von dem Cursoridentifikator und der Cursordefinition.

Definition

```
{ export | reexport } cursor <Bezeichner> <cursorSpec> | <variantDef>

cursorSpec ::= <cursorBitmap> | "<vordefinierter Cursor>" ;

cursorBitmap ::= <x>, <y>,
                 <cursorString>
             [ , <cursorString> ]

variantDef ::=
{
           0 : <cursorSpec>
  [ <Nummer> : <cursorSpec> ]
}
```

Beispiel

Nicht variant

```
cursor MyCursor "XC_cross";
```

Variant

```
cursor MyCursor
{
  0: "XC_cross";
  1: "Cross";
}
```

2.3.1 Bitmap-Cursor

<x> und <y> bezeichnen wie bei der Definition eines Musters die jeweilige Ausdehnung in X- bzw. Y-Richtung in Pixel.

Der IDM lässt beliebige Breiten (<x>) und Höhen (<y>) für die Cursor zu. Die meisten Grafiksysteme begrenzen die Größe eines Cursors jedoch auf 32 * 32 Pixel (z.B. MS Windows) bzw. 16 * 16 Pixel (z.B. Motif).

<cursorString> bezeichnet eine Zeichenkette mit genau <x> Zeichen. Sie sollten genau <y> Zeichenketten mit jeweils <x> Zeichen spezifizieren.

In <cursorString> zulässige Zeichen sind Punkt(„.“), Leerzeichen(„ “), Plus(„+“), Doppelkreuz („#“) und der Großbuchstabe „X“. Die einzelnen Zeichen haben folgende Bedeutung:

„.“

„ “ (Leerzeichen)

Übernehmen der Hintergrundfarbe, d.h. der Cursor ist an dieser Stelle transparent.

„+“

Diese Stellen werden mit der invertierten Hintergrundfarbe dargestellt. Dies ist wichtig für Fälle, bei denen die Vordergrundfarbe identisch mit der Hintergrundfarbe ist.

„#“

Diese Stellen werden mit der Vordergrundfarbe gezeichnet, die vom Fenstersystem definiert ist.

„X“

Wird ein Objekt mit dem Cursor angeklickt, so **muss** dieser Hotspot auf dem Objekt sein, andernfalls wird das Anklicken nicht registriert. Dieses Zeichen muss genau einmal in der Definition des Cursors vorkommen. Ist kein X-Zeichen angegeben, geht der DM davon aus, dass der Hotspot in der Mitte des angegebenen Objektes liegt.

Beispiel

```
cursor Pointer 16, 16,
  "........+.......",
  ".......+++......",
  "......++X++.....",
  ".....++#####....",
  "....++#######...",
  "...++#########..",
  "..++##########+.",
  "..+++++#########",
  ".....++#####....",
  ".....++#####....",
  ".....++#####....",
  ".....++#####....",
  ".....++#####....",
  ".....++#####....",
  ".....++#####....",
  ".....+++++++....";
```

Anmerkung zum IDM für Windows

Mit „Cursor“ ist der Mauszeiger gemeint – der Begriff „Cursor“ hat in Microsoft Windows eine andere Bedeutung.

Die Cursorgröße in Microsoft Windows ist 32 x 32 (mit EGA- oder VGA-Auflösung). Eine andere Größe ist nicht erlaubt! Daher passt der Dialog Manager den Cursor an, indem mit Transparenten aufgefüllt wird oder der rechte bzw. untere Rand abgeschnitten werden.

2.3.2 Vordefinierte Cursor

Eine Liste der auf dem jeweiligen System zur Verfügung stehenden vordefinieren Cursor kann über das Attribut .cursorname[integer] am setup Objekt erfragt werden.

2.3.2.1 Unabhängige UI-Cursor

Die UI-Cursorressourcen sind WSI-unabhängige vordefinierte Cursor, die auf allen Systemen in ähnlicher Ausprägung bzw. Bedeutung verfügbar sind. Herangezogen werden hierzu die Default-Cursor, die vom aktuellen Desktop/Window Manager oder Theme verwendet werden. Die einheitlichen CURSOR-Ressourcen sind wie folgt definiert:

- UI_CURSOR
  (Standardcursor meist Pfeil)
- UI_IBEAM_CURSOR
  (Edittext-Einfügemarke)
- UI_WAIT_CURSOR
  (Anzeige, dass die Anwendung beschäftigt ist)
- UI_CROSS_CURSOR
  (Kreuz)
- UI_UP_CURSOR
  (Pfeil nach oben)
- UI_SIZEDIAGF_CURSOR
  (Sizing-Pfeil von links oben nach rechts unten)
- UI_SIZEDIAGB_CURSOR
  (Sizing-Pfeil von links unten nach rechts oben)
- UI_SIZEVER_CURSOR
  (Sizing-Pfeil von links nach rechts)
- UI_SIZEHOR_CURSOR
  (Sizing-Pfeil von oben nach unten)
- UI_MOVE_CURSOR
  (Sizing-Pfeil in alle Richtungen)
- UI_STOP_CURSOR
  (Verbotszeichen/ durchgestrichener Kreis)
- UI_HAND_CURSOR
  (Handsymbol/ Zeigefinger)
- UI_HELP_CURSOR
  (Pfeil mit Fragezeichen)

2.3.2.2 Motif

Bei der Definition über den Namen wird der entsprechende X-Cursor aus den Cursor-Zeichensätzen des X-Windows geladen:

- XC_X_cursor
- XC_arrow
- XC_based_arrow_down
- XC_based_arrow_up
- XC_boat
- XC_bogosity
- XC_bottom_left_corner
- XC_bottom_right_corner
- XC_bottom_side
- XC_bottom_tee
- XC_box_spiral
- XC_center_ptr
- XC_circle
- XC_clock
- XC_coffee_mug
- XC_cross
- XC_cross_reverse
- XC_crosshair
- XC_diamond_cross
- XC_dot
- XC_dotbox
- XC_double_arrow
- XC_draft_large
- XC_draped_box
- XC_exchange
- XC_fleur
- XC_gobbler
- XC_gumby
- XC_hand1
- XC_hand2
- XC_heart

- XC_icon
- XC_iron_cross
- XC_left_ptr
- XC_left_side
- XC_left_tee
- XC_leftbutton
- XC_ll_angle
- XC_lr_angle
- XC_man
- XC_middlebutton
- XC_mouse
- XC_pencil
- XC_pirate
- XC_plus
- XC_question_arrow
- XC_right_ptr
- XC_right_side
- XC_right_tee
- XC_rightbutton
- XC_rtl_logo
- XC_sailboat
- XC_sb_down_arrow
- XC_sb_h_double_arrow
- XC_sb_left_arrow
- XC_sb_right_arrow
- XC_sb_up_arrow
- XC_sb_v_double_arrow
- XC_shuttle
- XC_sizing
- XC_spider
- XC_spraycan
- XC_star
- XC_target
- XC_tcross

- XC_top_left_arrow
- XC_top_left_corner
- XC_top_right_arrow
- XC_top_side
- XC_top_tee
- XC_trek
- XC_ul_angle
- XC_umbrella
- XC_ur_angle
- XC_watch
- XC_xterm

Beispiel

```
cursor Clockcursor "XC_clock";
```

2.3.2.3 Qt

Es gibt folgende benannte Cursors, bei denen der entsprechende Qt CursorShape geladen wird:

- arrow
- blank
- busy
- closedhand
- cross
- dragcopy ab Qt 4.7
- dragmove ab Qt 4.7
- draglink ab Qt 4.7
- forbidden
- ibeam
- openhand
- pointinghand
- sizeall
- sizebdiag
- sizefdiag
- sizehor
- sizever

- splith
- splitv
- uparrow
- wait
- whatsthis

Die Darstellung der Cursors kann je nach UI-Stil unterschiedlich sein.

2.3.2.4 Microsoft Windows

Bei der Definition über den Namen wird der entsprechende Cursor aus dem Bestand der Microsoft Windows Objekte genommen.

Für Microsoft Windows Systeme gibt es folgende vordefinierte System-Cursor:

- APPSTARTING
- ARROW
- CROSS
- HAND
- HELP
- IBEAM
- ICON
- NO
- SIZE
- SIZEALL
- UPARROW
- WAIT
- SIZENESW
- SIZENS
- SIZEWE
- SIZENWSE

Hinweise

- Wenn der Pfad HAND für einen **selbstdefinierten** Cursor verwendet wurde, wird ab IDM-Version A.05.02.f der System-Cursor und **nicht mehr** der selbstdefinierte Cursor geladen.

2.4 display

Die ***display***-Ressource dient zur späteren Zuordnung eines Fensters an einen bestimmten Screen in einer **Multiscreen-Umgebung** (nur MOTIF). Den tatsächlich gewählten Screen kann man über das Attribut *.real_screen* erfragen.

Definition

```
{ export | reexport } display <Bezeichner> { screen(<Nummer>) } ;
```

Attribute

Attribut	RSD	PSD	Eigenschaft	Kurzbeschreibung
.screen	integer	integer	S,G/-/-	Screen-Nummer
.real_screen	integer	integer	-,G/-/-	wirkliche Screen-Nummer

Die Verwendung von ungültigen Screen-Nummern werden automatisch auf den Default-Screen (meistens, aber nicht zwangsläufig, 0) weitergeleitet. Gültige Nummern sind >=0. Das heißt, dass z.B. mit der Angabe von -1 der Default-Screen benutzt wird.

Besonderheiten

Das *.screen*-Attribut kann auch dynamisch per Regelsprache umgesetzt werden.

Verfügbarkeit

Multiscreen-Support ist nur vorhanden beim IDM für MOTIF. Gültige Screen-Nummern können über **xpdyinfo** bzw. das .screen[integer] Attribut am setup-Objekt ermittelt werden.

Siehe auch

Kapitel „Multiscreen Ssupport untder Motif“ im Handbuch „Programmiertechniken“

2.4.1 Beispiel

Folgender Dialog zeigt, wie zwei Fenster auf verschiedenen Screens platziert und wie z.B. eine Behandlung für eine Single-Screen-Konfiguration durchgeführt werden kann:

```
dialog MultiScreen

display DpyDef screen(-1);
display DpyPrim screen(0);
display DpySec screen(1);

default window WINDOW
{
```

```
  .width 200;
  .height 100;

  on close { exit(); }
}

// primary window on screen#0
// (not necessarily the default screen!)
window WiPrim
{
  .display DpyPrim;
  .title "Primary Window";

  statictext StDefScreen { }
}

// secondary window on screen#1
window WiSec
{
  .display DpySec;
  .title "Secondary Window";

  listbox LbScreens
  {
    .xauto 0;
    .yauto 0;

    // move secondary window to another screen dynamically
    on select
    {
      this.window.display.screen := this.userdata[thisevent.index];
    }
  }
}

on dialog start
{
  variable integer I;
  StDefScreen.text := "Default Screen: " + setup.screen;

  // position second window below primary
  // when running in single-screen configuration
  if (setup.screencount = 1) then
    WiSec.ytop := WiPrim.ytop + WiPrim.real_width + 50;
  endif

  // list available screens
```

```
  for I:=1 to setup.screencount do
    LbScreens.content[I] := "Screen#" + setup.screen[I] + "  " +
      setup.screen_width[I] + "x" + setup.screen_height[I];
    LbScreens.userdata[I] := setup.screen[I];
  endfor

  LbScreens.activeitem := LbScreens:find(.userdata, DpySec.real_screen);
}
```

2.5 font (Zeichensatz)

Alle Zeichensätze, die verwendet werden sollen, müssen als Ressource deklariert werden. Die darstellbaren Zeichensätze sind von dem verwendeten Fenstersystem abhängig. Die Deklaration eines Zeichensatzes beginnt mit dem Schlüsselwort **font**, anschließend steht ein Identifikator, der den Zeichensatz innerhalb des DM beschreibt. Es folgt die Definition des Zeichensatzes.

Definition

```
{ export | reexport } font <Bezeichner> <fontSpec> | <variantDef> ;

fontSpec ::=
  "<fontName>" { , <size> { , <modifier> } } |
  "<Vordefinierte UI-Font>" |
  "<X Logical Font Description (XLFD)>" |                          // Motif,
Qt
  "<fontName> [ , <parameter> ]" { , <modifier> } |                // Qt
  "<fontName>" { , <size> { , <modifier> { + <modifier> } } } |
  "<size>.<fontName>" | "<systemFont>"                             // Windows
  { x:= * <xscale> % + <xoffset> | / <xdivider>} { y:= * <yscale> %
        + <yoffset> | / <ydivider>}
  { propscale }
  { r:= "<RefString >" } ;

variantDef ::=
{
          0 : <fontSpec>
  [ <Nummer> : <fontSpec> ]
}
```

Innerhalb der Zeichensatz-Definition wird der durch das Fenstersystem bestimmte Identifikator sowie die Größe und die Attribute des Zeichensatzes angegeben.

Abhängig von Ihrem Grafiksystem ist es eventuell nicht möglich, die Größe und/oder die Zeichensatzattribute unabhängig vom ausgewählten Zeichensatz zu definieren. In diesem Fall ist nur die erste Variante der Zeichensatzdefinition möglich, eventuell folgende Größen- oder Zeichensatzattribute werden ignoriert.

Neben den vom jeweiligen System bereitgestellten Möglichkeiten die verfügbaren Zeichensätze einzusehen, kann die Liste der Zeichensätze auch im IDM Editor eingesehen werden. Dort besteht - je nach System - die Möglichkeit die Liste der Zeichensätze nach UI Fonts, XFT und XLFD zu filtern.

Beispiele

Nicht variant

```
font BIG          "white_shadow-48";
font KilterBold   "-adobe-courier-bold-r-normal*";
font KilterItalic "kilter 12i";
```

```
font UI "UI_FONT";

// Definiert den Zeichensatz vtsingle des Fenstersystems X
// auf den Namen NORMAL im IDM
font NORMAL "vtsingle";

// Definiert den Zeichensatz Courier in der Größe 12 als Ressource Courier
font Courier "Courier", 12;

// Wie die vorhergehende Definition; zusätzlich wird das Attribut bold
// für fette Darstellung definiert
font Bold "Courier", 12, bold;

// Definiert den Zeichensatz mit der Schriftart, die standardmäßig
// vom System für Oberflächenelemente verwendet wird; zusätzlich wird das
// Attribut propscale für die Steuerung des Font-Rasters gesetzt
font UIpropscale "UI_FONT" propscale;

// Definiert einen XFT Zeichensatz (bspw. auf einem Ubuntu System);
// zusätzlich wird die Größe 12 und das Attribut bold für fette Darstellung
// definiert
font XFT "DejaVu Sans", 12, bold;
```

Variant

```
font BIG
{
  0: "white_shadow-48";
  1: "walla_walla-12";
  2: "helvetica", 24, bold;
  3: "harakiri",  48, roman;
}
```

2.5.1 Berechnung der Rastergröße aus einem Referenzfont

Bei der Berechnung der Rastergröße aus einem Referenzfont kann der Benutzer bei der Definition des Zeichensatzes Korrekturfaktoren angeben und kann so die aus dem Zeichensatz ermittelte Berechnungsgrundlage für das Raster beliebig ändern.

Außerdem ist es möglich, einen Referenzstring anzugeben, aus dem die Rasterbreite berechnet wird. Die Rasterbreite errechnet sich dann aus der Breite des gesamten Strings, dividiert durch die Anzahl der Zeichen (mit mathematischer Rundung). Die Angabe eines Referenzstrings macht die Breitenberechnung unabhängig von der mittleren und maximalen Zeichenbreite der Fontbeschreibung, die von System zu System unterschiedlich sein kann. Der Referenzstring wird durch den Zusatz `r:="<RefString>"` bei der Fontdefinition angegeben.

Ausgehend von dieser Zeichensatz-Definition, werden die Rastergrößen wie folgt berechnet:

```
xraster = ((Breite des Fonts) * xscale) / 100 + xoffset
yraster = ((Höhe des edittextes mit diesem Font) * yscale) / 100
          + yoffset
```

Das heißt die Zeichensatzgröße wird mit dem angegebenen Faktor versehen und dann um die angegebene Konstante vergrößert.

Der `<xdivider>/<ydivider>` erlaubt die Teilung eines Raster ohne Abrundungseffekte. Der IDM sorgt dafür, dass ein Raster in der vollen Teilergroße mindesten genaus gross ist wie das Raster ohne Teiler. Hiermit wird eine vollstaendige Objektdarstellung gewährleistet.

```
xraster = ((Breite des Fonts) + (xdivider - 1 )) / xdivider
yraster = ((Höhe des edittextes mit diesem Font) + (ydivider - 1))) /
ydivider
```

Beispiel

```
font Font "6x13" x:=*150%+2 y:=*200%+10;
```

In diesem Fall berechnet sich die aus diesem Zeichensatz resultierende Rastergröße wie folgt:

```
xraster = (6 * 150) / 100 + 2
        = 11
yraster = (13 * 200) / 100 + 10
        = 36
```

Wichtig: veränderte Berechnung der Rasterbreite unter Windows

Mit der IDM-Version A.06.03.a wurde die Berechnung der Rasterbreite grundlegend umgestellt. Wenn kein Referenzstring angegeben ist, wird die Rasterbreite jetzt aus einem internen Referenzstring („M“) berechnet, um ein übermäßiges Breitenwachstum durch überbreite Buchstaben innerhalb einer Schriftart zu vermeiden.

Aus Kompatibilitätsgründen kann jedoch durch die Option *opt_fontraster_compat*, die Startoption **-IDMfontraster_compat** oder die Umgebungsvariable `IDM_FONTRASTER_COMPAT` vorübergehend die veraltete Berechnung der Rasterbreite wieder reaktivieren werden (verbunden mit dem Nachteil, dass es wieder zu übermäßiges Breitenwachstum kommen kann).
Bei Verwendung der Option *opt_fontraster_compat* basiert die Größenberechnung zum Teil auf dem Systemfont, der nicht High DPI fähig ist, daher sollten High DPI fähige Anwendungen, die mit IDM FÜR WINDOWS 11 erstellt wurden, dies Option nicht verwenden.

2.5.2 Vordefinierte UI-Fonts

Die UI-Fontressourcen sind WSI-unabhängige vordefinierte Schriftarten, die auf allen Systemen in ähnlicher Ausprägung bzw. Bedeutung verfügbar sind. Herangezogen werden hierzu die Default-

Schriftarten, die vom aktuellen Desktop/Window Manager oder Theme verwendet werden. Die einheitlichen FONT-Ressourcen sind wie folgt definiert:

- UI_FONT
  (Schriftart, die standardmäßig für Oberflächenelemente verwendet wird)
- UI_FIXED_FONT
  (Schriftart mit einer festen Buchstabenbreite)
- UI_MENU_FONT
  (Schriftart, die für Menüs verwendet wird)
- UI_TITLE_FONT
  (Schriftart, die in der Fensterüberschrift verwendet wird)
- UI_SMALLTITLE_FONT
  (Schriftart, die in einer kleinen/niedrigen Fensterüberschrift verwendet wird - sofern vom WSI unterstützt)
- UI_STATUSBAR_FONT
  (Schriftart, die in der Statusbar verwendet wird - sofern vom WSI unterstützt)

Eine Liste der auf dem jeweiligen System zur Verfügung stehenden Schriftarten kann über das Attribut .fontname[integer] am setup Objekt erfragt werden.

2.5.3 Zeichensatz-Definition

Es gibt folgende Definitionsmethoden für einen Zeichensatz:

- Durch Angabe eines Strings und zusätzlichen Modifikatoren mit folgendem Schema:

```
fontSpec ::=
        "<fontName>" { , <size> { , <modifier> { + <modifier> } } } ;
```

  Dadurch wird ein Zeichensatz mit dem angegebenen Namen, der Größe (in Punkten) und entsprechenden Modifikatoren - in Form von Schriftstärke oder Schriftbild - ausgewählt.
  Zulässige Zeichensatz-Modifikatoren sind:
  - **normal**
  - **light** (dünner als normal)
  - **medium** (Schriftstärke zwischen **normal** und **demibold**)
  - **demibold** (Schriftstärke zwischen **medium** und **bold**)
  - **bold** (fett)
  - **black** (extra fett)
  - **italic** (kursiv)
  - **normal** (Standard)
  - **oblique** (wird ggf. auf **italic** abgebildet)
  - **roman** (wird ignoriert)

Hinweis:

Es ist zu beachten, dass es vom gewählten Zeichensatz abhängt ob und in welcher Umfang die ausgewählten Modifikatoren angewendet werden.

2.5.4 Zeichensatz-Definition für Microsoft Windows

Hier gibt es folgende Definitionsmethoden für einen Zeichensatz:

- Nur durch Angabe eines Strings mit folgendem Schema:

  ```
  fontSpec ::= "<size>.<fontName>" ;
  ```

  Dadurch wird ein Zeichensatz mit angegebenem Namen und Größe (in Punkten) ausgewählt.

- Verwendung eines Standard-Systemzeichensatzes durch Angabe eines Strings mit folgendem Schema:

  ```
  fontSpec ::= "<systemFont>" ;
  ```

  In diesem Fall sind die folgenden Systemzeichensätze möglich:

  - SYSTEM_FONT
  - ANSI_FIXED_FONT
  - ANSI_VAR_FONT
  - DEVICE_DEFAULT_FONT

  Anmerkung
  Eine Liste existierender und verwendbarer Systemzeichensätze zusammen mit ihrer Größe können in *„Control Panel“* → *„Fonts“* erhalten werden.

Anmerkung

Wenn der spezifizierte Zeichensatz nicht verfügbar ist, wird der Systemzeichensatz ausgewählt.

Warnung

Bei der Zeichensatzauswahl wird keine Beschränkung der Codepage vorgenommen, daher können alle im System verfügbaren Zeichensätze verwendet werden.

Dies kann jedoch dazu führen, dass Zeichensätze mit einer falschen Codepage ausgewählt werden, so dass falsche Zeichen dargestellt werden. Der IDM hat unter MICROSOFT WINDOWS keine Möglichkeit, eine Codepage-Konvertierung durchzuführen, da nur die WINANSI-Codepage definiert ist.

Siehe auch

Kapitel „Hinweis zur Verwendung von Symbol-Schriftarten unter Microsoft Windows“

2.5.5 Zeichensatz-Definition für Qt

Schriftarten können auf folgende Arten definiert werden:

Qt-Stil mit kommaseparierter Parameterliste

```
Family, PointSize, PixelSize, QFont::StyleHint, QFont::Weight, QFont::Style,
Flag underline, Flag strikeout, Flag pixedPitch, Flag rawMode
```

```
"Helvetica, 12, -1, 5, 75, 1, 0, 0, 0, 0"
```

Es müssen nicht alle Parameter angegeben werden, `"Helvetica, 12"` reicht zum Beispiel.

Schriftart mit unabhängigen Darstellungsattributen (Größe und Stil)

```
"Helvetica", 12, bold
```

Die Stile *bold*, *italic* und *oblique* werden unterstützt.

Angaben im Qt-Stil können mit unabhängigen Darstellungsattributen kombiniert werden:

```
"Helvetica, 12", bold
```

X Logical Font Description (XLFD)

```
"-adobe-helvetica-medium-r-normal--12-*-*-*-p-*-iso8859-1"
"*adobe-helvetica-bold-r-normal-*-100-*-iso8859-1"
```

Hinweise

- QT hat Probleme bei der Verarbeitung von Wildcards in XLFDs.
- Wenn QT eine Schriftart nicht genau zuordnen kann, wird die am ehesten passende Fontfamily genommen. Es kommt zwangsläufig zur Substitution von Werten.
- Font-Aliase (Datei **fonts.alias** im Font-Verzeichnis von X11) werden von Qt nicht unterstützt. Deshalb kann zum Beispiel Font *fixed* nicht verwendet werden.

Achtung

- Seit QT5 wurden unterstützende Funktionalitäten zum Verarbeiten von XLFD Fonts einstellt.
- Die Verwendung von XLFD Fonts wird nicht empfohlen.

2.5.6 Dynamisch änderbare Attribute

Bei Zeichensätzen können folgende Attribute zur Laufzeit dynamisch gesetzt und abgefragt werden:

Tabelle 3: *Änderbare Attribute der font-Ressource*

Attribut	Datentyp	Indexbereich	Beschreibung
.face	*enum face_ default*, *face_italic*, *face_obli- que*, *face_ roman*,	–	Schriftbild
.height[enum]	*integer*	*scale_factor*, *scale_offset*	Y-Skalierung und Offset
.name	*string*	–	Fontname
.refstring	*string*	–	Referenzstring für die Berechnung der Rasterbreite
.propscale	*boolean*	–	Rastersteuerung
.size	*integer*	–	Schriftgröße
.style	*enum face_ default*, *face_light*, *face_nor- mal*, *face_ medium*, *face_demi- bold*, *face_bold*, *face_black face_italic*, *face_obli- que face_ roman*,	–	Schriftart

Attribut	Datentyp	Indexbereich	Beschreibung
.weight	*enum face_default*, *face_light*, *face_normal*, *face_medium*, *face_demibold*, *face_bold*, *face_black*	–	Schriftstärke
.width[enum]	*integer*	*scale_factor*, *scale_offset*	X-Skalierung und Offset

2.6 format

Um Strings in editierbaren Texten und Tablefields formatieren zu können, können Format-Ressourcen definiert werden.

Eine Format-Ressource verknüpft einen String als Formatbeschreibung mit einer optionalen Format-Funktion, die den String interpretiert.

Definition

```
{ export | reexport } format <Bezeichner> <formatSpec> | <variantDef>

formatSpec ::=
  <Format-String> [ <Format-Funktion> ] ;

variantDef ::=
{
           0 : <formatSpec>
  [ <Nummer> : <formatSpec> ]
}
```

Wird keine Format-Funktion angegeben, wird der Format-String vom Dialog Manager intern behandelt. Wird eine Format-Funktion angegeben, muss diese als `formatfunc` definiert sein (siehe auch Kapitel „Format-Funktion“ im Handbuch „Regelsprache“).

Mit Hilfe von Varianten können z.B. Anpassungen an unterschiedliche Sprachen vorgenommen werden.

Bei der internen Behandlung des Formatstrings durch den Dialog Manager, sind folgende Formatbeschreibungen erlaubt:

Leerstring

Ist das Format ein Leerstring, werden die Eingabestrings nicht formatiert.

Eingabemuster

Die folgenden Zeichen stehen innerhalb des Formatstrings zur Verfügung:

A	Alphabetische Zeichen
C	Ziffern und Großbuchstaben
H	Hexadezimalziffern
N	Ziffern
U	Alphabetische Zeichen, nur Großbuchstaben
X	beliebige Eingabe

9	Ziffern (wie N)
/	Formatierungszeichen
,	Formatierungszeichen
.	Formatierungszeichen
-	Formatierungszeichen
:	Formatierungszeichen
Leerzeichen (Space)	Formatierungszeichen

Die Formatierungszeichen werden zur Darstellung und zur Eingabe in den Darstellungsstring eingefügt. Sie werden bei der Eingabe übersprungen und können daher nicht überschrieben werden. Sie werden nach der Eingabe nicht in den Inhaltsstring übernommen und sind dort nicht vorhanden.

Beispiel

Uhrzeiteingabe:

```
.format      "NN:NN:NN";
```

Eingabe von *„120300“*

```
Ergebnis    "12:03:00"
```

Verdeckte Formatierung

Wenn der Formatstring mit einem S beginnt, wird der Inhaltsstring verdeckt dargestellt und editiert.

Das dem "S" folgende Zeichen wird als Spezialzeichen zur Verdeckung interpretiert. Folgt dem "S" kein weiteres Zeichen, wird als Verdeckungszeichen ein * gewählt. Danach sind Formatstrings der Form

- Leerstring
- Eingabemuster
- numerisches Format

möglich.

Mit der verdeckten Formatierung kann z.B. eine Passwort-Eingabe realisiert werden.

Beispiel

```
.format     "SxAAA";
```

erlaubt die Eingabe von drei Buchstaben, die aber alle als "x" dargestellt werden.

Numerisches Format

Für die Formatierung von Zahlen, muss der Formatstring nach folgenden Schema aufgebaut werden:

```
% [<|>] [+|-] [ [0] 0] length [' [sepch] sepcount ] [ ( . | , ) [0] trail ]
[u|s] (b|d|f|h|o|x|X) [<|>] [+|-] [/(c[0]|p|t|z)+]
```

Bedeutung der verschiedenen Elemente

Formatzeichen	Zeichen	Bedeutung
%		Einleitungszeichen zur Erkennung des numerischen Formats.
[<|>]		Ausrichtung am linken Rand:
	keine Angabe	Auffüllen mit Leerzeichen vor dem Vorzeichen.
	>	Auffüllen mit Leerzeichen zwischen Vorzeichen und Ziffern.
	<	Kein Auffüllen, linksbündig.
[+|-]		Vorzeichen links:
	keine Angabe	Darstellung nur negativer Vorzeichen, falls kein Vorzeichen rechts definiert ist.
	-	Darstellung nur negativer Vorzeichen.
	+	Darstellung positiver und negativer Vorzeichen.
[[0]0]		Darstellung führender Nullen:
	keine Angabe	Es werden keine führenden Nullen dargestellt.
	0	Einzelne führende Null wird dargestellt, falls Vorkommaanteil 0 ist.
	00	Alle nicht belegten Vorkommastellen werden mit 0 dargestellt.
length		Anzahl aller Ziffernstellen (ohne Sonderzeichen).
[' [sepch] sepcount]		An jeder sepcount-Stelle wird im Vorkomma-Anteil ein Zeichen (sepch oder ') dargestellt, (z.B. als Tausendermarkierung). Wird als sepcount 0 angegeben, wird der Standardwert 3 verwendet.

Formatzeichen	Zeichen	Bedeutung
[(.|,) [0] trail]		Beschreibung des Nachkommaanteils: Es wird entweder ein Punkt oder ein Komma als Dezimaltrennzeichen dargestellt.
		trail gibt die Anzahl der Nachkommastellen an. Bei Angabe einer vorangestellten 0 werden die nicht belegten Stellen mit 0 aufgefüllt. Dabei gilt, dass die Anzahl der Nachkommastellen kleiner sein muss als die Anzahl der Ziffernstellen (trail < length).
[u|s]		Eingabe negativer Zahlen:
	u	unsigned, d.h. negative Zahlen sind nicht erlaubt.
	s	signed, d.h. es sind auch negative Zahlen erlaubt.
(b|d|f|h|o|x|X)		Formatzeichen zur Festlegung des Zahlensystems:
	b	Binärziffern.
	d	Dezimalziffern.
	f	Fließkommawerte mit Dezimaltrennzeichen im Eingabestring.
	h	Entspricht x.
	o	Oktalziffern.
	x	Hexadezimalziffern (mit Kleinbuchstaben).
	X	Hexadezimalziffern (mit Großbuchstaben).
		Außer beim Formatzeichen f darf im Inhaltsstring kein Dezimaltrennzeichen enthalten sein. Dieses wird nur zur Darstellung und Eingabe in den Darstellungsstring eingefügt.
[<|>]		Ausrichtung am rechten Rand:
	keine Angabe	Auffüllen mit Leerzeichen hinter dem Vorzeichen.
	>	Kein Auffüllen, rechtsbündig.
	<	Auffüllen mit Leerzeichen zwischen Vorzeichen und Ziffern.
[+|-]		Vorzeichen rechts:

Formatzeichen	Zeichen	Bedeutung
	keine Angabe	Keine Vorzeichendarstellung.
	-	Darstellung nur negativer Vorzeichen.
	+	Darstellung positiver und negativer Vorzeichen.
[/(c[0]|p|t|z)+]		Format-Modifikatoren:
		Durch die Angabe von einem oder mehreren Modifikatoren kann das Verhalten des Formats beeinflusst werden. Standardeinstellung: Kaufmännische Zahleneingabe: die Ziffern werden bei der Eingabe über das Dezimaltrennzeichen hinweg geschoben. Leerstrings sind bei der Eingabe erlaubt und von Null verschieden. Das Dezimaltrennzeichen wird immer angezeigt - auch wenn der Inhalt ein Leerstring ist.
	c	Ziffernlöschungs-Modus: Löschen der letzten verbliebenen Ziffer aus einem String führt zu einem leeren String und nicht zu einem String mit dem Wert "0". Der Modifikator hat keine Auswirkung, wenn er mit dem *z*-Format-Modifikator (Zero-Modus) kombiniert wird.
	c0	0-Löschungs-Modus: Wenn ein String nach einer Eingabe oder Löschung den Zahlenwert 0 repräsentiert, wird er durch einen leeren String ersetzt. Im 0-Löschungs-Modus wird ein Inhaltsstring zum Leerstring, wenn nur eine Ziffer ungleich 0 vorhanden ist und diese Ziffer gelöscht wird (Beispiel: 7000,00 wird nach Löschen der 7 durch DEL oder BACKSPACE zum Leerstring). Der Modifikator hat keine Auswirkung, wenn er mit dem *z*-Format-Modifikator (Zero-Modus) kombiniert wird.
	p	Punkt-Modus: wenn der Inhalt ein Leerstring ist, wird kein Dezimaltrennzeichen angezeigt.
	t	Technische Zahleneingabe: Vor- und Nachkommaanteil werden getrennt editiert.

Formatzeichen	Zeichen	Bedeutung
	z	Zero-Modus: Leerstrings werden in Nullen umgewandelt, es können keine Leerstrings eingegeben werden, abhängig vom Format können Nullen eventuell als Leerstring angezeigt werden.

Anmerkungen

- Führende und anhängende Nullen werden nur in den Darstellungsstring eingefügt. Sie werden auch nach der Eingabe nicht in den Inhaltstring übernommen und sind somit dort nicht vorhanden.
- Bei einem einziffrigen Inhaltsstring steht die enthaltene Ziffer im formatierten String nicht notwendigerweise ganz rechts sondern kann auch die erste Ziffer links vom Komma sein, z.B. bei der technischen Zahleneingabe.
- In einem Eingabefeld können Zeichen, die durch ein Format in den Darstellungsstring eingefügt werden, nicht mit der Rückschritt- bzw. Entfernen-Taste (BACKSPACE bzw. DEL) gelöscht werden.
  Ein Vorzeichen, das durch ein Format eingefügt wird, kann durch Eingabe von + oder - geändert werden. Dabei schaltet + auf ein positives Vorzeichen um bzw. behält dieses bei. Dagegen schaltet - zwischen positivem und negativem Vorzeichen hin und her (wie eine Multiplikation mit -1).

Regulärer Ausdruck

Verfügbarkeit

Ab IDM-Version A.06.02.g

Erlaubt ist ein Matching-Ausdruck in der Form `"/ … /"` um so die Prüfung des Inhalts bzw. der Eingabe zu erlauben.

Passt der Inhalt nicht zum Muster, so wird ein Leerstring dargestellt bzw. die Eingabe unterbunden.

Beispiele

- ```
  format FmtKommaZahl "/^(\\d+(\\.\\d*)?)?$/";
  ```
  Erlaubte Strings z.B. *""*, *"1"*, *"07654.321"*
- ```
  format FmtHexZahl "/^[abcdefABCDEF0123456789]*$/";
  ```
  Erlaubte Strings z.B. *""*, *"AF4513c"*, *"a"*, *"7EF"*
- ```
  format FmtSpecial "/^([[:alpha:]]*\n\\d*\n[01]*)?$/";
  ```
  Erlaubte Strings z.B. *""*, *"Stadt\n999\n01"*, *"Frankfurt\n60311\n01101110"*

Siehe auch

Eingebaute Funktion regex und Kapitel „PCRE-Bibliothek zur Unterstützung Regulärer Ausdrücke“.
  ```

2.7 text

Zur Unterstützung mehrsprachiger Dialoge hält der IDM die Ressource ***text*** bereit. Hierbei wird der Dialog zunächst in einer Sprache entwickelt.

Bei der Übersetzung in eine neue Sprache wird dann die Ressource ***text*** eingeführt. Dabei wird ein in der Originalsprache vorhandener String zusammen mit seinen Übersetzungen definiert.

Definition

```
{ export | reexport } text { <Bezeichner> }  "<Original-String>"
  <variantDef> | ;

variantDef ::=
{
          1 : "<übersetzter String>" | nlscat(<Message-Nummer>) ;
  [ <Nummer> : "<übersetzter String>" ; ]
}
```

Der Identifikator ist gewöhnlich optional, d.h. er muss nicht unbedingt angegeben werden. Mit Hilfe des Identifikators kann später auf den Text referenziert werden.

Als erste Übersetzung kann entweder ein <übersetzter String> oder ein Eintrag aus einem Message- oder Text-Katalog nlscat(<Message-Nummer>) angegeben werden, weitere Übersetzungen werden als <übersetzter String> definiert.

Der Zugriff vom Dialog Manager auf den Text ist wie folgt:

Die mittels der Funktion DM_InstallNlsHandler() installierte Funktion wird aufgerufen. Danach wird der Text intern gespeichert. Der Zugriff erfolgt nur für aktuell verwendete Texte!

Damit im Dialog möglichst viele unterschiedlichen Zeichen dargestellt werden können, muss die Definition der Texte im ISO 8859/1 Standard erfolgen, der für fast alle europäischen Sprachen geeignet ist. Der Dialog Manager wandelt diese Darstellung dann in die des jeweiligen Fenstersystems um, sodass die Zeichen korrekt dargestellt werden.

Die Eingabe der Zeichen erfolgt dabei nach folgendem Schema:

Alle 7-Bit-Zeichen werden direkt mit dem jeweiligen Editor eingegeben, d.h., alle ASCII-Zeichen können direkt eingegeben werden. Diese Art der Zeicheneingabe garantiert, dass alle Editoren diese Zeichen korrekt darstellen können, da viele Editoren nur 7-Bit-Zeichen darstellen können.

Alle 8-Bit-Zeichen müssen in oktaler Schreibweise eingegeben werden, d.h. Zeichen mit einem Code größer als 127 müssen als Oktalzahl mit einem führenden Backslash "\" eingegeben werden.

	xx0	xx1	xx2	xx3	xx4	xx5	xx6	xx7
00								
01x								
02x								
03x								
04x								
05x								
06x								
07x								
10x								
11x								
12x								
13x								
14x								
15x								
16x								
17x								
20x								
21x								
22x								
23x								
24x		¡	¢	£	¤	¥	¦	§
25x	¨	©	ª	«	¬	-	®	¯
26x	°	±	²	³	´	µ	¶	·
27x	¸	¹	º	»	¼	½	¾	¿
30x	À	Á	Â	Ã	Ä	Å	Æ	Ç
31x	È	É	Ê	Ë	Ì	Í	Î	Ï
32x	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×
33x	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
34x	à	á	â	ã	ä	å	æ	ç
35x	è	é	ê	ë	ì	í	î	ï
36x	ð	ñ	ò	ó	ô	õ	ö	÷
37x	ø	ù	ú	û	ü	ý	þ	ÿ

Abbildung 1: *Tabelle für octale Werte*

Um ein bestimmtes Zeichen (z.B. "A") zu beschreiben, müssen Sie zuerst den entsprechenden Wert auf der Y-Koordinate feststellen:

-> 10x

Das x muss nun durch den entsprechenden Wert auf der X-Koordinate ersetzt werden:

-> xx1 (xx steht für die oben gefundenen Zahlen 10)

Daraus ergibt sich:

=> \101

Weitere Beispiele

```
ö     36x     xx6     =>     \366

#     04x     xx3     =>     \043
```

Beispiel

```
text "Hello world"
{
  1:  "Hallo Welt";
  2:  "Allo le monde";
  3:  "Buon giorno il mondo";
  4:  "Holá mundo";
}
```

Die Anwahl einer der Sprachen 1–4 mit der Option **-IDMlanguage** wird der String *"Hello world"* in dem gesamten Dialog durch die entsprechende Übersetzung ersetzt. Die Verwendung dieses so definierten Textes kann dabei entweder durch die Angabe des Originaltextes oder durch die Angabe des Identifikators erfolgen.

Beispiel

```
text WindowText "Window"
{
  1:  "Fenster";
  }
Window.title := WindowText;
```

ist gleichbedeutend mit

```
text "Window"
{
  1: :"Fenster";
}
Window.title := "Window";
```

Anmerkung

Mit dem Attribut *.text* kann man den String der aktuellen Sprachvariante erhalten.

2.7.1 Hinweis zur Verwendung von Symbol-Schriftarten unter Microsoft Windows

Bei der Verwendung von Schriftarten, die Symbole enthalten (z.B. „Wingdings“) ist folgendes zu beachten:

1. Der Name muss dem tatsächlichen Namen der Schriftart entsprechen (kein Alias). Führt Windows eine Schriftart-Ersetzung durch, erkennt dies der IDM und erzwingt eine Ersetzung zu einer

Schriftart, die den WIN ANSI Character Set unterstützt. Dies ist nötig, damit nicht ungewollt eine unlesbare Schriftart gewählt wird.

2. Einer Schriftart, die Symbole enthält, ist der Windows SYMBOL Character Set zugeordnet. Dieser wird auf den Unicode-Bereich \uF000–\uF0FF abgebildet (benutzerdefinierter Bereich in Unicode). Um beispielsweise das Zeichen 0xFE darzustellen muss also \uF0FE angegeben werden.

   Anmerkung
   Auf einem westeuropäischen Windows-System ist dies im Allgemeinen nicht nötig, da die System-Codepage CP1252 fast identisch zum unteren Unicode-Bereich ist und fast alle Zeichen enthält. Deshalb kommt es zu keiner Konvertierung und es wird – quasi zufällig – das gewünschte Zeichen dargestellt. Auf anderssprachigen Windows-Systemen ist das aber nicht der Fall.

2.8 tile (Muster)

Grafiken, Bilder und Muster, die innerhalb des IDM verwendet werden, müssen als Ressource deklariert werden. Das Schlüsselwort hierzu ist **tile**, gefolgt von einem Bezeichner und der Musterdefinition. Die Musterdefinition ist dabei entweder direkt Bestandteil der IDM-Datei oder in einer separaten Grafikdatei enthalten.

Definition

```
{ export | reexport } tile <Bezeichner> <tileSpec> | <variantDef>

tileSpec ::= <tileBitmap> | "<Dateipfad>" | "<Grafikressource>" { scale |
               noscale | numscale | propscale | } ;

tileBitmap ::= <x>, <y>,
               "<Musterzeile>"
           [ , "<Musterzeile>" ]

variantDef ::=
{
           0 :  <tileSpec>
  [ <Nummer> :  <tileSpec> ]
}
```

Bei der Definition von Varianten wird empfohlen, immer eine Variante mit der Nummer *0* zu definieren, da diese als Standardvariante dient, wenn keine, eine nicht vorhandene oder eine ungültige Variante gesetzt wird. Die weiteren Nummern können beliebige natürliche Zahlen sein.

Die zu verwendende ***tile***-Variante kann beim Start der Anwendung mit der Startoption **-IDMtile** gesetzt werden. Zur Laufzeit der Anwendung kann sie mit dem Attribut .tile des setup-Objekts abgefragt und gesetzt werden.

2.8.1 Internes Muster (Bitmap)

<*x*> und <*y*> stehen jeweils für die Anzahl der Pixel in horizontaler bzw. vertikaler Richtung. Breite (<*x*>) und Höhe (<*y*>) können beliebig gewählt werden.

<*Musterzeile*> bezeichnet eine Zeichenkette mit <*x*> Zeichen, in der jedes Zeichen ein Pixel des Musters repräsentiert. Es sind nur rechteckige Muster möglich, deshalb müssen genau <*y*> Musterzeilen mit jeweils <*x*> Zeichen definiert werden.

Die Musterzeilen bestehen aus den Zeichen Punkt („.“) Leerzeichen („ “) und Doppelkreuz („#“), die folgende Bedeutung haben:

„.“
„ “ (Leerzeichen)

Übernimmt die Hintergrundfarbe, d.h. an dieser Stelle ist das Muster transparent.

Wenn die ***tile***-Ressource im *.picture*- bzw. *.picture[enum]*-Attribut eines ***image***-Objekts verwendet wird und dessen Attribut .imagebgc gesetzt ist, erscheint an dieser Stelle ein Pixel in der Farbe *.imagebgc*.

„#“

An dieser Stelle erscheint ein Pixel in der Vordergrundfarbe.

Bei Verwendung der ***tile***-Ressource im *.picture*- bzw. *.picture[enum]*-Attribut eines ***image***-Objekts ist das die Farbe, die im Attribut .imagefgc des ***image***-Objekts gesetzt ist.

Beispiel für die Definition eines internen Musters

```
tile TiGray 16, 16,
  "# # # # # # # # ",
  " # # # # # # # #",
  "# # # # # # # # ",
  " # # # # # # # #",
  "# # # # # # # # ",
  " # # # # # # # #",
  "# # # # # # # # ",
  " # # # # # # # #",
  "# # # # # # # # ",
  " # # # # # # # #",
  "# # # # # # # # ",
  "# # # # # # # # ",
  " # # # # # # # #",
  "# # # # # # # # ";
```

2.8.2 Externes Muster

Bei der Definition über einen Dateinamen (inklusive Pfad) geht der IDM davon aus, dass die Definition des Musters in einer separaten Grafikdatei erfolgt. Der IDM lädt die entsprechende Datei und stellt ihren Inhalt mit den Systemfunktionen des jeweiligen Toolkits dar. Deshalb ist es plattformabhängig, welche Grafikformate angezeigt werden können.

Die folgende Tabelle zeigt, welche Grafikformate auf den einzelnen Plattformen unterstützt werden:

Tabelle 4: *Von der tile-Ressource unterstützte Grafikformate*

Grafikformat	Dateiendung	Microsoft Windows	Motif	Qt
Graphics Interchange Format	.gif	ja	ja	ja
Device Independent Bitmap	.bmp	ja	nein	ja
X PixMap	.xpm	nein	ab Motif 2.1	nein
JPEG File Interchange Format	.jpg	ja	ab Motif 2.3	ja

Grafikformat	Dateiendung	Microsoft Windows	Motif	Qt
Portable Network Graphics	.png	ja	ab Motif 2.3	ja
Enhanced Metafile Windows Metafile	.emf .wmf	ja	nein	nein
Aldus Placeable Metafile	.apm	ja	nein	nein
Windows Icon Resource File	.ico	ja	nein	nein
Windows Icon Resource Windows Bitmap Resource	–	ja	nein	nein
Scalable Vector Graphics	.svg	ja	nein	ja

Die Definition über eine Grafikressource wird nur vom IDM für Windows unterstützt. Hier können die Bezeichner „IDM_AppIcon“ und „IDM_DefIcon“ der in den IDM-Bibliotheken definierten Ressourcen verwendet werden (siehe auch Attribut .icon in der „Attributreferenz“).

Unter Microsoft Windows können außerdem folgende Bezeichner für vordefinierte Bitmap-Ressourcen des Systems angegeben werden:

BTNCORNERS	BTSIZE	CHECK	CHECKBOXES	CLOSE
COMBO	DNARROW	DNARROWD	DNARROWI	LFARROW
LFARROWD	LFARROWI	MNARROW	OLD_CLOSE	OLD_DNARROW
OLD_LFARROW	OLD_REDUCE	OLD_RESTORE	OLD_RGARROW	OLD_UPARROW
OLD_ZOOM	REDUCE	REDUCED	RESTORE	RESTORED
RGARROW	RGARROWD	RGARROWI	SIZE	UPARROW
UPARROWD	UPARROWI	ZOOM	ZOOMD	

Hinweis

Die Bilder der ***tile*** Ressource werden nun automatisch entsprechend des eingestellten Vergrößerungsfaktor vergrößert. Es wird hierbei davon ausgegangen, dass die Bilder für einen DPI-Wert von *96* entworfen wurden. Sind die Bilder der Anwendung für eine höhere Auflösung entworfen worden, dann kann dies am setup Objekt mit dem Attribut .tiledpi eingestellt werden.

Bei der Spezifikation einer externen Datei wird empfohlen, den Pfad nicht absolut, sondern mit Hilfe einer Umgebungsvariablen zu definieren, z.B. "IDM_IMAGEPATH:Check.gif".

Weitere Grafikformate können mithilfe eines selbst implementierten Grafik-Handlers (GFX-Handler) verarbeitet werden (siehe Funktion DM_PictureHandler im Handbuch „C-Schnittstelle - Funktionen“).

2.8.3 Skalierung

Durch die Angabe eines **scalestyles** bei der Musterdefinition wird die Größe des Musters oder Bildes nach bestimmten Kriterien an den verfügbaren Platz angepasst.

Best Practice: Muster sollten wenn möglich bereits in der richtigen bzw. verwendeten Größe vorliegen, da jede Skalierung mit Verlusten an Schärfe oder Detailreichtum verbunden ist.

„scale | noscale | numscale | propscale“

Bestimmt die Art der Skalierung des Musters oder Bildes.

Definition am Tile	Dynamische Setzung	Bedeutung
noscale	scalestyle_ none	Das Muster bzw. Bild wird nicht skaliert., setup.tiledpi hat keine Auswirkungen.
scale	scalestyle_any	Höhe und Breite des Musters bzw. Bildes werden voll auf die verfügbare Fläche vergrößert.
propscale	scalestyle_ prop	Höhe und Breite des Musters bzw. Bildes werden auf die verfügbare Fläche vergrößert, wobei Höhen- und Breitenproportionen des Musters bzw. Bildes auf jeden Fall erhalten bleiben. D.h. es können oben und unten bzw. links und rechts Freiflächen entstehen.
numscale	scalestyle_ num	Die Skalierung des Musters bzw. Bildes wird durch einen numerischen Skalierungsteiler vorgenommen. Dabei wird die Skalierung in Viertel-Schritten vorgenommen, also 1.25-fach, 1.5-fach, 1.75-fach, 2-fach, 2.25-fach, 2.5-fach, usw. Eine Verkleinerung finded bis maximal 0.25-fach statt.
dpi	scalestyle_dpi	Das Muster bzw. Bild wird immer entsprechend der eingestellten Bildschirmskalierung skaliert.
Keine Angabe	scalestyle_ auto	Das Muster bzw. Bild wird entsprechend der eingestellten Bildschirmskalierung skaliert. Eine zur Vorgängerversion kompatible Skalierung findet statt. **Defaultwert**

Siehe auch

Attribut .scalestyle im Handbuch„Attributreferenz“

2.8.4 Dynamisch änderbare Attribute

Die änderbaren Attribute können der nachfolgenden Tabelle entnommen werden. Dabei ist zu beachten, dass sich *.name* einerseits sowie *.width*, *.height* und *.pattern* andererseits gegenseitig ausschließen, sodass „get“ zu einer Fehlermeldung *„can't get value“* führt.

Bei varianten ***tile***-Ressourcen wird auf die Attribute der aktuellen ***tile***-Variante zugegriffen, die mit dem Attribut .tile des setup-Objekts abgefragt und gesetzt werden kann.

Tabelle 5: *Änderbare Attribute der tile-Ressource*

Attribut	Datentyp	Indexbereich	Beschreibung
.name	*string*	–	Dateipfad einer Grafikdatei.
.width	*integer*	–	Breite des Musters in Pixeln.
.height	*integer*	–	Höhe des Musters in Pixeln.
.pattern	*string*	–	Vollständiges Muster als String.
.pattern[integer]	*string*	*1 … .height*	Einzelne Zeilen des Musters. Die Länge jedes Strings muss gleich *.width* sein.
.scale	*boolean*	–	Größenanpassung an den verfügbaren Platz. Standardwert ***false*** (keine Größenanpassung).
.scalestyle	*enum*	–	Steuert die Darstellung und Skalierung eines Tiles. Standardwert ***scalestyle_auto***

2.8.5 SVG-Support

Unter WINDOWS und QT unterstützt die Tile-Ressource nun auch Vektor-Bilder im Scalable Vector Graphics-Format (.svg).

2.8.5.1 QT

Zwar konnte der IDM für QT bereits SVG-Bilder anzeigen, allerdings nur über ein pixelbasiertes Stellvertreterbild das entsprechend skaliert wurde. Die Dateiendung .svg wird jetzt als SVG-Bildformat erkannt und bei der Darstellung in den diversen Objekten ein vektorbasiertes Zeichnen angewandt. Dadurch wird bei Anwendungen, die auf HiDPI-Bildschirmen dargestellt werden, eine möglichst optimale Abbildungen von Vektorbildern erzielt.
Aus Gründen der Performanz wird beim Zeichnen von Hintergrund-Tiles von Gruppierungsobjekten im Stil *tilestyle_tiled*, das pixelbasierte Stellvertreterbild verwendet.

> Installationshinweis: Das QT-Paket **qt5-qtsvg** muss bereits installiert sein um die entsprechende SVG-Unterstützungs-Bibliothek verfügbar zu machen und die Startfähigkeit einer IDM-Anwendung zu gewährleisten.

2.8.5.2 Windows

Zur Darstellung von Bildern im "Scalable Vector Graphics" (SVG) Format wird Microsoft Direct2D verwendet. Die SVG-Unterstützung von Microsoft Direct2D ist abhängig von der Microsoft Windows Version. Ab dem Windows 10 Creators Update ist die Unterstützung in eingeschränkter Form gegeben.

Hinweisezur Verwendung des SVG Formats als ICON

Das SVG-Format lässt sich nicht einwandfrei zu einem Icon wandeln, deshalb eignet es sich nicht für die Darstellung als *.picture* des treeview- oder notepage-Objekts. Ebenso wenig eignet es sich als *.icon* des window-Objekts. Es können unschöne Ränder am Übergang zwischen durchsichtigem und gezeichnetem Bereich auftreten.

Mögliche Änderungen zur Vorgängerversion:
Da dieses Grafikpaket andere Skalierungsmethoden als das Windows GDI verwendet, können Muster (Bilder), die zur Anzeige skaliert werden müssen, anders aussehen als in den Vorgängerversionen des IDM. Das gilt insbesondere für Muster die als Fenstericon oder als Bilder in den Objekten **treeview** und **notepage** verwendet werden, da hier Microsoft Windows den Icondatentyp voraussetzt.
Best practice: Die Muster (Bilder) liegen bereits in der benötigten Größe vor.

Die SVG-Unterstützung von Microsoft Direct2D

Die Unterstützung hängt von der Microsoft Windows-Version ab. Ab dem Windows 10 Creators Update werden die folgenden SVG-Elemente und -Attribute unterstützt:

Element	Unterstützte Attribute
circle	id, style, transform, cx, cy, r
clipPath	id, style, transform, clipPathUnits
defs	id, style, transform
desc	id
ellipse	id, style, transform, cx, cy, rx, ry
g	id, style, transform
image	id, style, transform, x, y, width, height, preserveAspectRatio, xlink:href
line	id, style, transform, x1, y1, x2, y2
linearGradient	id, style, x1, y1, x2, y2, gradientUnits, gradientTransform, spreadMethod, xlink:href
path	id, style, transform, d
polygon	id, style, transform, points

polyline	id, style, transform, points
radialGradient	id, style, cx, cy, r, fx, fy, gradientUnits, gradientTransform, spreadMethod, xlink:href
rect	id, style, transform, x, y, width, height, rx, ry
stop	id, style, offset
svg	id, style, x, y, width, height, viewBox, preserveAspectRatio
title	id
use	id, style, transform, x, y, width, height, xlink:href

SVG-Darstellungsattribute

- clip-path
- clip-rule
- color
- display
- fill
- fill-opacity
- fill-rule
- opacity
- overflow
- stop-color
- stop-opacity
- stroke
- stroke-dasharray
- stroke-dashoffset
- stroke-linecap
- stroke-linejoin
- stroke-miterlimit
- stroke-opacity
- stroke-width
- visibility

Unterstützte Längeneinheiten

Die Längenwerte und prozentualen Längenwerte des Benutzerbereichs sowie die absoluten Einheitenbezeichner: px, pt, pc, cm, mm und in.

Bildquellen

Das Imageelement wird nur unterstützt, wenn sein xlink:href-Attribut auf ein base64-codiertes Image festgelegt ist.

Siehe hierzu auch: **https://learn.microsoft.com/en-us/windows/win32/direct2d/svg-support**

3 Programmierressourcen

3.1 message (Nachricht)

Mit Hilfe dieser Ressource können Objekte definiert werden, welche zur Definition von externen Ereignissen verwendet werden können. Diese Ereignisse können beispielsweise mittels der Funktion sendevent() aus der Regelsprache genutzt werden.

Zusätzlich wird diese Ressource noch für die OLE Schnittstelle des ISA Dialog Managers verwendet; näheres hierzu siehe Handbuch „OLE-Schnittstelle“, Kapitel „Die Ressource message“ (Definition und Verwendung in der OLE-Schnittstelle).

Definition

```
{ export | reexport } message  <Bezeichner> { (<messageSpec>) };
```

Hinweis

Der Parameter *messageSpec* wird in der Regelsprache (bei Nutzung zur Definition eines externen Ereignisses) nicht ausgewertet und ignoriert.

Näheres zum Parameter *messageSpec* siehe Handbuch „OLE-Schnittstelle“, Kapitel „Die Ressource message“.

Beispiel

```
message EvInformation;

on dialog start
{
  ...
  sendevent(WnMain, EvInformation, “Dialogstart”);
}

window WnMain
{
  ...
  on extevent EvInformation (string Info)
  { ... }
}
```

3.2 source

Mit Hilfe dieser Ressource wird definiert, wie sich ein Objekt als Quelle einer Drag&Drop-Operation verhalten soll. Im Gegensatz zu den anderen Programmierressourcen können hier Varianten definiert werden.

Die Zuweisung zu einem Objekt erfolgt dann über das Attribut *.source*. Wenn dieses Attribut bei einem Objekt gesetzt ist, kann es als Quelle einer Drag&Drop-Operation vom Benutzer ausgewählt werden.

Hinweis

Drag&Drop wird vom IDM FÜR MOTIF nicht unterstützt.

Definition

```
{ export | reexport } source <Bezeichner>
{
             0 : .action <action_enum> [ , <action_enum> ] ;
                 .type   <type_enum>   [ , <type_enum> ] ;
  { <Variante> : .action <action_enum> [ , <action_enum> ] ;
                 .type   <type_enum>   [ , <type_enum> ] ; }
}
```

Als Aktionen (<action_enum>) sind die Werte *action_cut* und *action_copy* möglich.

Mögliche Werte für den Typ (<type_enum>) sind:

type_text
: Das Textattribut des Objekts wird übergeben.

type_object
: Eine DM-ID wird übergeben.

Beispiel

Im folgenden Beispiel wird die Quell-Ressource „Src“ mit zwei Varianten definiert.

Die erste Variante erlaubt das Kopieren und Ausschneiden von Texten. Die zweite Variante erlaubt das Kopieren der Objekt-ID.

Danach wird die Ressource der ***Listbox*** „Lb“ zugewiesen.

```
source Src
{
  0: .action action_cut, action_copy;
     .type   type_text;
  1: .action action_copy;
     .type   type_object;
}

listbox Lb
```

```
{
  .source Src;
  ...
}
```

3.3 target

Mit Hilfe dieser Ressource wird definiert, wie sich ein Objekt als Ziel einer Drag&Drop-Operation verhalten soll. Im Gegensatz zu den anderen Programmierressourcen können hier Varianten definiert werden.

Die Zuweisung zu einem Objekt erfolgt dann über das Attribut *.target*. Wenn dieses Attribut gesetzt ist, kann der Benutzer ein anderes Objekt auf dieses Objekt per Drag&Drop schieben.

Der Aufbau ist dem der ***source***-Ressource sehr ähnlich.

Hinweis

Drag&Drop wird vom IDM FÜR MOTIF nicht unterstützt.

Definition

```
{ export | reexport } target <Bezeichner>
{
             0 : .action <action_enum> [ , <action_enum> ] ;
                 .type   <type_enum>   [ , <type_enum> ] ;
  { <Variante> : .action <action_enum> [ , <action_enum> ] ;
                 .type   <type_enum>   [ , <type_enum> ] ; }
}
```

Als Aktionen (<action_enum>) sind *action_copy*, *action_cut* und *action_paste* möglich, mit *action_paste = action_cut, action_copy*.

Im Unterschied zu ***source*** wird die Reihenfolge der Typen und der Aktionen ausgewertet. Mit der Prioritätenliste der Typen wird das Übertragungsformat bestimmt. Mit der Prioritätenliste der Aktionen wird die Aktion bestimmt.

Beispiel

```
target Tar
{
  0: .action action_paste;
     .type   type_text;
}

listbox Lb
{
  .target Tar;
  ...
}
```

Index

L

M

N

O

P

U

V

W

X

Z