ISA Dialog Manager

Object Reference

A.06.03.b

In this manual all objects of the ISA Dialog Manager are described. For each object its predefined attributes and methods are listed, as well as the events and child objects it supports.

ISA Informationssysteme GmbH
Meisenweg 33
70771 Leinfelden-Echterdingen
Germany

Notation Conventions

DM will be used as a synonym for Dialog Manager.

The notion of UNIX in general comprises all supported UNIX derivates, otherwise it will be explicitly stated.

< >	to be substituted by the corresponding value
color	keyword
.bgc	attribute
{ }	optional (0 or once)
[ ]	optional (0 or n-times)
<A> | <B>	either <A> or <B>

Description Mode

All keywords are bold and underlined, e.g.

variable **integer** **function**

Indexing of Attributes

Syntax for indexed attributes:

[I]

[I,J] meaning [row, column]

Identifiers

Identifiers have to begin with an uppercase letter or an underline ('_'). The following characters may be uppercase or lowercase letters, digits, or underlines.

Hyphens ('-') are ***not*** permitted as characters for specifying identifiers.

The maximal length of an identifier is *31* characters.

Description of the permitted identifiers in the Backus-Naur form (BNF)

<identifier>	::=	<first character>{<character>}
<first character>	::=	_ | <uppercase>
<character>	::=	_ | <lowercase> | <uppercase> | <digit>

```
<digit>         ::=    1 | 2 | 3 | … 9 | 0

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

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

Table of Contents

1 Introduction

In this manual you will find a summary of the definition of the Dialog Manager objects, their attributes and methods.

The following **objects** can be used in the dialog definition:

application

This object for the processing of distributed applications contains information about the application, its functions, and the kind of communication. Distributed applications can be started and be terminated individually.

canvas

The canvas is used for the display of dynamic information. The DM provides the "frame" for the object; the actual contents then have to be provided by the application.

checkbox

By means of a checkbox a setting may be carried out, e.g. that of the layout. It is possible to select several alternatives if more than one is given.

control

The control object defines a client or server for communicating with external objects through the MICROSOFT WINDOWS OLE protocol.

datetime

The ***datetime*** object (class name ***dmw_datetime***) provides a simple and intuitive way to enter dates and times.

The object is implemented as a user-defined widget (USW).

dialog

The dialog object is the first object in each dialog script. It provides the root for the object hierarchy of this script.

document

The document object is the container for an XML Document that is saved as a DOM tree.

doccursor

The doccursor object points to a node of the DOM tree in an XML Document whose child the doccursor is, and can be moved in the DOM tree by its methods.

edittext

By means of edittext you make it possible that new text can be input at a specified place during runtime.

filereq

The object filereq is a modal file selection window. It distinguishes three modes:

- choosing a file to open
- specification for saving a file and
- choosing a directory.

groupbox

A groupbox groups objects in a window. All objects within one groupbox are on the same logical level.

image

Here you can load an image which has been created by a graphics program.

import

The use of a module in a different module or dialog will be introduced by an import. For further information see chapter “Modularization” in manual “Programming Techniques”.

layoutbox

The layoutbox can arrange its children automatically. It is therefore used if an (unknown) number of similar objects shall be displayed.

listbox

Written information can be listed in a space-saving and clear manner. Thus it is made available for selection in a listbox.

listview

The ***listview*** (class name ***dmw_listview***) is an advanced list object that supports various display types.

The object is implemented as a user-defined widget (USW).

mapping

A mapping object defines a semantic action that is performed for a specific node during a transformation of an XML Document or an IDM object hierarchy.

menubox

The menubox is used to control the dialog; by means of a menu commands may be called or actions be performed.

menuitem

A menuitem is the actual object which the user can select from a menu.

menusep

The second element within a menu is the menuseparator. It merely serves for optical separation of menuitems.

messagebox

The standard messageboxes of the window systems can be used.

module

A module is a partial dialog, an independent unit. Such a dialog part can serve as a library

- for standard resources which are used in a company (company-specific style guides)
- for models used across dialogs and projects
- for functionality that is needed over and over again (e.g. searches, calling a database…)

See Also

Chapter “Modularization” in manual “Programming Techniques”

notebook

"notebook", like a real notebook, has several pages, the notepages. It is divided into different sections which are marked by tabs.

notepage

This is a page of the notebook.

poptext

The combobox is a combination of input field and selection list. With this object, information can be displayed space-savingly if the user needs only one piece of information. By means of attributes the object behavior is set, e.g. whether the selection list is displayed constantly or only if necessary.

progressbar

The progress bar is used to show the progress of a task. For this, its values can be set to indicate the relative progress of a task in relation to the overall actions.

pushbutton

The pushbutton can be used to confirm or perform certain actions during the dialog.

radiobutton

Radiobuttons serve to take an "either/or" decision; i.e. only one alternative may be selected from a number of alternatives.

record

With records, information objects can be defined, which comprise different types of information belonging together logically.

rectangle

The rectangle is merely an object for the optical separation of various objects. If height or width are set to 1, the rectangle can be used as a line.

scrollbar

The scrollbar is an indication object. Certain adjustments can be made or displayed by a scrollbar.

setup

The setup object can be used to query the system configuration out of the dialog.

spinbox

The spinbox is a combination of input field and pushbuttons. The pushbuttons (arrow up, arrow down) are used to scroll through the contents within a value range.

splitbox

The splitbox is an object used to divide an area in a window into subareas that can be adjusted by the user.

statictext

With statictext you define a text which is not editable during run time but rather serves as fixed information.

statusbar

The statusbar giving information is displayed at the bottom line of the window.

subcontrol

A subcontrol represents an OLE child object that is managed directly or indirectly by an OLE Server.

tablefield

Lists of any format can be included into the application with this object. Data which is clearly displayed can also be processed there.

timer

Using this object, actions can be executed at specified points in defined intervals.

transformer

The transformer object allows to traverse an XML Document or an IDM object hierarchy and perform semantic actions on selected nodes.

treeview

With this object you can display information hierarchically. The visibility of subtree information can be toggled by selecting the relevant item.

window

> The window acts as optical and logical frame on screen for the visual objects that users interact with.

Objects can be defined with or without their own identifier. Objects without their own identifier inherit the identifier of their model but will be distinguished from objects having their own identifier. Objects having their own identifier are called **named objects**, objects with an inherited identifier are called **unnamed objects**.

Within a parent the identifier of a named object may be used for only one named object. One and the same inherited identifier may be used to define any number of unnamed objects. In addition the inherited identifier of unnamed objects may be used one more time as identifier for a named object.

If several named objects are defined with one and the same identifier within a parent object in the dialog script, only the first object will get this identifier. For all other objects there will be a warning on loading and the identifier will be ignored. These objects are thus treated as unnamed objects. The identifier will not be available later on when saving the object.

If objects are referenced via path names, it is possible to access the named object *Child* which was defined as child of the object *Parent* with *Parent.Child* (this object exists only once, cf above).

If no named child object having the identifier *Child* exists, then the first unnamed child object of *Parent* which has inherited the identifier *Child* will be accessed. With *Parent.Child[4]* the fourth child of *Parent* can be accessed. This child has inherited the identifier *Child* as unnamed object from its model. If there are a named and a unnamed object having the identifier *Child* within a parent object, then also the first unnamed child must be accessed with an index, i.e. *Parent.Child[1]*.

Indexes are automatically assigned in the correct way on writing a dialog file.

See Also

Chapter “Overview” in manual “Development Environment” for information about the Dialog Manager structure

2 Conventions for Object Description

In this chapter you will find a definition of the objects available for the Dialog Manager. Following the keyword and the syntax, we describe the object and its attributes.* Furthermore, we will state the object events and menus as well as the possible child objects and parent objects.

*** Note:**

The keyword **export** indicates that the object can be referenced also outside the module where it has been defined. **export** is only allowed if the object or the resource has been defined within a module.

The following objects allow exporting only if the corresponding parent child has been exported!

In the attribute tables you will find the following information on the attribute objects:

Attribute	name of the attribute notation: Rule Language: attribute prefix = "." C: attribute prefix = "AT_" COBOL: attribute prefix = "AT-"
RLD	**d**efinition in the **r**ule **l**anguage
PID	**d**efinition in the **p**rogram **i**nterface data type notation: Rule Language: data type prefix = none C: data type prefix = "DT_" COBOL: data type prefix = "DT-"
Properties	**S** = access: "set" **G** = access: "get" S & G = readable in Rule Language **D** = definable **C** ="changed", i.e. the attribute can be used to trigger rules.
Short Description	short description of the attribute

See Also

Chapter “Modularization” in manual “Programming Techniques”

Should you need further information on an attribute, please refer to the “Attribute Reference”, in which all DM attributes are described

3 application

The network version makes it possible to split the DM applications in several parts.

In doing so, the capacity of the display machine can be better used and the server can be relieved from overload. In addition, any actions like for example creating graphics, querying external interfaces can be carried out via the display machine.

The object ***application*** for distributed applications contains information about the application, its functions, and the type of communication. Individual applications can be started and terminated.

Definition

```
{ export | reexport } application  { <Identifier> }
{
  <object-specific attributes>
}
```

Events

extevent

finish

start

Children

document

Function

record

transformer

Parent

dialog

module

Menu

none

3.1 Attributes

Attribute	RLD	PID	Properties	Short Description
active	boolean	boolean	S,G/D/C	active state of an object

Attribute	RLD	PID	Properties	Short Description
certificatefile (since A.06.02.g)	string	string	S,G/D/C	defines the certificate file used for an SSL connection
codepage	enum	enum	S,G/D/C	defines the code page used to call application functions
connect	string	string	S,G/D/C	defines command line the application has been started with (IDM-connect)
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
exec	string	string	S,G/D/C	defines command line the application has been started with (IDMexec)
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
extevent	boolean	boolean	S,G/-/C	sends asynchronous events to client in network and MS Windows
firstrecord	object	record	S,G/-/C	accesses the first record of an object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name / identifier of the object
lastrecord	object	record	S,G/-/C	accesses the last record of an object

Attribute	RLD	PID	Properties	Short Description
local	boolean	boolean	S,G/D/C	local application
options[enum]	boolean	boolean	S,G/D/C	options for an SSL connection » *opt_cert_required* » *opt_no_ssl_v2* » *opt_verify_peer* (since IDM version A.06.02.h)
password (since A.06.02.g)	string	string	S,G/D/C	password for starting the application side with the RSH or SSH protocol
privatekeyfile (since A.06.02.h)	string	string	S,G/D/C	file with the private key used for the SSH protocol
publickeyfile (since A.06.02.h)	string	string	S,G/D/C	file with the public key used for the SSH protocol
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
transformer[I]	object	transformer	S,G/-/-	accesses the I-th transformer of an object
transport	string	string	S,G/D/C	defines transport mechanism
username (since A.06.02.g)	string	string	S,G/D/C	user name for starting the application side with the RSH or SSH protocol

3.2 Specific Attributes

The object ***application*** has no display attributes. The following attributes are available:

- *.active*
  This attribute defines and queries whether the application is currently active. Changing this attribute from false to true has the effect that the application is started. The application is terminated when the attribute is set from true to false.
- *.connect*
  This attribute defines that the application connects to a running server process which has been started on the host (defined by the host name or IP address and the port number).
- *.exec*
  This attribute defines that the application starts a process given by the path on the host (defined by the host name or IP address). It contains the program name, path, host name, and miscellaneous information.
- *.label*
  Internal identifier of the application.
- *.local*
  Defines whether the application runs locally or on a network.
- *.transport*
  .transport defines the internal transport mechanism that the communication layer is to use. This is possible by the "tcpip" protocol.

The attributes *.transport*, *.connect*, *.local* and *.exec* can only be changed if *.active* is set to false.

The attributes *.connect* and *.exec* depend on the used transport mechanism, i.e. future versions of the transport layer may have different types of connection establishment.

If the "tcpip"-protocol is used, the syntax used for *.connect* is:

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

The "host"-parameter contains the host name, the "port" parameter contains the port number.

If the "tcpip" protocol is used, the syntax for *.exec* is:

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

The "host" parameter contains the host name on which the program is to be started.

The "username" parameter may contain a user name which exists on the host. The "password" parameter may contain the valid password of the user. These two parameters are optional.

The "path" parameter contains the path of the program to be started.

As of IDM version A.05.02.i, the Distributed Dialog Manager (DDM) supports the IPv6 protocol on all architectures that **natively** support IPv6.

3.3 Example

```
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);
}
```

4 canvas

Highly specialized existing graphic applications (e.g. process visualization) which have to have direct access to the underlying window system can use the dialog object ***canvas*** as their “individual” drawing area.

The DM provides this object for the application. The DM does not consider the contents of this object; the application itself is responsible for the contents. For this reason, a ***canvas*** must not contain any child objects.

Definition

```
{ export | reexport } { model } canvas  { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

The DM uses the attributes listed in chapter “Attributes” in the usual manner. This object is typically used when the application has to display dynamic charts or diagrams.

Events

cut

extevent

focus

help

paste

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

4.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
bgc	identifier	color	S,G/D/C	background color
bordercolor	identifier	color	S,G/D/C	border color
borderraster	boolean	boolean	S,G/D/-	controls computation of geometry with active grid
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
borderwidth	integer	integer	S,G/D/C	borderwidth
canvasfunc	identifier	func	S,G/D/C	canvas function
class	class	class	-,G/-/-	class/id of object
control	identifier	instance	-,G/-/-	control currently belonging to object

Attribute	RLD	PID	Properties	Short Description
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object

Attribute	RLD	PID	Properties	Short Description
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage of object
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object

Attribute	RLD	PID	Properties	Short Description
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from the left
xright	integer	integer	S,G/D/C	x-coordinate, distance from the right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

4.2 Specific Attributes

With Motif 1.1, you cannot navigate over an object with keyboard navigation, if this object is a “composite widget” and if this widget has no children. A program abort is also likely.

Therefore, the attribute *.options[enum]* which decides whether a DrawingArea-Widget ("composite", i.e. it accepts children) or a DrawnButton-Widget ("primitive", i.e. it does not accept children) is to be used has been made available for the object canvas.

The DrawnButton-Widget is used as default.

Applications which include their own widgets or special widgets in the canvas, have to set the option *opt_accept_child*.

If *opt_accept_child* is set to *true*, a DrawingArea-widget (composite) will be used for the canvas. Default is a DrawnButton-widget.

The following *enum* options are only evaluated if *opt_accept_child* is not set.

If *opt_focus_frame* is set to *false*, the "location-cursor-border" (keyboard focus) will not be drawn if the canvas does not have the focus. The default is *true*.

option_index	Meaning
opt_accept_child	canvas accepts child widgets (i.e. no focus)
opt_focus_frame	no focus frame will be drawn
opt_motif_shadow	canvas draws motif shadow frame

4.3 Example

```
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;
}
```

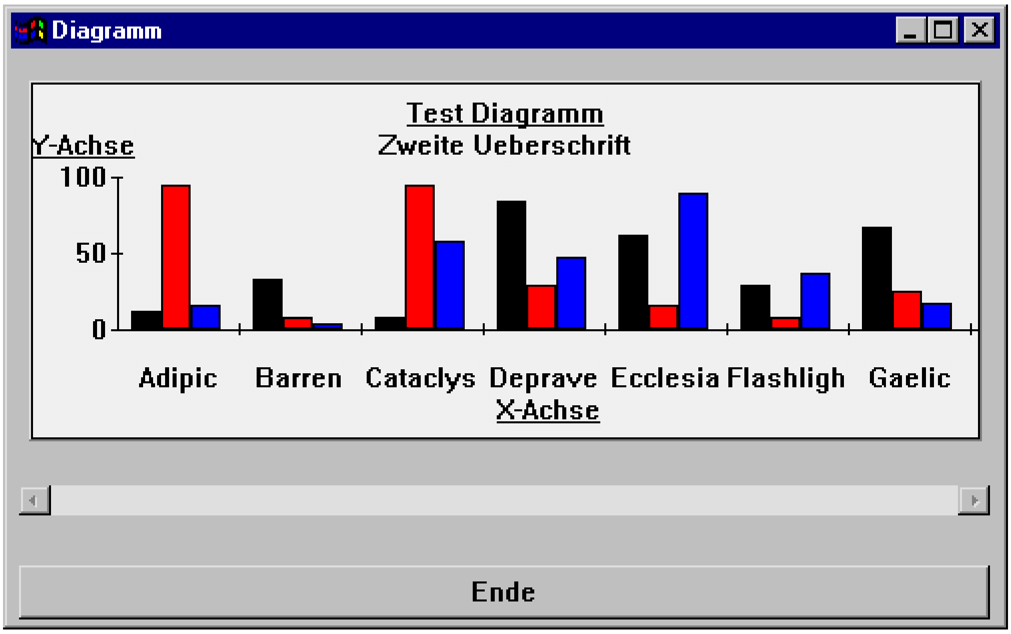


Figure 1: Canvas

5 checkbox

The ***checkbox*** is a type of button with the activation states “on” and “off”. A particular visual representation is linked to each of these states, so that the checkbox state is recognizable at any time. A filled square represents the “on” state, an empty square represents the “off” state.

Definition

```
{ export | reexport } { model } checkbox { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Important note on tab navigation

Checkboxes are assigned to `Tab` groups by the window systems. The group is mostly determined by the parent object. Single checkboxes can be reached individually with the `Tab` key. For checkboxes within a group, only one group element can be reached via the `Tab` key. The cursor keys are used to navigate within the group.
The .navigation attribute can be used to influence the `Tab` navigation. It determines whether a checkbox in the `Tab` navigation belongs to a group of elements or is treated as a stand-alone element.

Events

activate

cut

deactivate

extevent

focus

help

key

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up Menu

5.1 attributes

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[I]

membercount

menu

model

navigation

notepage

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

real_x

real_y

record[I]

recordcount

scope

sensitive

sizeraster

state

statushelp

style

text

toolhelp

toolbar

userdata

visible

width

window

xauto

xleft

xright

yauto

ybottom

ytop

5.2 Specific attributes

attribute	Description
active	Defines at *.style* = *checkbox* (*2*) whether the object should have the state “on” (*true*) or “off” (*false*). See chapter “Checkbox as a tristate button”
height	Height of the object. At value *0* the IDM calculates the height automatically based on the current character set.
navigation	Determines whether a checkbox belongs to a group of elements or is treated as a stand-alone element in the `Tab` navigation. Stand-alone checkboxes can be reached individually with the `Tab` key. For checkboxes within a group, only one group element can be reached by `Tab` key. The cursor keys are used to navigate within the group. » *navi_default* (default) » *navi_grouped* » *navi_tab*
options[enum]	Options of the object. Indices: » *opt_use_widget* (only Motif) » *opt_push_like* (only Microsoft Windows) » *opt_center_toolhelp* (only Microsoft Windows)

attribute	Description
state	Current activation state of the object. Possible values: » *state_unchecked* » *state_checked* » *state_indeterminate* See chapter “Checkbox as a tristate button”
style	Sets the appearance and behavior of the object. Possible values: » *checkbox* (2, default) » *tristate* (3) See chapter “Checkbox as a tristate button”
text	Displayed text (caption).
width	Width of the object. At value *0* the IDM calculates the width automatically based on the current character set.

5.3 Checkbox as a tristate button

Checkbox and Tristatebutton are implemented in ISA Dialog Manager using the same object. The .style attribute controls what type of object is displayed. The representation of the tristate button depends on the corresponding window system.

The states of the checkbox (as a tristate button) can be queried and set via the .state attribute.

For a tristate button, the .active attribute always returns false. Therefore, it is not suitable for querying the state of the checkbox. This does not change even if the checkbox is dynamically changed from *.style* = *tristate* (*3*) to *.style* = *checkbox* (*2*) at runtime.

The activate and deactivate events are not available for a tristate button.

Whether the interactive selection of the tristate button switches between all three states or only between “on” (*state_checked*) and “off” (*state_unchecked*), as well as the sequence when switching between the three states, is defined by the respective window system. The IDM has no influence on this.

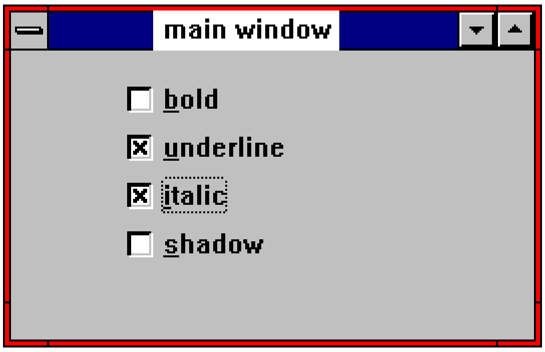
5.4 Notefor checkbox for Microsoft Windows

Concerning the background color, please refer to the notes on attribute .bgc in the “Attribute Reference”.

5.5 Example

Four navigable checkboxes within one window. The first two within a group:

```
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;
  }
}
```



The checkboxes underline and italic have been selected.

Figure 2: Checkbox

6 control

The tasks of the control object depend on the way the control object is used. There are two possibilities:

- If the control object is used as OLE client, then the communication with the server will be established with this object. In addition, this object defines the area in which the server is to appear within the client. All calls made to the server to set and query properties or to call methods will thus be carried out via this control object.
- If the control object is used as OLE server, it is used to define the interface and to communicate with the client. Dialog Manager is not able to make all its objects, methods, resources and attributes, etc. available as interfaces. This would go beyond the scope of the interfaces in OLE and, in addition, it is not recommendable offering the entire dialog to the external program. The attributes, methods and events of the control object describe the interface of the OLE server to its clients.

Definition

```
{ export | reexport } { model } control { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <grid attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

extevent

finish

help

help

paste

select

start

Children

canvas

checkbox

edittext

groupbox

image

listbox

menubox

menuitem

menusep

notebook

poptext

pushbutton

radiobutton

rectangle

scrollbar

spinbox

statictext

tablefield

treeview

window

Parent

dialog

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

6.1 Attributes

Attributes	RSD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
active	boolean	active	S,G/D/C	state of server or client
bgc	identifier	color	S,G/D/C	background color of object
bordercolor	identifier	color	S,G/D/C	border color of object
borderwidth	integer	integer	S,G/D/C	width of object border
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class of object
connect	boolean	boolean	S,G/D/C	state of connection to OLE server or OLE client
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation is still pending
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog for object
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class

Attributes	RSD	PID	Properties	Short Description
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color of object
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
firstsubcontrol	object	object	S,G/D/C	accesses the first sub-control
focus	object boolean	instance boolean	-,G/-/-	keyboard focus of object
font	identifier	font	S,G/D/C	font of object
function	identifier	func	S,G/D/C	function of object
groupbox	identifier	instance	-,G/-/-	groupbox the object belongs to
height	integer	integer	S,G/D/C	indicates height of object
help	string identifier	string text	S,G/D/C	helptext of object
index	integer	integer	-,G/-/-	current index of object in the children list of its parent
label	string	string	S,G/D/C	name (identifier) of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
lastsubcontrol	object	object	S,G/D/C	accesses the last sub-control
license_key	string	string	S,G/D/C	license key for an ActiveX control

Attributes	RSD	PID	Properties	Short Description
member[I]	attribute	attribute	-,G/-/-	i-th user-defined attribute of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	menu of object
message[I]	identifier	message	-,G/D/-	messages to be sent to client
mode	enum	enum	S,G/D/C	mode of control object, either as client or as server
model	identifier	instance	S,G/D/C	model belonging to object
name	string	string	-,G/D/-	name or ProgID of server or of client
notepage	identifier	instance	-,G/-/-	notepage the object belongs to
parent	identifier	instance	S,G/-/-	parent of object
picture	identifier	instance	S,G/D/C	picture to be displayed in inactive state
posraster	boolean	boolean	S,G/D/C	indication of position refers to raster
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectability of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from the left (in pixel)
real_xraster	integer	integer	-,G/-/-	width of internally used raster
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)

Attributes	RSD	PID	Properties	Short Description
real_yraster	integer	integer	-,G/-/-	height of internally used raster
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
reffont	identifier	font	S,G/D/C	reference font of object
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectability of object
sizeraster	boolean	boolean	S,G/D/C	size refers to raster of parent object
statushelp	string identifier	string text	S,G/D/C	text to be displayed in the statusbar
subcontrol[I]	object	object	S,G/D/C	accesses the I-th subcontrol
subcontrolcount	integer	integer	-,G/-/-	queries the number of subcontrols
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	userdata of object of any datatype
uuid	string	string	-,G/D/-	unambiguous UUID of OLE server
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer	integer	S,G/D/C	current width of object
window	identifier	instance	-,G/-/-	window the object belongs to

Attributes	RSD	PID	Properties	Short Description
xauto	integer (-1, 0, 1)	integer	S,G/D/C	type of x-coordinates definition
xleft	integer	integer	S,G/D/C	x-coordinate, distance from the left
xraster	integer	integer	S,G/D/C	unit in x direction
xright	integer	integer	S,G/D/C	x-coordinate, distance from the right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	type of y-coordinates definition
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
yraster	integer	integer	S,G/D/C	unit in y direction
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

6.2 Specific Attributes

6.2.1 connect

Identifier:

.connect

Classification: object-specific attribute

Definition

argument type:	boolean
C definition:	AT_connect
C datatype:	DT_boolean
COBOL definition:	AT-connect
COBOL datatype:	DT-boolean
access:	set, get

“changed”, i.e. attribute can be used to trigger rules.

This attribute defines the state of the connection to the OLE client or to the OLE server. The services of the OLE server can only be used if there is a connection.

6.2.2 mode

Identifier:

.mode

Classification:	object-specific attribute

Definition

argument type:	enum
value range:	mode_none, mode_client, mode_server
C definition:	AT_mode
C datatype:	DT_enum
COBOL definition:	AT-mode
COBOL datatype:	DT-enum
access:	set, get

“changed”, i.e. attribute can be used to trigger rules.

This attribute defines how to use the control object. There are three possibilities:

- mode_none: The mode is not defined, i.e. control is not used, neither as client nor as server. For example, you can set this value if you have implemented an OLE server, but if, for some reason, you are not able to act as a server, e.g. because the program was started as a normal program.
- mode_client: The control object will be regarded as OLE client, i.e. an OLE server will be accessed via this object.
- mode_server: The control object is used as OLE server and may be accessed by other clients.

6.2.3 name

Identifier:

.name

Classification:	object-specific attribute

Definition

argument type:	string
C definition:	AT_name
C datatype:	DT_string
COBOL definition:	AT-name
COBOL datatype:	DT-string
access:	set, get

“changed”, i.e. attribute can be used to trigger rules.

This attribute defines the name of the corresponding server at the control object which is used as OLE client. You may deposit either the name or the ProgID of the server in this attribute.

If the control object is used as OLE server, the given string in this attribute will be registered as server name in the registry list.

Example

```
model control CtTest
{
  .mode mode_client;
  .name "InternetExplorer.Application.1";
  .visible true;
  .active true;
  .connect false;
}
```

6.2.4 picture

Identifier:

.picture

Classification:	object-specific attribute

Definition

argument type:	object
C definition:	AT_picture
C datatype:	DT_tile
COBOL definition:	AT-picture

COBOL datatype:	DT-tile
access:	set, get

“changed”, i.e. attribute can be used to trigger rules.

This attribute is used to define the picture which is to be displayed in the inactive state of the control object.

See Also

Attribute picture

6.2.5 uuid

Identifier:

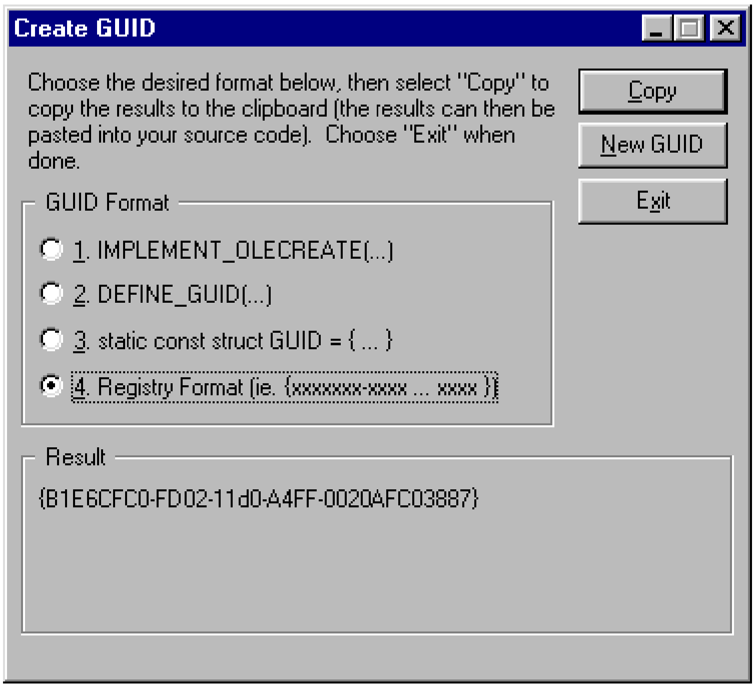
.uuid

Classification:	object-specific attribute

Definition

argument type:	string
C definition:	AT_uuid
C datatype:	DT_string
COBOL definition:	AT-uuid
COBOL datatype:	DT-string
access:	get

This attribute must be set when the control object is used as OLE server. Via this UUID the object can be clearly identified by other programs. This UUID will be generated by using the program guidgen.exe.



Choose the desired format below, then select "Copy" to copy the results to the clipboard (the results can then be pasted into your source code). Choose "Exit" when done.

GUID Format

○ 1. IMPLEMENT_OLECREATE(...)

○ 2. DEFINE_GUID(...)

○ 3. static const struct GUID = { ... }

● 4. Registry Format (ie. {xxxxxxxx-xxxx ... xxxx })

Result

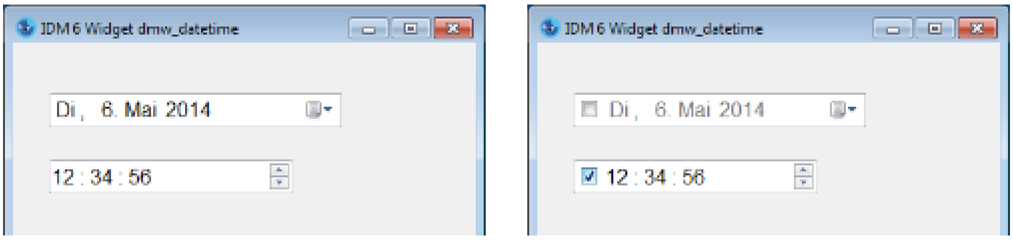
{B1E6CFC0-FD02-11d0-A4FF-0020AFC03887}

Figure 3: Generation of GUID

If the control object is used as OLE client, you may deposit the server UUID in this attribute. This attribute thus replaces the attribute .name which is usually used to deposit the server name.

7 datetime

The ***datetime*** object (class name ***dmw_datetime***) provides a simple and intuitive way to enter dates and times. It can display date or time values or both in different formats. Besides keyboard input, the user can change the values, for example by selecting a date from a calendar (upper ***datetime*** object in “Figure 4”) or using spin buttons (lower ***datetime*** object in “Figure 4”). In addition, the user can be enabled to set the value to “indefinite” using a checkbox (right part of “Figure 4”).



Di , 6. Mai 2014

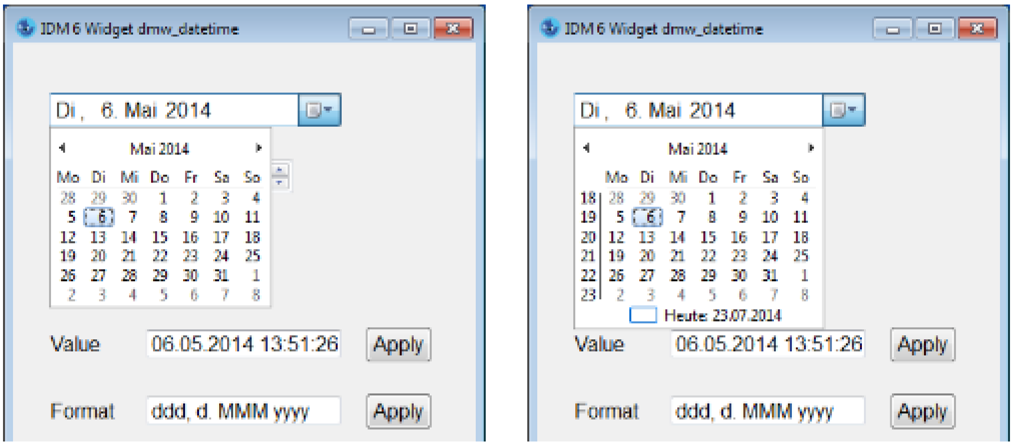
12 : 34 : 56

□ Di , 6. Mai 2014

☑ 12 : 34 : 56

Figure 4: *datetime objects*

“Figure 5” shows a ***datetime*** object with the calendar unfolded. The appearance of the calendar can be adjusted by several attributes. For example, in the right part of “Figure 5”, the numbers of the calendar weeks and the current date are displayed in the calendar.



Di , 6. Mai 2014

Mai 2014

Mo	Di	Mi	Do	Fr	Sa	So
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

Value 06.05.2014 13:51:26

Format ddd, d. MMM yyyy

Di , 6. Mai 2014

Mai 2014

	Mo	Di	Mi	Do	Fr	Sa	So
18	28	29	30	1	2	3	4
19	5	6	7	8	9	10	11
20	12	13	14	15	16	17	18
21	19	20	21	22	23	24	25
22	26	27	28	29	30	31	1
23	2	3	4	5	6	7	8

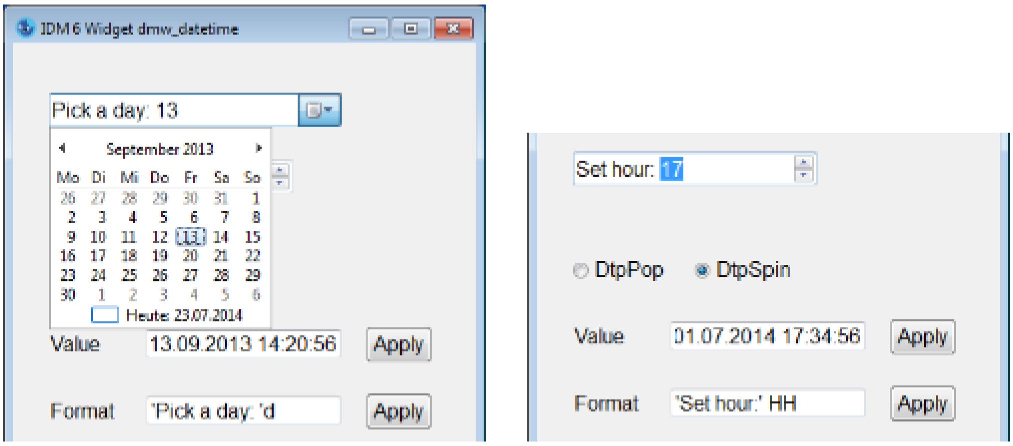
Heute: 23.07.2014

Value 06.05.2014 13:51:26

Format ddd, d. MMM yyyy

Figure 5: *datetime object with opened calendar*

The presentation of the date and time values can be controlled using a format string. You can use predefined system formats or define own display formats using the corresponding formatting characters. In both cases, the language and region settings of the system are taken into account (for example, for the month names). “Figure 6” shows ***datetime*** objects with formats that contain extra, non-editable text in addition to parts of the date or time.



Pick a day: 13

Value 13.09.2013 14:20:56

Format 'Pick a day: 'd

Set hour: 17

○ DtpPop ● DtpSpin

Value 01.07.2014 17:34:56

Format 'Set hour:' HH

Figure 6: datetime objects with different formats

Availability

- Microsoft Windows only.
- Can only be used with the USW option of the ISA Dialog Manager.
- The **idmwidgets.dll** must be in the USW class path (default: *<IDM installation directory>\uswclasses*).

7.1 Definition

```
{ export | reexport } { model } dmw_datetime { <Identifier> }
{
  <standard attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events	*changed*	*deselect*	*extevent*
	focus	*help*	*select*
Children	None		
Parent	***groupbox***	***layoutbox***	***notepage***
	splitbox	***toolbar***	***window***
Menu	Pop-up Menu		

7.2 Description of Events

7.2.1 deselect

The *deselect* event occurs when the ***datetime*** has lost the focus. This is an indication that the user has completed entering or selecting the date or time. However, it should be noted that this event is not only triggered by direct user actions, but may also occur as a side effect of other actions:

- The user sets the focus with the mouse or keyboard to another object.
- The user activates another application or another application pushes itself into the foreground and captures the focus.

Note

Opening a menu or pop-up does not necessarily move the focus. Therefore, it cannot be predicted whether a *deselect* event will occur for these actions. For example, a *deselect* event may not be triggered until the *select* event rule of a ***menuitem*** is executed, if it is triggered at all.

7.2.2 select

The *select* event will occur each time the content of the *.value* attribute changes. The event can be triggered by various actions of the user:

- The user edits the value in the input field of the ***datetime***.
- The user navigates in the calendar to reach a specific date.
- The user selects a value with mouse or keyboard.
- The user scrolls the spinbox of the ***datetime***.

This means that multiple *select* events will typically occur before the user has entered, set or selected the final value. Therefore, the event rule for the *select* event should not perform any extensive actions.

The *.real_modified* attribute can be used to check whether the value of the ***datetime*** has actually changed due to the user actions, compared to the value of the ***datetime*** when it received the focus.

The *select* event may also occur if *.minvalue* or *.maxvalue* are violated. This depends on the system object, but here too a content change of *.value* must be present.

7.3 Inherited Attributes

Attribute	Data Type	Notes
.accelerator	object	
.bgc	object	is ignored
.borderraster	boolean	

Attribute	Data Type	Notes
.control	object	
.count[attribute]	anyvalue	
.cursor	object	
.cut_pending	boolean	
.cut_pending_changed	boolean	
.dialog	object	
.document[integer]	object	
.export	boolean	
.fgc	object	is ignored
.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	
.menu[integer]	object	
.model	object	
.module	object	

Attribute	Data Type	Notes
.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	
.userdata	anyvalue	
.visible	boolean	
.width	integer	

Attribute	Data Type	Notes
.window	object	
.xauto	integer	
.xleft	integer	
.xright	integer	
.yauto	integer	
.ybottom	integer	
.ytop	integer	

7.4 Specific Attributes

Attribute	Short Description	Data Type	Default	Access		C	I
				get	set		
.allowundefined	acceptance of indefinite values	*boolean*	*false*	x	x	x	x
.calendaralignment	position of the calendar	*integer* (*-1*, *0*, *1*)	*1*	x	x	x	x
.format	display format	*string*	""	x	x	x	x
.maxvalue	maximum value	*string*	""	x	x	x	x
.minvalue	minimum value	*string*	""	x	x	x	x
.real_modified	indicates actual change of *.value*	*boolean*		x	–	–	–
.shortdaynames	display of short weekday names	*boolean*	*false*	x	x	x	x
.style	value input through a fold-out calendar or a spinbox	*class* (*poptext*, *spinbox*)	*poptext*	x	x	x	x
.today	display of current date	*boolean*	*true*	x	x	x	x
.todaymarker	highlighting of current date	*boolean*	*true*	x	x	x	x

Attribute	Short Description	Data Type	Default	Access		C	I
				get	set		
.trailingdates	display days of previous and next month	*boolean*	*true*	x	x	x	x
.value	date and/or time value	*string*	*""*	x	x	x	x
.weeknumbers	display of calendar week	*boolean*	*false*	x	x	x	x

C *changed* event on modification

I Attribute is inherited

8 dialog

In every DM file the dialog identifier has to be defined. This can be done with the keyword **dialog**, followed by the dialog identifier and the attributes of the dialog.

Definition

```
dialog { <Identifier> }
{
  <standard attributes>
  <grid attributes>
  <object-specific attributes>
}
```

Events

deactivate (if a timeout occurred)

extevent

finish

help

key

start

Children

document

import

messagebox

record

transformer

window

Parent

none

Menu

none

8.1 Attributes

Attribute	RLD	PID	Properties	Short Description
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class/id of object
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
ignorecursor	boolean	boolean	S,G/D/C	cursor will not be used
label	string	string	S,G/D/C	name/id of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
msgboxtext[enum]	string	text	S,G/D/C	contents of a messagebox
options[enum]	boolean	boolean	S,G/D/-	special options of object

Attribute	RLD	PID	Properties	Short Description
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
reffont	object	font	S,G/D/C	reference font of object
timeout	integer	integer	S,G/D/C	time specification for timeout
xraster	integer	integer	S,G/D/C	unit in x-direction
yraster	integer	integer	S,G/D/C	unit in y-direction

8.2 Specific Attributes

Grid Attributes

With the grid values defined in the dialog, the windows contained in the dialog can be positioned relatively to a user-determined size or to a fixed font. A grid can be specified with the attributes *.xraster* and *.yraster*.

Dialog Attributes

The attribute *.msgboxtext[I]* defines the text of messagebox buttons.

If a button text is not supported by the window system, the attribute will be ignored.

Some window systems (e.g. Microsoft Windows) have fixed button texts and do not allow labeling by the user.

The attribute *.timeout* defines the time delay after which the timeout is to become active when the user doesn't work with the program. The delay is specified according to the same schemes that are used for the timer increments.

If *.timeout* of the dialog is not 0, the DM will set up this timer each time it reads the window system blocking it. A deactivate event is sent to the setup object when the timer becomes active. Every event other than another timer resets the timeout.

9 doccursor (XML Cursor)

The ***doccursor*** object is always a child of an XML Document. It refers to a node in the DOM tree, which is saved in the XML Document (the parent).

There are methods to set the reference to a different node of the DOM tree. In other words this means that the XML Cursor can be moved in the DOM tree through methods. It will remain a child of the XML Document of course.

Definition

```
{ export | reexport } { model } doccursor { <Identifier> }
{
  [ <atribute definition> ]
  [ <method definition> ]
}
```

Besides the “normal” ISA Dialog Manager attributes, the XML Cursor attribute contains additional attributes which make it possible to access properties such as name, value or other attributes of the DOM nodes. Because these are runtime attributes, they cannot be passed down. In addition, many of these attributes can only be read because the corresponding properties of the DOM nodes cannot be changed.

The XML Cursor is initially invalid, but as soon as it is accessed for the first time it will be automatically positioned to the root of the DOM tree. Note that this also happens when the XML Cursor gets invalid through an action. The attribute *.mapped* shows if the XML Cursor is valid or not.

Events

None

Children

document

record

transformer

Parents

document

Menu

None

Methods

:add()

:delete()

:match()

:reparent()

:select()

:transform()

9.1 Attributes

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[I]

userdata

value

xml

9.2 Object-specific Attributes

attribute[I]
attribute[string]

Depending on the indexing, the attribute “attribute” serves for querying the name or the value of a DOM nodes’ attributes .

If the index is a number, the name of the corresponding attribute of a DOM node will be delivered. Take note that attributes of DOM nodes usually are unsorted.

If the index is a string, then the index is regarded as a name of an attribute and the value of this attribute is returned. An assignment to an attribute indexed with a string creates a corresponding attribute for the DOM node. The assignment of an empty string deletes the corresponding attribute of the DOM node.

This attribute is not passed down because it refers to a runtime characteristic.

data

Is for setting and retrieving the data within a DOM node. The attribute is only available when the node type is either *nodetype_cdata_section* or *nodetype_processing_instruction*.

This attribute is not passed down because it refers to a runtime characteristic.

idispatch

The IDispatch COM interface pointer of the XML Cursor can be accessed through the attribute idispatch under Microsoft Windows.

In the Rule Language the attribute can only be assigned to the same attribute of a different IDM object. Be aware, that in the programming interface the COM object is only valid during the time when it is in use by the ISA Dialog Manager. Therefore, an application should increase its reference counter (COM Method: IUnknown->AddRef). When the object is no longer needed, the counter should be decreased (COM Method: IUnknown->Release). Please note, the counter cannot be decreased more often than it is increased. Otherwise, the COM object will be released. The ISA Dialog Manager cannot recognize this situation and this will lead to a system crash.

This attribute is not passed down because it refers to a runtime characteristic.

ixmldomnode

The IXMLDOMNode COM interface pointer of the XML Cursor can be accessed through the ixmldomnode attribute under Microsoft Windows.

In the Rule Language the attribute can only be assigned to the same attribute of a different IDM object. Be aware, that in the programming interface the COM object is only valid while it is in use by the ISA Dialog Manager. Therefore, an application should increase its reference counter (COM Method: IUnknown->AddRef). When the object is no longer needed, the counter should be set back again, or decreased (COM Method: IUnknown->Release). Please note, the counter cannot be decreased more often than it is increased. Otherwise, the COM object will be released. The ISA Dialog Manager cannot recognize this situation and this will lead to a system crash.

This attribute is not passed down because it refers to a runtime characteristic.

ixmldomnodelist

The IXMLDOMNodeList COM interface pointer of the XML Cursor can be accessed through the ixmldomnodelist attribute under Microsoft Windows. The direct children of the XML Cursor can be accessed through the interface pointer.

In the Rule Language the attribute can only be assigned to the same attribute of a different IDM object. Be aware, that in the programming interface the COM object is only valid while it is in use by the ISA Dialog Manager. Therefore, an application should increase its reference counter (COM Method: IUnknown->AddRef).). When the object is no longer needed, the counter should be decreased (COM Method: IUnknown->Release). Please note, the counter cannot be decreased more often than it is increased. Otherwise, the COM object will be released. The ISA Dialog Manager cannot recognize this situation and this will lead to a system crash.

This attribute is not passed down because it refers to a runtime characteristic.

mapped

Is true when the XML Cursor points to a node in the DOM tree. Please take note that an XML Cursor, which does not reference any node in the DOM tree, is automatically positioned on the root of the DOM tree when any object-specific attribute is accessed or when an object-specific method is invoked.

This attribute is not passed down because it refers to a runtime characteristic.

name

Refers to the name or tag of the DOM node.

This attribute is not passed down because it refers to a runtime characteristic.

nodetype

This serves for querying the type of DOM node.

This attribute is not passed down because it refers to a runtime characteristic.

path

Delivers a string representation for the position of the XML Cursor in the DOM tree. The select method can be called with this string in order to position an XML Cursor to the nodes in the DOM tree.

If the value of the path attribute is stored somewhere else (i.e. in the userdata attribute), then it is important to know that these stored values will not be adjusted when the structure of the DOM tree changes. When the select method is invoked with the stored value afterward, the XML Cursor will point to an incorrect DOM node.

This attribute is not passed down because it refers to a runtime characteristic.

publicid

Is the public identifier of the DOM node. The attribute is only available, when the node type is either *nodetype_entity* or *nodetype_notation*.

This attribute is not passed down because it refers to a runtime characteristic.

specified

This reveals if an attribute of a DOM node was explicitly given, or if it was inherited from a standard value. The attribute is always *true* except for the node type *nodetype_attribute*.

This attribute is not passed down because it refers to a runtime characteristic.

systemid

Is the system identifier of DOM nodes. This attribute is only available when the node type is either *nodetype_entity* or *nodetype_notation*.

This attribute is not passed down because it refers to a runtime characteristic.

target

Is the name of the instruction for a DOM node. The value is the same as the value of the name attribute. This attribute is only available when the node type is *nodetype_processing_instruction*.

This attribute is not passed down because it refers to a runtime characteristic.

text

Is the value of all sub-nodes within a DOM node. A string is delivered which represents the text of all sub-nodes. This attribute is mainly helpful when only the text of an XML element is needed, as it is not required to navigate to the child nodes containing the actual text.

Please note that setting this attribute automatically deletes all child nodes and inserts a new text node.

This attribute is not passed down because it refers to a runtime characteristic.

value

Is the value of a DOM node. This attribute is only available when the node type is *nodetype_attribute*, *nodetype_text*, *nodetype_cdata_section*, *nodetype_processing_instruction* or *nodetype_* *comment*.

This attribute is not passed down because it refers to a runtime characteristic.

xml

String representation of a DOM node and all of its child nodes.

This attribute is not passed down because it refers to a runtime characteristic.

9.3 Object-specific Methods

:add()

Inserts a child node as the last node to the current DOM node. The XML Cursor is then set to the new element.

:delete()

Deletes the DOM node and all of its child nodes. The XML Cursor is then positioned to the father node. The XML Cursors, which point to the deleted DOM nodes in the sub-tree, become invalid. The attribute *.mapped* has the value *false* for invalid XML Cursors.

When the value of the path attributes is stored elsewhere (i.e. in userdata attribute), then it is important to know that these stored values will not be adjusted when the structure of the DOM tree changes. When the select method is invoked with the stored value afterward, the XML Cursor will point to an incorrect DOM node.

:match()

Tests if the DOM node satisfies the given pattern (see chapter “Pattern for the Methods :match() and :select()”).

:reparent()

Reallocates the DOM node with all of its child nodes.

When the value of the path attributes is stored elsewhere (i.e. in userdata attribute), then it is important to know that these stored values will not be adjusted when the structure of the DOM tree changes. When the select method is invoked with the stored value afterward, the XML Cursor will point to an incorrect DOM node.

:select()

Moves the XML Cursor in the given direction or moves the XML Cursor to the first DOM node that matches the given pattern (see chapter “Pattern for the Methods :match() and :select()”).

:transform()

Transforms the XML Cursor with the given scheme. When the target is an XML Document, the saved DOM tree is deleted, and a new tree is built. All existing XML Cursors become invalid. The attribute *.mapped* has the value *false* for invalid XML Cursors.

Alternatively, the target can be a text or file. If the result of the transformation is no legal XML format, then it needs to be directly converted into a text or file, as the result cannot be assigned to an XML Document. This is true, for example, for conversions to HTML.

9.4 Pattern for the Methods :match() and :select()

A pattern is very similar to a ISA Dialog Manager identifier. The pattern represents a path of element names. The path starts at the root of the tree. Each hierarchical level is compared to the corresponding part of the path. Here it is possible to define certain characteristics of the father such as existence of an attribute or the position within the children.

Basically all characters except **.**, **[**, **"**, **]**, **<**, **>**, **=**, ****, tab, space and line break can be used in the pattern. If one of the characters mentioned above, without the page break, shall be used, then it has to be escaped with a **** before the character. Between double quotes (**"**), that is within a string, the following symbols are also allowed: **.**, **[**, **]**, **<**, **>**, **=**, tab and line break. Please note that in the dialog script the symbol **** within a string already is an escape symbol. Due to this it is important to use \\ in a dialog script whenever a **** is needed. Also in a dialog script **\"** has to be used whenever the character **"** is needed.

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

<Name>

Is compared to the name attribute of the XML Cursor. The XML Cursor must possess the node type *nodetype_element*. Alternatively, the character ***** can be used. In this case the name attribute will not be taken into consideration.

The name of an XML element starts with a letter or an underline. It may contain letters, digits, hyphens, underlines and dots. The exact definition of an XML element name can be found in the XML specifications (**www.w3c.org/XML**).

<Attr>

The DOM node that points to the XML Cursor must possess the given attribute.

The name of an XML attribute starts with a letter or an underline. It may contain letters, digits, hyphens, underlines and dots. The exact definition of an XML attribute name can be found in the XML specifications (**www.w3c.org/XML**).

<Op>

Is a comparison operator which compares the attribute given in <Attr> with <Value>. It can be tested for equality **=** and inequality **<>**.

<Value>

Is the value to which the <Attr> is compared.

<Idx>

The DOM node must be at this position within the child nodes of the same parent. The first child has position 1.

<Idx> consists only of figures (**0** – **9**), which are interpreted as number.

Particularities

.

When the pattern start with a dot, it is relative to the current DOM node. This means that the path begins at the current DOM node.

..

Two consecutive dots define, that an arbitrary number of hierarchy levels may be skipped.

[<Idx>]

An index without any further information selects the DOM node at this position. In this case, the type of DOM node is insignificant. Thus each DOM node can be uniquely referenced by an expression like {**[**<Idx>**]**[**.[**<Idx>**]**]} (path attribute).

Additions for Microsoft Windows

XPath can also be used as pattern syntax. Here, the pattern must start with either **/** or **./**. The use of XPath patterns is not supported on all platforms and therefore not portable.

10 document (XML Document)

The document object is the container for an XML Document. An XML Document is saved as a DOM tree. This DOM tree can be traversed with the help of a doccursor, which must be a child of the document object.

Definition

```
{ export | reexport } { model } document { <Identifier> }
{
  [ <atribute definition> ]
  [ <method definition> ]
}
```

Events

None

Children

doccursor

document

record

transformer

Parents

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

Menu

None

Methods

:load()

:save()

:transform()

:validate()

10.1 Attributes

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

10.2 Object-specific Attributes

doccursor[I]

It is possible to access the XML Cursor of the XML Document through the doccursor attribute. The attribute is indexed with the object index (similar to child).

idispatch

The Idispatch COM interface pointer in XML Documents can be accessed through the idispatch attribute under Microsoft Windows.

In the Rule Language the attribute can only be assigned to the same attribute of a different IDM object. Please note, in the programming interface a COM object will remain valid only as long as it remains in use in the ISA Dialog Manager. For this reason it is important to increase the reference counter of an application (COM Method: IUnknown->AddRef) . When the object is no longer needed, the counter must be decreased again (COM Method: IUnknown->Release). In any case, it is not allowed to decrease the counter more than it is increased. If this happens, the COM object will be released. The ISA Dialog Manager cannot recognize this situation, which will lead to a system crash. The ISA Dialog Manager may also crash when the given pointer does not point to a COM interface.

This attribute is not passed down because it refers to a runtime characteristic.

ixmldomdocument2

The IXMLDOMDocument2 COM interface pointer in XML Documents can be accessed through the ixmldomdocument2 attribute under Microsoft Windows.

In the Rule Language the attribute can only be assigned to the same attribute of a different IDM object. Please note, in the programming interface a COM object will remain valid only as long as it remains in use in the ISA Dialog Manager. For this reason it is important to increase the reference counter of an application (COM method: IUnknown->AddRef). When the object is no longer needed, then the counter must be decreased again (COM Method: IUnknown->Release). In any case, it is not allowed to decrease the counter more than it is increased. If this happens, the COM object will be released. The ISA Dialog Manager cannot recognize this situation, which will lead to a system crash. The ISA Dialog Manager may also crash when the given pointer does not point to a COM interface.

This attribute is not passed down because it refers to a runtime characteristic.

xml

The string representation of XML Documents can be accessed with this attribute. When a new value is set, the saved DOM tree is deleted and a new DOM tree is built from the newly set value. All existing XML Cursors become invalid. When an XML Cursor is invalid, the attribute *.mapped* has the value *false*.

10.3 Object-specific Methods

:load()

This loads an XML Document from the given file or URL. The saved DOM tree is deleted, and a new DOM tree is built. All existing XML Cursors become invalid. When an XML Cursor is invalid, the attribute *.mapped* has the value *false*.

:save()

Saves the XML Document to a file or URL.

:transform()

Transforms the XML Document with the given scheme. When the target is an XML Document, the saved DOM tree is deleted and a new tree is built. All existing XML Cursors become invalid. When an XML Cursor is invalid, the attribute *.mapped* has the value *false*.

Alternatively, the target of the transformation can be a text or file. If the result of the transformation is no legal XML format, then it needs to be directly converted into a text or file, as the result cannot be assigned to an XML Document. This is true, for example, for conversions to HTML.

:validate()

This checks if the XML Document conforms to the document type given in the XML Document. If it does not have a document type, an error will occur.

11 edittext

The keyword **edittext** is used to handle texts editable by the user.

Definition

```
{ export | reexport } { model } edittext { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <scrollbar attributes>
  <object-specific attributes>
}
```

Within the object edittext the text cursor can be moved by the cursor control keys on the keyboard.

The `Backspace` key erases the character to the left of the text cursor. The `Delete` key erases the character to the right of the text cursor. All alphabetic and numeral keys are processed as “input character”.

Events

activate

charinput

cut

deselect

deselect_enter

extevent

focus

help

key

modified

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

11.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
active	boolean	boolean	S,G/D/C	activation state of an object
alignment	integer (-1, 0, 1)	integer	S,G/D/C	alignment of text
bgc	identifier	color	S,G/D/C	background color
borderraster	boolean	boolean	S,G/D/-	controls computation of geometry with active grid

Attribute	RLD	PID	Properties	Short Description
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
borderwidth	integer	integer	S,G/D/C	borderwidth
class	class	class	-,G/-/-	class/id of object
content	string identifier	string	S,G/D/C	contents of edittext
control	identifier	instance	-,G/-/-	control currently belonging to object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
editable	boolean	boolean	S,G/D/C	editability of editable text
endsel	integer	integer	S,G/D/C	end of the selection
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object

Attribute	RLD	PID	Properties	Short Description
font	identifier	font	S,G/D/C	object font
format	string identifier	string format	S,G/D/C	format string of text
formatfunc	identifier	func	S,G/D/C	linking of application function to edittext (for formatting)
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
hinttext	string	string	S,G/D/C	placeholder text of edittext (see also chapter “Notes for attribute .hinttext”)
hsb_visible	boolean	boolean	S,G/D/C	visibility of horizontal scrollbar
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
maxchars	integer	integer	S,G/D/C	maximal number of input characters for edittext
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object

Attribute	RLD	PID	Properties	Short Description
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
multiline	boolean	boolean	S,G/D/C	multiline edittext
navigable	boolean	boolean	S,G/D/C	focusability of edittext
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_modified	boolean	boolean	-,G/-/-	actual change of content
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object

Attribute	RLD	PID	Properties	Short Description
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
startsel	integer	integer	S,G/D/C	start of the selection
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
.textwidth	integer	integer	S,G/D/C	text width of an RTF edittext (see also chapter “Attribute .textwidth”)
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
vsb_visible	boolean	boolean	S,G/D/C	visibility of vertical scrollbar
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xmargin	integer	integer	S,G/-/-	right and left spacing between the edges of the layoutbox and the children can be set
xright	integer	integer	S,G/D/C	x-coordinate, distance from right

Attribute	RLD	PID	Properties	Short Description
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ymargin	integer	integer	S,G/-/-	upper and lower spacing between the edges of the layoutbox and the children can be set
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

11.2 Specific Attributes

Standard Attributes

The attribute *.active* defines whether the edittext is to have the current focus.

Text Attributes

The text displayed in the edittext array is defined with *.content*.

The attribute *.real_modified* indicates, whether the content actually has changed since the edittext has received the focus.

Scrollbar Attributes

The edittext can be supplied with the scrollbar attributes only if *.multiline* = *true*:

- .hsb_visible
- .vsb_visible

The attributes *.hsb_visible* and *.vsb_visible* define whether an edittext is to have scrollbars or not.

Layout Attributes

.borderstyle

Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.

11.2.1 Notes for attribute .hinttext

This attribute is supported only by WINDOWS and QT.

For the ***edittext*** object on WINDOWS, .hinttext is supported only if the attributes .multiline and .options [opt_rtf] have the value *false*.

11.3 Editable Text with Formatting (RTF edittext)

On MICROSOFT WINDOWS, the edittext has an additional mode to support formatted text in Rich Text Format (RTF). This mode is activated by setting *.options[opt_rtf] = true*.

RTF is a text format specified by Microsoft that contains formatting instructions. RTF files can be edited, for example, with the Windows program WordPad, Microsoft Office Word and the Writer of OpenOffice or LibreOffice.

11.3.1 Particularities of Certain Attributes

The function of the attributes in the RTF edittext corresponds largely to their function in an edittext without formatting. However, for some attributes, there are particularities to be considered that are related to the fact that the content string – that is, the value of the *.content* attribute – and the displayed text differ.

The content string contains formatting instructions such as \i, \ul, and \par, which are not displayed as text, but in the mentioned examples cause the text in the display to be italic, underlined, or in a new paragraph. The RTF edittext may also restructure the content string, for example, by adding information to the RTF header.

This means that there are particularities for attributes where the text length or position in the text matters:

- Compliant with Windows, *.startsel* and *.endsel* refer to the position of the cursor or the selection in the **displayed** text. It is not possible to infer the position in the content string because of the formatting instructions contained therein. For this reason, *.startsel* and *.endsel* cannot be used directly to edit the contents (*.content*) of the edittext.
- *.format*, *.formatfunc* and *.maxchars* are permitted, but not useful for multi-line texts.

11.3.2 Attribute .textwidth

The maximum width for the text in the RTF edittext can be defined independently from the width of the edittext. Setting the text width for example may be useful to make the alignment of texts (left-aligned, right-aligned or centered) visible.

With the *.textwidth* attribute the maximum text width can be set and queried. If *.textwidth* is set to a value <= *0*, the text width is calculated automatically and depends on the visibility of the horizontal scrollbar. The details are described at the attribute .textwidth in the “Attribute Reference”.

11.3.3 Editing and Formatting Content

The content of an RTF edittext can be modified with the method :replacetext(). The method :gettext() can be used to retrieve text and :findtext() can be used to search for text in the content. **:replacetext()** can insert plain, unformatted text or text in RTF format, **:gettext()** can return the retrieved text in both formats. To format text or query its formatting, the methods :setformat() and :getformat() are available.

These methods are explained in the “Method Reference”.

Depending on the font, it may happen that formatting instructions or formatting attributes set with **:setformat()** are ignored by the Windows object that the IDM uses for the RTF edittext. On Windows 7, this happens, for example, with *text_bold* if no font is set explicitly and therefore the “System” font is used implicitly.

11.4 Example

Single-line Edittext

```
window MAIN
{
  child edittext Input
  {
    .xleft    10;
    .ytop     10;
    .width    100;
    .content  "This is the edittext";
  }
}
```

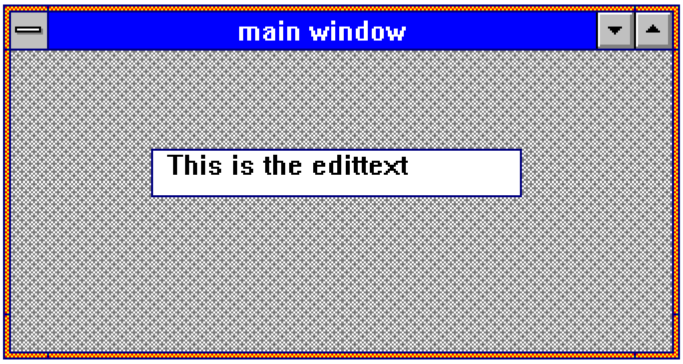
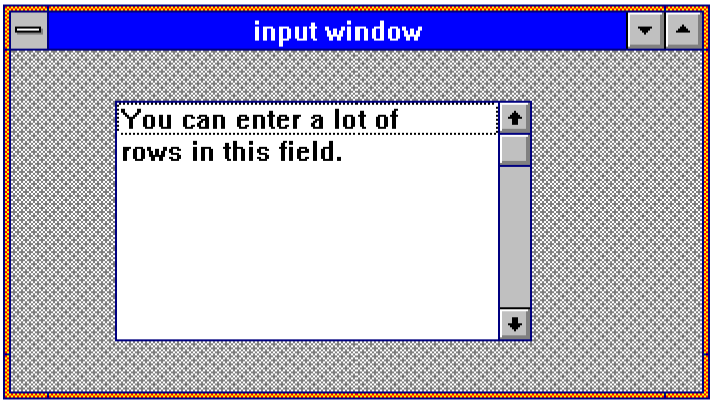


Figure 7: Edittext

Multi-line Edittext:

```
window Input
{
  child edittext PE_Editor
  {
    .xleft      20;
    .ytop       20;
    .visible    true;
    .width      80;
    .height     15;
    .maxchars   2000;
    .multiline  true;
    .content    "You can enter a lot of rows in this field.";
  }
}
```



You can enter a lot of
rows in this field.

Figure 8: *Multi-line Edittext*

12 filereq (File Requester, File Dialogs)

The object ***filereq*** (file requester) is a modal file selection window. It distinguishes three modes:

- choosing a file to open
- specification for saving a file and
- choosing a directory.

In addition, selectable files and directories can be restricted by a pattern, or the selection can be limited to existing files or directories.

Definition

```
{ export | reexport } { model } filereq { <Identifier> }
{
  <object-specific attributes>
}
```

Events

extevent

Children

document

record

transformer

Parents

dialog

module

Menu

None

12.1 Attributes

bgc

directory

changedir

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[**I**]

extension

external

external[**I**]

fgc

firstrecord

font

groupbox

help

label

lastrecord

mapped

model

module

multisel

mustexist

navigable

notepage

options[enum]

parent

pattern

record[**I**]

recordcount

scope

self

startsel

statushelp

style

text[enum]

title

userdata

value

value[I]

window

12.2 Specific Attributes

Please also take note of the subsequent chapters, especially chapter "Description of the filereq Object".

Attribute	Description
changedir	Controls whether the attribute *.directory* shall be set to the directory that last had been used in a file dialog in which the user actually did a selection.
directory	Initial directory when the file dialog is opened with the querybox function.
extension	Defines a file extension that is appended to file names without an extension.
multisel	Controls whether multiple files may be selected (Microsoft Windows only; with the styles *fr_load* and *fr_save* only).
mustexist	Ensures that only existing files or directories can be selected; otherwise paths and file names are not checked.
navigable	Sets if the object can receive the focus and therefore be navigated by keyboard.
options[enum]	Options of the filereq object; please refer to the "Attribute Reference" for details. Allowed indexes: *fro_createprompt*, *fro_overwriteprompt*, *fro_relativepath*
pattern	Definition of a pattern for file and directory names to restrict which files or directories are selectable.
startsel	With the value *true*, the path from the attribute *.value* is taken as initial value. The default value is *false*.
style	Sets the filereq object to a certain mode so that it can be used to pick a file, save a file or choose a directory. Possible values are: *fr_load*, *fr_save* and *fr_directory*

Attribute	Description
text[enum]	Sets the labels for several dialog elements (Motif only).
title	Text displayed in the titlebar of the file dialog.
value	Contains the full path of the file or directory that the user has selected.
value[I]	List with the paths of the selected files or directories when multi-selection was turned on.

12.3 Description of the ***filereq*** Object

The ***filereq*** object has no attribute *.visible*. File dialogs can be opened in the Rule Language with the built-in function **querybox()** and in the application with the interface functions **DM_QueryBox()** or **DMcob_QueryBox()**.

The start-up position of the file dialog can be influenced through the parent parameter of these functions. The table below shows the effect of the parent parameter on different window systems.

Parent Parameter	Start-up Position of the File Dialog		
	Motif	Microsoft Windows	
		File Selection	Directory Selection
Not specified or Null	Upper left corner of the desktop	Upper left corner of the desktop when no application window is visible, or in the upper left corner of the last application window	Upper left corner of the desktop
Window Object	Centered in the window	Upper left corner of the desktop	

During selection, no input is possible in other windows of the application. The user can navigate through the directory tree and select files with the mouse or keyboard. The file dialogs can be closed by aborting the selection with the “Cancel” Button, confirming it with the “OK” button or selecting by double clicking the left mouse button. The corresponding return values are:

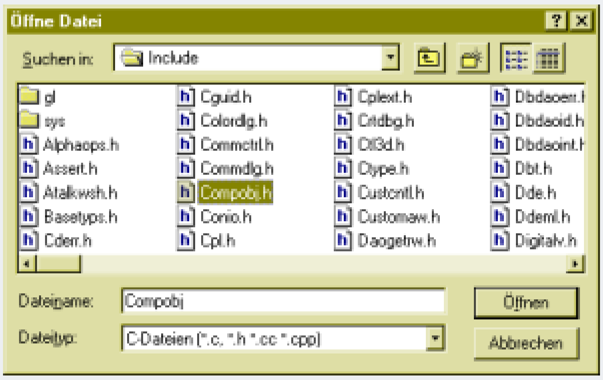
- *button_ok*
  The user actually selected something. The full paths of the selected files or directories are contained in *.value* in case of single-selection or *.value[]* in case of multi-selection. If the attribute *.changedir* had been set to *true*, the attribute *.directory* contains the path to the parent directory of the selected files or directories.
- *button_cancel*
  The user aborted the selection.

- *nobutton*
  An error occurred that prevented the file dialog from being opened.

The selection options for the user can be limited by indicating an initial directory in the *.directory* attribute and by a pattern given in the *.pattern* attribute that determines which items are displayed and selectable by the user. The file extension set in the *.extension* attribute will be appended to the input name if the name does not already have an extension.

Depending on the window system and the mode specified in the .style attribute, the file dialogs appear differently:

Microsoft Windows



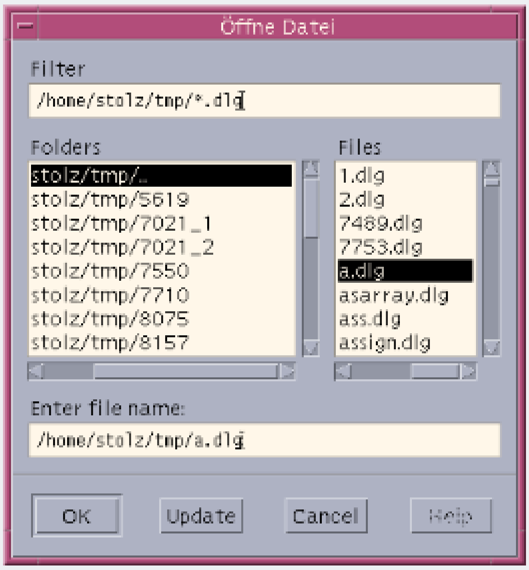
Suchen in: Include

Dateiname: Compobj

Dateityp: C-Dateien (*.c, *.h *.cc *.cpp)

***Figure 9:** Selecting a file to be loaded*

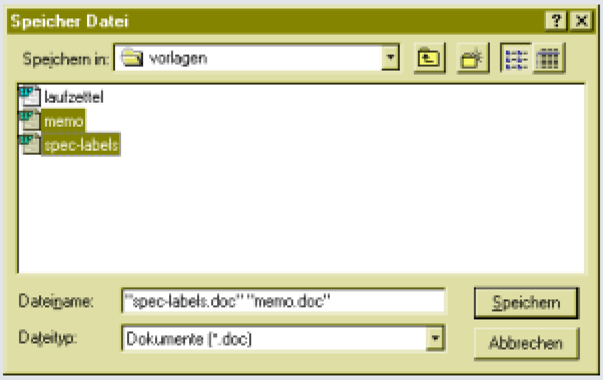
Motif



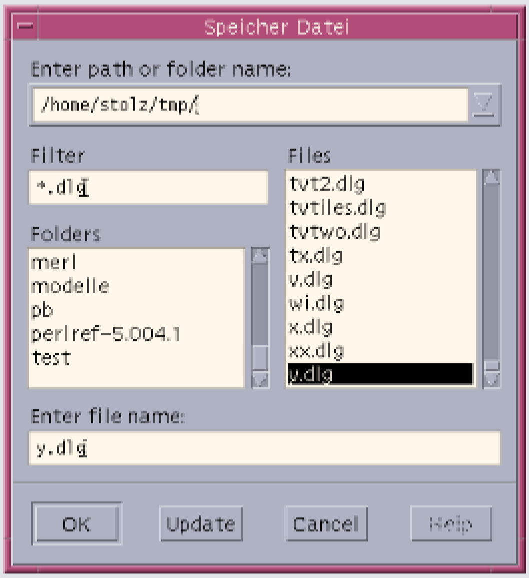
Filter
/home/stolz/tmp/*.dlg

Enter file name:
/home/stolz/tmp/a.dlg

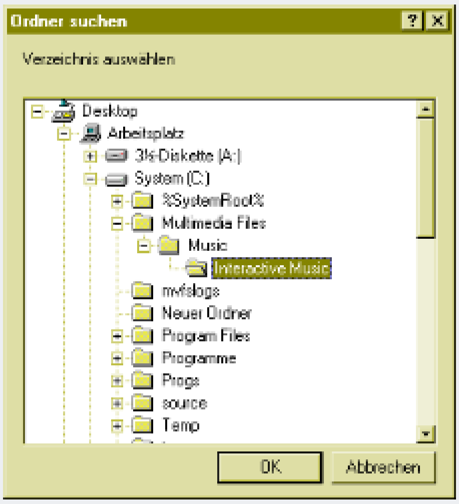
***Figure 10:** Selecting a file to be opened*

<table>
<tr><th>Microsoft Windows</th><th>Motif</th></tr>
<tr><td>


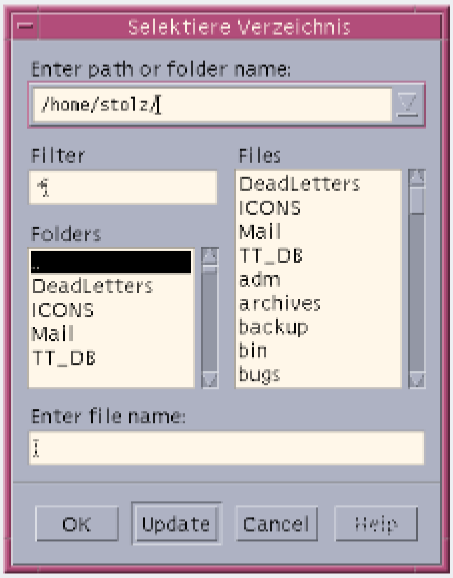
vorlagen
laufzettel
memo
spec-labels
"spec-labels.doc" "memo.doc"
Dokumente (*.doc)

Figure 11: Selecting files to be saved (with multi-selection)
</td><td>


/home/stolz/tmp/
*.dlg
tvt2.dlg
tvtiles.dlg
tvtwo.dlg
tx.dlg
v.dlg
wi.dlg
x.dlg
xx.dlg
y.dlg
merl
modelle
pb
perlref-5.004.1
test
y.dlg

Figure 12: Selecting a file to be saved (relative path, Motif 2.1)
</td></tr>
<tr><td>


Desktop
Arbeitsplatz
3½-Diskette (A:)
System (C:)
%SystemRoot%
Multimedia Files
Music
Interactive Music
mvfslogs
Neuer Ordner
Program Files
Programme
Progs
source
Temp

Figure 13: Selecting a directory
</td><td>


/home/stolz/
*
.
DeadLetters
ICONS
Mail
TT_DB
DeadLetters
ICONS
Mail
TT_DB
adm
archives
backup
bin
bugs

Figure 14: Selection of a directory
</td></tr>
</table>

Some attributes (*.options[enum]*, *.multisel*, *.text[enum]*) enable the use of system-specific functions, offer additional functionality, or configure the selection. They are not available on all platforms or have no effect on some platforms.

12.4 Notes

The ***filereq*** object is mapped to the file and directory dialogs offered by the systems. Therefore there are different appearances and feature sets available on the different systems. On the other hand, one automatically benefits from improvements of the systems.

The feature to provide a “Help” button, which all systems would offer, is not supported.

The ***filereq*** object does not affect the current working directory of the application.

12.4.1 Motif

Most of the appearance (font, color, texts) is controlled by the Window Manager; only the font and some texts can be overwritten by the application.

The same dialog is used for all modes of the filereq object. Therefore its concrete function must be conveyed by the dialog title and the labels of dialog elements. Some labels in the file dialogs can be defined through the *.text[enum]* attribute of the filereq object.

The IDM assumes checking the existence of files and directories and attaching a file extension if it is not present.

12.4.2 Microsoft Windows

The appearance (font, color, texts) is completely controlled by Windows system settings with the exception of the dialog title.

Multi-selection is provided for choosing files and directories. Clicking the mouse button while holding down `Shift`, `Ctrl`, or both, allows for selecting a range of items or toggling the activation of items.

Selection of directories is, in contrast to Motif, limited to existing directories and does not offer the possibility to use patterns. Additionally in IDM versions below A.05.01.e, giving an initial directory limits the selection to the tree beneath this directory. This restriction has been omitted since version A.05.01.e.

Another feature of Microsoft Windows is that the user is prompted with confirmation dialogs before files are created or overwritten.

12.4.3 How to Increase Portability

To achieve a high level of consistency for functionality and appearance, adhere to the following rules:

- Do not use multi-selection.
- Set the *.title* attribute to a value that clearly describes the action (open, save, select…).
- For consistency of language, the language of your texts must comply with the system language.
- Use various patterns (attribute *.pattern*). Avoid using patterns for directory selection.
- Use the directory selection without an initial directory.
- Unix and Microsoft Windows use different path separators. This should be taken into account when processing paths.
- The attribute *.opsys_type* of the setup object can be used to query the system.

12.5 Example

A file dialog can be displayed like this:

```
!! Example for Microsoft Windows
filereq Fr
{
  .style fr_load;
  .title "Show GIF image";
  .directory "c:\";  /* Motif: "/" */
  .pattern "GIF File\t*.gif\tAll\t*.*";  /* Motif: "*.gif"; */
  .extension "gif";
}
rule void OpenFile()
{
  if button_ok = querybox(Fr) then
    ViewGif(Fr.value);
  endif
}
```

13 groupbox

The ***groupbox*** is an auxiliary object which defines a subordinate logical structure in the object hierarchy. All objects defined in a groupbox are on the same logical level. The ***groupbox*** has to be used if several groups of radiobuttons are to be realized in one window. Moreover the ***groupbox*** can always be used if objects of any kind are to be grouped.

Definition

```
{ export | reexport } { model } groupbox { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <grid attributes>
  <hierarchy attributes>
  <layout attributes>
  <scrollbar attributes>
  <object-specific attributes>
}
```

Events

extevent

help

hscroll

key

paste

scroll

select

vscroll

Children

canvas

checkbox

document

edittext

groupbox

image

layoutbox

listbox

notebook

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

tablefield

transformer

treeview

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

13.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation

Attribute	RLD	PID	Properties	Short Description
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for Microsoft UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
bgc	identifier	color	S,G/D/C	background color
bordercolor	identifier	color	S,G/D/C	border color
borderraster	boolean	boolean	S,G/D/-	controls computation of geometry with active grid
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
borderwidth	integer	integer	S,G/D/C	border width
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class/id of object
control	identifier	instance	-,G/-/-	control currently belonging to object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document

Attribute	RLD	PID	Properties	Short Description
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
focus_on_click	boolean	boolean	S,G/-/-	defines if a mouse click into an object's client area activates the object
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
hsb_arrows	boolean	boolean	S,G/-/-	defines whether horizontal scrollbar has arrows at its end
hsb_linemotion	integer	integer	S,G/D/C	horizontal scroll value for scrolling line by line
hsb_optional	boolean	boolean	S,G/D/C	horizontal scrollbar is only displayed if necessary
hsb_pagemotion	integer	integer	S,G/D/C	horizontal scroll value for scrolling page by page
hsb_visible	boolean	boolean	S,G/D/C	visibility of horizontal scrollbar

Attribute	RLD	PID	Properties	Short Description
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_xraster	integer	integer	-,G/-/-	width of the grid internally used

Attribute	RLD	PID	Properties	Short Description
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
real_yraster	integer	integer	-,G/-/-	height of the grid internally used
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
reffont	identifier	font	S,G/D/C	reference font of object
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
tile	object	tile	S,G/D/C	tile resource used as background image
tilestyle	enum	enum	S,G/D/C	controls how the background image set in *.tile* is arranged
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
vheight	integer	integer	S,G/D/C	internal (virtual) height
visible	boolean	boolean	S,G/D/C	visibility of object

Attribute	RLD	PID	Properties	Short Description
vsb_arrows	boolean	boolean	S,G/-/-	defines whether vertical scrollbar has arrows at its end
vsb_linemotion	integer	integer	S,G/D/C	vertical scroll value for scrolling line by line
vsb_optional	boolean	boolean	S,G/D/C	vertical scrollbar will be displayed, if necessary
vsb_pagemotion	integer	integer	S,G/D/C	vertical scroll value for scrolling page by page
vsb_visible	boolean	boolean	S,G/D/C	visibility of vertical scroll-bar
vwidth	integer	integer	S,G/D/C	internal (virtual) width of object
width	integer	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xorigin	integer	integer	S,G/D/C	shift of the origin along the x-axis in objects with scrollbars
xraster	integer	integer	S,G/D/C	units in x-direction
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom

Attribute	RLD	PID	Properties	Short Description
yorigin	integer	integer	S,G/D/C	shift of the origin along the y-axis in objects with scrollbars
yraster	integer	integer	S,G/D/C	units in y-direction
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

13.2 Specific Attributes

Standard Attributes

The attribute *.font* defines the font to be used by objects contained in the groupbox.

Geometry Attributes

The attributes *.vwidth* and *.vheight* define the internal object size.

The attributes *.xorigin* and *.yorigin* define the origin shift. If these values are set to 0, the origin will be located in the top left corner of the groupbox. If the values are not 0, they specify the positions of the scrollbar sliders.

The grid attributes *.xraster*, *.yraster* and *.reffont* define the unit which is used for the size and position of the objects in the groupbox.

Layout Attributes

.bordercolor defines the color of the groupbox border.

Scrollbar Attributes

The scrollbar attributes *.hsb_...* and *.vsb_...* define whether the groupbox is to be provided with scrollbars.

The groupbox can have the following scrollbar attributes:

- .hsb_linemotion
- .hsb_optional
- .hsb_pagemotion
- .hsb_visible
- .vsb_linemotion
- .vsb_optional
- .vsb_pagemotion
- .vsb_visible

Operation mode of scrollbars in windows and groupboxes:

The visibility of scrollbars is influenced by the attributes

- .hsb/vsb_optional
- .hsb/vsb_visible

and

- .width/height
- .vwidth/vheight

The following rules are valid:

- A groupbox and a window can have horizontal scrollbars only if the virtual size *.vwidth* is set.
- A groupbox and a window can have vertical scrollbars only if the virtual size *.vheight* is set.

If no scrollbars are set, the virtual size will be ignored.

- The attributes *.hsb/vsb_visible* and *.hsb/vsb_optional* control whether the scrollbars are to be visible permanently or only if necessary.
- As mentioned above, scrollbars require the setting of the virtual size, i.e., if no virtual size is set, there will not be any scrollbars available:

If the scrollbars are visible,

- and *.hsb/vsb_optional = true*
  => the scrollbar is visible only if necessary.
- and *.hsb/vsb_optional = false*
  => the scrollbar is constantly visible.

Note

.hsb/vsb_optional is ignored by the Motif version, i.e. *.hsb/vsb_optional* implicitly will be always set to true.

To query or to change the scrollbar slider position, the attributes *.xorigin* and *.yorigin* have to be used. These attributes specify the shift of the object origin.

The attributes *.pagemotion* and *.linemotion* define the shift (in pixels) of object contents during the scrollbar action. Value *0* means that the system default value is to be used; for *.pagemotion* the scrolling is made page by page.

13.3 Example

The following example describes two groups of radiobuttons in one window.

```
window Radio_Window
{
  .title "radio window";
  ...
  child groupbox group_1
  {
    child radiobutton First
    { ...
    }
    child radiobutton Second
    { ...
    }
    child radiobutton Third
    { ...
    }
  }
  child groupbox group_2
  {
    child radiobutton Four
    { ...
    }
    child radiobutton Five
    { ...
    }
    child radiobutton Six
    { ...
    }
  }
}
```

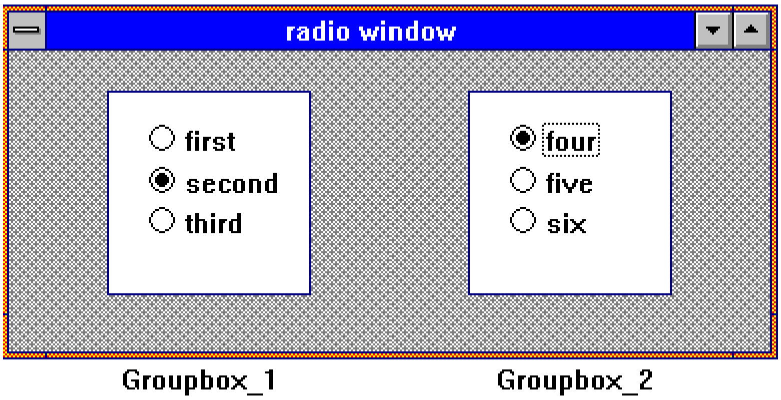


Figure 15: Groupbox

14 image

This object is used to present images or charts. These images may be defined directly in the DM file or they may be imported from an external file.

Definition

```
{ export | reexport } { model } image { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

cut

dbselect

extevent

focus

help

key

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

statusbar

toolbar

window

Menu

Pop-up Menu

14.1 attributes

acc_label

acc_text

accelerator

.active

.alignment

bgc

borderstyle

borderwidth

class

control

cursor

cut_pending

cut_pending_changed

dialog

document[I]

external

external[I]

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

14.2 Specificattributes

attribute	Description
.active	Indicates the active/inactive states. See also chapter “Image display and mouseover events”.
.alignment	Horizontal presentation of the text. Default: *0*. (See also spacing and in particular the description in the „Attributreferenz“ zu tilestyle).
.borderstyle	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a) Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
.focus_on_click	Determines whether the object gets the focus by mouse click.

attribute	Description
.height	Height of the object. At value *0* the DM automatically calculates the correct object size for the object.
.imagebgc	Background color of the image. Evaluated only if the displayed image is not loaded from an external file.
.imagefgc	Foreground color of the image. Evaluated only if the displayed image is not loaded from an external file.
.index	Index of the object in the child vector of the parent object.
.mouseover	Determines whether the object responds to mouseover events. See also chapter “Image display and mouseover events”.
.options[enum]	Options of the object. Index: *opt_center_toolhelp* (only IDM FOR WINDOWS).
.picture	Specifies the tile pattern or image of the object. See also chapter “Image display and mouseover events”.
.picture[enum]	Specifies the tile pattern or image of the object for the different states. See also chapter “Image display and mouseover events”.
.spacing	Space between image and text. (See also alignment and in particular the description in the „Attributreferenz“ for tilestyle).
.style	Defines whether the object can represent the active/inactive states by different images or not. See also chapter “Image display and mouseover events”. In addition, under Windows you can specify whether the image should behave as a menu (see also chapter “Usage as menuitem replacement”).
.text	Specifies the text associated with the object. It will be displayed below the image.
.tilestyle	Position of the image in the object. Please also note detailed description „Attributreferenz“ for tilestyle and attributes alignment and spacing.
.width	Width of the object. At value *0* the DM automatically calculates the correct object size for the object.
.xmargin	Vertical distance between content and margin. See also detailed description „Attributreferenz“ for xmargin.

attribute	Description
.ymargin	Horizontal distance between content and border. See also detailed description at „Attributreferenz“ for ymargin.

14.2.1 Combination of the layout attributes

The attributes .xmargin and .ymargin define the distances between image border and display area (border shown dashed in the following image). The attribute .spacing defines the distance between tile and text within the display area of the image.

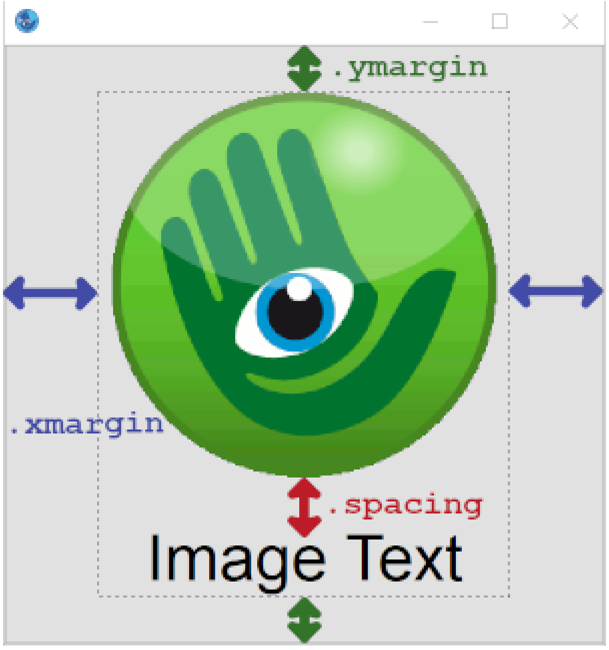


Figure 16: Image with text

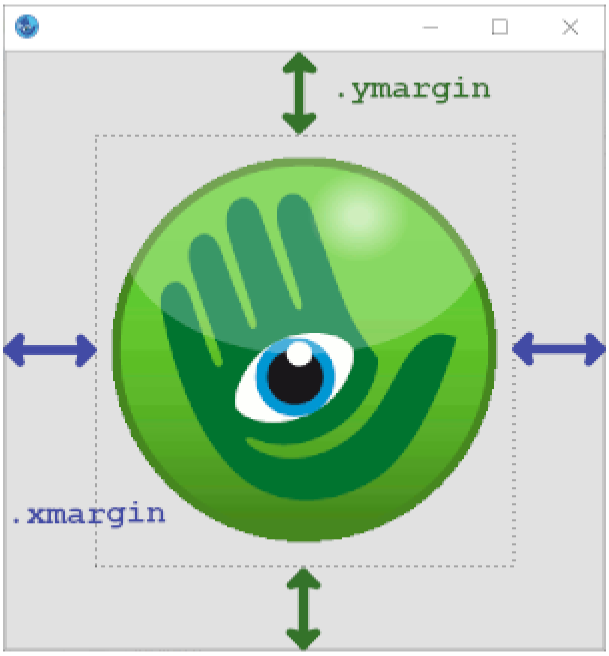


Figure 17: Image without text

The following is an example of the effect of some attributes (the tile has the .scalestyle *propscale*):

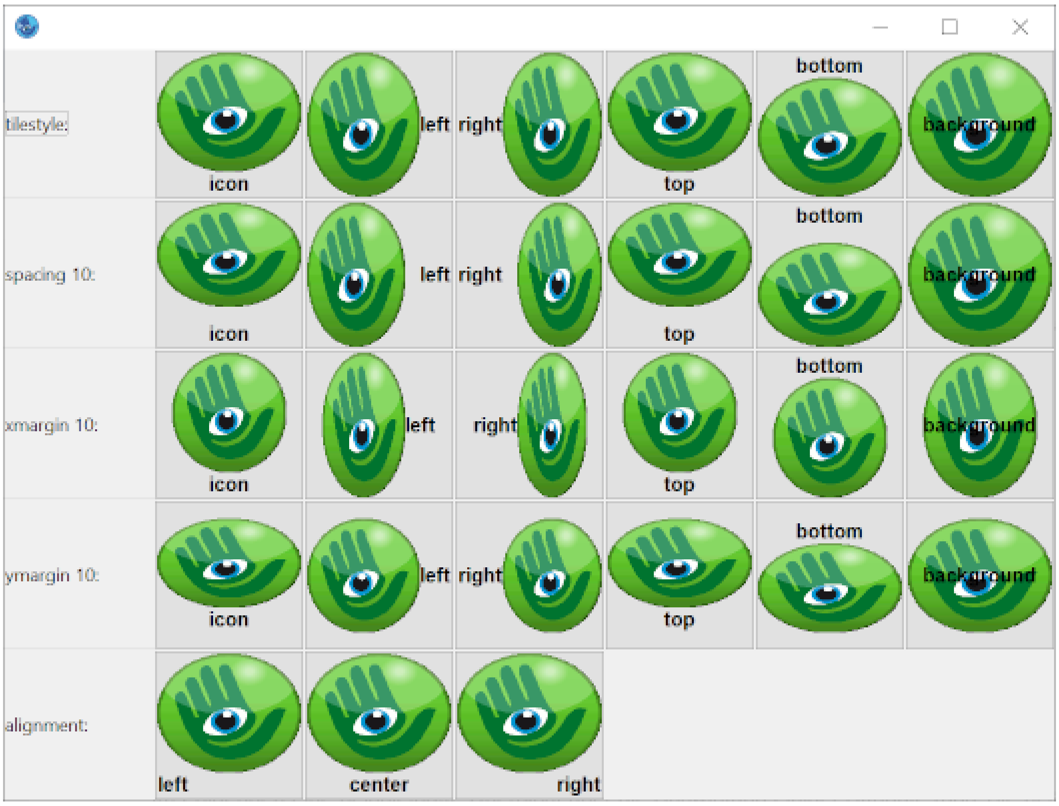


Figure 18: effects of attributes

See alsochaper„HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

14.2.2 Image display and mouseover events

The image to be displayed (e.g. a tile resource) is usually defined in the picture attribute.

If the picture object is to display different pictures in different states (active, inactive...) (see attributes .style and .active) or react to mouseover, the necessary settings must be made in the attributes .picture[enum] or mouseover (see „Attributreferenz").

14.2.3 Usage as *menuitem* replacement

The ***image*** object can be used with a menu box style under Microsoft Windows. To do this, the attribute .style must be given the value *menubox*. Menus similar to Microsoft Office can then be designed. It makes sense to define all ***image*** objects with *.style* = *menubox* as children of a ***toolbar*** object. Furthermore, such a window should not have any other menus.

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;
}
```

14.3 Example

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

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

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

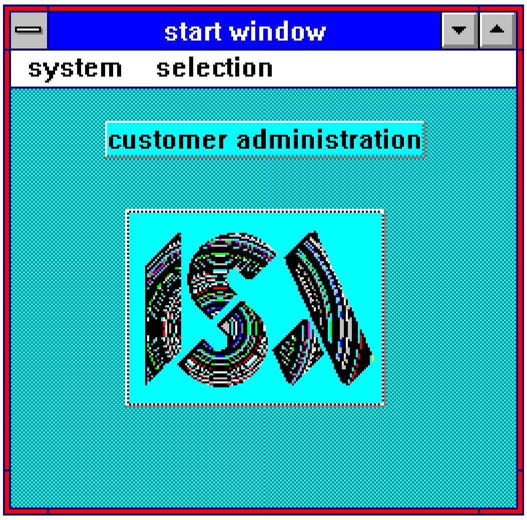


Figure 19: Bild

15 import

The use of a module in another module or dialog is started with the keyword **import**.

The keyword is followed by the logical name of the module. The logical name usually is not the name of the module itself. Sometimes it is even possible to use one and the same module with different logical names in a module although this can be described in a better and more elegant way by using models. The logical name of the imported module is followed by a string bearing the name of the interface file.

Definition

```
{ export | reexport } import { <Identifier> } "<filename>"
{
  object-specific attributes>
}
```

See also

For further information please read about the object ***module*** or refer to chapter “Modularization” of manual “Programming Techniques”.

Events

none

Children

document

record

transformer

Parent

dialog

module

Menu

none

15.1 Attributes

Attribute	RLD	PID	Properties	Short Description
application	application	application	S,G/D/C	functions of the module to be loaded are to be linked to this application
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
firstrecord	object	record	S,G/-/C	accesses the first record of an object
index	integer	integer	-,G/-/-	current index of object in the children list of its parent
lastrecord	object	record	S,G/-/C	accesses the last record of an object
load	boolean	boolean	S,G/D/C	loading state of the module
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
static	boolean	boolean	S,G/D/C	module can be unloaded

15.2 Specific Attributes

A module can be loaded into another module in different ways. Loading here implies that not only the object names but also their definitions are known to the module and can thus be displayed and changed. Internally there are three different kinds of loading some of which are already determined by the objects defined or exported in the module.

- **load on use**
  The module is loaded immediately. You cannot influence the time when the module is loaded, as objects from the module are needed immediately (default); others are needed when superordinate modules are read. This applies when the module contains exported necessary models and resources.
- **implicit load**
  The module will be loaded only if an exported object out of the module is really needed, e.g. if the object is accessed in a rule by reference or query.
- **explicit load**
  In this case you yourself decide whether you want the module to be loaded. For this purpose you have to set the attribute *.load* to *true* at the logical module name (import name). The module will then be loaded and made available by the Dialog Manager.

Example

```
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 is preferred to the other loading types. **Implicit load** is preferred to **explicit load**. In the example above **load on use** is carried out, i.e. the other types have no effect any longer as the module is already in the memory and the rules can be executed directly.

Explicit load can be specified as attribute already at the import.

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

15.3 Example

```
!! Interfacefile: module.if
module Module
```

```
export import StandardColor "color.if";
...
import Colors "PRIVLIB:mycolor.if";
  // The file mycolor.if is only searched
  // in the path PRIVLIB
```

16 layoutbox

The ***layoutbox*** is a grouping object in which the objects and user interface components that are to be arranged can be thrown into. They are then automatically arranged.

Definition

```
{export | reexport} { model } layoutbox { <Identifier> }
{
  <standard attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

dbselect

extevent

help

hscroll

key

paste

scroll

select

vscroll

Children

canvas

checkbox

control

document

edittext

groupbox

image

layoutbox

listbox

notebook

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

tablefield

transformer

treeview

Parents

control

dialog

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menus

Pop-up menu

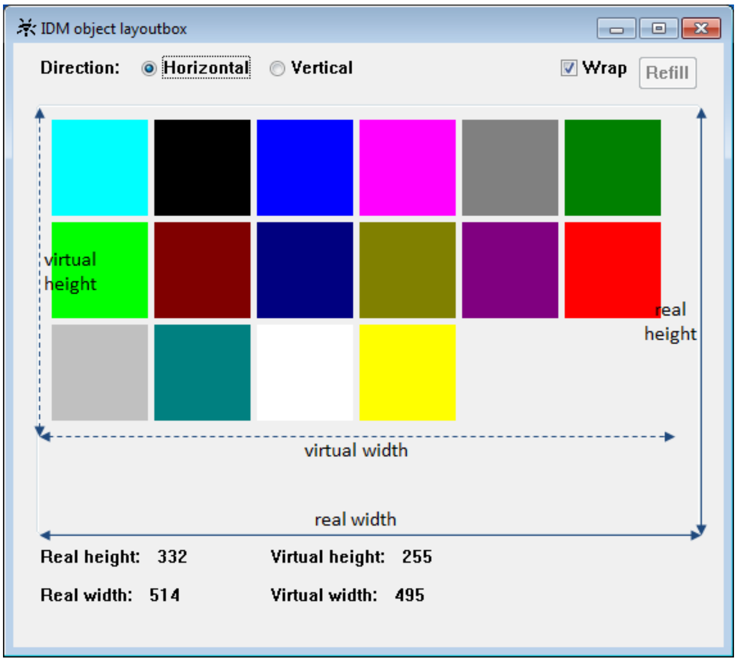
16.1 Using the layoutbox

Often one is faced with the problem that objects or user interface components must be arranged. It is possible to do this by hand, however it is laborious and error-prone when some of the objects change in position or size.

The layoutbox can be of help in such situations. It is a grouping object in which the objects and user interface components that are to be arranged can be thrown into. They are automatically arranged without overlapping according to their sequence in the child vector.

The layoutbox tries to arrange the objects one after the other. If an object does not fit into a row or column anymore, it will be automatically placed in the next one.

In addition, it is ensured that all objects are accessible, i.e. when not all objects can be placed in the visible area (e.g. because the parent window is too small) scrollbars are displayed. The virtual sizes of layoutboxes are always set, provided that *.wrap* is *true*. They cannot be influenced but be queried to determine the size of the working area. The virtual sizes can be smaller or larger than the real sizes of the layoutbox.



IDM object layoutbox

Direction: ● Horizontal ○ Vertical ☑ Wrap

Real height: 332 Virtual height: 255

Real width: 514 Virtual width: 495

Figure 20: Layoutbox sizes

The column width or row height is always determined by the widest or highest object in the respective column or row. All other objects are aligned accordingly.

When objects are arranged row-by-row, *.yauto* refers to the row height and *.xauto* is irrelevant as objects are arranged side by side with even spacing. Likewise *.xauto* refers to the column width when objects are arranged column-by-column and *.yauto* is irrelevant in this case.

16.2 Attributes

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

document[I]

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

transformer[I]

userdata

vheight

visible

vwidth

width

window

wrap

xauto

xleft

xmargin

xorigin

xraster

xright

xspacing

yauto

ybottom

ymargin

yorigin

yraster

yspacing

ytop

16.3 Specific Attributes

Attribute	Description
borderraster	Determines how geometry is computed when a grid is used. Only effective for *.borderwidth* > *0*.
borderstyle	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
direction	Controls the arrangement of objects within the layoutbox. *.direction* = *2* (default) sets row-by-row arrangement. *.direction* = *1* sets column-by-column arrangement.
mincolwidth	Minimum column width; only effective with column-by-column arrangement (*.direction* = *1*).
minrowheight	Minimum row height; only effective with row-by-row arrangement (*.direction* = *2*).
options[enum]	Options of the layoutbox; please refer to the “Attribute Reference” for details. Allowed indexes: *opt_center_toolhelp* (Microsoft Windows only) *opt_scroll_on_focus* (Motif only)
tile	Sets a background image for the object.
tilestyle	Controls how the background image set in *.tile* is arranged.

Attribute	Description
wrap	Sets, whether the children are rearranged automatically when the layoutbox is resized.
xmargin	Sets the distance of the children from the left and right edges of the layoutbox.
xspacing	Defines the horizontal spacing between children of the layoutbox.
ymargin	Sets the distance of the children from the top and bottom edges of the layoutbox.
yspacing	Defines the vertical spacing between children of the layoutbox.

16.4 Notes

For the children of the ***layoutbox*** the meaning of some attributes changes:

Attribute	Description
.xauto	With column-by-column arrangement the column widths are determined by the widest objects in the respective columns. Objects with *.xauto* = *-1* are aligned on the left side of the column; with *.xauto* = *1* they are aligned on the right side of the column. Objects with *.xauto* = *0* are spread to the column width. If there are only objects with *.xauto* = *0* in a column, the width of the objects is set to *mincolwidth*. With row-by-row arrangement the attribute is ignored.
.yauto	With row-by-row arrangement the row heights are determined by the highest objects in the respective rows. Objects with *.yauto* = *-1* are aligned at the bottom of the row; with *.yauto* = *1* they are aligned at the top of the row. Objects with *.yauto* = *0* are spread to the row height. If there are only objects with *.yauto* = *0* in a row, the height of the objects is set to *minrowheight*. With column-by-column arrangement the attribute is ignored.

16.4.1 Position Raster

As the purpose of the layoutbox is to arrange its children automatically, the kind of arrangement is determined by the layoutbox and not by its child objects. Therefore the attribute *.posraster* is set to *false* on the children. Grid layouts are not supported within a layoutbox and have to be avoided.

16.4.2 Virtual Sizes

The layoutbox computes its virtual width and height itself. Both can be queried and set, but the layoutbox will ignore the settings. The virtual sizes define the working area, i.e. the rectangular area that the children and their outer margins encompass within the layoutbox.

When using a background image with the layoutbox the following should be noted:

For the value *tilestyle_centered* and *tilestyle_stretched* of the *.tilestyle* attribute, the working size of the layoutbox is used for computation; the background image gets centered or stretched with regard to the virtual sizes. If this is not desired, the background image should be set for the parent and *tilestyle_parent_tile* should be set for the layoutbox (possibly insert a groupbox as “intermediate object”).

16.4.3 Scrollbar Attributes

The layoutbox provides scrollbar attributes, but the management and display of scrollbars should be left to the layoutbox. Therefore it is not recommended to use the scrollbar attributes:

- Do not set values for the scrollbar attributes, even if it apparently leads to the desired result.
- Do not rely in your code on the values returned by the scrollbar attributes.

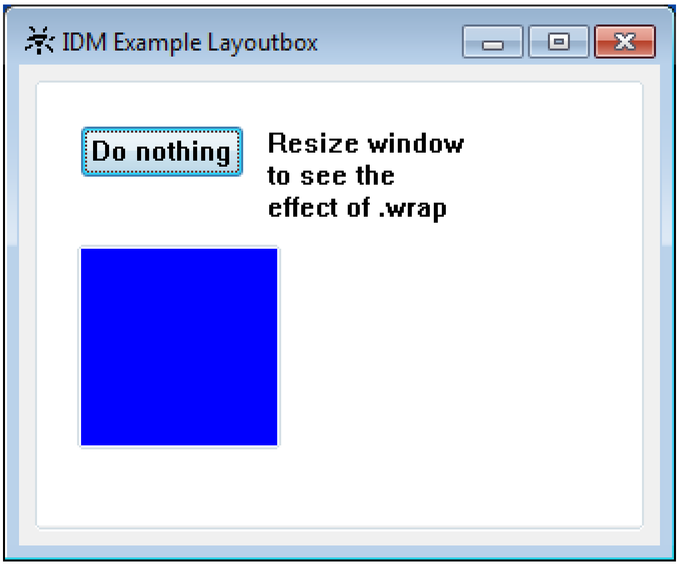
16.5 Example

A simple use of the ***layoutbox*** can look like this:

```
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(); }
 }
```

The example produces this window:



Resize window
to see the
effect of .wrap

***Figure 21:** layoutbox-Beispiel*

17 listbox

The ***listbox*** is a selection mechanism which makes it possible to select one or more element out of a number of text elements by a mouse click.

Definition

```
{ export | reexport } { model } listbox { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

activate

cut

dbselect

deactivate

extevent

focus

help

hscroll

key

paste

scroll

select

vscroll

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

Methods

:delete()

:exchange()

:find()

:insert()

17.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
activeitem	integer	integer	S,G/D/C	active listbox item
active[I]	boolean	boolean	S,G/D/C	selection state of listbox object
bgc	identifier	color	S,G/D/C	background color
bordercolor	identifier	color	S,G/D/C	border color

Attribute	RLD	PID	Properties	Short Description
borderwidth	integer	integer	S,G/D/C	width of object border
class	class	class	-,G/-/-	class/id of object
content[I]	string	string	S,G/D/C	listbox is filled from rules
control	identifier	instance	-,G/-/-	control currently belonging to object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstchar	integer	integer	S,G/D/C	number of the first displayed character (horizontal scroll position).
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
focusitem	integer	integer	S,G/-/C	focusitem in a listbox
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object

Attribute	RLD	PID	Properties	Short Description
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
itemcount	integer	integer	S,G/D/C	number of listbox items
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
multisel	boolean	boolean	S,G/D/C	multiple selection in listbox
nextactive[I]	integer	integer	-,G/-/-	next selected item in listbox
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
picheight	integer	integer	S,G/D/-	height of the space for pictures displayed left of the items

Attribute	RLD	PID	Properties	Short Description
picture[I]	object	tile	S,G/D/C	picture for each item (IDM for Windows and IDM for Qt)
picture_hilite[I]	object	tile	S,G,D,C	picture for each item when the item is selected (IDM for Windows and IDM for Qt)
picwidth	integer	integer	S,G/D/-	width of the space for pictures displayed left of the items
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
selstyle	enum	enum	S,G/-/C	multisel has become superfluous for tablefield and listbox by selstyle
sensitive	boolean	boolean	S,G/D/C	selectivity of object

Attribute	RLD	PID	Properties	Short Description
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
topitem	integer	integer	S,G/D/C	first displayed listbox item
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
userdata[I]	anyvalue	anyvalue	S,G/D/C	userdata of listbox items
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

17.2 Specific Attributes

Layout Attributes

The color of the listbox border can be defined with *.bordercolor*.

Listbox-specific Attributes

In order to realize the data exchange between application and listbox, there are the following additional listbox attributes.

To change the selectivity of individual listbox items and their selection states, the attributes *.active* and *.sensitive* can be indexed. If *.sensitive* is used without an index the entire listbox will be affected; *.active* is not allowed without an index in the listbox.

Example

```
listbox.multisel    :=  true;
listbox.active[i]   :=  false;
```

These commands deselect the listbox item.

Note

The attribute *.sensitive[index]* is ignored in the Motif version. If the attributes *.active* and *.sensitive* are not explicitly specified, the default for *.active* is false, the default for *.sensitive* is true .

The number of texts to be displayed in the listbox can be specified with *.itemcount*.

17.2.1 Selection Control in Listbox Object

The following values are valid for the attribute *.selstyle*:

single	maximal one item is active at the same time
multiple	compatible to *.multisel* = *true*
extended	see description below

The element onto which you click is the current element.

Action	Selection
“Click”	delete previous selection, activate current element
`Shift` + “Click”	maintain previous selection, activate area from last “Click” without `Shift` up to current element
`Ctrl` + “Click”	maintain previous selection, toggle activation of current element

Action	Selection

`Ctrl`
+
`Shift`
+ “Click”
	maintain previous selection, area from last “Click” without `Shift` up to current element maintains the activation state of the first one

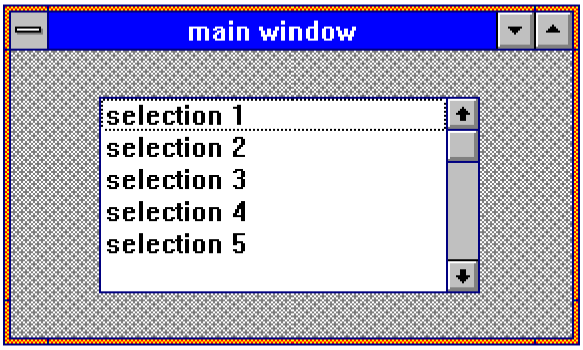
Interaction of *.selstyle* with attribute *.selection[]* which is used to select a selection type in the ***tablefield***.

Selection Type	Current Element
sel_single	a single element
sel_header	a header element
sel_row	a row
sel_column	a column

Please note that a single-column ***tablefield*** behaves like a ***listbox*** with *.selstylesingle* and *extended*.

17.3 Example

```
window main
{
  child listbox selection
  {
    .xleft       10;
    .ytop        10;
    .xauto       1;
    .yauto       1;
    .width       50;
    .height      100;
    .sensitive   true;
    .multisel    false;
    .itemcount    6;
    .borderwidth  1;
    .content[1]   "selection 1";
    .content[2]   "selection 2";
    .content[3]   "selection 3";
    .content[4]   "selection 4";
    .content[5]   "selection 5";
  }
}
```



selection 1
selection 2
selection 3
selection 4
selection 5

__Figure 22:__ Listbox

18 listview

The ***listview*** (class name ***dmw_listview***) is an advanced list object that supports various display types. The object is known in principle by the “Windows Explorer”, where it is used to display the directories of the file system.

The following figures show the different display types of the ***listview*** object:

1. The icon view displays large icons along with a caption. The first content column is used as the caption.

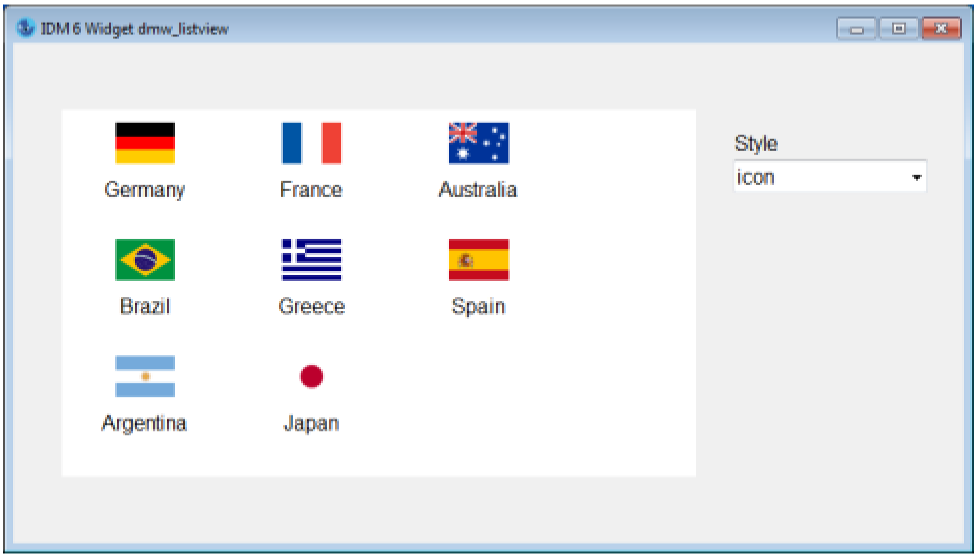
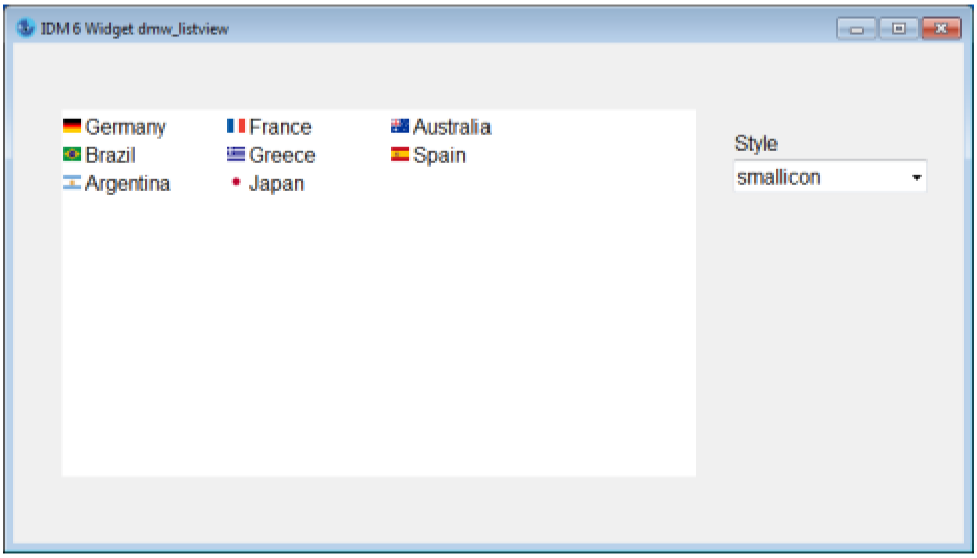


Figure 23: *Icon view with large icons of the listview*

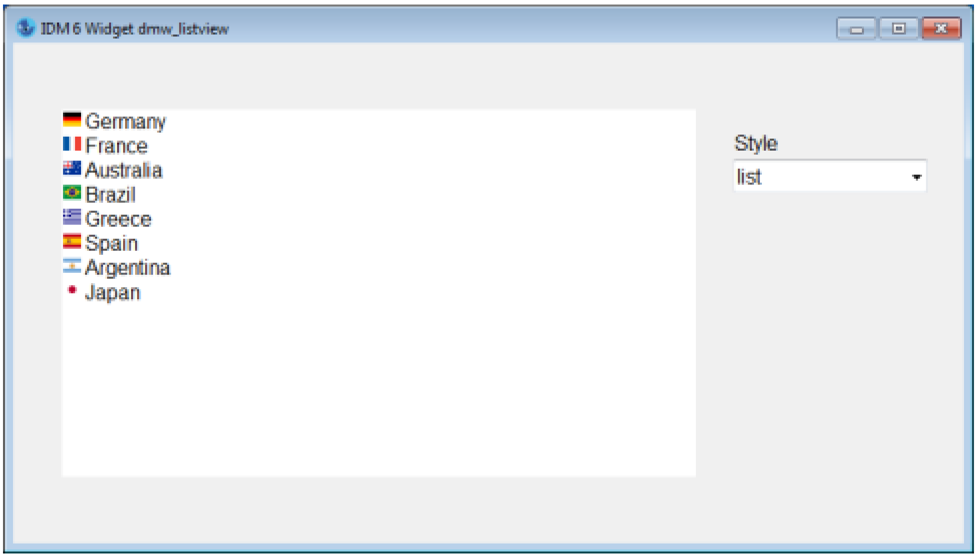
2. The small icon view is very similar to the icon view, the only difference is that small icons are used.



Style: smallicon

Figure 24: Icon view with small icons of the listview

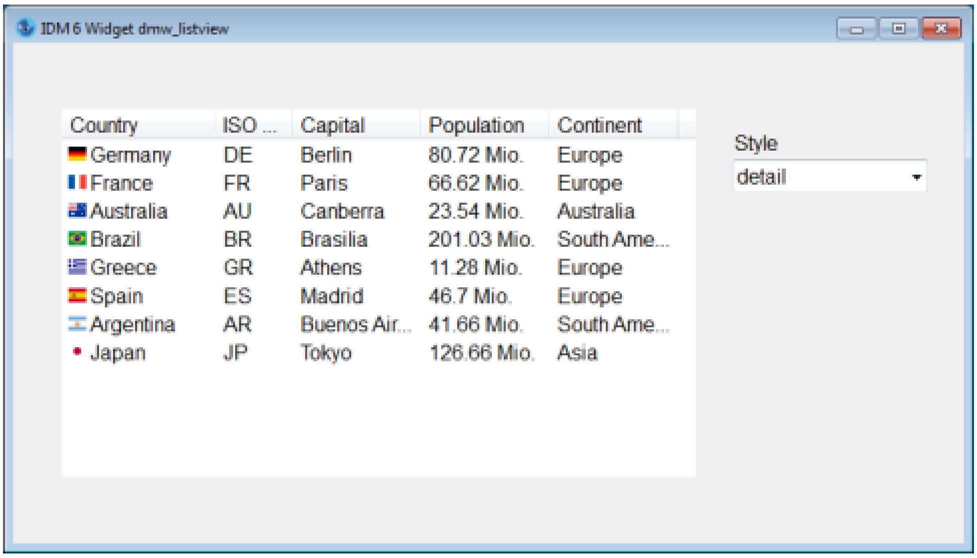
3. The list view also uses the small icons and the caption, but the items are arranged in a list from top to bottom. If necessary, additional columns are added.



Style: list

Figure 25: List view of the listview

4. The detail view displays all available information in a table. Now the ***listview*** object resembles a table.

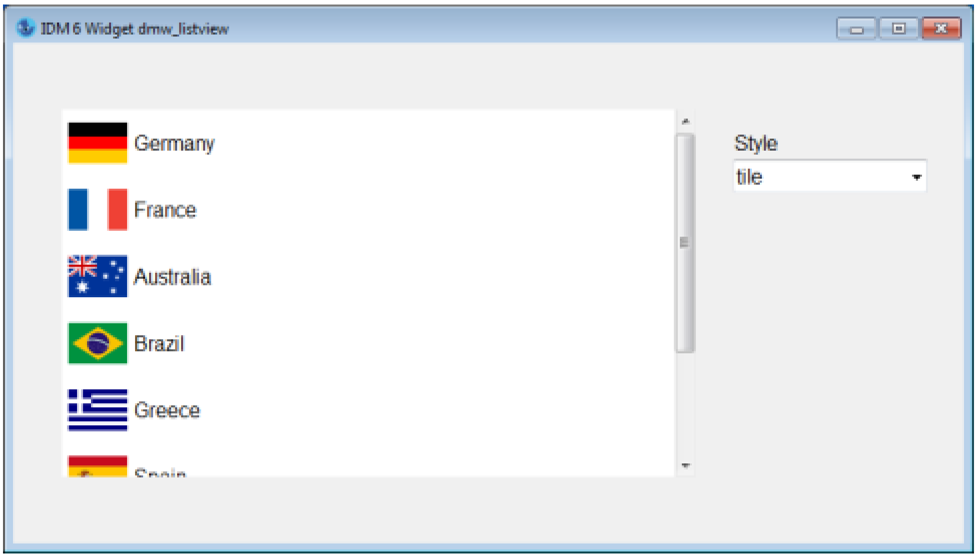


Country	ISO ...	Capital	Population	Continent
Germany	DE	Berlin	80.72 Mio.	Europe
France	FR	Paris	66.62 Mio.	Europe
Australia	AU	Canberra	23.54 Mio.	Australia
Brazil	BR	Brasilia	201.03 Mio.	South Ame...
Greece	GR	Athens	11.28 Mio.	Europe
Spain	ES	Madrid	46.7 Mio.	Europe
Argentina	AR	Buenos Air...	41.66 Mio.	South Ame...
Japan	JP	Tokyo	126.66 Mio.	Asia

Style
detail

__Figure 26:__ Detail view of the listview

5. The tile view is like the list view, but shows large icons. The caption appears next to them.



Germany

France

Australia

Brazil

Greece

Spain

Style
tile

__Figure 27:__ Tile view of the listview

Availability

- Microsoft Windows only.
- Can only be used with the USW option of the ISA Dialog Manager.
- The **idmwidgets.dll** must be in the USW class path (default: *<IDM installation directory>\uswclasses*).

18.1 Definition

```
{ export | reexport } { model } dmw_listview { <Identifier> }
{
  <standard attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events	*activate*	*changed*	*dbselect*
	deactivate	*extevent*	*focus*
	help	*resize*	*select*
Children	None		
Parent	***groupbox***	***layoutbox***	***notepage***
	splitbox	***toolbar***	***window***
Menu	Pop-up Menu		

18.2 Description of Events

18.2.1 activate

The *activate* event is triggered when a list item is selected (*.selected[I] := true*).

The *.index* attribute of ***thisevent*** contains the index of the item that was selected. The data type of *thisevent.index* is *index* and thus consistent with the *select* event. The column value of *thisevent.index* is always *1*, since only this column has a selection status.

18.2.2 dbselect

The *dbselect* event occurs when a double-click is carried out in the ***listview***.

The *.index* attribute of ***thisevent*** contains the index of the item that was clicked. The data type of *thisevent.index* is *index* and thus consistent with the *select* event. If no item was hit, *thisevent.index* is not set.

The header in the detail view (column headings) does not have a double-click event.

18.2.3 deactivate

The *deactivate* event is triggered when a list item loses its selection (*.selected[I] := false*).

The *.index* attribute of ***thisevent*** contains the index of the item that has lost its selection. The data type of *thisevent.index* is *index* and thus consistent with the *select* event. The column value of *thisevent.index* is always *1*, since only this column has a selection status.

18.2.4 resize

The *resize* event occurs when a column width has been changed in the detail view.

The event does **not** arise for columns whose width has been set to *0* and is calculated by the ***listview***.

18.2.5 select

The *select* event occurs when a click is carried out in the ***listview***.

The *.index* attribute of ***thisevent*** contains the index of the item that was clicked. The data type of *thisevent.index* is *index* and thus consistent with the *activate* event. The column value can be different from *1* only in the detail view.

If no item was hit, *thisevent.index* is not set. When the header (column headings) of the detail view is clicked, the row value of *thisevent.index* is *0*.

18.3 Inherited Attributes

Attribute	Data Type	Notes
.accelerator	object	
.bgc	object	
.borderraster	boolean	
.control	object	
.count[attribute]	anyvalue	
.cursor	object	
.cut_pending	boolean	
.cut_pending_changed	boolean	
.dialog	object	
.document[integer]	object	

Attribute	Data Type	Notes
.export	boolean	
.fgc	object	
.firstrecord	object	
.focus	boolean	
.focus_on_click	boolean	
.font	object	
.function	object	
.function[integer]	object	
.groupbox	object	
.height	integer	has to be > *0*
.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	
.real_sensitive	boolean	

Attribute	Data Type	Notes
.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	has to be > *0*
.window	object	
.xauto	integer	
.xleft	integer	
.xright	integer	
.yauto	integer	
.ybottom	integer	
.ytop	integer	

18.4 Specific Attributes

Attribute	Short Description	Data Type	Default	Access		C	I
				get	set		
.colcount	number of columns	*integer*	*1*	x	x	x	x
.coltitle[integer]	column headings	*string*	""	x	x	x	x
.colwidth[integer]	column widths in the detail view	*integer*	*0*	x	x	x	x
.content[index]	list entries, caption in column 1	*string*	""	x	x	x	x
.mincolwidth [integer]	minimum column width in the detail view	*integer*	*0*	x	x	x	x
.picheight	height of the large icons	*integer*	*0*	x	x	x	x
.picture[integer]	large icons	*object* (*tile*)	""	x	x	x	x
.picwidth	width of the large icons	*integer*	*0*	x	x	x	x
.rowcount	number of rows	*integer*	*0*	x	x	x	x
.selected[integer]	selection of the list entries	*boolean*	*false*	x	x	x	x
.smallpicheight	height of the small icons	*integer*	*0*	x	x	x	x
.smallpicture [integer]	small icons	*object* (*tile*)	""	x	x	x	x
.smallpicwidth	width of the small icons	*integer*	*0*	x	x	x	x
.style	presentation mode	*integer* (*0…4*), *string*	*0*	x	x	x	x

C *changed* event on modification

I Attribute is inherited

19 mapping

The purpose of the mapping object is to define a semantic action, which is called during a transformation for a specific node in an XML tree or in an IDM object hierarchy), when a match between the mapping object and the node is found. Mapping objects are defined as children of a transformer object and describe a transformation of data in conjunction with it.

Definition

```
{ export | reexport } { model } mapping { <Identifier> }
{
  [ <atribute definition> ]
  [ <method definition> ]
}
```

Events

None

Children

document

record

transformer

Parents

transformer

Menu

None

Methods

:action()

19.1 Attributes

document[I]

external

external[I]

firstrecord

label

lastrecord

name

model

parent

record[I]

recordcount

scope

transformer[I]

userdata

19.2 Object-specific Attributes

name

Here, a pattern is given (similar to an XPath expression) to define the nodes within an XML tree or an IDM object hierarchy which the mapping object should match. This determines if the :action method should be called for the node during the transformation or not.

19.3 Pattern for .name

A pattern is very similar to an IDM identifier. The pattern represents a path of element names. The path starts at the root of the document or IDM object hierarchy (i.e. dialog or module). Each hierarchy level is compared to the corresponding part of the path. In addition, certain properties of the parent, such as the existence of an attribute or the position within the children or the parent can be given:

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

<Name>

XML: Is compared to the name attribute of the cursor. The cursor must have the node type *node-type_element*. Alternatively, * can be used. In this case the name attribute is ignored.

IDM: Is compared to the label of an IDM object. Alternatively, * can be used, which means that every label matches.

<Attr>

XML: The XML node that the cursor points to must have the indicated attribute.

IDM: The IDM object must have the indicated attribute.

<Op>

XML and **IDM:** Comparison operator which compares the attribute given in <Attr> with <Value>. I can be tested for equality = and inequality <>.

<Value>

XML and **IDM:** The value to which <Attr> is compared.

<Idx>

XML: The XML node must be at this position within the child nodes of the same parent.

IDM: The IDM object must be at this position within the children of the same parent. Here, all children of hierarchical attributes are regarded as combined.

Particularities

.

XML: If the pattern starts with a dot, it is relative to the current node. This means that the path begins at the current node.

..

XML and **IDM:** Two consecutive dots define, that an arbitrary number of hierarchy levels may be skipped.

[<Idx>]

XML: An index, without any further information, selects the XML node at this position. In this case, the type of node is insignificant. Thus each DOM node can be uniquely referenced by an expression like {**[**<Idx>**]**[**.[**<Idx>**]**]} (path attribute).

All comparisons are case sensitive.

Example

The pattern “..CD[.Title = Yellow]” matches all nodes within an XML tree (regardless of their hierarchy level) that have the tag “CD” and an attribute “.Title” with the value *Yellow*.

When the same pattern is applied to IDM objects, all objects possessing the label “CD”, and an user-defined attribute “.Title” with the value *Yellow* will match.

19.4 Object-specific Methods

:action()

This method is invoked from the **:action** method of the parent ***transformer***, when a match between a node in an XML tree or an IDM object hierarchy and the pattern in the *.name* attribute of the ***mapping*** objects is found. The default implementation of this method does nothing other than always returning *true*.

Because this method can be redefined, it is possible for the programmer to determine what should happen with the data from the node.

20 menubox

Most of the objects can be assigned exactly one active menu. Since a menubox is not directly linked to the object as a child, the definition has to be started with **menu** instead of **child**. If the menu belongs to a window, this menu must be linked to the window with **child**.

The actual menu definition then starts with the keyword **menubox** followed by an identifier. The menu-items, the menu separators, and the menu attributes are defined in braces.

Definition

```
{ export | reexport } { model } menubox { <Identifier> }
{
  <standard attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

extevent

open

Children

document

menubox

menuitem

menusep

record

transformer

Parent

canvas

checkbox

dialog

edittext

groupbox

image

layoutbox

listbox

menubox

module

notepage

poptext

pushbutton

radiobutton

rectangle

scrollbar

spinbox

splitbox

statictext

tablefield

toolbar

treeview

window

Menu

none

Note

window is the only object to which more than one menu can be assigned. This menu has to be linked to the window menu bar with the keyword **child**.

The menu objects in windows can be defined with the attribute *.menu*.

With the keyword **child**, an unspecified number of ***menuboxes***, ***menuitems*** and ***menu separators*** can be assigned to one ***menubox***.

20.1 Attributes

Attribute	RLD	PID	Properties	Short Description
bgc	identifier	color	S,G/D/C	background color
child[I]	object	object	S,G/-/C	accesses the I-th child object

Attribute	RLD	PID	Properties	Short Description
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class/id of object
control	object	control	-,G/-/-	control currently belonging to object
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstchild	object	object	S,G/-/C	accesses the first child object
firstmenu	identifier	instance	S,G/-/-	first menu of window
firstrecord	object	record	S,G/-/C	accesses the first record of an object
font	identifier	font	S,G/D/C	object font
groupbox	identifier	instance	-,G/-/-	object groupbox
help	string identifier	string text	S,G/D/C	help text of object
helpmenu	boolean	boolean	-,G/D/C	right-justified menubox
label	string	string	S,G/D/C	name/identifier of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastmenu	identifier	instance	S,G/-/-	last menu of window

Attribute	RLD	PID	Properties	Short Description
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menucount	integer	integer	-,G/D/C	number of menu elements in window
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
order	integer	integer	S,G/D/-	defines the order relative to other menuboxes or menuitems
parent	identifier	instance	S,G/-/-	parent of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object

Attribute	RLD	PID	Properties	Short Description
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
title	string identifier	string text	S,G/D/C	title of entry
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
window	identifier	instance	-,G/-/-	window to which object belongs

20.2 Specific Attributes

The attribute .options can have the following values:

option_index	Meaning
opt_enable_tearoff	You can place this menu as Tearoff menu wherever you like. (on Motif only)

20.3 Popup-Menus with Microsoft Windows

When a context menu is opened, the object on which the context menu (popup menu) is defined receives the keyboard focus. If this object is a grouping object (for example, a window or groupbox object), then the keyboard focus is set to the first child object that can receive the focus. However, keyboard focus is implemented only if the object on which the context menu is defined or one of the direct or indirect child objects does not already have keyboard focus.

Remarks

- Popup menus under Windows are also opened via the system menu key (`Alt` key). If you want to open the window menu directly, you should use the `F10` key.

20.4 Popup-Menus with Motif

Starting with Motif 2.1, it is standard for pop-up menus with mnemonics to open automatically when a mnemonic character is entered. However, for objects with text input, this behavior can interfere with text input.

As of IDM version A.05.02.f4, for ***edittexts*** with a **popup menu** where ***mnemonics*** are defined, the popup menu no longer opens automatically when a mnemonic character is entered.

This makes the edit text behave like other objects **with** text input. For focusable objects without text input, such as pushbutton, checkbox, radiobutton, image, scrollbar and spinbox with statictext, the popup menu is automatically opened when a mnemonic character is entered.

Remarks

- For all object classes, as of Motif 2.1, the popup menu can be opened by default with the key combination `Shift + F10`.
- Accelerators can be used as an alternative to mnemonics.

20.5 Example

Window with menu bar and pop-up menu

```
window  Main_window
{
  child menubox Menu_1
  {
    .title "Start Menu";
  }

  child menubox Menu_2
  {
    .title "File";
  }

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

    child menuitem Menu_3
    {
      .text "Popup 1";
    }

    child menuitem Menu_4
    {
      .text "Popup 2";
    }
  }
}
```

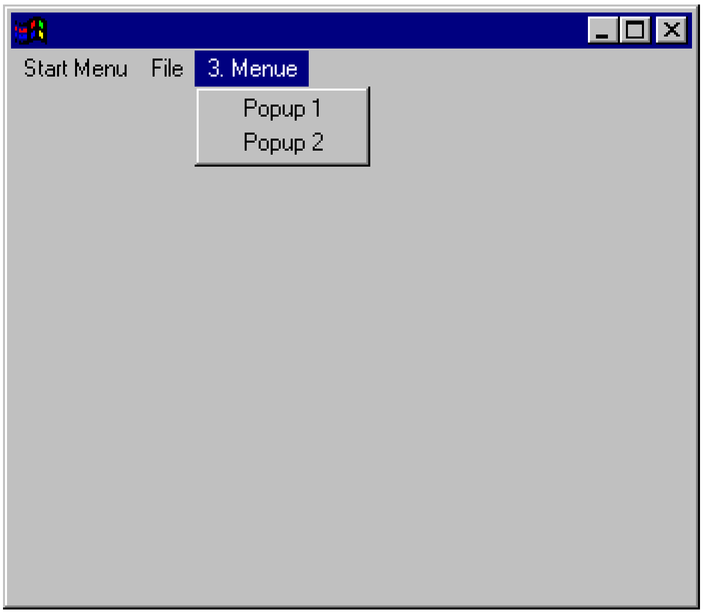


Figure 28: Fenster mit Menüleiste

Note

If a further menu is to be contained in a menu (cascade menu), this menu has to be defined as the child of the first one with the keyword **menubox**.

```
child menubox Pulldown
{
  .title      "Edit file";
  .sensitive  true;
  .visible    true;
  child menuitem Load
  {
    .text     "Load";
    .visible  true;
  }
  child menubox Save  /* Start cascade menu */
  {
    .title  "Save";
    child menuitem As
    {
      .text  "Save as";
    }
    child menuitem Save
    {
      .text  "With current name";
    }
  }
}
```

21 menuitem

An unspecified number of menuitems can be defined in a ***menubox***. The keyword for the definition of a menuitem is **menuitem** followed by an identifier. Object attribute specifications are in braces after that. A menuitem can be defined only as child of a ***menubox*** or of a ***window***.

Definition

```
{ export | reexport } { model } menuitem { <Identifier> }
{
  <standard attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

activate

deactivate

extevent

help

key

select

Children

document

record

transformer

Parent

menubox

module

window

Menu

none

21.1 Attributes

Attribute	RLD	PID	Properties	Short Description
accelerator	identifier	accel	S,G/D/C	accelerator of object
active	boolean	boolean	S,G/D/C	active state of object
bgc	identifier	color	S,G/D/C	background color
class	class	class	-,G/-/-	class/id of object
control	object	control	-,G/-/-	control currently belonging to object
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object

Attribute	RLD	PID	Properties	Short Description
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage of object
options[enum]	boolean	boolean	S,G/D/-	special options of object
order	integer	integer	S,G/D/-	defines the order relative to other menuitems in a menubox
parent	identifier	instance	S,G/-/-	parent of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
style	integer	integer	S,G/D/-	kind of display/of behavior
text	string identifier	string text	S,G/D/C	static text of object

Attribute	RLD	PID	Properties	Short Description
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
window	identifier	instance	-,G/-/-	window to which object belongs

21.2 Specific Attributes

The attribute *.style* defines whether a pushbutton, radiobutton or checkbox menu is to be used.

21.3 Example

two-step menu:

```
window main_window
{
    :
    child menubox    File
    {
        .title                    "file";
        .sensitive             true;
        .visible               true;

        child menuitem    Mi_Open
        {
            .text           "open";
            .sensitive       true;
            .visible           true;
        }
        child menuitem    Mi_Save
        {
            .text                "save";
            .sensitive            true;
            .visible           true;
        }
        child menubox    Mi_Exit
        {
            .title            "exit";
            .sensitive            true;
            .visible            true;

            child menuitem    SSS_Yes
```

```
                {
                    .text       "yes";
                    .sensitive    true;
                    .visible        true;
                }
                child menuitem        SSS_No
                {
                    .text             "no";
                    .sensitive        true;
                    .visible        true;
                }
            }    /*End menu Cancel*/
        }    /*End menubox File*/
    }    /*End window main_window*/
```

The above example generates the following menu:

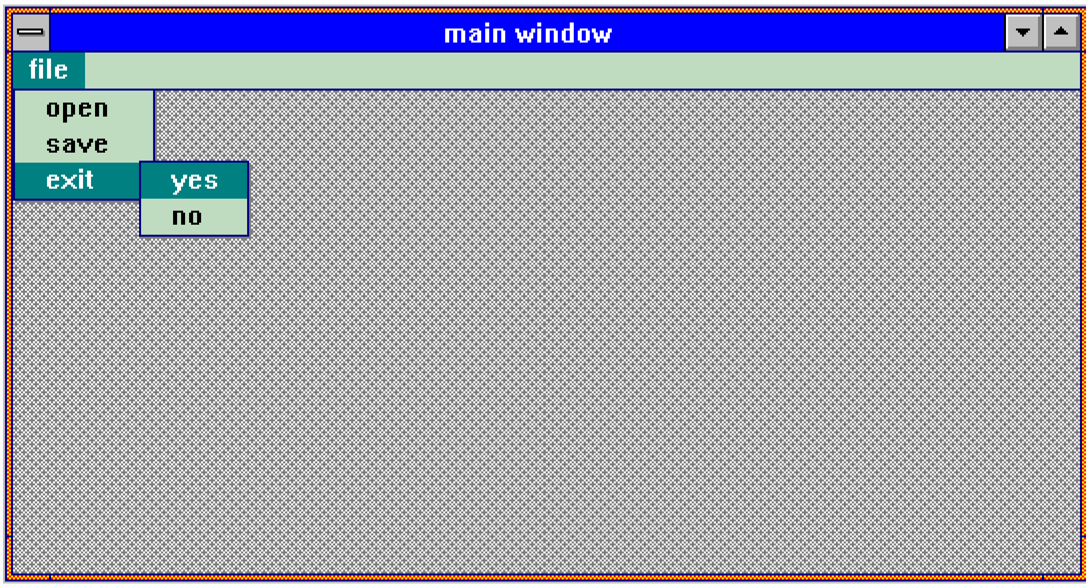


Figure 29: Menu

22 menusep (Menu Separator)

Menu separators are the second type of objects which can be used in a menu. They are defined with the keyword **menusep**.

Definition

```
{ export | reexport } { model } menusep { <Identifier> }
{
  <standard attributes>
  <hierarchy attributes>
  <object-specific attributes>
}
```

Events

extevent

Children

document

record

transformer

Parent

menubox

module

Menu

none

22.1 Attributes

Attribute	RLD	PID	Properties	Short Description
bgc	identifier	color	S,G/D/C	background color
class	class	class	-,G/-/-	class/id of object
control	object	control	-,G/-/-	control currently belonging to object

Attribute	RLD	PID	Properties	Short Description
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
groupbox	identifier	instance	-,G/-/-	object groupbox
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to

Attribute	RLD	PID	Properties	Short Description
order	integer	integer	S,G/D/-	defines the order relative to other menuitems in a menubox
parent	identifier	instance	S,G/-/-	parent of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
style	integer	integer	S,G/D/-	kind of display/of behavior
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
window	identifier	instance	-,G/-/-	window to which object belongs

22.2 Specific Attributes

The attribute *.style* defines whether a single-line or a double-line separator is to be used.

22.3 Example

In the following example, the menuitem “End” is visually separated from other items by a separating line.

```
child menubox       FileWithSeparator
{
  .title            "File";
```

```
    .sensitive      true;
    .visible        true;

    child menuitem  Mi_Open
    {
      .text         "Open";
      .sensitive    true;
      .visible      true;
    }
    child menuitem  Mi_Save
    {
      .text         "Save";
      .sensitive    true;
      .visible      true;
    }
    child menusep   Separator_Line
    {
    }
    child menuitem  Mi_End
    {
      .text         "End";
      .sensitive    true;
      .visible      true;
    }
  }
```

23 messagebox

This object uses the standard messageboxes of the used window system.

A messagebox is opened via the built-in function **querybox()** or the function **DM_QueryBox()**. Please note that the attribute *.visible* is not available for this object!

Definition

```
{ export | reexport } { model } messagebox { <Identifier> }
{
  <object-specific attributes>
}
```

See Also

Built-in function querybox in manual “Rule Language”

C function DM_QueryBox in manual “C Interface - Functions”

Events

extevent

Children

document

record

transformer

Parent

dialog

module

Menu

none

23.1 Attributes

Attribute	RLD	PID	Properties	Short Description
bgc	identifier	color	S,G/D/C	background color

Attribute	RLD	PID	Properties	Short Description
button[1-3]	enum	enum	S,G/D/C	display of buttons for messagebox » button_abort » button_cancel » button_ignore » button_no » button_ok » button_retry » button_yes » nobutton
class	class	class	-,G/-/-	class/id of object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
defbutton	integer	integer	S,G/D/C	default pushbutton in a dialogbox
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
font	identifier	font	S,G/D/C	object font

Attribute	RLD	PID	Properties	Short Description
icon	object	tile	S,G/D/C	assigns object to icon: » icon_asterisk » icon_error » icon_exclamation » icon_hand » icon_information » icon_query » icon_question » icon_warning » noicon
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
model	identifier	instance	S,G/D/C	model belonging to object
record[I]	object	record	S,G/-/C	accesses the **I**-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sysmodal	boolean	boolean	S,G/D/C	specifies whether the ***messagebox*** is displayed in front of all other windows on the desktop
text	string	string	S,G/D/C	static text of object

Attribute	RLD	PID	Properties	Short Description
title	string identifier	string text	S,G/D/C	title of entry

23.2 Specific Attributes

The texts of messagebox buttons can be defined by the attribute *.msgboxtext[I]* for the object dialog.

23.2.1 button[1-3]

The following values are available for the attribute *.button*.

- button_abort
- button_cancel
- button_ignore
- button_no
- button_ok
- button_retry
- button_yes
- nobutton

A button will not be displayed if it is defined as *nobutton*.

A messagebox can display a maximum of three buttons. The assignment of each of these buttons is regulated by the attribute *.button[I]*.

The following button combinations are allowed for the attribute *.button[1-3]*.

.button[1]	.button[2]	.button[3]
button_ok	nobutton	nobutton
button_ok	button_cancel	nobutton
button_cancel	nobutton	nobutton
button_retry	button_abort	nobutton
button_retry	button_abort	button_ignore
button_yes	button_no	nobutton
button_yes	button_no	button_cancel

If an non-existing button combination is used it might be treated as an error by the window system. The messagebox is then not displayed, and the return value is *nobutton*.

Note for DM on Microsoft Windows

The combination "button_cancel - nobutton - nobutton" is not supported by Microsoft Windows.

Warning

Combinations differing from the table above and valid for Motif are not allowed on Microsoft Windows!

Note for DM on Motif

The `Escape` key usually activates a "Cancel" button. However, with Motif 1.1, the second button is always activated. In order to get the regular button behavior it is recommended to define the second button as "Cancel" button.

23.2.2 defbutton

The attribute *.defbutton* sets the default button. If the value is not valid because the button is invisible the window system will set a default button.

The following values are available for the attribute *.icon*.

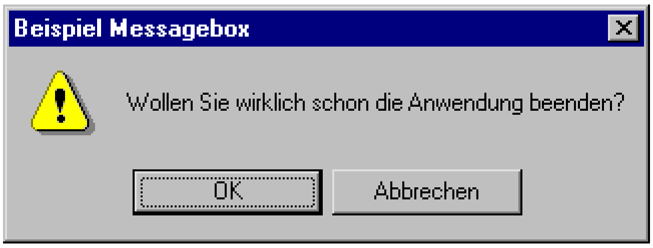
- icon_asterisk
- icon_error
- icon_exclamation
- icon_hand
- icon_information
- icon_query
- icon_question
- icon_warning
- noicon

See Also

Attribute icon

23.3 Example

```
dialog Mess

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



Beispiel Messagebox

Wollen Sie wirklich schon die Anwendung beenden?

Figure 30: *Messagebox*

24 module

A module is a partial dialog within a comprehensive dialog. It is an independent unit and can be compared with the object ***dialog***. It constitutes the brackets for all objects contained because these objects can be saved in a file. The module functionality represented by resources, rules, defaults, models or instances can be defined in a module.

Definition

```
module { <Identifier> }
{
  <standard attributes>
  <hierarchy attributes>
}
```

Such a part of a dialog can for example be used as a library

- for standard resources which are to be implemented in a company (realization of company-specific style guides)
- for models which go beyond the dialog or the project
- for functionality which is needed again and again (e.g. search, calling a database…)

See also

For further information please read about the object import or refer to chapter “Modularization” in manual “Programming Techniques”.

Events

extevent

finish

start

Children

document

import

module

record

transformer

window

Parents

dialog

module

Menu

none

24.1 Attributes

Attribute	RLD	PID	Properties	Short Description
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class / id of object
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
index	integer	integer	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name / identifier of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object

Attribute	RLD	PID	Properties	Short Description
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records

25 notebook

notebook symbolizes a real notebook consisting of different pages, the ***notepages***. It is divided up into different sections which are marked by "tabs".

This object can be used to make certain basic settings for parameters. Moreover, data can be displayed which can be divided into logical groups.

Definition

```
{ export | reexport } { model } notebook  { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

extevent

focus

help

key

paste

Children

document

notepage

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

25.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
activeobject	object	instance	-,G/-/-	relevant notepage on top
alignment	integer (-1, 0, 1)	integer	S,G/D/C	justification of status text of notepage
backpage	enum	enum	S,G/D/C	corner in which visible borders meet
bgc	identifier	color	S,G/D/C	background color of notebook
binding	enum	enum	S,G/D/C	kind of notebook binding, e.g. spiral binding
bordercolor	identifier	color	S,G/D/C	border color of notebook
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class of object
control	identifier	instance	-,G/-/-	control the object currently belongs to
cursor	identifier	cursor	S,G/D/C	cursor belonging to object

Attribute	RLD	PID	Properties	Short Description
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog of object
direction	integer	integer	S,G/D/C	direction in which the notebook is bound
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	text color of status line and tabs
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object

Attribute	RLD	PID	Properties	Short Description
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
majortabheight	integer	integer	S,G/D/C	height of all majortabs (main index)
majortabwidth	integer	integer	S,G/D/C	width of all majortabs (main index)
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-/-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
minortabheight	integer	integer	S,G/D/C	height of all minortabs (side index)
minortabwidth	integer	integer	S,G/D/C	width of all minortabs (side index)
model	identifier	instance	S,G/D/C	model belonging to object
multiline	boolean	boolean	S,G/-/C	multiline edittext
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
picheight (IDM for Windows only)	integer	integer	S,G/D/-	height of the pictures displayed in the tabs for the ***notepages***

Attribute	RLD	PID	Properties	Short Description
picwidth (IDM for Windows only)	integer	integer	S,G/D/-	width of the pictures displayed in the tabs for the ***notepages***
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
tabalignment	integer	integer	S,G/D/C	justification of text in tabs
tabshape	enum	enum	S,G/D/C	shape of major- and minort-abs
tile	object	tile	S,G/D/C	tile resource used as background image
tilestyle	enum	enum	S,G/D/C	controls how the background image set in *.tile* is arranged
toolbar	object	object	-,G/-/-	toolbar of object

Attribute	RLD	PID	Properties	Short Description
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	left position of notebook; the furthest point will be positioned
xleft	integer	integer	S,G/D/C	right position of notebook; the furthest point will be positioned
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	bottom position of notebook; the furthest point will be positioned
ytop	integer	integer	S,G/D/C	top position of notebook; the furthest point will be positioned

25.2 Specific Attributes

Standard Attributes

Please note that the mouse pointer set with the *.cursor* is independent of the window system and that it cannot always be set where you want. We therefore recommend to set the mouse pointer for each notepage individually.

Geometry Attributes

.height defines the height of the entire notebook object including all its elements. If the height is set to low, a minimal value depending on the window system will be used instead. The height must not be 0, since the required height cannot be calculated.

.menu: The availability of a popup menu exclusively for the notebook depends on the window system. This is why - if a pop-up menu is required for the notebook - only two variants should be chosen:

1. a pop-up menu is set for each notepage, but none for the notebook.
2. a pop-up menu is set for the notebook, but none for the notepages.

If *.posraster* = *true*, the notebook will be shifted by a half grid unit (border object!).

If *.sizeraster* = *true*, a grid unit will be deducted from the notebook height and width (border object!).

Layout Attributes

The attribute *.bgc* defines the background color of the notebook.

The attribute *.bordercolor* (default: 0) defines the border color of the notebook.

Object-specific Attributes

.activeobject defines the active, relevant notepage on top.

.alignment defines the alignment of a status text which every notepage can have (default = 1).

.backpage defines the corner in which the visible page borders meet. "backpage" describes the realization of invisible notepages. The borders of the invisible notepage are displayed in a corner.

.binding defines the way a notebook is to be bound.

.direction specifies the direction which the notebook is oriented to (default = 1, vertically).

.majortabheight defines the height of all majortabs. "majortab" stands for the labeling of a notepage in the main index.

.majortabwidth defines the width of all majortabs.

.minortabheight defines the height of all minortabs. "minortab" stands for the labeling of a notepage in the side index.

.minortabwidth defines the width of all minortabs (side index).

.tabalignment defines how the text is justified in the tabs of the notebook.

.tapshape defines the shape of major- and minortabs.

.width defines the width of the entire notebook object including all its elements. If the width is specified too small, a minimal value independent of the window system is chosen instead. The width must not be 0 because in that case the necessary width cannot be calculated.

25.2.1 backpage

In the following table the effects of the attributes *.backpage* and *.direction* on the positioning of major tabs and minor tabs as well as on the binding are shown:

.backpage	*.direction*	Major Tab	Minor Tab	Binding
bp_bottomright	*1*	right	bottom	left
bp_bottomright	*2*	bottom	right	top
bp_bottomleft	*1*	left	bottom	right
bp_bottomleft	*2*	bottom	left	top
bp_topright	*1*	right	top	left
bp_topright	*2*	top	right	bottom
bp_topleft	*1*	left	top	right
bp_topleft	*2*	top	left	bottom

25.3 Keyboard Control

Within the notebook, the keyboard control only affects the child objects of the top notepage, and not - as is usual - all child objects.

A notepage change has to be made by system-dependent keyboard methods.

Whether the notebook is in the focus order also depends on the system.

25.4 Example

```
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";
    }
  }
}
```

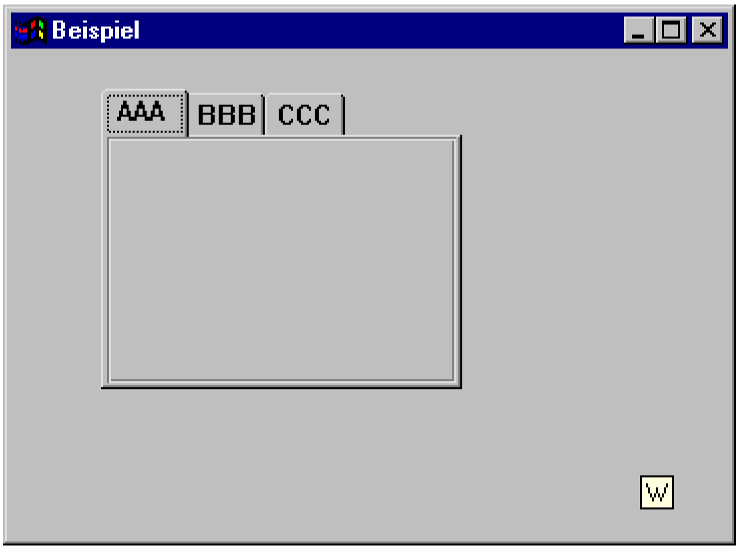


Figure 31: notebook

26 notepage

The object ***notepage*** is a page of a ***notebook***.

A ***notepage*** has to be defined in the same way as a child. The keyword is **notepage** and is followed optionally by a label name.

The ***notepage*** is always positioned in such a way that the entire page of the ***notebook*** except for the tab row will be covered.

A ***notepage*** has child objects (like ***groupbox***).

Definition

```
{ export | reexport } { model } notepage { <Identifier> }
{
  <standard attributes>
  <hierarchy attributes>
  <layout attributes>
  <scrollbar attributes>
  <object-specific attributes>
}
```

Events

activate

cut

deactivate

extevent

focus

help

hscroll

key

paste

scroll

select

vscroll

Children

document

canvas

checkbox

dialog

edittext

groupbox

image

layoutbox

listbox

notebook

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

statictext

tablefield

transformer

treeview

Parent

notebook

Menu

Pop-up menu

26.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation

Attribute	RLD	PID	Properties	Short Description
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object, brings a certain notepage on the top
active	boolean	boolean	S,G/D/C	true for the top notepage
bgc	identifier	color	S,G/D/C	background color of object
bordercolor	identifier	color	S,G/D/C	border color of object
borderwidth	integer	integer	S,G/D/C	border width of object
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class of object
control	identifier	instance	-,G/-/-	control the object currently belongs to
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color

Attribute	RLD	PID	Properties	Short Description
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
help	string identifier	string text	S,G/D/C	help text of object
hsb_arrows	boolean	boolean	S,G/-/-	defines whether horizontal scrollbar has arrows at its end
hsb_linemotion	integer	integer	S,G/D/C	horizontal scroll value for scrolling line by line
hsb_optional	boolean	boolean	S,G/D/C	horizontal scrollbar is only displayed if necessary.
hsb_pagemotion	integer	integer	S,G/D/C	horizontal scroll value for scrolling page by page
hsb_visible	boolean	boolean	S,G/D/C	visibility of horizontal scroll-bar
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object

Attribute	RLD	PID	Properties	Short Description
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
picture	object	tile	S,G/-/C	picture for i-th item
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
reffont	identifier	font	S,G/D/C	reference font of object
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
tabtype	enum	enum	S,G/D/C	tab type
text	string identifier	string text	S,G/D/C	text of status line

Attribute	RLD	PID	Properties	Short Description
tile	object	tile	S,G/D/C	tile resource used as background image
tilestyle	enum	enum	S,G/D/C	controls how the background image set in *.tile* is arranged
title	string identifier	string text	S,G/D/C	labeling of tabs
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
vheight	integer	integer	S,G/D/C	internal (virtual) height
visible	boolean	boolean	S,G/D/C	visibility of object
vsb_arrows	boolean	boolean	S,G/-/-	defines whether vertical scrollbar has arrows at its end
vsb_linemotion	integer	integer	S,G/D/C	vertical scroll value for scrolling line by line
vsb_optional	boolean	boolean	S,G/D/C	vertical scrollbar will be displayed, if necessary
vsb_pagemotion	integer	integer	S,G/D/C	vertical scroll value for scrolling page by page
vsb_visible	boolean	boolean	S,G/D/C	visibility of vertical scrollbar
vwidth	integer	integer	S,G/D/C	internal (virtual) width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xorigin	integer	integer	S,G/D/C	shift of the origin along the x-axis in objects with scrollbars

Attribute	RLD	PID	Properties	Short Description
xraster	integer	integer	S,G/D/C	units in x-direction
yorigin	integer	integer	S,G/D/C	shift of the origin along the y-axis in objects with scroll-bars
yraster	integer	integer	S,G/D/C	units in y-direction
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

26.2 Specific Attributes

Standard Attributes

The attribute *.accelerator* can be used to activate a certain notepage, i.e. the notepage can be brought on top. To do so, a select-activate event will be triggered.

For a notepage *.active* is true for the page which is then on top, for all other notepages .active is false. This attribute can be true for one single notepage only.

(The notepage label is saved in the .activeobject.)

Note for the attribute *.focus* that only one child object can have the focus (as it is the case with the object "groupbox").

.sensitive defines the selectivity of the notepage so that the child objects are specified as insensitive, that the notepage itself, however, can still be activated.

Geometry Attributes

.menu: Whether a pop-up menu is available depends on the window system (as it is the case with "notebook"). This is why - if a pop-up menu is required - only two variants should be chosen:

1. a pop-up menu is set for each notepage, but none for the notebook.
2. a pop-up menu is set for the notebook, but none for the notepages.

With the attributes *.real_height* and *.real_width* the actual size of a notepage can be queried.

Layout Attributes

.fgc and *.font*, the color and the font are specified by the notebook attributes.

For the layout of notepages in Windows the following has to be considered:

- The foreground color (.fgc) of the ***notebook*** resp. ***notepage*** object will be ignored. The texts are displayed in a system color (like ***pushbutton*** object).

- The background color (.bgc) of the ***notepage*** object changes only the background color of the actual display area. The background color of the title or tab is a system color.
- The background color (.bgc) of the ***notebook*** object changes only the color outside the tabs.The background color of the tabs is a system color.
- The font (.font) of the ***notepage*** object will be ignored. The font of the ***notebook*** will be used for the labeling of the tabs.

Object-specific Attributes

.tabtype: The kind of tab for "notepage" can be described with this attribute.

The value range of some attributes depends on the window system. According to the window system either the attributes will be considered or the value which is marked as default will be chosen. This default value will also be chosen if nothing is specified.

Text Attributes

The text which is to appear in the status line has to be defined by the attribute *.text* (default = 0). If no text is defined, the notepage will get no status line. The position at which the status line appears depends on the window system and on the .backpage and .direction.

.title defines the labeling of tabs.(default = 0). There will only be a text, if a value <> *0* is specified here. The labeling can be either a text or a bitmap.

Note

It is allowed to create an unlimited number of notepages as long as memory is available. (Attention: There is a limit of 64kByte on MS-Windows!)

The order of notepages is defined by the order you create them in the dialog file. The principle is the same as with a normal file: the first notepage is the one on top. The order is in the opposite direction to the child objects, but it is in the "natural" order of the tabs.

26.3 Keyboard Control

Within the notebook, the keyboard control only affects the child objects of the top notepage, and not - as is usual - all child objects.

A notepage change has to be made by system-dependent keyboard methods.

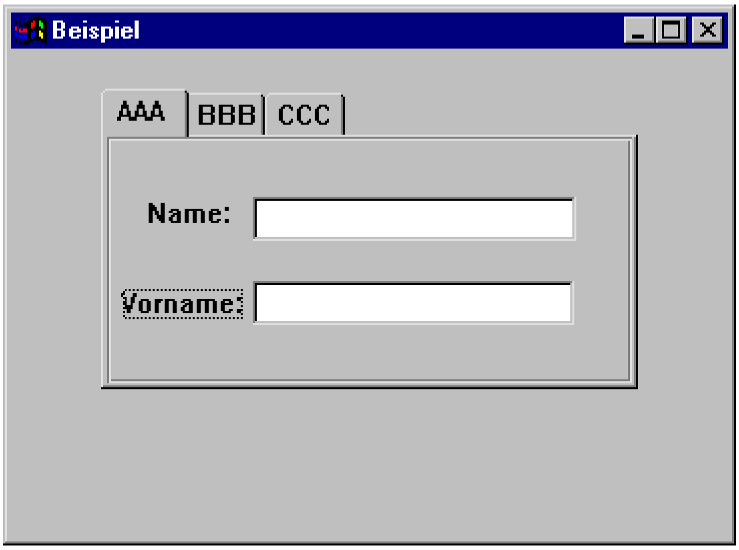
Whether the notebook is in the focus order also depends on the system.

26.4 Example

```
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";
      }
    }
  }
```



Name:

Vorname:

Figure 32: notepage

27 poptext (Combobox)

With this object textual information can be displayed in a space-saving way, if only one piece of information out of several pieces of information is needed.

Definition

```
{ export | reexport } { model } poptext { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

In the description “combobox” is used as a synonym for ***poptext***.

Events

activate

charinput

cut

deselect

deselect_enter

extevent

focus

help

key

modified

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

27.1 Attributes

Attribute	RSD	PSD	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
activeitem	integer	integer	S,G/D/C	active text item of combobox
alignment	integer (-1, 0, 1)	integer	S,G/D/C	text alignment
bgc	identifier	color	S,G/D/C	background color of object
borderraster	boolean	boolean	S,G/D/-	controls computation of geometry with active grid
class	class	class	-,G/-/-	object class
content	string	string	S,G/-/-	displayed text of object
control	identifier	instance	-,G/-/-	control which the object belongs to

Attribute	RSD	PSD	Properties	Short Description
cursor	identifier	cursor	S,G/D/C	cursor belonging to the object
cut_pending	boolean	boolean	S,G/-/-	cut operation has not been executed yet
cut_pending_changed	boolean	boolean	-,G/-/-	state of change during cut operation
dialog	identifier	instance	-,G/-/-	dialog for object
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
editable	boolean	boolean	S,G/D/C	editability of editable text
endsel	integer	integer	S,G/D/C	end of the selection in the input field
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color of object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	-,G/-/-	keyboard focus of object
font	identifier	font	S,G/D/C	font of object
function	identifier	func	S,G/D/C	function of object
format	string identifier	string format	S,G/D/C	formatstring of a text
formatfunc	identifier	func	S,G/D/C	link of application function to editable text
groupbox	identifier	instance	-,G/-/-	groupbox which the object currently belongs to
height	integer	integer	S,G/D/C	indicates height of object

Attribute	RSD	PSD	Properties	Short Description
help	string identifier	string text	S,G/D/C	helptext of object
hinttext	string	string	S,G/D/C	placeholder text of object (see also chapter “Notes for attribute .hinttext”)
index	integer	integer	-,G/-/-	current index of object in the child list of its parent
itemcount	integer	integer	S,G/D/C	number of elements in combobox
label	string	string	S,G/D/C	name (identifier) of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
maxchars	integer	integer	S,G/D/C	maximum number of characters to be input in edittext
member[I]	attribute	attribute	-,G/-/-	i-th user-defined attribute of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	menu of object
model	identifier	instance	S,G/D/C	model belonging to model
notepage	identifier	instance	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
picheight	integer	integer	S,G/D/-	height of the space for pictures displayed left of the items

Attribute	RSD	PSD	Properties	Short Description
picture[I]	object	tile	S,G/D/C	picture for each item (IDM for Windows and IDM for Qt)
picture_hilite[I]	object	tile	S,G,D,C	picture for each item when the item is selected (IDM for Windows and IDM for Qt)
picwidth	integer	integer	S,G/D/-	width of the space for pictures displayed left of the items
posraster	boolean	boolean	S,G/D/C	indication of positions refers to raster
real_height	integer	integer	-,G/-/-	actual height of object
real_modified	boolean	boolean	-,G/-/-	actual change of content
real_sensitive	boolean	boolean	-,G/-/-	actual selectability of object
real_visible	boolean	boolean	-,G/-/-	actual visibility of object
real_width	integer	integer	-,G/-/-	actual width of object
real_x	integer	integer	-,G/-/-	real distance to left (in pixel)
real_y	integer	integer	-,G/-/-	real distance to top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectability of object
showitem	integer	integer	S,G/D/C	number of visible items

Attribute	RSD	PSD	Properties	Short Description
sizeraster	boolean	boolean	S,G/D/C	size referring to raster of parent object
startsel	integer	integer	S,G/D/C	start of the selection in the input field
statushelp	string identifier	string text	S,G/D/C	text to be displayed in statusbar
style	class	class	S,G/D/C	type of poptext, normal, editable (combobox) or list-box-like
text	string identifier	string text	S,G/D/C	sets individual texts
toolbar	object	object	-,G/-/-	toolbar of object
userdata	anyvalue	anyvalue	S,G/D/C	userdata of object of any datatype
userdata[I]	anyvalue	anyvalue	S,G/D/C	userdata of combobox items of any datatype
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	current width of object
window	identifier	instance	-,G/-/-	window which object currently belongs to
xauto	integer (-1, 0, 1)	integer	S,G/D/C	type of x-coordinatedefinition
xleft	integer	integer	S,G/D/C	x-coordinate, distance to the left
xright	integer	integer	S,G/D/C	x-coordinate, distance to the right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	type of y-coordinate definition
ybottom	integer	integer	S,G/D/C	y-coordinate, distance to bottom

Attribute	RSD	PSD	Properties	Short Description
ytop	integer	integer	S,G/D/C	y-coordinate, distance to top

27.2 Specific Attributes

Standard Attributes

In the combobox there is a .userdata attribute for each item which can be accessed by index. Without an index the usual .userdata attribute will be accessed.

Plain Attributes

These attributes are used to define size and position of the combobox. If the attributes *.width* or *.height* are set to 0, the DM will calculate the corresponding size automatically.

Attribute to control the creation of the activate and select events

The attribute *.options[opt_old_select]* controls the creation of the activate and select events, it is normally set to *false*.

If the attribute *.options[opt_old_select]* is set to *false*, then the select/activate events will be generated analog to the other sample objects, such as the listbox and treeview.

First modification to the “old” behavior is that the activate events no longer indicate the receiving of the focus (edittext and listbox style), but rather show a change in the activation status (.activeitem attribute). Secondly, a select event is only created by a "completed" selection of the user. This is true even when the same entry is selected a second time. In this sense "completed" means that the list is closed (exception: comment (1) and (2)) or when the modification does not require that the list be made visible.

Data type	*boolean*
Value range	*false*, *true*
Default value	*false*
Access	get, set
Supported on	MICROSOFT WINDOWS, UNIX

Summary

For a poptext with the attribute *.options[opt_old_select]=false*, the select and activate where triggered under the following conditions:

Action	Motif 1.2	Motif 2.1	MS Win	Event
cursor selection (close list, on different entry)	*	*	*	- activate and select
cursor selection (open list, on different entry)	*	*	*	- activate (and select in listbox style)
mouse selection (list open, same entry)	*	*	*	select
mouse selection (list open, different entry)	*	*	*	activate and select
close list (i.e. `Return` but no mouse selection)	*	*	*	select
abort list (2) (i.e. `Esc`)	*	*	*	activate in order to return to original entry

Comments

1. The poptext list in the style listbox is always open. Therefore the keyboard selection behaves analog to the mouse selection. Naturally, in this style the list cannot be "aborted".
2. Only on some window systems is it possible to abort an open list.
3. The keys which can be used for cursor selection is dependent upon the system in use; normally these are the `Up` and `Down` cursor keys.

Example for the use of separate events with .options[opt_old_select] = false

» In order to be aware of all changes that have taken place (even in open lists)

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

» In order only to react to the selection of another list item

```
poptext Pt {
  .options[opt_old_select] false;
  on select {
    if (thisevent.value <> thisevent.index) then
      ...
    endif
  }
}
```

- In order to react to every selection

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

- "on activate" is no longer to be used when the receipt of the focus is meant. Now this should be written as follows

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

Text Attributes

Text attributes are active in the styles "edittext" and "listbox" only. Usually (.style poptext) these attributes are ignored. .content is only readable in Style poptext. .content is a dynamic attribute in a combobox. It may be set during runtime only. This attribute cannot be defined statically in the dialog file.

The combobox contains three different types of representation and editability. These types are controlled with the attribute *.style*:

- *poptext*
  The combobox is not editable and is only used to represent a usually fixed number of items.
- *edittext*
  The combobox is editable and may be changed by the user. Basically this type is used to administrate dynamic lists. You may add further items to the list and select items.
- *listbox*
  The combobox is editable and is always open, allowing you to quickly select an item from a given list (similar to listbox) or to specify an item yourself. This is helpful for search processes.

The event processing in the combobox (.style edittext or listbox) is different from the actual edittext. The select event for the combobox behaves different from the one for the edittext.

With the styles *.style* = *edittext* and *.style* = *listbox*, the attribute *.real_modified* indicates, whether the content actually has changed since the combobox has received the focus.

Notes for attribute .hinttext

This attribute is supported only by WINDOWS and QT. The ***poptext*** object must have the *edittext* style set.

On WINDOWS, .hinttext is also supported if the attribute .style has the value *listbox*.

Note for Dialog Manager on Microsoft Windows

The height of a combobox cannot be set. The corresponding window system will automatically choose a value depending on the font.

The poptext supports Dialog Manager **opt_hscroll** option as of IDM version A.06.01.h. The horizontal scrollbar appears within the display area and thus covers the lowest lines. A vertical scrollbar is then also displayed for this purpose. Since the operation of scrollbars in an open list is not easy for the user, this option should only be used if there is no other solution.

Note for Dialog Manager on Motif

The attribute *.showitem* is not available since Motif version 2.1

27.3 Example

```
window PoptextDemo
{
  .active false;
  .title "PoptextDemo";
  child poptext Poptext
  {
    .xleft 5;
    .ytop 1;
    .text[1] "Item 1";
    .text[2] "Item 2";
    .text[3] "Item 3";
    .text[4] "Item 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] "List 1";
    .text[2] "List 2";
    .text[3] "List 3";
    .text[4] "List 4";
    .text[5] "List 5";
    .text[6] "List 6";
```

```
      .text[7] "List 7";
      .text[8] "List 8";
      .activeitem 1;
      .style listbox;
    }
  }
```

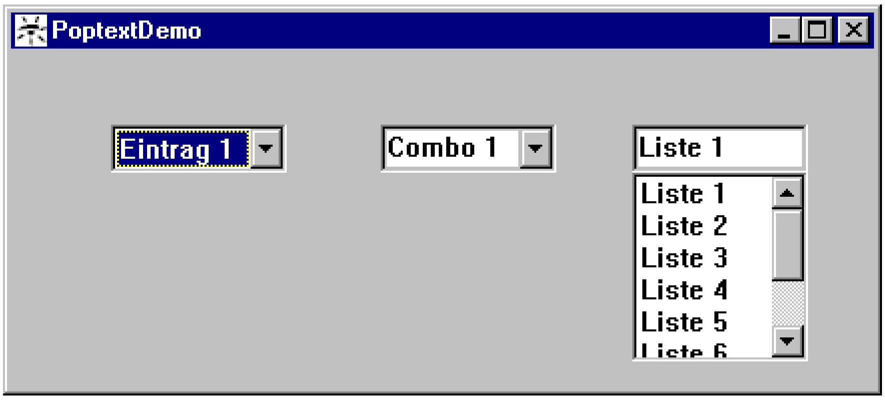
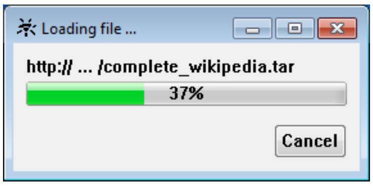


Figure 33: Window showing the different combobox shapes

28 progressbar

The ***progressbar*** object enables to represent the process of a long lasting action visually. For this a kind of bar is shown, whose length is adapted to the progress of the action. Thereby the user can estimate how far the action has advanced and how long it will last.



Definition

```
{ export | reexport } { model } progressbar { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

cut

extevent

focus

help

paste

Children

document

record

transformer

Parents

groupbox

layoutbox

module

notepage

splitbox

statusbar

toolbar

window

Menu

Pop-up menu

28.1 Attributes

accelerator

active

bgc

borderstyle

borderwidth

class

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

28.2 Specific Attributes

Attribute	Description
curvalue	Defines the position of the progressbar. See chapter “Calculation of the Progressbar”.
borderstyle	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a) Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
borderwidth	Width of the object border. On Microsoft Windows a standard border appears if the value of the attribute is greater than *0*. Therefore it can only be defined if the border appears or not. The width cannot be influenced.
direction	Direction of the progressbar. *1* means vertical direction, the bar proceeds from bottom to top. *2* means horizontal direction, the bar proceeds from the left to the right.
maxvalue	Maximum value of the progress indicator. See chapter “Calculation of the Progressbar”.
minvalue	Minimum value of the progress indicator. See chapter “Calculation of the Progressbar”.
options[enum]	Options of the object.
style[enum]	Appearance of the progress indicator.
textfgc	Color of the label (provided that it is supported by the system), with *null* the default value of the system is used.

28.3 Calculation of the Progressbar

The attribute curvalue specifies the position of the progress indicator. Furthermore this value is used to calculate an percentage value that may be displayed optionally.

If the value is equal to minvalue, the progress indicator is invisible and the percentage value is *0%*. If the value is equal to maxvalue , the progress indicator covers the entire bar area and the percentage value is *100%*. The values in between are calculated accordingly. The progress indicator grows always from left to right or from bottom to top, no matter if *.minvalue* is larger or smaller than *.maxvalue*.

28.4 Notes for the Progressbar on Microsoft Windows

Note for active “Visual Styles”:

The coloring of “Visual Styles” cannot be adjusted to the color set. To avoid inapplicable color combinations, it is recommended to omit setting colors.

Especially it is discouraged to use *.fgc* with the progressbar (as this would lead to a display of the progressbar with a wide border caused by “Visual Styles”).

28.5 Example

```
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  "Modules are loaded ...";
  .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;
  ...
}
```

29 pushbutton

The ***pushbutton*** is a type of button that can be used to make a decision, such as “Yes”, “No”, “Cancel” or “Continue”. The activation state of the ***pushbutton*** is not stored, it remains only as long as the user is actively selecting the button.

Definition

```
{ export | reexport } { model } pushbutton { <Identifikator> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

cut

extevent

focus

help

key

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

29.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
bgc	identifier	color	S,G/D/C	background color
class	class	class	-,G/-/-	class of object
control	identifier	instance	-,G/-/-	control the object currently belongs to
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
defbutton	boolean	boolean	S,G/D/C	default pushbutton in dialogbox
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class

Attribute	RLD	PID	Properties	Short Description
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to

Attribute	RLD	PID	Properties	Short Description
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
text	string	string	S,G/D/C	static text of object
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)

Attribute	RLD	PID	Properties	Short Description
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

29.2 Specific Attributes

Plain Attributes

These attributes define the size and position of the pushbutton. If *.height* or *.width* is set to 0, the DM calculates the size of the pushbutton on the basis of the actual font.

The attribute .options can have the following values:

option_index	Meaning
opt_use_widget	Pushbutton is displayed as widget. (on Motif version only)

29.3 Example

```
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";
  }
}
```

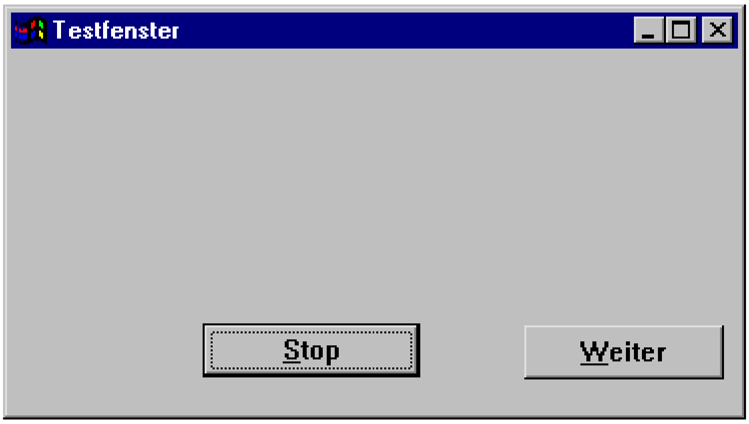


***Figure 34:** window with pushbuttons*

30 radiobutton

The ***radiobutton*** is a type of button that can be used to select an item out of a group of options (either-/or). Only one ***radiobutton*** can be selected. If another one is selected, the state belonging to the previously selected ***radiobutton*** will be deleted.

By selecting a ***radiobutton***, all ***radiobuttons*** on the same hierarchical level are deselected. If more than one ***radiobutton*** is to be selected at the same time in one window, their direct parents have to be different. This can be achieved by using ***groupboxes*** in which these groups of ***radiobuttons*** are located.

Definition

```
{ export | reexport } { model } radiobutton { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

activate

cut

deactivate

extevent

focus

help

key

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

30.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for Microsoft UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for Microsoft UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
active	boolean	boolean	S,G/D/C	activation state of an object
bgc	identifier	color	S,G/D/C	background color
class	class	class	-,G/-/-	class/id of object
control	identifier	instance	-,G/-/-	control currently belonging to object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs

Attribute	RLD	PID	Properties	Short Description
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	S,G/-/C	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes

Attribute	RLD	PID	Properties	Short Description
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
text	string identifier	string text	S,G/D/C	static text of object
toolbar	object	object	-,G/-/-	toolbar of object

Attribute	RLD	PID	Properties	Short Description
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

30.2 Specific Attributes

Standard Attributes

The attribute *.active* defines the “on” state (*.active* = *true*) or the “off” state of the radiobutton.

Plain Attributes

These attributes define the size and position of the radiobutton. If *.height* or *.width* is specified as 0, the DM calculates the size of the radiobutton on the basis of the actual font.

The attribute .options can have the following values:

option_index	Meaning
opt_use_widget	Radiobutton is displayed as widget. (on Motif only)
opt_push_like	Radiobutton is displayed like a pushbutton. (on Microsoft Windows only)

30.3 Notes on the radiobutton under Microsoft Windows

- Under Microsoft Windows a radiobutton gets activated when it obtains the focus. If an activated radiobutton gets insensitive, the focus moves to the next radiobutton, activating that one. Therefore it should be avoided to set the activated radiobutton insensitive.
- Concerning the background color, please refer to the notes on attribute .bgc in the “Attribute Reference”.

30.4 Example

Two groups of radiobuttons in one window.

```
window Radiowindow
{
  child groupbox Group_1
  {
    child radiobutton First
    {
    }
    child radiobutton Second
    {
    }
    child radiobutton Third
    {
    }
  }
  child groupbox Group_2
  {
    child radiobutton Four
    {
    }
    child radiobutton Five
    {
    }
    child radiobutton Six
    {
    }
  }
}
```

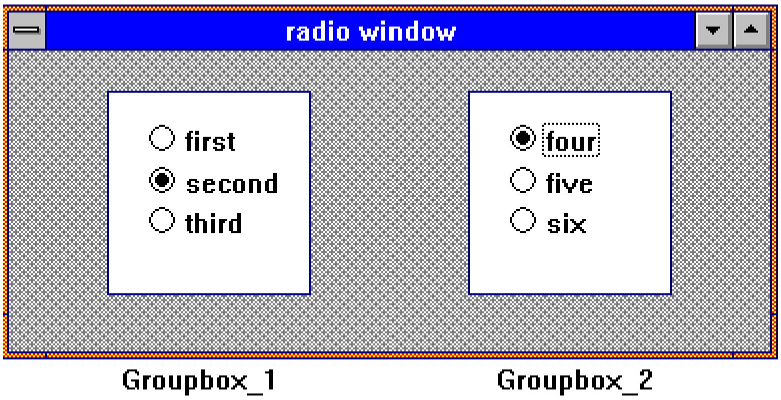


Figure 35: Radiobuttons

31 record

By means of the ***record*** objects, any structure can be built in the rules.

These ***record*** objects can be defined directly as children of the dialog (globally available) or as children of arbitrary other objects.

record objects can also be defined for models and are passed on to instances.

A ***record*** can contain user-defined attributes as well as subrecords.

Definition

```
{ export | reexport } { model } record <Identifier>
{
  <hierarchy attributes>
  <object-specific attributes>
  [ <definition of user-defined attribute> ; ]
  [ { export | reexport } record <Identifier> <record definition> ]
}
```

Events

none

Children

document

record

transformer

Parent

all objects

Menu

none

31.1 Attributes

Attribute	RSD	PSD	Properties	Short Description
class	class	class	-,G/-/-	object class
configurable	boolean	boolean	S,G/D/C	configurability via profile

Attribute	RSD	PSD	Properties	Short Description
control	identifier	instance	-,G/-/-	control the object currently belongs to
dialog	identifier	instance	-,G/-/-	object dialog
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
firstrecord	object	record	S,G/-/C	accesses the first record of an object
groupbox	identifier	instance	-,G/-/-	groupbox the object currently belongs to
index	integer	integer	-,G/-/-	current index of object in child list of its parent
itemorder	string	string	-,G/-/-	order of child records and attributes
label	string	string	S,G/D/C	name (identifier) of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
member[I]	attribute	attribute	-,G/-/-	i-th user-defined attribute of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
model	identifier	instance	S,G/D/C	model belonging to object
notepage	identifier	instance	-,G/-/-	notepage the object currently belongs to
parent	identifier	instance	S,G/-/-	parent of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object

Attribute	RSD	PSD	Properties	Short Description
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
toolbar	object	object	-,G/-/-	toolbar of object
window	identifier	instance	-,G/-/-	window the object currently belongs to

31.2 Specific Attributes

With the attribute *.configurable* you can define whether a record can be configured, i.e. whether it can be set in the configuration file loaded with DM_LoadProfile().

By using the attribute *.itemorder* the order of definitions for attributes and records may be queried within records. In the string there will be deposited an “A” for each attribute and an “R” for each child record at the corresponding position.

31.3 Example

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

In doing so, you can define a structure consisting of three elements and two functions which get this record as parameter.

Record with child record:

```
record Data
{
  string   Name;
  integer  State;

  record Startvalues
  {
    integer  Position;
    integer  Alphavalue;
  }
  record Endresult
  {
    boolean  OK;
    integer  Betavalue;
    integer  Gammavalue;
  }
}
```

Querying the definition order in a record:

```
record R
{
  string Name;
  record Child1
  {
    integer Number;
    boolean On;
  }
  boolean Off;
}

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

31.4 Integrating Records in Application

C modules generated by DM have to be compiled and linked to be able to provide functions for dialogs with records with an application.

These modules are generated by calling the simulation with the option **+writetrampolin**:

```
idm +writetrampolin <outfile> <dialogscript>
```

This statement generates the necessary modules for the call of the functions from a dialog file. According to the type of the functions which use such records, the corresponding header files (C and/or COBOL) are generated.

If such structures are used in the DM object "application", i.e. if the Distributed Dialog Manager is used, you additionally have to indicate the application for which the files are generated.

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

32 rectangle

This object can be used for the graphic design of the dialog only. Normally it has no functionality and meaning for the dialog process.

Definition

```
{ export | reexport } { model } rectangle { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

dbselect

extevent

focus

help

key

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

32.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
bgc	identifier	color	S,G/D/C	background color
borderraster	boolean	boolean	S,G/D/-	controls computation of geometry with active grid
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
borderwidth	integer	integer	S,G/D/C	width of the object border
class	class	class	-,G/-/-	class/id of object
control	identifier	instance	-,G/-/-	control currently belonging to object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs

Attribute	RLD	PID	Properties	Short Description
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
filled	boolean	boolean	S,G/D/C	rectangle is filled with background color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
focus_on_click	boolean	boolean	S,G/-/-	defines if a mouse click into an object's client area activates the object
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	current index of object in the child list of its parent
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object

Attribute	RLD	PID	Properties	Short Description
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object

Attribute	RLD	PID	Properties	Short Description
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	string object	string text	S,G/D/C	gives a short explanation of object at the cursor
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

32.2 Specific Attributes

Standard Attributes

The attribute *.menu* is ignored in the Motif version, i.e. it is not possible to attach a popup menu to the object rectangle.

Layout Attributes

A cursor does not exist for the object rectangle in the Motif version!

32.3 Example

```
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;
  }
}
```

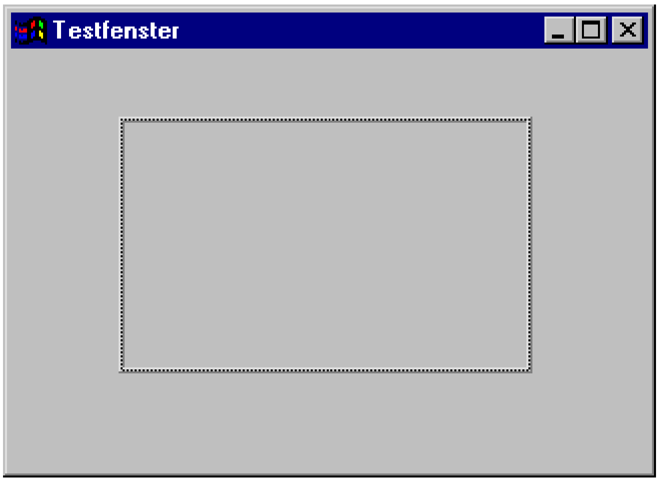


***Figure 36:** window with rectangle*

33 scrollbar

A ***scrollbar*** can be used as an analog announcing object or as an analog regulator. It may be horizontally or vertically aligned.

Definition

```
{ export | reexport } { model } scrollbar { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

cut

extevent

focus

help

key

paste

scroll

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

33.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
arrows	boolean	boolean	S,G/D/C	scrollbar arrows
bgc	identifier	color	S,G/D/C	background color of object
class	class	class	-,G/-/-	class/id of object
control	identifier	instance	-,G/-/-	control currently belonging to object
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
curvalue	integer	integer	S,G/D/C	position of scrollbar slider
cut_pending	boolean	boolean	S,G/-/-	cut operation not yet carried out
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
direction	integer	integer	S,G/D/C	justification of scrollbar
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class

Attribute	RLD	PID	Properties	Short Description
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	height of object
help	string identifier	string text	S,G/D/C	help text of object
index	integer index	integer index	-,G/-/-	returns or sets the first record of an object
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
linemotion	integer	integer	S,G/D/C	value by which scrollbar changes when scrolling line by line
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
maxvalue	integer	integer	S,G/D/C	maximal scrollbar value
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu

Attribute	RLD	PID	Properties	Short Description
minvalue	integer	integer	S,G/D/C	minimal scrollbar value
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	standard value of attribute is true
options[enum]	boolean	boolean	S,G/D/-	special options of object
pagemotion	integer	integer	S,G/D/C	value by which scrollbar changes when scrolling page by page
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object

Attribute	RLD	PID	Properties	Short Description
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

33.2 Specific Attributes

Note for IDM on Microsoft Windows

If the attribute .arrows is set to false, the IDM uses the Windows object "slider" instead of the object "scrollbar".

See Also

Attribute arrows

Scrollbar attributes:

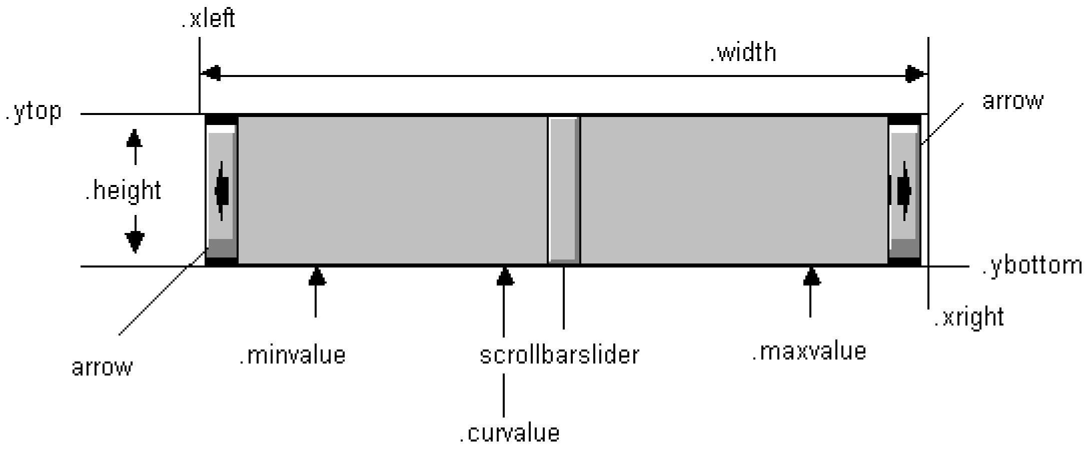


Figure 37: Attributes of the scrollbar

Note

The size of a scrollbar slider is calculated on the basis of .maxvalue, .minvalue and .pagemotion. If the entire area over which the slider can navigate is 1 pixel long, the calculation is as follows:

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

if necessary the result is rounded up or down.

The relation of the slider size to the display area is similar to the relation of .pagemotion to the difference of .maxvalue and .minvalue.

This calculation is valid for scrollbars as objects as well as for scrollbars attached to objects.

However, this calculation is not valid for the so-called "scale widgets" (.arrows = false), which have a fixed size.

33.3 Example

```
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;
  }
}
```

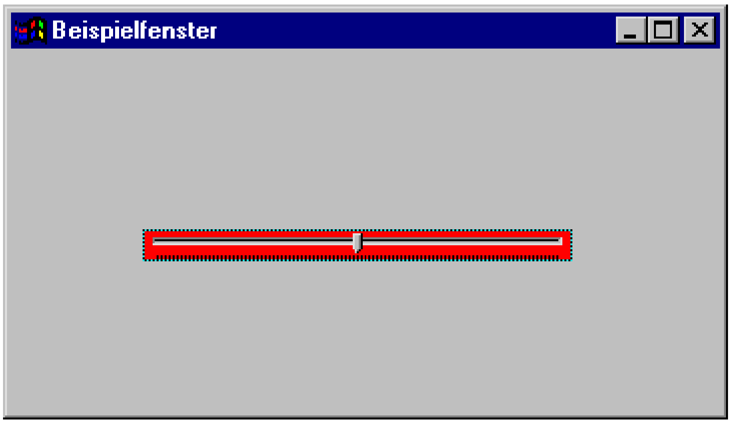


Figure 38: scrollbar

34 setup

The ***setup*** object can be used to query the system configuration out of the dialog. The attributes query the window system, the screen resolution, the operating system, the color model, etc.

Definition

```
This object cannot be defined; it can only be queried in the rules!
```

Events

none

Children

none

Parent

none

Menu

none

See Also

C function DM_ParsePath in manual “C Interface - Functions”

34.1 Attributes

Attribute	RLD	PID	Properties	Short Description
color	integer	integer	-,G/-/-	color variant
colorcount	integer	integer	-,G/-/-	number of colors
colorcount[I]	integer	integer	-,G/-/-	number of colors for screen I in multiscreen systems (IDM for Motif only)
color_type	enum	enum	-,G/-/-	type of screen possible values: » coltype_bw » coltype_color » coltype_grey

Attribute	RLD	PID	Properties	Short Description
color_type[I]	enum	enum	-,G/-/-	type of screen for screen I in multiscreen systems (IDM for Motif only) possible values as for *.color_type*
cursor	identifier	cursor	S,G/D/C	cursor variant
env[string]#	string	string	S,G/-/-	accesses environment variables set through the option **-IDMenv**
envvar[string]	string	string	S,G/-/-	accesses environment variables
errfile	string	string	-,G/-/-	absolute path of the error file
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
font	identifier	font	S,G/D/C	font variant
format	string identifier	string format	S,G/D/C	format variant
keyboard	integer	integer	-,G/-/-	accelerator variant
language	integer	integer	-,G/-/-	text variant
logfile	string	string	-,G/-/-	absolute path of the log file
mouse_buttons	integer	integer	-,G/-/-	number of logical mouse buttons
opsys_string	string	string	-,G/-/-	identification of operation system

Attribute	RLD	PID	Properties	Short Description
opsys_type	enum	enum	-,G/-/-	type of operation system possible values: » os_dos » os_mpe » os_nt » os_os2 » os_unix » os_vms
options[enum]	boolean	boolean	S,G/D/-	special options of object
overridecursor	identifier	cursor	S,G/-/-	temporary cursor for the entire dialogs in general
pointer_height	integer	integer	-,G/-/-	maximal height of mouse cursor
pointer_height[I]	integer	integer	-,G/-/-	maximal height of mouse cursor for screen I in multiscreen systems (IDM for Motif only)
pointer_width	integer	integer	-,G/-/-	maximal width of mouse cursor
pointer_width[I]	integer	integer	-,G/-/-	maximal width of mouse cursor for screen I in multiscreen systems (IDM for Motif only)
scale	integer	integer	-,G/-/-	specified system scaling
screen	integer	integer	-,G/-/-	screen number of the default screen in multiscreen systems (IDM for Motif only)
screen[I]	integer	integer	-,G/-/-	screen number of screen I in multiscreen systems (IDM for Motif only)
screen_height	integer	integer	-,G/-/-	screen height in pixels

Attribute	RLD	PID	Properties	Short Description
screen_height[I]	integer	integer	-,G/-/-	screen height in pixels for screen I in multiscreen or multimonitor systems (IDM for Motif & Windows only)
screen_width	integer	integer	-,G/-/-	screen width in pixels
screen_width[I]	integer	integer	-,G/-/-	screen width in pixels for screen I in multiscreen or multimonitor systems (IDM for Motif & Windows only)
screencount	integer	integer	-,G/-/-	number of available screens (IDM for Motif & Windows only)
searchpath	string	string	S,G/-/-	search path for dialog, module, interface and binary files for imports with **use**
terminal	string	string	-,G/-/-	terminal for AlphaWindows, e.g. vt220
terminaltype	string	string	-,G/-/-	terminal type for AlphaWindows, e.g. "/dev/tty"
tile	integer	integer	-,G/-/-	tile variant
tildpi	integer	integer	S,G/D/C	specified for which dpi value the images were created
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
toolkit	enum	enum	-,G/-/-	query of tookit type possible values: » toolkit_alpha » toolkit_isa » toolkit_motif » toolkit_pm » toolkit_windows
toolkit_string	string	string	-,G/-/-	query of toolkit

Attribute	RLD	PID	Properties	Short Description
toolkit_version	integer	integer	-,G/-/-	query of toolkit version
tracefile	string	string	-,G/-/-	absolute path of the trace file
tracetime	integer	integer	S,G/-/-	
tracing	boolean	boolean	S,G/-/-	executing tracing completely
version_string	string	string	-,G/-/-	query of DM version string
winsys	enum	enum	-,G/-/-	query of kind of window system possible values: » winsys_none » winsys_pm » winsys_windows » winsys_x11
winsys_string	string	string	-,G/-/-	query of window system
winsys_version	integer	integer	-,G/-/-	query of version of window system
xdpi	integer	integer	-,G/-/-	horizontal DPI (dots per inch) of screen
xdpi[I]	integer	integer	-,G/-/-	horizontal DPI (dots per inch) of screen I in multiscreen systems (IDM for Motif only)
ydpi	integer	integer	-,G/-/-	vertical DPI (dots per inch) of screen
ydpi[I]	integer	integer	-,G/-/-	vertical DPI (dots per inch) of screen I in multiscreen systems (IDM for Motif only)

34.2 Specific Attributes

The object messagebox ignores the .overridecursor set at the object setup and always displays the cursor defined for this messagebox or the default cursor.

Example for the Display of the Overridecursor

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

You may change the variant of a text resource also during runtime, namely in the

- Rule Language
- programming language

So far the language was determined on starting an application with the command line option **-IDMlanguage**.

Rule Language

```
setup.language := <variant number of language>
```

Example

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

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

C and COBOL Interface

The corresponding variant number of the chosen language has to be assigned to the setup object with DM_SetValue.

Note

It is helpful to set the language before loading the dialog as the text of every visible object has to be changed. If no object is visible yet the language may be changed very quickly.

34.3 Access to Environment Variables

The environment variables of the program can be set using the ***setup*** object. For this purpose there are the two attributes *.env[]* and *.envvar[]*.

.env[] contains all environment variables. Values which have been set with the option **-IDMenv** overwrite the values set in the environment.

Variables which have been set with the option **-IDMenv** are valid for IDM functions only. With the attribute *.envvar[]*, however, you receive environment variables only.

The attributes *.env[]* and *.envvar[]* are gettable and settable. They are indexed with the names of the environment variables. If the environment variable has not been set, a `fail` is returned on querying.

In addition you may view all variables which have been set with **-IDMenv**. Use the attribute *.count[]* for this purpose. It is indexed via *.env* whose return value is be the number of IDM environment variables. Now *.env[]* can be indexed with numbers from *1* to *.count[.env]* in order to find out the names of the environment variables.

Example

```
!! Reading the PATH variable
print setup.envvar["PATH"];
!! Setting an IDM environment variable
setup.env["DIALOGPATH"] := "/usr/local/idm/examples";
load("DIALOGPATH:draw.dlg");
!! Reading all IDM variables
for I:=1 to setup.count[.env] do
  print setup.env[I]+"="+setup.env[setup.env[I]];
endfor
!! Deleting the variable
setup.env["DIALOGPATH"] := null;
!! Setting an unset environment variable
if fail(setup.env["SERVER_OPT"]="") then
  setup.env["SERVER_OPT"] := "true";
endif
```

Note

Environment variables which have been set during a process will not result in changes in the environment variables of parent processes (e.g. shell) or existing child processes. Only new child processes can receive the changed environment variables.

34.4 Attributes for Controlling the Tracing

The setup object provides several attributes to control the tracing:

errfile, logfile, tracefile, tracetime and tracing.

See Also

Chapter “Tracing” in manual “Development Environment”

35 spinbox

A ***spinbox*** is a container object consisting of two arrow keys and which has **exactly one** child. This child must be a single-line ***edittext*** or ***statictext*** used to indicate the currently set value. If a value of the ***spinbox*** is being changed – by using an arrow key or by assigning a new value – the contents of the field will be updated automatically.

Definition

```
{ export | reexport } { model } spinbox  { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

changed

cut

extevent

help

key

paste

scroll

Children

document

1 *edittext* or **1 *statictext***

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

Methods

:delete()

:exchange()

:find()

:insert()

35.1 Attributes

Attribute	RSD	PSD	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
activeitem	integer	integer	S,G/D/C	current value at style = string
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects

Attribute	RSD	PSD	Properties	Short Description
class	class	class	-,G/-/-	object class
control	identifier	instance	-,G/-/-	control the object currently belongs to
curvalue	integer	integer	S,G/D/C	current value at style <> string
cut_pending	boolean	boolean	S,G/-/-	cut operation has not been executed yet
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during a cut operation
deltavalue	integer	integer	S,G/D/C	value that indicates the number of steps
dialog	identifier	instance	-,G/-/-	object dialog
direction	integer	integer	S,G/D/C	alignment of spinbuttons
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
focus	object	instance	S,G/-/-	key focus of object
firstchild	object	object	S,G/-/C	accesses the first child object
firstrecord	object	record	S,G/-/C	accesses the first record of an object
font	identifier	font	S,G/D/C	object font
height	integer	integer	S,G/D/C	object height
itemcount	integer	integer	S,G/D/C	number of text elements
index	integer	integer	-,G/-/-	current index of object in children list of its parent

Attribute	RSD	PSD	Properties	Short Description
label	string	string	S,G/D/C	name (identifier) of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
maxvalue	integer	integer	S,G/D/C	maximum value at style <> string
member[I]	attribute	attribute	-,G/-/-	i-th user-defined attribute of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
minvalue	integer	integer	S,G/D/C	minimum value of style <> string
model	identifier	instance	S,G/D/C	model belonging to object
notepage	identifier	instance	-,G/-/-	notepage the object currently belongs to
parent	identifier	instance	S,G/-/-	parent of object
real_height	integer	integer	-,G/-/-	real height of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectability of object (= always false)

Attribute	RSD	PSD	Properties	Short Description
statushelp	string identifier	string text	S,G/D/C	text to appear in the statusbar
style	datatype	datatype	S,G/D/C	defines the spinbox type
text[I]	string identifier	string text	S,G/D/C	individual text
tile	object	tile	S,G/D/C	tile resource used as background image
tilestyle	enum	enum	S,G/D/C	controls how the background image set in *.tile* is arranged
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
userdata	anyvalue	anyvalue	S,G/D/C	userdata of object of any datavalue
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer	integer	-,G/-/-	width of object
window	identifier	instance	-,G/-/-	queries window of object
wrap	boolean	boolean	S,G/D/C	determines whether a wraparound is possible
xraster	integer	integer	S,G/D/C	unit in x direction
yraster	integer	integer	S,G/D/C	unit in y direction

35.2 Specific Attributes

.wrap

This attribute for the spinbox controls whether a circular spinning is to be carried out (.wrap = true), or if the spinning is to be stopped once the spinning limits are reached (.wrap = false).

This attribute has the value true per default.

.deltavalue

This attribute specifies the difference value by which .curvalue is to be increased or decreased each step.

.borderstyle

Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.

35.3 Example

```
dialog Spinbox
{
}
color White "WHITE", grey(0);
window Wn
{
  .xleft 73;
  .width 260;
  .ytop 29;
  .height 222;
  .title "TestWindow";
  child spinbox Sp1
  {
    .xleft 62;
    .ytop 64;
    .curvalue 1;
    child edittext
    {
      .xauto 0;
      .xleft 0;
      .xright 0;
      .yauto 0;
      .ytop 0;
      .ybottom 0;
      .content "1";
    }
  }
  child spinbox Sp2
  {
    .xleft 62;
    .width 100;
    .ytop 113;
    .style string;
    .maxvalue 7;
    .text[1] "Monday";
```

```
      .text[2] "Tuesday";
      .text[3] "Wednesday";
      .text[4] "Thursday";
      .text[5] "Friday";
      .text[6] "Saturday";
      .text[7] "Sunday";
      .activeitem 1;
      child edittext Et2
      {
        .xauto 0;
        .xleft 0;
        .xright 0;
        .yauto 0;
        .ytop 0;
        .ybottom 0;
        .content "Monday";
      }
    }
  }
```

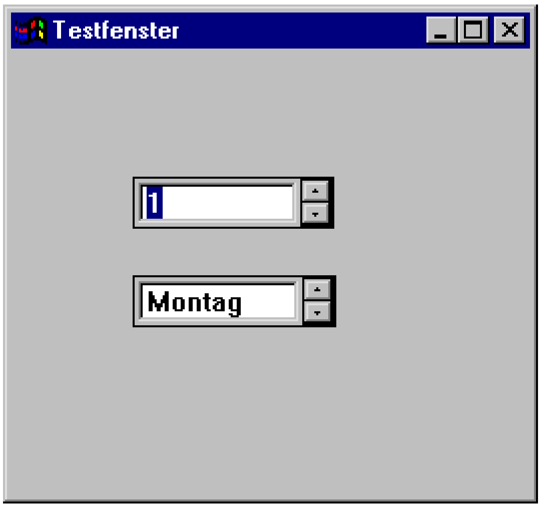


Figure 39: Spinbox

36 splitbox

The ***splitbox*** is a grouping object (similar to a ***groupbox***) that divides the display area of its parent object into smaller, adjustable areas. The areas are separated by splitbars, which can be interactively moved with the mouse by the user in order to size them individually. Each visible child of a ***splitbox*** is assigned to a separate area and in each area only one direct child can be present. The size and the visibility of the children are set according to the size of the areas they are assigned to. To place more than one child in a split area, a ***groupbox*** or ***layoutbox*** must be used, which can then contain further children. With ***groupboxes*** as children split areas can have virtual sizes.

Definition

```
{ export | reexport } { model } splitbox { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <grid attributes>
  <hierarchy attributes>
  <hierarchy attribute>
  <object-specific attributes>
}
```

For each split area the size can be set individually. Additionally, minimum and maximum values can be set for the size that cannot be underrun or exceeded when moving the splitbars with the mouse.

The splitbars can be either horizontally or vertically directed.

Events

cut

extevent

help

key

paste

resize

select

Children

canvas

checkbox

control

document

edittext

groupbox

image

layoutbox

listbox

notebook

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

tablefield

transformer

treeview

Parents

dialog

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

36.1 Attributes

acc_label

acc_text

accelerator

active

barwidth

bgc

bordercolor

borderraster

borderstyle

borderwidth

child[**I**]

childcount

class

control

cursor

cut_pending

cut_pending_changed

dialog

direction

document[**I**]

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

toolbar

toolhelp

userdata

visible

width

window

xauto

xleft

xraster

xright

yauto

ybottom

yraster

ytop

36.2 Specific Attributes

Attribute	Desciption
barwidth	Determines the width of the splitbar in pixels.
borderraster	Determines how geometry is computed when a grid is used. Only effective for *.borderwidth > 0*.
borderstyle	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
direction	Sets the direction of the splitbars: » *1* (default): vertical » *2*: horizontal
maxsize[I]	Sets the maximum size for split area I, which cannot be exceeded when resizing the split area with the mouse. *.maxsize[0]* acts as default value for all split areas that have no *maxsize* defined. See also chapters “Setting the Sizes of the Split Areas” and “Particularities”.

Attribute	Desciption
minsize[I]	Sets the minimum size for split area I, which cannot be underrun when resizing the split area with the mouse. *.minsize[0]* acts as default value for all split areas that have no *minsize* defined. See also chapters “Setting the Sizes of the Split Areas” and “Particularities”.
options[enum]	Options of the splitbox; please refer to the “Attribute Reference” for details. Allowed indexes: *opt_center_toolhelp* (Microsoft Windows only)
real_size[I]	Returns the actual size of split area I. See also chapters “Setting the Sizes of the Split Areas” and “Particularities”.
size[I]	Defines the width (*.direction* = *1*) or height (*.direction* = *2*) of split area I. *.size[0]* is the default value for all split areas that have no size set. See also chapters “Setting the Sizes of the Split Areas” and “Particularities”.
tile	Sets a background image for the object.
tilestyle	Controls how the background image set in *.tile* is arranged.

36.2.1 Setting the Sizes of the Split Areas

The sizes of the individual split areas are defined through the size[I] attribute.

If interactive resizing by the user shall be limited between minimal and maximal sizes, these limits can be set in the attributes minsize[I] and maxsize[I].

For all three attributes, the value of the item with *I = 0* is the default value for items that are not set. The values of the items *1 … .childcount* are used for the particular split areas. The details of the interaction between these attributes, especially with resizing by the user, are explained in the subsequent chapters.

The current size of the split areas can be queried through the attribute real_size[I].

36.3 Particularities

Concerning the splitbox, special attention has to be paid to the layout, the geometry management, and constraints like an active size raster or the adherence of minimum and maximum sizes.

36.3.1 Size Raster

In contrast to other objects, which allow for the use of attributes like *.xraster*, *.yraster* and *.reffont*, the raster grid of the splitbox is not applied to the entire object but rather to the individual split areas. This avoids problems with rounding differences, which would be inevitable if one had to align separate split areas to a global raster while at the same time having to fit splitbars (with widths given in pixels) into the grid. Adapting the splitbar widths to raster units would look really bad in most cases. Relating the raster grid to the individual split areas solves these problems. Additionally this simplifies the usual calculation rule for pixel values. The common rule is *pixel value = (raster value – 1) * raster*. For calculating the sizes of split areas it is *pixel value = raster value * raster*.

36.3.2 How the Sizes of the Split Areas are Calculated

The algorithm outlined below is used to calculate the actual sizes of the split areas:

1. The first approach is to assign each split area the size given in the *.size[]* attribute.
2. If the first step yields, that more space is needed for all split areas including their splitbars than the object can provide with its constellation of geometry attributes like *.width*, *.height*, *.xauto*, *.yauto*, etc., the split areas are scaled down beginning at the right or bottom and obeying minimum sizes until all areas fit into the available space. If this cannot be achieved, for example because the sum of the minimum sizes (*.minsize[I]*) already exceeds the available space, the split areas are gone through once more starting at the right or bottom. This time they are diminished ignoring their minimum values. This may lead to split areas with size 0 at the right or bottom of the splitbox, for which only the splitbars remain visible. If the sizes of all split areas have been set to 0 and there still is too little space (the width of the splitbars altogether is larger than the splitbox dimension), the outermost split areas with their splitbars, for which the space does not suffice, are moved into the invisible area to the right or bottom.
3. If after the first step less space is occupied than has been provided through the settings of the geometry attributes, the split areas are expanded until the entire space is utilized, but not beyond the maximum values given in *.maxsize[I]*. If the available space cannot be utilized completely with this approach, because the sum of all maximum sizes and splitbar widths is smaller than the splitbox dimension, the last split area is enlarged under defiance of its maximum size to fill the remaining space.

During this process the attributes *.size[I]*, *.minsize[I]*, and *.maxsize[I]* are not changed. The attributes keep the values given to them by the programmer, but these can be only partially respected due to the existing constraints. This property enables to recompute the widths of the split areas whenever the dimensions of the splitbox change. If, for example, a splitbox is attached to a window with *.xauto=0* and *.yauto=0*, it can be observed how the split areas one after the other are reduced with and then without considering of the minimum values, when the window size is reduced interactively. When the window is enlarged again, the splitbars return to their original positions.

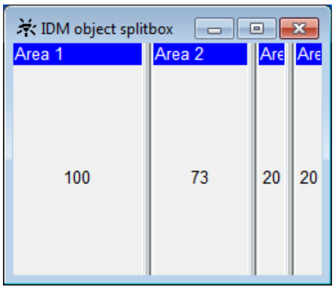
The values in *.size[I]* are only changed when they conflict with the minimum and maximum values in *.minsize[I]* and *.maxsize[I]*. The values in *.minsize[I]* have a higher priority for the IDM as the values in *.maxsize[I]*.

In the case that the minimum and maximum values are violated, as described in steps 2 and 3 above, the affected splitbars can no longer be interactively moved since they already have illicit positions. However, dragging these splitbars changes the size of the respective split areas and sets new values in the *size* vector. As the attribute values in *.size[I]* change, *resize* events are triggered, although there is no visible change. So it can be handled programmatically that the original values in the *size* vector have been changed through user interaction. When the same splitbars are dragged once more, again there will be not visible change as the restrictions resulting from the minimum and maximum values are still in force. This time no further *resize* events are sent since the suitable values in *size[I]* had already been set with the first drag.

A particularity has to be noted concerning size raster. Since it is virtually impossible to perfectly fit the sizes of all split areas to the size raster with the adaptation process described above, normally there will be one (and no more than one) split area whose size is no multiple of the grid size. This applies to the split area that has lastly been adapted by the described algorithm.

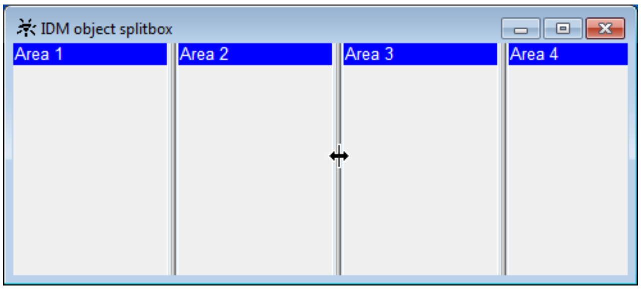
The following is an example for this situation:

Let there be a splitbox with *.direction* = *1*, *.size[0]* = *10* and *.minsize[0]* = *2*. Size raster is activated and *.xraster* = *10*. The splitbox is tied to its father object with *.xauto* = *0* and *.yauto* = *0*. This splitbox is depicted below. Due to external constraints not all values from the .size[I] vector can be satisfied. The two split areas to the right already reached their minimum sizes of 20 pixels. The first split area has the desired size of 100 pixels. The second split area from the left has been changed last by the described algorithm and could not be fitted into the size raster.



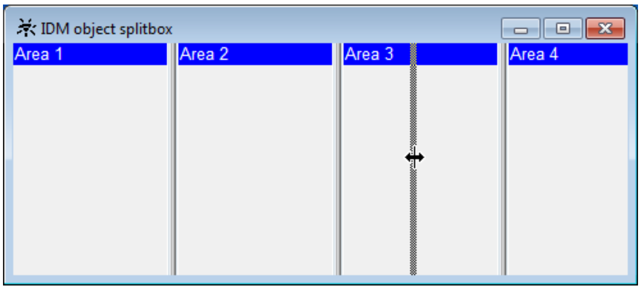
36.3.3 Interactive Resizing

A typical use of the splitbox is that the user can interactively change the sizes of the split areas. Therefor the mouse cursor has to be positioned on one of the splitbars. As shown in the image below the shape of the mouse cursor changes on the splitbars.

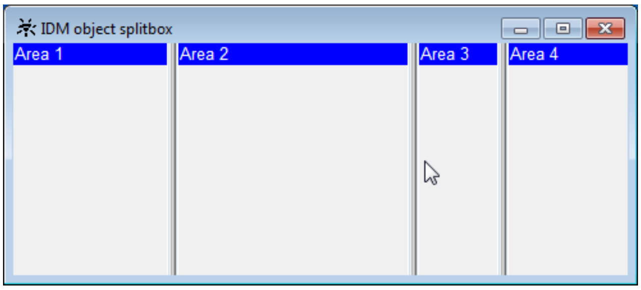


Then the splitbar can be dragged by moving the mouse right or left (in this case) with the left button pressed until the desired size of the split areas is reached. The following image shows that as a visual feedback a dimmed splitbar is displayed which follows the mouse cursor while dragging. If one of the maximum or minimum sizes is violated while the splitbar is moved (i.e. the maximum size of the second area is exceeded or the minimum size of the third area is underrun), the dimmed splitbar remains at the last valid position.

On MICROSOFT WINDOWS the resizing of a split area can be canceled with the `Escape` key.



When the desired size has been reached, the left mouse button can be released. Now the splitbox sets the new sizes of the split areas. The result is shown in the next image.



Now, up to two *resize* events are sent. One, which notifies that the size of the second split area has changed (of relevance is a new value in the *size* vector; no event is triggered if the respective value remains unchanged). In *thisevent.index* the (one-based) index of the child object in the affected split area is registered. The second *resize* event belongs to the change of the third split area. Accordingly, *thisevent.index* is set to the index of the third splitbox child.

Note

When *.sizeraster* is set to true on the splitbox, this has no effect while dragging a splitbar. It is moved simultaneously with the mouse cursor. Only after the splitbar is released, it snaps to the closest upper or left grid position so that the adjacent split areas fit into the size raster.

36.4 Example

A simple use of the splitbox can look like this:

```
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 section 1";
  }

  child statictext St1
  {
    .text  "in section 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]  "one";
    .content[2]  "a";
    .content[3]  "b";
```

```
      .content[4]  "c";
      .content[5]  "two";
      .content[6]  "d";
      .content[7]  "three";
      .content[8]  "four";
      .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(); }
}
```

37 statictext

The keyword **statictext** is used for the definition of the text which is not interactively modifiable by the user. Single-line texts can be positioned and displayed in one window.

Definition

```
{ export | reexport } { model } statictext { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

cut

extevent

focus

help

key

paste

select

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

spinbox

splitbox

toolbar

window

Menu

Pop-up menu

37.1 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
alignment	integer (-1, 0, 1)	integer	S,G/D/C	alignment of text
bgc	identifier	color	S,G/D/C	background color
class	class	class	-,G/-/-	object class
control	identifier	instance	-,G/-/-	control the object currently belongs to
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation has not been executed yet
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during a cut operation
depth	integer	integer	S,G/-/C	controls appearance of statictext
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
document[I]	object	document	S,G/-/-	accesses the I-th XML Document

Attribute	RLD	PID	Properties	Short Description
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
focus_on_click	boolean	boolean	S,G/-/-	defines if a mouse click into an object's client area activates the object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
groupbox	identifier	instance	-,G/-/-	object groupbox
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu

Attribute	RLD	PID	Properties	Short Description
model	identifier	instance	S,G/D/C	model belonging to object
notepage	object	object	-,G/-/-	notepage which object currently belongs to
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string object	string text	S,G/-/C	text to be displayed in statusbar
text	string identifier	string text	S,G/D/C	static text of object
toolbar	object	object	-,G/-/-	toolbar of object

Attribute	RLD	PID	Properties	Short Description
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM datatype)
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	actual width of object
window	identifier	instance	-,G/-/-	window to which object belongs
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xright	integer	integer	S,G/D/C	x-coordinate, distance from right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

37.2 Specific Attributes

These attributes define the position and the size of the statictext. If *.height* or *.width* is specified as 0, the DM calculates the size of the statictext on the basis of the actual font.

The statictext if it is the child of a statusbar requires the additional attribute *.depth*. This attribute controls the appearance of the statictext.

The value range of this attribute is integer, the default is 0. Negative values mean that the statictext will be displayed **intaglio** in contrast to the parent object. Positive values mean that the statictext will be displayed in highlighted form. The value corresponds to the negative values used for the highlighting. The attribute is inherited as all other attributes.

This attribute will only have an effect if the statictext is a child of an statusbar.

The attribute .options can have the following values:

option_index	Meaning
opt_use_widget	the statictext will be displayed as a widget (only with Motif)

37.3 Example

Text in a window

```
window MAIN
{
  .title "main window";
  child statictext FixText
  {
    .xleft  10;
    .ytop   10;
    .text   "Smith Ltd.";
  }
}
```

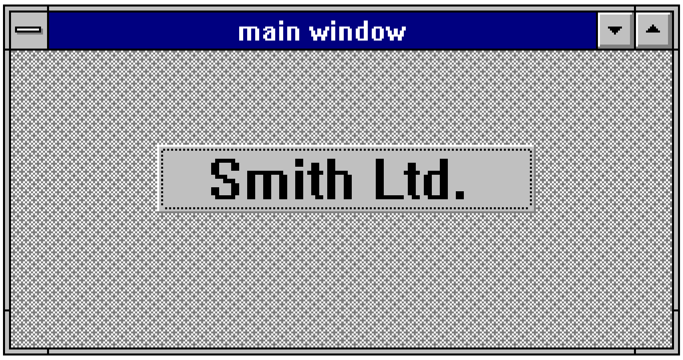


Figure 40: Statictext

38 statusbar

The ***statusbar*** is used to display status information and helptexts. It is situated at the bottom line of the window and will not alter the size of the work area. The frame window is enlarged by the ***statusbar***.

Definition

```
{ export | reexport } { model } statusbar { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Events

extevent

Children

document

image

progressbar

record

statictext

transformer

Parent

module

window

Menu

none

38.1 Attributes

Attribute	RSD	PSD	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	object class
control	identifier	instance	-,G/-/-	control the object currently belongs to
dialog	identifier	instance	-,G/-/-	dialog for object
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color of object
firstchild	object	object	S,G/-/C	accesses the first child object

Attribute	RSD	PSD	Properties	Short Description
firstrecord	object	record	S,G/-/C	accesses the first record of an object
font	identifier	font	S,G/D/C	font of object
helppos	integer	integer	S,G/D/C	position of helptext
height	integer	integer	S,G/D/C	will be ignored
index	integer	integer	-,G/-/-	current index of object in the children list of its parent
label	string	string	S,G/D/C	identifier of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
layoutbox	object	object	-,G/-/-	layoutbox of object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-/-	i-th user-defined attribute of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
model	identifier	instance	S,G/D/C	model belonging to the object
parent	identifier	instance	S,G/-/-	parent of object
real_height	integer	integer	-,G/-/-	real height of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object

Attribute	RSD	PSD	Properties	Short Description
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectability of object, is always false
sizeraster	boolean	boolean	S,G/D/C	size refers to the raster of parent object
tile	object	tile	S,G/D/C	tile resource used as background image
tilestyle	enum	enum	S,G/D/C	controls how the background image set in *.tile* is arranged
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
userdata	anyvalue	anyvalue	S,G/D/C	userdata of object of any data type
visible	boolean	boolean	S,G/D/C	visibility of object
vsb_visible	boolean	boolean	S,G/D/C	visibility of vertical scrollbar
window	identifier	instance	-,G/-/-	window the object currrently belongs to

38.2 Specific Attributes

Statusbars can be used in top-level windows and child windows. If a child window has no visible statusbar, statushelp texts will be displayed in the statusbar of the parent window (if available).

The helptexts are set in the corresponding objects with the attribute .statushelp. The helptexts will be displayed if a menu is being selected or if the mouse is being moved to an object.

Only statictexts are valid as child objects. The attribute .width of the statictext specifies whether the statustext will be flat, sunken or raised. The texts will be displayed in the order of the children from the left to the right with the width specified in .width. The last child is enlarged up to the right margin. To avoid this you can add a “dummy”.

A special feature of the statictexts in the statusbar is the possibility to position texts with "\t".

The following graphic shows the schematic structure of a statusbar:

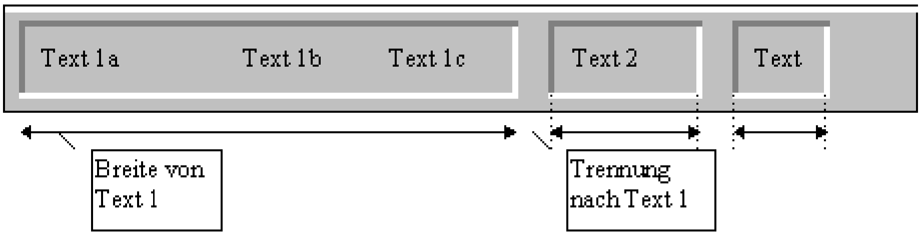


Figure 41: Schematic structure of a statusbar

A statusbar consists of up to 255 text fields which can either be connected to or separated from each other. In front of the first and after the last text field there is a "separation". The text is left-aligned within a text field. The "separation" between two text fields has the width of one raster unit, if the raster is switched on, otherwise the width will depend on the system.

The statusbar can also display a helptext. This helptext can overlap all or several children, with the result that important information is still visible.

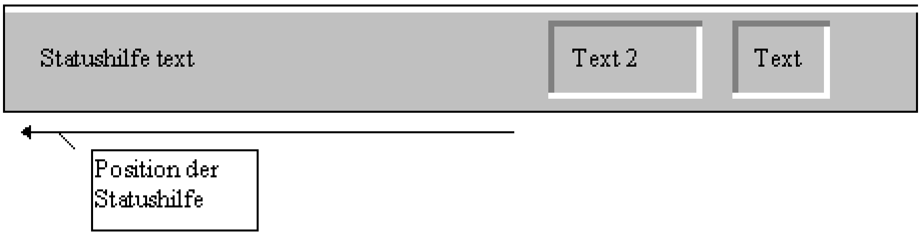


Figure 42: Display of helptext in a statusbar

Geometry Attributes

The usual geometry attributes (width, height, position) will all be ignored, as the position of the statusbar is invariable. The statusbar is situated at the bottom line of the window, below the horizontal scrollbar if available. The statusbar extends over the entire window width; its height is defined by the indicated font.

The statusbar is not scrolled along with the window contents. This is why you have to note that the statusbar information remains visible in its totality. When calculating the geometry of the window the statusbar is not considered as a part of the window contents like the scrollbar but of the window frame.

Only the attributes .xraster and .yraster will be considered as they are useful when calculating the position of children.

Standard Attributes

Only the standard attribute .visible is considered. With this attribute the statusbar can be made visible or invisible. The attribute .sensitive is ignored as the statusbar is not selectable.

Layout Attributes

.borderstyle

Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.

.font

The font to be used for the text display is defined with the attribute .font. Fonts which have been defined at the corresponding objects will be ignored.

Specific Attributes

.helppos: This attribute indicates in which position the helptext *.statushelp* (see below) is to be displayed. *0* (default) defines a full-line representation of the help information. The value n defines the representation of the helptext via all children up to the **n-th** child. If the value is *<0* no help information will be displayed. This attribute may vary from window system to window system. The values *0* and *-1* can always be set and will be displayed according to the description above.

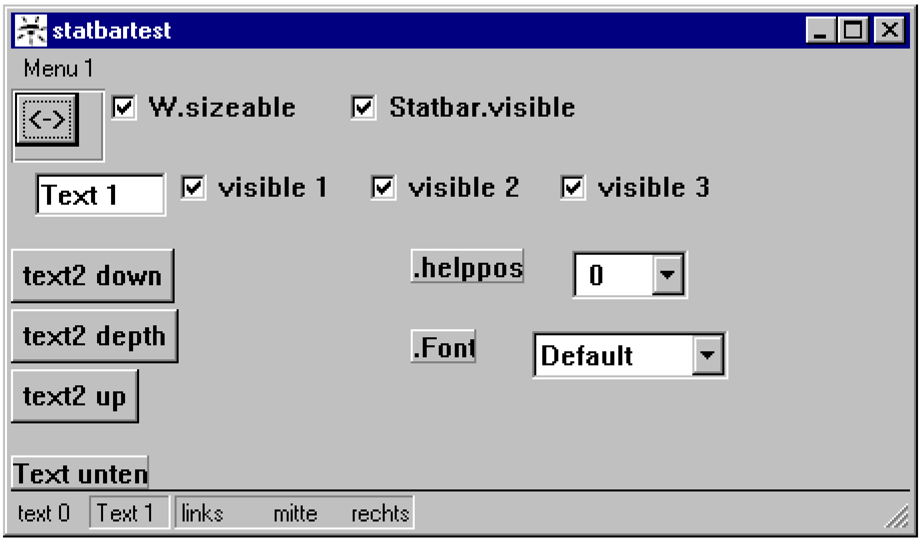
.sizeraster: This attribute is used to calculate the sizeraster of the fields within the statusbar. The following values are valid:

- *false* (default): The size of the fields will be indicated in pixel. The distance between two text fields depends on the system.
- *true*: The size of the fields will be indicated in raster coordinates. The distance between two text fields is a raster.

38.3 Example

Window with ***statusbar*** which may be changed by window settings.

```
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;
    }
  }
}
```



☑ W.sizeable ☑ Statbar.visible

Text 1 ☑ visible 1 ☑ visible 2 ☑ visible 3

.helppos 0

.Font Default

Figure 43: *Fenster mit statusbar-*

Figure 44: *window with statusbar*

39 subcontrol

OLE objects that can be used as servers by the IDM through the IDM object ***control*** often have a rather complex structure and can consist of several further child objects. For example, the OLE control “Word.Application” has a collection of “Documents”, which manages the open documents. In turn, these documents have “Words”, “Sentences”, “Ranges”, etc. as their own children. To be able to access these sub-objects from the Rule Language, i.e. to call methods or query attributes, the ***subcontrol*** object can be used. The ***subcontrol*** represents an OLE child object which can be directly or indirectly managed by an OLE server. Therefore the starting point for accessing such an object is always the ***control***.

Definition

```
{ export | reexport } subcontrol { <Identifier> }
{
  <hierarchy attributes>
  <object-specific attributes>
}
```

Events

None

Children

document

record

subcontrol

transformer

Parents

control

subcontrol

Menu

None

39.1 Attributes

connect

control

dialog

document[I]

firstrecord

firstsubcontrol

groupbox

label

lastrecord

lastsubcontrol

layoutbox

model

module

notepage

parent

record[I]

recordcount

scope

subcontrol[I]

subcontrolcount

toolbar

transformer[I]

userdata

window

39.2 Implicit Creation

One notable feature of subcontrols is that these objects can be implicitly created without being statically defined as object in a dialog or being dynamically created with **create()** before. But please pay attention to chapter “Dynamic OLE Properties and Subcontrols” when you work with dynamic OLE properties which create subcontrols.

Probably the implicit use is the most common use of these objects in practice. For example, when using Microsoft Word as an OLE server the individual can be accessed like this:

Let ***control***“Co” be defined somewhere in the dialog.

```
child control Co
{
  .mode mode_client;
  .name "Word.Application";
  .visible true;
  .connect true;
}
```

Then control Co can be used in a rule like this:

```
rule OLETest ()
{
  variable object Doc;
  Doc := Co.Documents:Add();
  // Word has a collection that is entitled "Documents".
  // Co.Documents is used to access this OLE object. In order to
  // address the object in IDM, a subcontrol with the identifier
  // "Documents" is implicitly created and registered as a child
  // of control Co. The :Add() method is a member of the
  // "Documents" collection. Therefor, the method can be applied
  // to the just created subcontrol. It creates a new document
  // in Word and returns a pointer to its IDispatch interface.
  // As a counterpart on the IDM side another subcontrol is
  // created as child of the first subcontrol and connected
  // with the Word document. Finally this object is assigned
  // to the variable "Doc".

  // The subcontrol in "Doc" now can be used to call methods of
  // the OLE document and to query or set its properties.
  print Doc.FullName;
}
```

In general, if a method or a property returns a pointer to an IDispatch interface, or if this is passed to the IDM as a parameter, the IDM creates a subcontrol that is connected to the IDispatch interface. This means that the *.connect* attribute of such a subcontrol is already set to *true*.

Normally the identifier (label) of such a created object is not set so that something similar to *"subcontrol Co.SUBCONTROL[2]"* appears in the trace file. One exception is the access of properties that return OLE objects. Since these are usually collections whose names can be known to the IDM, an implicitly created subcontrol receives the name of the property as identifier.

Example

```
Doc1 := Co.Documents:Add();
Doc2 := Co.Documents:Item(1); // Assumption: No other Word documents
                              // were open before.
```

In this example, two subcontrols are created in the first line: One with the identifier "Documents", another with the identifier "SUBCONTROL". The reason for the identifier of the second subcontrol is that the IDM has no further information when processing the Add() method. In the second line only one more subcontrol is created with "SUBCONTROL[2]" as its identifier. As the IDM recognizes that a subcontrol for "Documents" already exists, it is unnecessary to create it once more. All of this leads to the situation, that on the OLE side there are two objects: one is the "Documents" collection and the other one an empty document. On the IDM side there are three objects: a subcontrol for the

"Documents" collection and two subcontrols in the variables "Doc1" and "Doc2", both connected to the empty document on the OLE side. It is a matter of good programming style to avoid heaps of unnecessary subcontrol objects.

One may wonder at this point, what happens to all these subcontrols if they are not needed anymore. Chapter "Garbage Collection" deals with this.

39.3 Dynamic OLE Properties and Subcontrols

When dynamic properties of OLE objects are queried, this may lead to the creation of new subcontrols when these properties return objects (e.g. "ActiveSheet" of Microsoft Excel). At first this is no problem. In subsequent queries however, the IDM will not refer to a current, newly returned object but to the already existing subcontrol which has been cached by the IDM.

If this caching is unwanted or leads to problems, it can be avoided by two means:

- After the access of a OLE property that creates a subcontrol, this subcontrol can be destroyed with the **:destroy()** method before the next access of the OLE property (provided that it is no longer needed).
- After the access of a OLE property that creates a subcontrol, the *.label* attribute of this subcontrol can be set to an empty string (""). Hereby the subcontrol object remains connected and available. It will be destroyed within the usual garbage collection (see chapter "Garbage Collection"). Further accesses of the OLE property will create a new subcontrol as the prior object will not be found (internal search relies on *.label*).

Setting *.connect* of the concerned subcontrol to *false* provides no solution however, because with this the subcontrol persists, only all further accesses will fail.

39.4 Garbage Collection

All implicitly created subcontrols have to be deleted again. For this purpose, the IDM remembers by how many objects a subcontrol is referenced. If it is determined that a subcontrol is no longer in use, neither by a variable (local, global, static) nor by a user-defined attribute, then the subcontrol is destroyed. Currently two strategies are applied, which are explained below.

1. If an implicitly created subcontrol is assigned to a variable (or something similar), and later this variable is overwritten with another value, the subcontrol can be released, provided that it has no children.

   Example

```
Doc := Co.Documents:Add();  // 2 Subcontrols are implicitly created (A for
Documents
                            // and B for the document created by the Add()
method).
Doc := Co;                  // Subcontrol B is no longer used by the
variable Doc
```

```
                            // and therefore can be destroyed. Now
subcontrol A can
                            // determine that it has no more children and
can destroy
                            // itself too.
```

2. In awkward situations it may happen that a subcontrol does not notice it can be destroyed. This happens for example, when a newly created subcontrol has not been assigned anywhere. This situation lacks a trigger to throw away the object. For this reason the IDM occasionally starts a cleanup in which implicitly created subcontrols are checked if they are still required. Otherwise they are discarded.

Important

Since the IDM garbage collection only takes into consideration the references from the Rule Language, great care should be taken, if ObjectIDs of subcontrols are managed from the C interface (as well as the COBOL and C++ interfaces). If such an ID is temporarily stored in the application, this is not realized by the IDM. After the call of interface functions, the IDM may determine that the subcontrol is not needed anymore and dispose it accordingly. If control is transferred back to the application side, it can no longer be assumed that the previously saved ObjectID is still valid. Therefore it should be avoided to save subcontrols in the application. If this is wanted still, it is important to ensure that the object is still referenced in the Rule Language (for example by a global or static variable, or in a user-defined attribute). This guarantees that the subcontrol will not be thrown away.

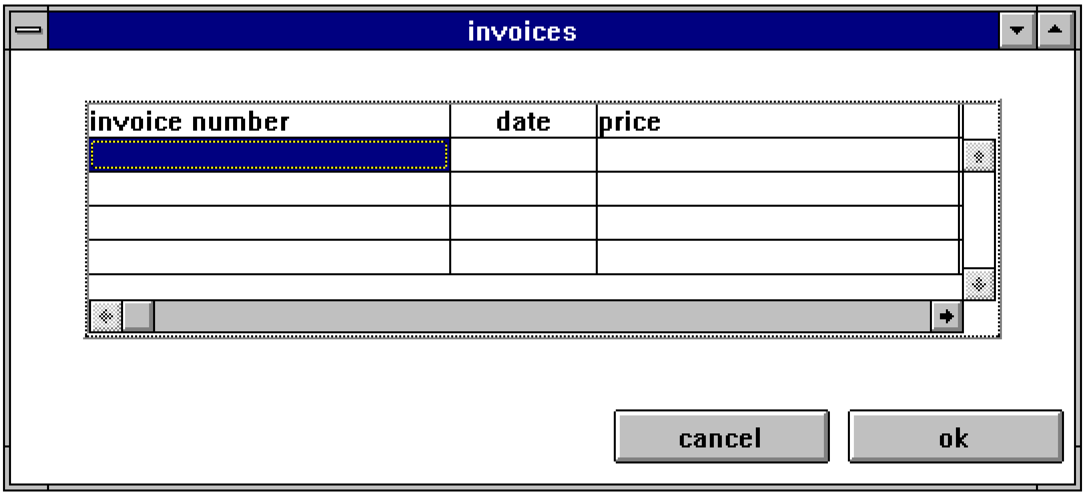
39.4.1 Example

The following example shows how subcontrols can be used.

```
dialog Word

window WnWindow
{
  .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;

      // Assumes that the file actually exists
      Doc := Co.Documents:Open("c:/tmp/olddocument.docx");

      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
    {
      // Assumes that the file actually exists
      Co.Documents:Add("c:/tmp/newdocument.docx");
    }
  }

  child pushbutton PbClose
  {
    .xleft 180;
    .width 92;
    .yauto -1;
    .height 27;
    .ybottom 10;
    .text "Close Doc";
    on select
    {
      // Close first document (of the document list)
      Co.Documents:Item(1):Close();
    }
  }

  child pushbutton PbQuit
  {
    .xauto -1;
    .width 85;
    .xright 20;
    .yauto -1;
    .height 27;
    .ybottom 10;
    .text "Quit";
    on select
    {
      // Close Word
      Co:Quit();
    }
 }
}

on dialog start
{
  Co.Visible := true;
}
```

40 tablefield

By means of the ***tablefield*** object, arbitrarily formatted lists can be output and edited.



invoice number	date	price

Figure 45: *tablefield*

Definition

```
{ export | reexport } { model } tablefield { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <grid attributes>
  <object-specific attributes>
}
```

Using the two scroll bars, the contents of the tablefield can be scrolled both horizontally and vertically. The content of the tablefield consists of static texts that can be selected.

Events

activate

cut

charinput

deactivate

dbselect

extevent

focus

help

hscroll

key

modified

paste

scroll

select

vscroll

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up Menu

Methods

:clear()

:delete()

:exchange()

:find()

:insert()

Remark

To allow the dialog programmer to distinguish more precisely what the user has really done, additional querying of attributes may be necessary. The editable text belonging to the tablefield does not send any events. These are all sent to the tablefield and can be queried there.

Remarks for important events of the tablefield

The select event is triggered whenever the user clicks the mouse (exception see below “Release mouse capture” or when the user presses the `space` bar.

The *select* event is available in two forms:

- With Index
  A sensitive field has been selected.
- Without Index
  A *select* event has occurred that cannot be assigned to a single field. I.e. the user has either clicked in an insensitive field or in an area where there are no fields.

The dbselect event is triggered whenever the user double-clicks the mouse (exception see“Release capture” below) or when the user presses the `Ctrl` and `Return` keys simultaneously.

The *dbselect* event comes in two forms:

- With Index
  A *dbselect* event has occurred in a field that is sensitive.
- Without Index
  A *dbselect* event has occurred that cannot be assigned to a single field. I.e. the user has either clicked into an insensitive field or into an area where there are no fields.

However, for DM with Motif, the *dbselect* event is always sent with index. *dbselect* for insensitive fields does not exist.

The focus event is generated whenever the focus object or the focus field changes. The *focus* event comes in two forms:

- With Index
  A new field that can be focused has received the focus.
- Without Index
  The tablefield has received the focus. There is no field that can receive the focus.

The modified event contains the index of the field that was modified.

The *activate* or *deactivate* events are generated without an index, since the activation state usually changes for more than one field.

Handling of events with index

It must always be ensured that - before accessing an index - it is checked whether an index is set at all. A check can be done with the following code fragment:

```
if (typeof (thisevent.index) = index) then
  /*
   * here you can work with the index.
   * "thisevent.index.first" corresponds to the line and
   * "thisevent.index.second" .corresponds to the column
   */
else

endif
```

Implicit reset of tablefield states when changing tablefield attributes

For MOTIF, the content is not discarded for any of the attributes listed below.

The following list is for MICROSOFT WINDOWS.

The list does not claim to be complete. The index attributes for the inside of the tablefield are not included in the list; however, the corresponding applies.

Deactivate the edit text without saving the content

Regardless of edittype, the edit text is deactivated and the possibly changed content is discarded for the following attribute changes:

colcount, colvisible[I], edittext, rowcount, rowvisible[I]

As of IDM version A.05.02.h, using a method that changes the content (e.g. **:insert**, **:exchange**) causes the linked edit text to be deactivated without saving its content. In previous IDM versions, the edit text remained enabled, but not all changes were displayed immediately.

Deactivating the edit text with saving the content

With the following attribute changes the edit text is deactivated and the possibly changed content is saved or discarded. If *.edittype = locking*, the edit text is disabled and the content is discarded.

If the active attribute is implemented without an index, the edit text is enabled or disabled, and the content is saved only if *.edittype* is not *locking*.

Release mouse capture

When the left mouse button is pressed, a mouse capture (exclusive mouse access) is fetched. When the mouse button is released, this mouse capture is released again and a *select* event is generated. **No** *select* event is generated if no mouse capture is active when the left mouse button is released. The mouse capture is also released on the attribute changes listed below. Thus, it is possible that *select* events are prevented. This risk exists especially when processing *focus* events, since the focus is triggered when the mouse buttons are pressed. This complicated processing is justified to be able to assign mouse clicks correctly.

Remark

This behavior does not apply to DM with MOTIF.

40.1 attributes

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

40.2 Specific attributes

attribute	Description
accelerator	Accelerator of the object. If an accelerator is assigned to a tablefield and this is pressed, the focus is set to the field in the tablefield that last had the focus. If no element had the focus yet, the focus is set to the upper left field inside the tablefield.
active[I,J]	Activity state of the individual cell of a tablefield.
activeitem	Active element of the object.
bgc[I,J]	Background color of a cell of the tablefield.
borderraster	Defines the type of geometry calculation when the grid is active. The detailed description can be found in the attribute description in the „Attribute referenz“.
colalignment[I]	Text alignment in a column. See also chapter “Text alignment in tablefield”.

attribute	Description
colcount	Number of columns.
colfirst	First column of the tablefield displayed next to the column headers.
colheader	Number of column headers.
colheadfgc	Foreground color of the column headers.
colheadfont	Character set used in the column headers.
colheadshadow	Shadow display (similar to a button) of the column headers.
colheadvisible	Visibility of column headers.
collinewidth[I]	Width of the closing line of a column. See also chapter “Size calculation in tablefield”.
colsizeable[I]	Controls the interactive magnification of the column.
colwidth[I]	Specification of the column width.
colvisible[I]	Visibility of the column.
content[I,J]	Contents of a tablefield cell.
contentfunc	Defines the function to be called for dynamic reloading of the tablefield.
direction	Controls the alignment of the tablefield. **See also** Chapter “Alignment of the tablefield”
editpos	Defines how and where the tablefield is edited.
editable	Determines the general editability of the whole tablefield.
editable[I,J]	Determines the editability of a cell of the tablefield.
edittext	Defines the identifier of the editable text associated with a tablefield. This editable text must have the same father as the tablefield.
edittype	Describes how the editable text associated with the tablefield and the contents of the tablefield work together. Values: *edit_locking*, *edit_offline*, *edit_online*.
fgc	Foreground color of the object.
fgc[I,J]	Foreground color of a cell of the tablefield.

attribute	Description
field[I,J]	Sets each individual text field in the tablefield, but not column or row headers.lefields.
fieldactive[I,J]	Activation state of each text field in the tablefield, but without column or row headers.
fieldfocus	Query or set the field inside the table that has or should have the focus.
fieldfocus[I,J]	Sets the focus to a single text field in the tablefield, but without column or row headers.
fieldfocusable	Defines whether the cursor control should also consider non-sensitive elements in the tablefield or not. This attribute is only evaluated if .sensitive of this field is false.
fieldshadow	Display of the sensitive fields of the tablefield with or without shadows.
focus	The tablefield gets the focus.
focus[I,J]	A certain cell of the tablefield gets or has the focus.
font	Specification of the character set belonging to the object.
font[I,J]	Specifying the character set belonging to a cell of the object.
format[I,J]	Specify the format belonging to a cell of the tablefield.
formatfunc	Specifies the format function associated with the object.
function	Defines the function belonging to the object.
height	Height of the object. See also chapter “Size calculation in tablefield”.
hsb_optional	Determines whether the object's horizontal scrollbar should always be displayed or only when necessary.
maxchars[I,J]	Maximum number of possible characters in the input string of a tablefield cell.
navigable	Controls the focyusability (accessibility of the object by keyboard).
nextactive[I,J]	Next active cell at tablefield and selstyle *single*.

attribute	Description
options[enum]	Options of the object. Indexes: » *opt_center_toolhelp* (only MS Windows). » *true*: If there is enough space for it, the toolhelp is displayed centered below the object it belongs to. » *false* (default): The toolhelp is displayed at the mouse pointer. » *opt_new_align* (Motif and MS Windows). » *true*: The attributes .xalignment[I,J] and .yalignment[I,J] are decisive for the text alignment. » *false* (default): The attributes .colalignment[I] and .rowalignment[I] are decisive for the text alignment. » See also chapter “Text alignment in tablefield”. » *opt_new_colwidth* (only Motif). In older versions of IDM (before A.03.10.f), the column width was calculated incorrectly under Motif for tables with a *reffont*. There was a **second** fix for this in version A.03.10.f, where the *opt_new_colwidth* option was introduced. » *true* (default): Corrected column width (calculation based on grid sizes, as in the layout of other objects). » *false*: Calculate the column width according to the *opt_old_colwidth* option. » *opt_old_colwidth* (only Motif). In older versions of IDM (before A.03.04.a) the columns were too wide under Motif for tables with a *reffont*. There was a **first** fix for this in version A.03.04.a, where the *opt_old_colwidth* option was introduced. » *true*: Column width as before correction. » *false* (default): Corrected, smaller column width. **Note** » As of version A.03.10.f, the column width has been corrected again, introducing the opt_new_colwidth option (see above). If opt_new_colwidth has the value true, then this setting takes precedence over opt_old_colwidth. » *opt_old_select* (only Motif, since IDM-Version A.05.02.h). This option can be used to set a select-oriented or an action-oriented dispatch of *select* events. This enables a consistent behavior of the Motif and Windows version of the IDM. » *true*: Select events are sent when the activation state changes (selec-

attribute	Description
	tion-oriented). This corresponds to the behavior up to and including IDM version A.05.02.g. » *false* (default): Select events are sent when the mouse is clicked or the selection key is pressed (action-oriented). This corresponds to the behavior under Microsoft Windows. Thus, the select event no longer indicates a change in the activation state. In order to react to changes in the selection even when *.options[opt_old_select] = false*, the "activate" and "deactivate" events can be used.
preeditsel	Controls the initial selection of the text when the edit text of a tablefield is explicitly or implicitly activated. Values: *presel_default*, *presel_all*, *presel_begin*, *presel_end*.
reffont	Reference font for size calculation of the object. See also chapter "Size calculation in tablefield".
rowalignment[I]	Text alignment in one row. See also chapter "Text alignment in tablefield".
rowcount	Number of rows.
rowfirst	First row of the tablefield displayed below the header rows.
rowheader	Number of tablefield headers.
rowheadfgc	Foreground color of the headers.
rowheadfont	Character set used in the headers.
rowheadshadow	Shadow display (similar to a button) of the headers.
rowheadvisible	Visibility of the headers.
rowheight[I]	Specification of the row height. See also chapter "Size calculation in tablefield".
rowlinewidth[I]	Width of the closing line of a row.
rowsizeable[I]	Controls the interactive magnification of the line.
rowvisible[I]	Visibility of the line.
selection[enum]	Allowed type of selection in the tablefield. Indexes: *sel_column*, *sel_header*, *sel_row*, *sel_single*.

attribute	Description
selstyle	Control of the selection type within the tablefield. Values: *single*, *multiple*, *extended*.
sensitive	Defines whether the tablefield should be selectable. This allows the object to be made insensitive in one step.
sensitive[I,J]	Sensitivity control of each cell of the object.
sizeraster	Activation of the size raster. See also chapter “Size calculation in tablefield”.
userdata[I,J]	Userdata for any data about any cell of the object.
vsb_optional	Determines whether the vertical scrollbar of the object should always be displayed or only when necessary.
width	Width of the object. See also chapter “Size calculation in tablefield”.
xalignment[I,J]	Describes the text alignment within a cell in horizontal direction. See also chapter “Text alignment in tablefield”.
xraster	Horizontal raster of the object See also chapter “Size calculation in tablefield”.
yalignment[I,J]	Describes the text alignment within a cell in vertical direction. See also chapter “Text alignment in tablefield”.
yraster	Vertical raster of the object. See also chapter “Size calculation in tablefield”.

40.2.1 Size calculation in tablefield

The height and width attributes can also have the value 0. This means that the Dialog Manager should calculate the corresponding size so that all elements can be displayed in the corresponding direction without the need for a scrollbar. For example, if the width of the tablefield is set to 0, it will be so wide that all columns can be completely displayed in the object.

If the tablefield has sizeraster set, and both xraster/yraster and reffont = 0, the raster definition of the direct parent object is taken. So the tablefield calculates its own size like a normal object from the raster factors of its parent. Only the size definitions inside the tablefield, collinewidth[I] and rowheight[I], refer to the raster attributes defined for the tablefield.

However, the pixel values are calculated according to the method defined for the tablefield.

40.2.2 Text alignment in tablefield

The horizontal and vertical alignment of text in a tablefield cell is set by the colalignment[I] and rowalignment[I] attributes, but the horizontal alignment is set only column-wise and the vertical alignment is set only row-wise.

Using the options[opt_new_align] option with the value true, the *.xalignment[I,J]* and *.yalignment[I,J]* attributes are taken into account instead of *.colalignment[I]* and *.rowalignment[I]*. These have the same value range/meaning as *.colalignment[I]* and *.rowalignment[I]*; however, *.xalignment[I,J]* and *.yalignment[I,J]* are double-indexed and inheritance is by direction as with all other double-indexed attributes.

The default value of *.options[opt_new_align]* is *false*.

Supported under Motif and MS Windows.

40.2.3 Attribute default values

For some attributes *.attribute[0]* is defined as default. This is always taken if no more value is available for the really needed value. The first correctly usable element is always *.attribute[1]*!

For some attributes *.attribute[0,0]* or *.attribute[y,0]* and *.attribute[0,x]* is defined as default. These are always taken if no more value is available for the really needed value. The first correctly usable element is always *.attribute[1,1]*! With the use of these Defaults two-stage one proceeds:

- If no value is defined for the required index, the index defined by the *.direction* attribute is set to 0. If a value is defined here, it will be taken
- However, if no value is defined here either, the global default for the attribute at *.attribute[0,0]* is taken.

Because of this approach, you should always define a default *.attribute[0,0]* for used attributes.

40.2.4 Indexing the table

All attributes provided with two indices are always indexed with [I] = Row/row and [J] = Column/column, i.e. *.attribut[I, J]*.

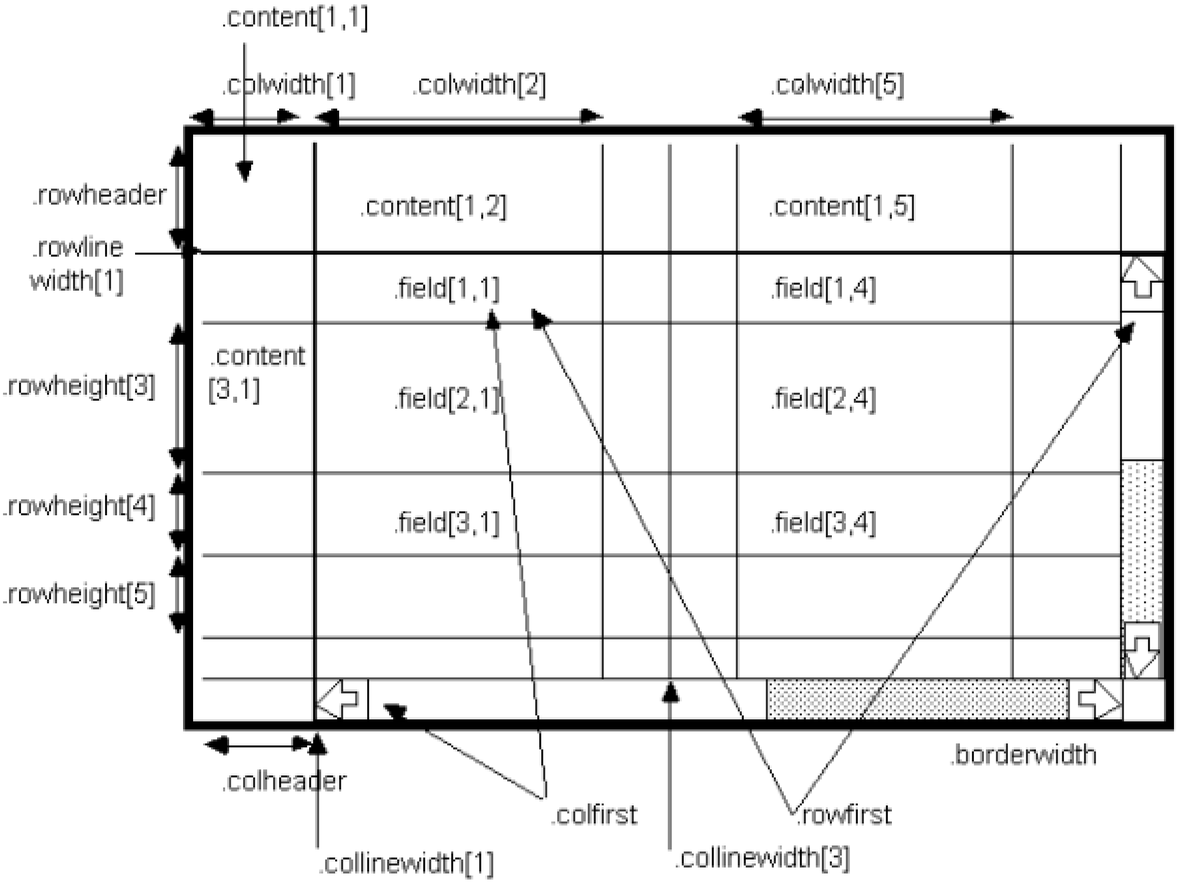


Figure 46: Schematic representation of the tablefield

In the image above, all indices are specified as this must be defined.

40.3 Note on deprecated attributes

The deprecated *.focusable* attribute is no longer supported and generates an ignoring warning when setting and resetting (setinherit) the attribute, and a FAIL when reading the attribute.

40.4 Alignment of the tablefield

The *.direction* attribute defines the orientation of the ***tablefield*** and controls whether rows or columns take precedence. The attribute influences the adopting of row and column default values, the selection and navigation as well as certain methods.

Value range

1 (default)

Vertical orientation with column priority.

2

Horizontal orientation with row priority.

Effects

Default values

With *. direction* = *1*, the column defaults, i.e. the attribute values with the indices *[0,<C>]*, will be used for cell values (*.content[index]*, *.bgc[index]*, *.fgc[index]*…) that are not set. With *.direction* = *2,* the row default values, i.e. the attribute values with the indices *[<R>,0]*, will be used.

Selection

If both *.selection[sel_column]* and *.selection[sel_row]* are set to true and the cell *[<R>,<C>]* is selected, then with *. direction* = *1* the column *<C>* and with *.direction* = *2* the row *<R>* will be selected.

In a ***tablefield*** with multiple selection .nextactive[index] searches for the next selected cell row by row if *.direction* = *1* and column by column if *.direction* = 2. For example, if cells *[4,2]* and *[3,5]* are selected, *.nextactive[1,1]* will return *[3,5]* if *.direction* = *1* and *[4,2]* if *.direction* = 2.

Navigation

If *.direction* = *1*, pressing `Return` will move the input focus to the next selectable cell to the right of the current cell. `Home` and `End` will jump to the beginning or end of the row with the currently focused cell.

If *.direction* = 2, pressing `Return` will move the input focus to the next selectable cell below the current cell. `Home` and `End` will jump to the beginning or end of the column with the currently focused cell.

:find() method

The *.direction* attribute affects the search direction of the **:find()** method. If *.direction* = *1*, the ***tablefield*** is searched row by lrow, if *.direction* = 2, it is searched column by column. For example, if the searched value is located in cells *[4,2]* and *[3,5]*, **:find()** without specifying a search range will return the location *[3,5]* if *.direction* = *1* and the location *[4,2]* if *.direction* = 2.

Methods :clear(), :delete(), :exchange(), :insert() and :move()

The attribute *.direction*, together with the optional *Direction* parameter of the methods, controls whether the methods are applied to rows or columns. If the *Direction* parameter is not specified, the methods are applied to rows if *.direction* = *1* and to columns if *.direction* = 2.

40.5 Mouse interactions

If available, the following interactions are possible with the mouse:

- selection
- scrolling

40.5.1 Selection in the tablefield

The selection of a field is only possible if it is sensitive. This is defined via the attribute *.sensitive* for the entire tablefield and via the field-specific attribute *.sensitive[I,J]*. In addition to these attributes, the *.selstyle* attribute also influences the selectability of a field. Normally, the field on which the user has clicked is selected. The input focus is set to this field and marked as selected and with focus. The content of the field is immediately transferred to the associated editable text and released there for modification.

In order to distinguish the single field selection from a row/column selection, two mouse selection types must be defined. The **single field selection** is done as the user normally selects an object, i.e. by clicking on the object with the left mouse button. The **row/column selection** should be done by pressing the left mouse button and simultaneously pressing the `shift` key. When determining the action to be triggered, the weight of each selection type must be taken into account.

The following order applies (> = "more important than", "takes precedence"):

- For mouse single selection
  single > row/column
- For the row/column selection
  row/column > single

This results in the following procedure for the evaluation:

- Single field selection
  First, it is checked whether the mouse click can be interpreted as a "single" selection or a "header" selection. If this is not possible, it is interpreted as a "row/column" selection. The direction specified in the *.direction* attribute is examined first.
- Row/column selection
  First, it is checked whether the mouse click can be interpreted as a "row/column" selection. The direction specified in the *.direction* attribute is examined first. If this is not possible, an attempt is made to interpret it as a "single" selection or a "header" selection.

Actions on the individual tablefield selection types:

.selection[I] true	Reaction
sel_single	The field that currently has the focus loses the focus and passes it to the element under the mouse if these two elements are different. If the two objects are identical, nothing happens.
sel_header	The field that currently has the focus loses the focus and passes it to the element under the mouse if these two elements are different. If the two objects are identical, nothing happens.
sel_column	The column/row under the mouse is selected. The field that has been clicked on receives the focus. If the column/row is already selected, nothing happens except that the focus is set to the field under the mouse.
sel_row	

40.5.2 Scrolling the tablefield

Scrolling in all directions is always done in sense units. Scrolling is always done in such a way that entire rows or columns are deleted from the display area and replaced by new ones. This applies to both row/column scrolling and page scrolling. It is always possible to scroll so far that the last column or row is just completely visible and there might be an empty area on the right and at the bottom. This is because the Dialog Manager guarantees that an element is visible in the upper left corner.

In detail, this looks as follows:

Scroll action	Reaction
Scroll Right	The left column of the tablefield interior is scrolled out of the visible area. The second column so far is placed next to the row headers. The rest of the display area is filled with the following columns. If only one column can be displayed, the following column is brought into the display area.
Scroll Left	The previously invisible column before the first visible column is brought into the display area and the remaining visible columns are moved accordingly. If only one column can be displayed, the following column is brought into the display area.
Scroll Down	The first row of the tablefield interior is scrolled out of the visible area. The second line so far is set below the headings. The rest of the display area is filled with the following rows. If only one column can be displayed, the following column is brought into the display area.
Scroll Top	The previously invisible row of the tablefield interior is moved in front of the first row in the display area and the remaining visible rows are moved accordingly. If only one column can be displayed, the following column is brought into the display area.
Page Right	The previously last visible column becomes the first visible column and the rest of the display area is filled with the subsequent columns. If only one column can be displayed, the following column is brought into the display area.
Page Left	The previously first visible column becomes the last visible column and the rest of the display area is filled with the preceding columns. If only one column can be displayed, the previous column is brought into the display area.
Page Down	The last visible line so far becomes the first visible line and the rest of the display area is filled with the following lines. If only one line can be displayed, the following line is brought into the display area.
Page Top	The previously first visible line becomes the last visible line and the rest of the display area is filled with the preceding lines. If only one line can be displayed, the previous line is brought into the display area.

Scroll action	Reaction
Slider	The user's scroll action is always corrected so that the first row/column in the tablefield interior is correctly visible. Of course, it can happen that the bottom row and the right column are only partially visible.

Through this scrolling, Dialog Manager guarantees that the first row and the first column in the tablefield interior are always displayed correctly and completely, if the tablefield is only large enough. The last visible row and the last visible column can only be partially displayed if the tablefield has unfavorable dimensions. If the scrolling action is not feasible in the described form, the scrolling is performed as far as possible.

Example

The user wants to scroll page by page, but only one column is not displayed. It is then scrolled so that this column just comes completely into the display area. It can happen that an empty space remains on the right and below, because the upper left corner is always displayed correctly.

The calculation of the slider size and the maximum value of the slider is based on pixels, not on sense units of the tablefield. When calculating the slider size, the available space in the tablefield interior is set in relation to the actually required area.

This is done according to the following formula:

- for the horizontal scrollbar
  Slidersize = Scrollbarsize * (usable width /sum of the width of all columns (without row headers))
- for the vertical scrollbar
  Slidersize = Scrollbarsize * (usable height /sum of heights of all lines (without headers))

When scrolling, the focus in the tablefield is not changed. This makes it possible for the focus to sit on an object in the tablefield that is not currently visible. To visualize this, the focus frame is then painted around the entire object. If the object with the focus is scrolled back into the visible area of the tablefield, the focus is only painted around the field and no longer around the entire object.

40.6 Interactions via the keyboard

The tablefield can be operated completely without a mouse. With tablefield cursor control, the tablefield must automatically perform all necessary scroll operations so that the field reached by cursor control is in the visible area of the tablefield. The cursor control is not cyclic for any direction; i.e. if there is no field allowed for cursor control in the actual search direction, the cursor stops. A field that is allowed for cursor control is either *.sensitive* = *true* or the attribute *.fieldfocusable* was set to true for the tablefield (i.e. non-sensitive objects can also be reached with the cursor control).

40.6.1 Keyboard mapping for navigation

The keyboard layout for navigation looks like this:

Taste	Aktion
Cursor `Down`	With the help of this key, the input focus is to be set to the next field lying in the rising y-direction that is permitted for cursor navigation. This causes the input focus to move optically downwards.
Cursor `Up`	With the help of this key, the input focus is to be set to the next field lying in the falling y-direction that is permitted for cursornavigation. This causes the input focus to move optically upwards.
Cursor	With the help of this key, the input focus is to be set to the next field lying in the falling x-direction that is permitted for cursor navigation. This causes the input focus to move optically to the left.
Cursor `Right`	With the help of this key, the input focus is to be set to the next field lying in the rising x-direction that is permitted for cursor navigation. This causes the input focus to move optically to the right.
	With the help of this key the input focus is to be set to the next field lying in the direction defined by .direction. If the end of a column/row is reached, the focus remains on this element.

`Tab`
`Ctrl` + `Tab`
	These two buttons are used to exit the tablefield. This is to go to the next/previous object, according to the usual behavior of the window system.

`Pos 1` (Home)	This button sets the focus to the first element of a row (.direction=2) or column (.direction =1) in the tablefield interior (rowindex > rowheader or colindex>-colheader). It is searched starting with the element [rowheader + 1] [?] in the row to the right (.direction = 2) or starting with the element [?] [colheader + 1] in the column to the bottom (.direction = 1) until an element is found that can get the focus (.sensitive and .fieldfocusable). This element is then scrolled into the visible area of the tablefield.
`Ctrl` + `Pos 1`	This button sets the focus on the first element (top left) in the tablefield interior (rowindex > rowheader or colindex > colheader). Starting with the element [rowheader + 1] [colheader + 1] in the row to the right (.direction = 2) or in the column downwards (.direction = 1) is searched until an element is found that can receive the focus (.sensitive and .fieldfocusable). If no element is found in the row or column, the search continues in the row below or column to the right. The found element is then scrolled into the visible area of the tablefield.

Taste	Aktion
`End`	This button sets the focus to the last element of a row (.direction =2) or column (.direction = 1) in the tablefield interior (rowindex > rowheader or colindex > colheader). It is searched starting with the element [rowcount] [?] in the row to the left (.direction = 2) or starting with the element [?] [colcount] in the column to the top (.direction = 1) until an element is found that can get the focus (.sensitive and .fieldfocusable). This element is then scrolled into the visible area of the tablefield.
`Ctrl` + `End`	With the help of this button the focus is set to the last element (bottom right) in the tablefield interior (rowindex > rowheader or colindex > colheader). Starting with the element [rowcount] [colcount] in the row to the left (.direction = 2) or in the column to the top (.direction = 1) is searched until an element is found that can receive the focus (.sensitive and .fieldfocusable). If no element is found in the row or column, the search continues in the row above or column to the left. The found element is then scrolled into the visible area of the tablefield.
logical: Show	This button can be used to bring into the display area the element in the tablefield that currently has the focus. If the object is already in the visible area, nothing happens.

Remark for Cursor Up, Down, Right, Left

If the cursor control is used to move from the header area to the tablefield area inside, the cursor is placed on the first element visible below/next to the headers without triggering scrolling.

If the cursor control is triggered in the direction of the header (cursor `Up`, cursor `Left`) in the first element below/next to the headers, the cursor is not positioned on the header until the corresponding scrollbar is at its initial value.

(this->tablefield.?->first = this->tablefield.?->header + 1)

As long as this is not the case, scrolling is performed accordingly.

This means that the actions cursor `Left` followed by cursor `Right` and cursor `Down` followed by cursor `Up` do **not** have to be inverse operations.

40.6.2 Keyboard mapping for scrolling

The keyboard layout for scrolling looks like this:

Key	Action
Page Up	This button should be used to move the contents of the tablefield up by a maximum of one page.

Key	Action
Page Down	This button should be used to move the contents of the tablefield down by a maximum of one page.
Page Left	This button is used to move the contents of the tablefield to the left by a maximum of one page.
Page Right	This button is used to move the contents of the tablefield to the right by a maximum of one page.
Line Up	This button is used to move the contents of the tablefield up one line.
Line Down	This button is used to move the contents of the tablefield down one line.
Row Left	This button is used to move the contents of the tablefield one column to the left.
Row Right	This button is used to move the contents of the tablefield one column to the right.

For all these scrolling actions, the scrolling behavior described in the chapter “Mouse interactions” applies.

Remark Remark on scrolling via keyboard

If scrolling is triggered via the keyboard, the focus does not change its position on the screen. However, this can change its internal position or the element that currently has the focus. For example, if the cursor is in a heading and scrolling vertically, the cursor will remain in the same position both on the screen and internally. However, if the cursor is inside the tablefield, it will change its internal position in any case; the position visible on the screen, however, will remain largely unchanged.

40.6.3 Keyboard mapping for the selection

Key	Action
Space, log. selection	This button selects the field with the focus, if it is not already selected.
log.row/column selection	This key triggers the row/column selection.

The two keys "logical selection" and "logical row/column selection" depend on the window system.

40.7 Interactions with the editable text

The editable text belonging to the tablefield is automatically provided with the content of the field that currently has the focus. Additionally, the attributes *.maxchars[I,J]* and *.format[I,J]* corresponding to the current focus object are taken over.

Depending on the *.edittype* attribute, the actions in the editable text are transferred to the tablefield.

- With *.edittype online*, all changes in the editable text are immediately applied to the tablefield.
- With *.edittype offline*, the changes are not applied to the tablefield until the editable text is deselected.
- With *.edittype locking* the tablefield is locked after the first input into the editable text and only after input of Return released again and the content is taken over.
  If the editable text is left without a return and only by a normal deselection or the input of `ESC`, the changes are not accepted, but the tablefield is released again. The editable text itself does not send any events.

Remarks

- An editable text is not activated by the cursor navigation, but only by the first character designated for the editable text.
- When an editable text is enabled, the following navigation applies:

Key	single-line ET	multiline ET
Cursor `Left`, `Right`	ET receives key	ET receives key
Cursor `Up`, `Down`	ET receives key (can also be ignored)	ET receives key
`Return`	Next (previous) entry	ET receives key
`Tab` (`Shift` + `Tab`)	Next (previous) entry	Next (previous) entry

- A multiline editable text always has scrollbars, regardless of *.editpos*.
- The height of an entry does not affect the number of text lines that can be entered by the user.
- Since the *.format* attribute is controlled by the tablefield, the attribute of the bound edit text should **not** be used.

40.8 Interactive column and row resizing

When the mouse is moved over the edge of a manipulable field, the mouse cursor changes and shows a resize icon. The appearance of the cursor cannot be specified by the application programmer. For columns, resizing can be done at the right border of each column. For the rows, the lower boundary is relevant.

The user can initiate the resizing with the left mouse button. A gray hatched line appears, indicating the position of the mouse (the width of the hatched line depends on the attributes *.collinewidth[I]* and *.rowlinewidth[I]*). If the mouse is moved (with the left mouse button pressed), the hatched line moves with it. The resizing is completed by releasing the mouse button.

The new attribute values of the enlarged or reduced rows/columns are automatically applied and the tablefield is redrawn accordingly (The attribute values of *.rowheight[I]* and *.colwidth[I]* are changed). This is consistent with resizing the window.

The indexed event *resize[I,J]* is sent to the tablefield and can be reacted to. One component of thisevent.index is zero at a time. If a row was changed, second(thisevent.index) is zero and with first(thisevent.index) the changed row can be queried. If a column was changed, then the changed column can be queried with second(thisevent.index).

The assignment at a glance:

Changes on	first(thisevent.index)	second(thisevent.index)
Row	Index of the changed row	0
Column	0	Index of the changed column

40.9 Note for the tablefield with Microsoft Windows

40.9.1 Application code page selection

The “utfwin” code page is not suitable as an application code page when strings containing <Carriage-Return> characters must be processed. The reason is that <Linefeed> characters (line breaks) are converted to the <Carriage-Return><Linefeed> string. To avoid duplicate <Carriage-Return> characters, they are skipped. Instead of “utfwin”, the code pages “utf16” or “utf16l” should be used, which do not perform this special handling.

40.9.2 Use of colors and “Visual Styles”

If no colors are set for the ***tablefield***, the IDM uses the system colors of the respective state. If *.fieldshadow* = *true*, those “Visual Styles” of the Windows object “pushbutton” are used. It defines the assigned colors. If *.rowheadshadow* or *.colheadshadow* are switched on, the colors are taken from those “Visual Styles” of the Windows object “headeritem”.

If colors are set, they are used. For an active field, *.fgc* determines the background color and *.bgc* the foreground color. If **only one** of the two colors is set, the corresponding system color is used for the **other** color. Thus, if only one color is set, the foreground and background colors are not swapped, but the set color is assigned differently. There is no separate color assignment for an active **and** insensitive field. It is displayed in the same way as an active field.

Recommendation

Set either **both** colors – *.fgc* and *.bgc* – or **neither** of the two colors.

Remark

The colors were also already assigned as described in the IDM for Windows 2000. However, for active objects practically only foreground and background colors were swapped, which brought mostly a good result even with only one color set. Only the “Visual Styles” (especially that of the Windows push-button) partly do not swap the colors anymore, but only use a different background.

40.10 Note for the tablefield with Motif

The Motif tablefield ignores leading line breaks in the table cells (one or more \n at the beginning of the texts). To nevertheless display blank lines before a text, a space can be inserted in the text before the line breaks.

Example

`"  \nXYZ"` instead of `"\nXYZ"`

40.11 Note for the tablefield with Qt

Scrollbars are present in an extremely small tablefield where the header rows and columns cannot be displayed completely. However, these have no effect.

The tablefield has an empty row and column at the bottom and right, which fill the free area when scrolling all the way down or to the right. If the tablefield is dimensioned in such a way that the last content line or content column cannot be displayed completely, then the user can scroll to this empty line or empty column. This position cannot be queried or set programmatically.

The selection model of Qt is used, so there may be slight differences to the other window systems. However, the behavior is typical for Qt. If several of the elements *sel_column*, *sel_row* and *sel_single* have the value true for the attribute .selection[enum], then *sel_single* is used. If none of the elements *sel_column*, *sel_row* and *sel_single* has the value true for the .selection[enum] attribute, *sel_single* is used.

Activated cells are deactivated when a header cell is clicked at *.selstyle[sel_header] = false*.

With *.fieldfocusable = true* a click on an insensitive cell also causes a reactivation. The corresponding *activate* and *deactivate* events are generated, but there is no *select* event with index.

The following attributes are not implemented:

- .colheadshadow
- .fieldshadow
- .rowheadshadow
- .xmargin
- .ymargin

40.12 Example

The following example shows how to use the extensions:

The tablefield headings can be manipulated interactively.

```
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";
}
```

41 timer

This object executes actions at fixed times in intervals specified before. It is possible, for example, to start a process on a fixed day in the week, or to be reminded of certain dates (days, hours,…).

With the ***timer*** times can be specified in months, weeks, days, hours, minutes, and seconds. Intervals can also be defined in these units.

Definition

```
{ export | reexport } { model } timer { <Identifier> }
{
  <object-specific attributes>
}
```

Events

select

Children

document

record

transformer

Parent

dialog

module

Menu

none

41.1 Attributes

Attributes	RLD	PID	Properties	Short Description
active	boolean	boolean	S,G/D/C	active state of object
class	class	class	-,G/-/-	class/id of object
count	integer	integer	S,G/D/-	repetition of timer
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs

Attributes	RLD	PID	Properties	Short Description
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
firstrecord	object	record	S,G/-/C	accesses the first record of an object
incrtime	string	string	S,G/D/C	intervals of timer
label	string	string	S,G/D/C	name/identifier of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
model	identifier	instance	S,G/D/C	model belonging to object
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
starttime	string	string	S,G/D/C	specified time for timer

41.1.1 Definition of Time Spans

There are two possibilities for time definition:

- absolute time definition
- relative time definition

Absolute Time Definition

```
{<hour>}:{<minute>}{:{<second>}}
{<day>}.{<month>}.{<year>} {<hour>}:{<minute>}{:{<second>}}
{<month>}/{<day>}/{<year>} {<hour>}:{<minute>}{:{<second>}}
{<DoW>} {<hour>}:{<minute>}{:{<second>}}

DoW: day of the week (1 = Monday, 2 = Tuesday, .... , 7 = Sunday)
```

Time Interval

```
+ {<hour>}:{<minute>}{:{<second>}}
+ n d
+ n D
+ n m
+ n M
+ n y
+ n Y
```

d, D:	days
m, M:	months
y, Y:	years

If no value is specified in the **absolute time definition**, the corresponding value will be taken from real time, i.e. "now".

If no **seconds** are specified, they will be set to 0 if : is missing, or to the seconds in real time if : is given.

Day numbers up to 31 are allowed for each **month**. If a month has less days, the last day specified will be considered the last day of that month.

Years can be specified in the range of 1901 to 2099. Alternatively, values from 0 to 99 are allowed, which will be mapped to the period 1970 to 2069.

Examples

» Every Friday at noon

```
.starttime  "5 12:00";
.incrtime   "+7D";
```

» Every day at 16:00

```
.starttime  "16:00";
.incrtime   "+1D";
```

» Christmas Eve 19:00

```
.starttime  "24.12 19:00";
.incrtime   "+1Y";
```

» Four hours after program start

```
.starttime  "+04:00";
```

Note

The attribute *.starttime* can be used for the absolute time definition or for the time interval.

The attribute *.incrtime* can be used for the time interval only.

41.2 Example

Use of a ***timer*** for an autosave function:

```
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.
}
```

42 toolbar

As implied by its name, the ***toolbar*** object is a grouping object with the main purpose to define toolbars. Only windows can have one or more toolbars as children.

A toolbar can either be docked to its parent window or appear as a separate tool window.

The background color of toolbars is that of the menus; the child objects do not have focus frames and can only be operated by mouse or accelerators. The frame around a docked toolbar shows its scope.

Definition

```
{ export | reexport } { model } toolbar { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <object-specific attributes>
}
```

Events

activate (only for undocked toolbars)

close

cut

deactivate (only for undocked toolbars)

extevent

move

paste

resize

Children

canvas

checkbox

document

edittext

groupbox

image

layoutbox

listbox

menubox

notebook

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

tablefield

transformer

treeview

Parents

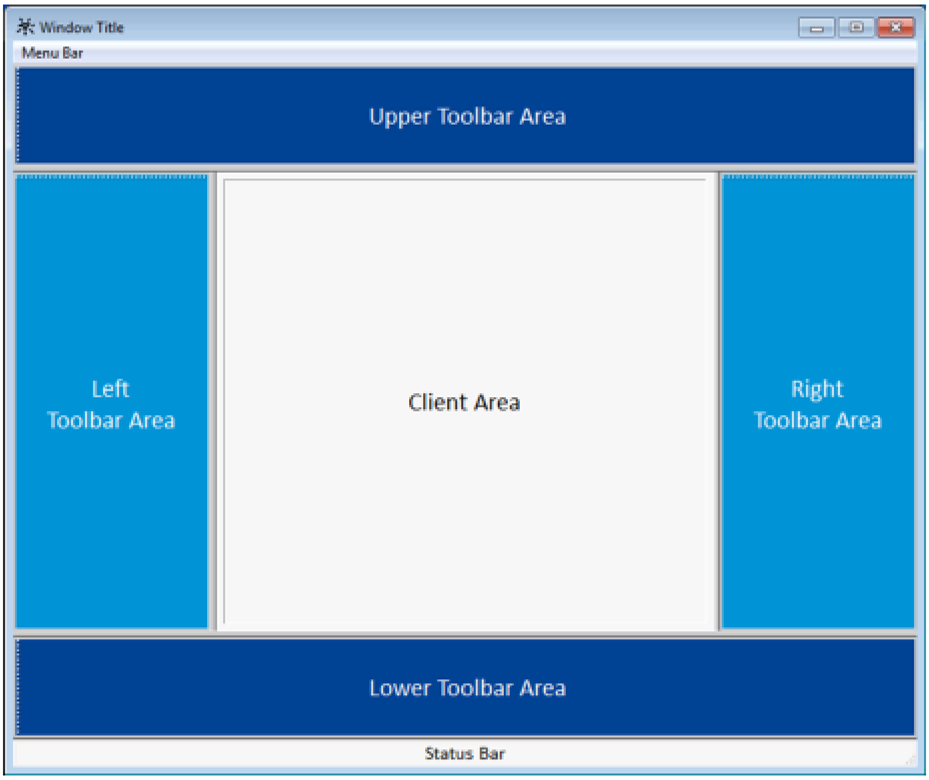
module

window

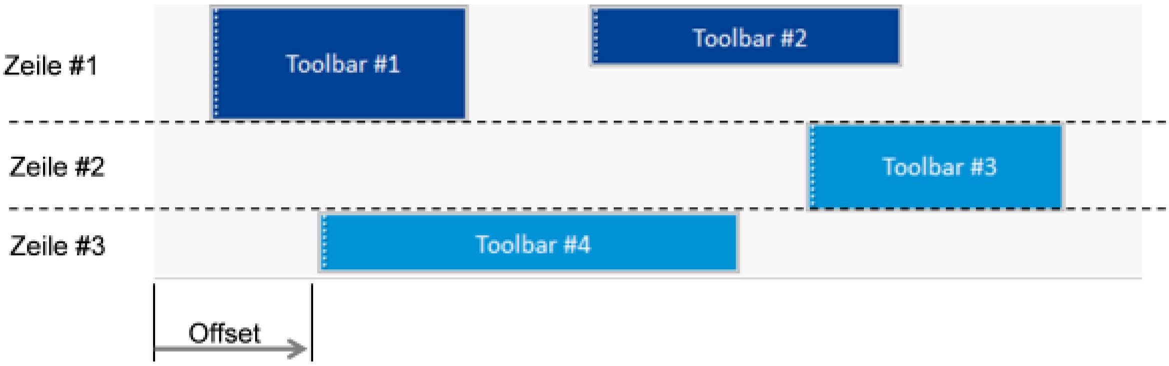
Menu

Pop-up menu

The following schematic picture of a window illustrates the arrangement of the several areas. Adding and removing toolbars does not change the size of the window but affects the size of the client area where the actual contents of the window are displayed.



One toolbar area can contain one or more toolbars, which can be arranged on several rows or columns (exclusively). The size of a toolbar area depends on the number, sizes and distribution of the toolbars within; there cannot be rows or columns without at least one toolbar in it.



The offset is measured from the left or top edge of the toolbar area, depending on whether it is a horizontal (*dock_up*, *dock_down*) or a vertical (*dock_left*, *dock_right*) toolbar.

With docked toolbars *move* events only occur when the attributes *.dock_offset* or *.dock_line* change. As an example there is no *move* event when a docked toolbar is moved down but remains in the same toolbar row. Indirect changes of *.dock_offset* or *.dock_line*, caused for instance through resizing another toolbar, are only performed transiently without altering the defined positions persistently. Therefore toolbars take back their defined positions once the required space becomes available.

42.1 Attributes

accelerator

active

autoalign

autosize

bgc

bordercolor

borderraster

borderstyle

borderwidth

child[**I**]

childcount

class

cursor

cut_pending

cut_pending_changed

dialog

dockable[enum]

docking

dock_line

dock_offset

document[**I**]

external

external[**I**]

fgc

firstchild

firstrecord

focus

focus_on_click

font

function

height

height[class]

help

label

lastchild

lastrecord

mapped

maxheight

maxwidth

minheight

minwidth

model

module

options[enum]

parent

posraster

real_height

real_sensitive

real_visible

real_width

record[**I**]

recordcount

scope

sensitive

sizeable

sizeable[class]

sizeraster

source

target

tile

tilestyle

title

titlebar

titlebgc

titlefgc

toolhelp

transformer[I]

userdata

visible

width

width[class]

window

xauto

xleft

xraster

xright

yauto

ybottom

yraster

ytop

42.2 Specific Attributes

A particularity of the toolbar-specific attributes is that they heavily influence each other. Most inherited attributes behave like on the other objects, e.g. the window.

Toolbar-specific features of the respective attributes are described in the table below and the next sections.

Attribute	Description
autoalign	Toggles the automatic positioning of the object after the last toolbar or at the beginning of the row or column.
autosize	Defines whether the toolbar shall be enlarged in the respective dimension to cover the entire toolbar row or column.
borderraster	Determines how geometry is computed when a grid is used. Only effective for *.borderwidth* > *0* or *.docking* = *dock_window*.

Attribute	Description
borderstyle	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a) Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
dockable[enum]	Controls in which toolbar areas a toolbar can be docked. Possible indexes: » *dock_window* » *dock_up* » *dock_down* » *dock_left* » *dock_right*
docking	Sets or returns the current docking position of the toolbar. Possible values: » *dock_window* (default) » *dock_up* » *dock_down* » *dock_left* » *dock_right* With instances it is only allowed to set values for which the attribute dockable [enum] is *true*.
dock_line	Defines the order of toolbars within one toolbar area.
dock_offset	Sets the distance of the toolbar from the left or top edge of the toolbar area.
focus_on_click	Defines if a mouse click into the client area of the object activates the object, i.e. sets the focus on it (Microsoft Windows only). See also chapter ".focus_on_click".
height	Height of the object. See chapter ".height, .width" for specifics concerning inheritance.
height[class]	Height of the object in a specific state. Indexes: *toolbar*, *window*. See also chapter ".height[class], .width[class]".

Attribute	Description
maxheight	Maximum height of the object. See also chapter ".minwidth, .maxwidth, .minheight, .maxheight".
maxwidth	Maximum width of the object. See also chapter ".minwidth, .maxwidth, .minheight, .maxheight".
minheight	Minimum height of the object. See also chapter ".minwidth, .maxwidth, .minheight, .maxheight".
minwidth	Minimum width of the object. See also chapter ".minwidth, .maxwidth, .minheight, .maxheight".
options[enum]	Options of the toolbar; please refer to the "Attribute Reference" for details.
sizeable	Defines if the toolbar can be resized interactively; the default value is *true*. See also chapter ".sizeable and .sizeable[class]".
sizeable[class]	Defines if the toolbar can be resized interactively when it is in a specific state. Indexes: *toolbar*, *window*. See also chapter ".sizeable and .sizeable[class]".
tile	Sets a background image for the object.
tilestyle	Controls how the background image set in *.tile* is arranged.
title	Title of the tool window when the toolbar is undocked.
titlebar	Toggles whether the tool window of an undocked toolbar has a title bar.
width	Width of the object. See chapter ".height, .width" for specifics concerning inheritance.
width[class]	Width of the object in a specific state. Indexes: *toolbar*, *window*. See also chapter ".height[class], .width[class]".
xauto	Defines the horizontal dimension that will be calculated by IDM. Only effective for tool windows, the attribute is ignored for docked toolbars.
xleft	Distance from the left edge. Only effective for tool windows, the attribute is ignored for docked toolbars.
xright	Distance from the right edge. Only effective for tool windows, the attribute is ignored for docked toolbars.

Attribute	Description
yauto	Defines the vertical dimension that will be calculated by IDM. Only effective for tool windows, the attribute is ignored for docked toolbars.
ybottom	Distance from the lower edge. Only effective for tool windows, the attribute is ignored for docked toolbars.
ytop	Distance from the upper edge. Only effective for tool windows, the attribute is ignored for docked toolbars.

42.2.1 .dockable, .docking[enum]

For the attributes *.dockable* and *.docking[enum]* inheritance to instances is ignored in cases it would lead to inconsistencies.

This also applies if inheritance of the *.docking* attribute would change the size attributes of the parent window and the parent window of the toolbar is visible but not sizeable.

Nevertheless the inheritance relations persist, meaning that no call of **setinherit()** is necessary to pass down future value changes from the model.

42.2.2 .focus_on_click

With the attribute *.focus_on_click* a toolbar can be defined that does not pull the focus out of the related window when the toolbar is clicked with the mouse. As in the ISA Dialog Manager all sensitive child objects of the toolbar may also obtain the focus, the attribute is available for the objects ***groupbox***, ***image***, ***rectangle*** and ***statictext*** too. Take into consideration that a grouping object with *.focus_on_click = false* should not contain objects, which may gain the focus or even require it for input (e.g. ***edittext***).

The attribute can only be used on Microsoft Windows.

42.2.3 .height, .width

These attributes are the defaults for height[class] and width[class]. The attributes *.height* and *.width* therefore act like other predefined or user-defined attributes that have default values. The respective sizes can be set simultaneously for the docked and undocked states. Also *changed* events are triggered for these attributes.

For docked toolbars in visible but not sizeable parent windows, the heights of horizontally docked toolbars and the widths of vertically docked toolbars cannot be changed as this would change the *.width* or *.height* attributes of the parent windows.

42.2.4 .height[class], .width[class]

The attributes height[class] and width[class] specify the heights and widths of a toolbar in its docked (class = *toolbar*) and undocked (class = *window*) states. When theses sizes are changed while the toolbar has the corresponding state, *changed* events are triggered for the affected attributes, with the respective index though.

42.2.5 .minwidth, .maxwidth, .minheight, .maxheight

These attributes have the same meanings as those of the window object and are effective in the docked as well as in the undocked states. If applicable they have to be adapted when a toolbar is moved to another docking area (e.g. from the top to the right).

42.2.6 .sizeable and .sizeable[class]

The sizeable[class] attribute prevents a toolbar from being interactively resized by the user. It is a vector attribute like *.height[class]* with a default value in sizeable. The possibility of interactive resizing can be set individually for docked and undocked state whereas *.sizeable* affects both states.

42.2.7 .xleft, .xright, .xauto, .ytop, .ybottom, .yauto

These attributes are only effective for tool windows and are ignored with docked toolbars.

42.3 Interactive Handling

The child objects of a toolbar behave like in a window.

Handling of toolbars allows interactive closing, moving and resizing of tool windows with the mouse or by means of the system menu; regardless of the docking state, the docking state of a toolbar can be toggled by double-clicking the left mouse button in a free area or the non-client area of the toolbar.

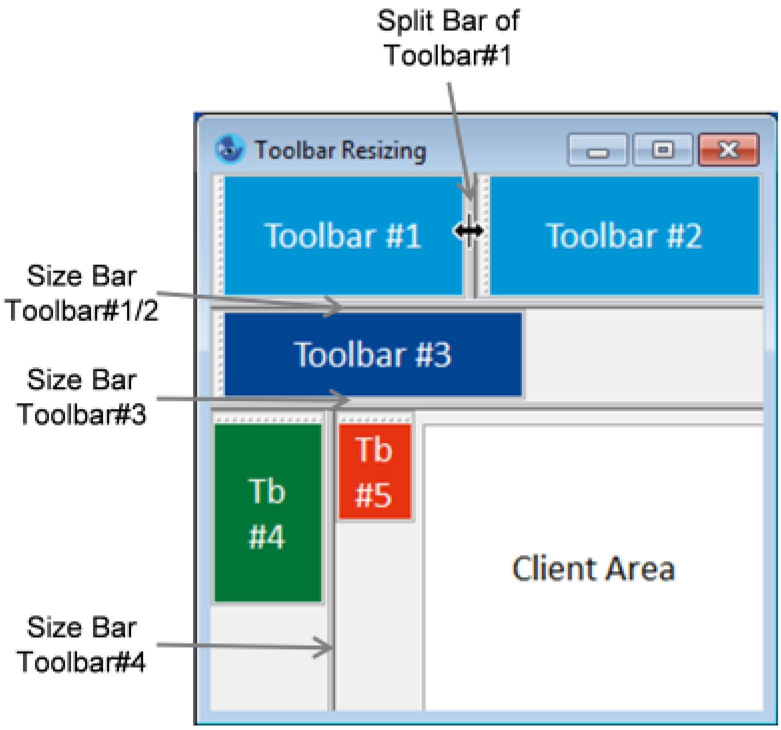
If the docking state is toggled by double-click, the toolbar jumps to the position it had the last time in that docking state. Toolbars that are docked for the first time get docked at the top by default.

Interactive docking and undocking can be locked through the *.dockable* attribute. If the parent window is visible and not sizeable it is also impossible to dock or undock toolbars.

The handle of a docked toolbar has no function yet.

42.4 Interactive Toolbar Resizing

For the resizing of objects there is the *.sizeable* attribute. This also exists for the toolbar but is only effective in the undocked state (similar to the window). The following deals with the resizing of docked toolbars.



Every docked toolbar whose size can be changed provides bars for resizing. Bars that separate consecutive toolbars within one row or column will be called split bars in the following. In contrast, bars that influence the height of a toolbar row or the width of a toolbar column will be called size bars. Moving size bars typically affects several other toolbars and the size of the client area. Moving a split bar at first only affects the size of the respective toolbar but may have an impact on other toolbars in the same row or column as well as on the client area (cue: wrap).

In the picture above all toolbars but the smallest toolbar #5 can be resized. Thus the following split bars and size bars exist:

- Size bars at the right or the bottom of a toolbar (depending on its direction), except for last toolbar in a row or column.
- Split bars if a toolbar can be resized within a row or column. The split bars are located at the edge of the toolbar that is closest to the client area.

The bars are handled like the splt bars of the ***splitbox***, which means:

- As visual feedback the cursor shape changes when the mouse cursor is over a bar (bidirectional horizontal or vertical arrows).
- When the left mouse button is pressed and held down over the bar, a dimmed bar appears which follows the moves of the mouse within a toolbar row or column.
- The toolbar is finally resized when the mouse button is released.

A *resize* event is triggered when the size of a toolbar has changed (that is the *width* or *height* attribute has changed).

Minimum and maximum values are only taken into account with sizeable toolbars, leading to this behavior:

- Interactive resizing: A toolbar row or column can never be smaller than the largest minimum size and never be larger than the smallest maximum size. Exceptional cases are conflicting minimum and maximum sizes or window sizes which prohibit the defined behavior.
- Of course the values influence the arrangement and layout of toolbars.

42.4.1 Automatic Sizing and Positioning of Toolbars

Toolbars provide two additional features:

- *.autoalign = true*
  Automatically places a docked toolbar after the last toolbar in a row or column or at the beginning of empty rows and columns.
- *.autosize = true*
  Automatically enlarges toolbars to cover the entire row or column. Beginning with the last toolbar in a row or column, toolbars are expanded if empty space is available, regardless of *.sizeable*. Expansion is limited to the docking direction: width is increased for horizontal direction and height for vertical direction.

42.5 Coordinates and Sizes

The toolbars and the free space between them constitute the toolbar area of the parent window. This area is located outside the client area of the window and is visually separated from it by a border. Menu and statusbar, if present, are placed further outside than the toolbar area, scrollbars further inside. The priority of overlapping toolbars is top to bottom first and then left to right.

Toolbars support size and position grids. If size raster is turned on for the parent window, the toolbar areas are expanded to fit grid units. This may cause gaps between the toolbars and the borders of the toolbar area.

In visible windows docking, undocking, and changing the docking position of toolbars, creating or destroying, showing or hiding, and resizing of docked toolbars lead to an adjustment of the parent window’s client area. That is, for the parent window the values of the attributes *.width* and *.height* change. This prevents the window from “jumping” when toolbars are docked or undocked interactively. Hence a *resize* event is triggered for the window (of course a *move* event for the toolbar is triggered too).

If the window is invisible, the size of its client area remains the same, rather the outer dimensions of the window are changed.

There are exceptions to this general behavior:

- If the parent window is visible but not sizeable, no toolbars can be docked or undocked. But attention, as in general the following is valid: when toolbars are manipulated from the Rule Language, the client area of the parent window may change even if the window is not sizeable.

- When minimum or maximum sizes are set for the parent window and these values would be violated by undocking or destroying, docking or creating a toolbar, contrary to the standard behavior the outer dimensions of the window are adjusted as far as necessary.
- If no maximum attributes but a virtual size is set for the parent window, undocking or destroying a toolbar may cause the outer dimensions of the window to be reduced in order to prevent the client area from exceeding the virtual size.

42.6 Particularities with Focusing

Important: In contrast to other toolbars, objects in a IDM toolbar can obtain the focus. On Microsoft Windows this feature can be controlled through the focus_on_click attribute for the toolbar as well as for the objects groupbox, image, rectangle and statictest. The attribute does not exist on Motif.

In general the focus behavior of the toolbar is quite similar to that of the window object.

A undocked toolbar (tool window) can be activated with a mouse click. This moves the focus to the toolbar child that had it the last time the toolbar was active. If no child ever had the focus, the first child obtains it. In a toolbar without children, no object will be focused, which shows through no object on the screen possessing a focus frame.

A docked toolbar cannot be activated as in this case the parent window is active. However it is possible to set the focus on the toolbar with a mouse click, or to be more accurate, to set the focus on one of the toolbar children. With regard to which child will receive the focus, the same rules as for undocked toolbars apply. If the toolbar has no children, the focus moves to or remains at a child of the parent window. In this situation the toolbar cannot obtain the focus.

To preserve the focused object in a docked toolbar when the window is activated, the attribute *.focus_on_click* of the toolbar must be set to *true*. Anyhow a mouse click into the client area of the parent window shifts the focus to a child of the parent window.

As customary, within a toolbar the focus can be moved from one child to the next with the `Tab` key. The sequence is determined through the order in which the children were generated in the dialog. When the last child is reached, the focus jumps back to the first child.

When the focus is on a child object of a toolbar and the toolbar becomes insensitive, the previously active window is activated again and the focus shifts to a child of this window or the window itself if it has no children.

42.7 Remark Windows 10 and 11 with multiple monitors

In WINDOWS, after adding/removing a monitor or changing the display settings, you may encounter the following problems:

- No moving frame appears when moving a toolbar object. It can also happen that a rectangle the size of the **toolbar** frame is displayed with an incorrect background on another monitor.
- Undocked **toolbar** objects do not adjust to changes in DPI value. Both when moving to another monitor or changing the magnification.

The problem is due to Windows, which on the one hand incorrectly converts the coordinates when drawing to the desktop and on the other hand does not send important messages (*WM_DPICHANGED*) to so-called tool windows and does not change the window DPI value.

After the screen saver was active or you logged in again (and after a reboot), the problem no longer occurs.

42.8 Particularities of the Toolbar on Motif

Undocked toolbars cannot be docked by dragging their titlebar with the mouse.

42.9 Particularities of the Toolbar on Qt

The ***toolbar*** can have two different forms on Qt, which are set via the *.style* attribute. By default, the style *toolbar* is active, which visually matches the familiar toolbars.

Table 1: *Attribute .style of the toolbar*

Value	Description
.style = toolbar (default value)	Conventional toolbar, corresponding to the well-known IDM toolbar.
.style = notepage	Toolbars use a dock area that lies between toolbars and inner area. “Tabbed toolbars” and “nested toolbars” are possible (see chapter “Particularities of the Window on Qt” at the ***window*** object).

“Figure 47” shows the division of a window with different toolbars in the docked and undocked state. “Tb16” and “Tb18” as well as “Tb19” and “Tb20” are “tabbed toolbars”that share their space. “Tb17” is a “nested toolbar” embedded in the second row of the right pane.

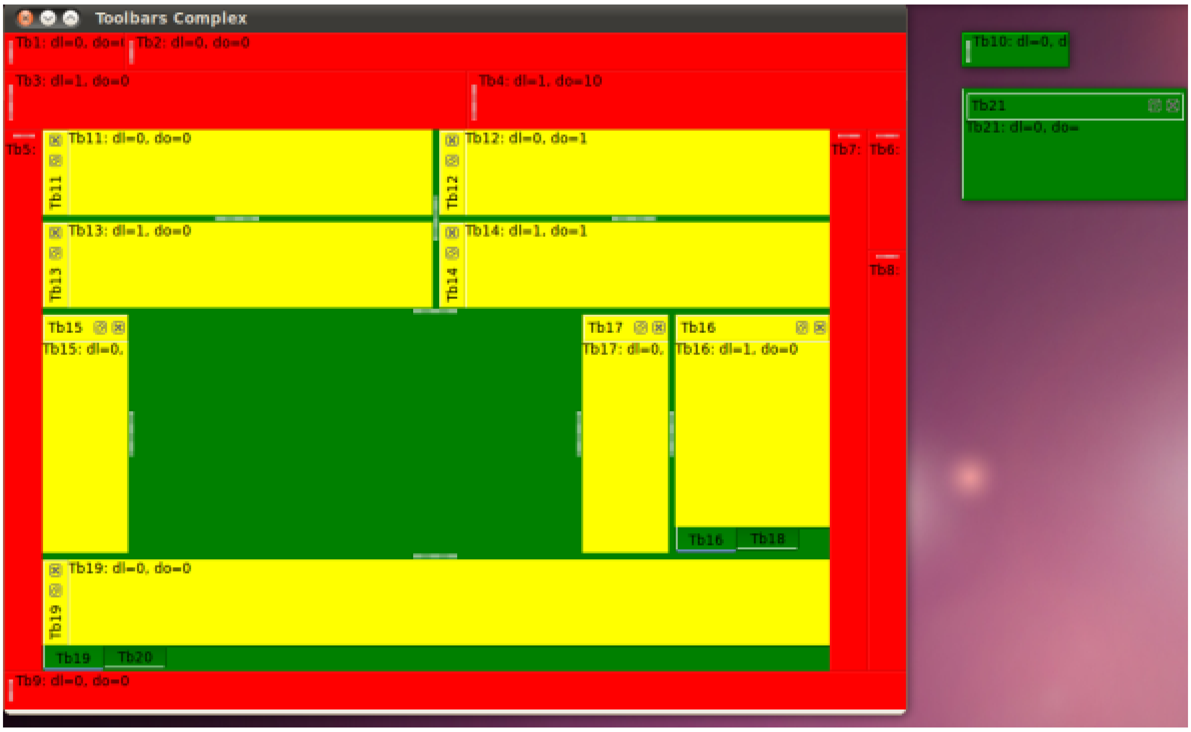


Figure 47: *Toolbar types in the IDM for Qt*

The style *notepage* allows to nest multiple ***toolbars*** in in one docking area and arrange them as tabs (see chapter “Particularities of the Window on Qt” for the corresponding control options). The toolbars then have a title bar and can be undocked and closed using their title buttons.

Toolbars of different styles defined in the same docking area cannot be mixed and are grouped according to their style. Toolbars with the style *toolbar* are always positioned at the outer edge of the window and toolbars with the style *notepage* are always positioned between the client area of the window and the toolbars with the style *toolbar* (see “Figure 48”).

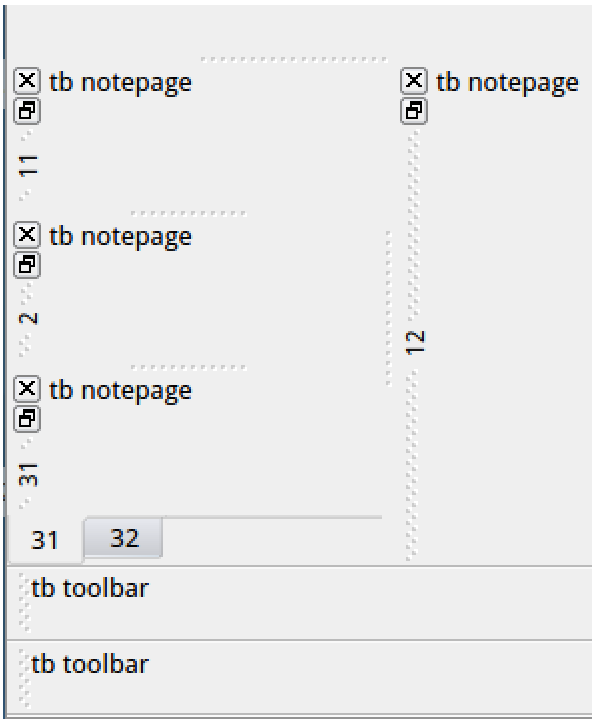


Figure 48: Toolbars of different styles in the lower docking area (dock_down)

Toolbars cannot be positioned with spaces between them. The attribute *.autoalign* has no function.

The attribute *.sizeable* behaves when *true* like the attribute *.autosize*, that is, the entire available width or height is filled. If *false*, the set size is retained under all circumstances.

Undocked toolbars with the *toolbar* style cannot be resized with the mouse and do not create *move* events when moved.

Docked toolbars with the style *toolbar* and *.sizeable* = *true* can be manually enlarged in one direction only, depending on their orientation: Horizontal toolbars only in width, vertical toolbars only in height. The size in the other direction is determined by the largest toolbar in the same row or column. A smaller toolbar is automatically adjusted by Qt to the height or width of the highest or widest toolbar. With *.sizeable* = *true* it is therefore recommended not to define different heights or widths within a toolbar row or column.

Furthermore, manual enlargement in one direction is only possible at the toolbar junctions.

Toolbars with style *notepage* and *.sizeable* = *true*ignore depending on their orientation the set *.width* for vertical and the set *.height* for horizontal orientation. They are always displayed with their minimum width or height. Therefore it is recommended to always set the attribute *.minwidth* or *.minheight*, otherwise Qt will use certain default values.

Toolbars are **not** automatically wrapped in the next row or column if not all of them fit into the given window width or height. Depending on *.sizeable* the following behavior may occur:

- *.sizeable = true*
  Widths respectively heights of the toolbars are ignored by Qt and reduced so that all toolbars fit into the given window width or height.
- *.sizeable = false*
  If necessary, Qt automatically increases the width or height of the window to provide space for all toolbars of a row. Thus the set window width or height will be overridden by Qt.

A new row or column can only be forced through the *.dock_line* attribute.

For ***toolbars*** with the *notepage* style, positioning depends on the *opt_nested_docks* and *opt_tabbed_docks* ***window*** options. If both options are deactivated (default), then *.dock_line* is ignored and all toolbars are arranged side by side (horizontal) or one above the other (vertically).

If the option *opt_nested_docks* is active, then *.dock_line* is considered. Nesting is always relative to the previous toolbar of the same docking area. In “Figure 49” the toolbars “eins”, “zwei” and “drei” each have *.dock_line = 1*, “vier” has *.dock_line = 2* and is embedded into „“drei”“.

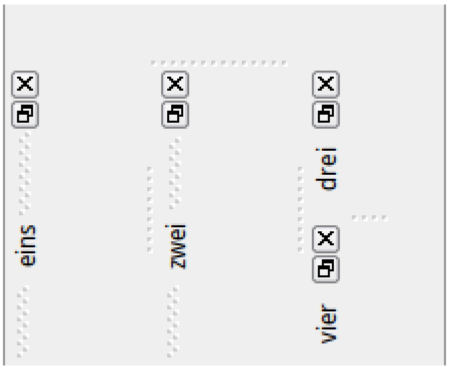


Figure 49: *Toolbars with style notepage and opt_nested_docks active*

If the *opt_tabbed_docks* option is active, all toolbars with the same *.dock_line* are displayed as consecutive tab pages (“Figure 50”).

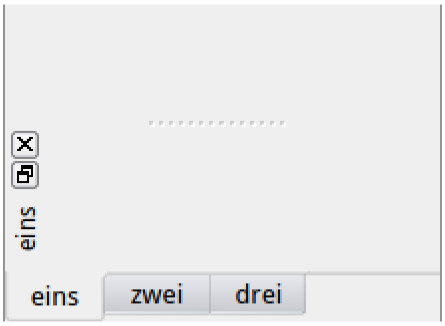


Figure 50: *Toolbars with style notepage and opt_tabbed_docks active*

Depending on the UI style, no boundary from the client area may be displayed for *.style = notepage*.

When switching between horizontal and vertical docking, the width and height are not reversed, because depending on the type of the toolbar, changes of orientation may already occur during the move.

The attributes *.height[]* and *.width[]* are only supported and applied in the *notepage* style.

42.9.1 Behavior

With “tabbed toolbars”, activating a tab or the associated visualization sends an *activate* event (similar to notebook and notepage).

Sending events such as *activate*, *deactive*, *close*, *move* and *resize* strongly depends on the Qt toolbar object in use. Internal states such as “User is still dragging” cannot be taken into account.

“Table 2” summarizes the differences and limitations of the ***toolbar*** in the IDM FOR QT.

Table 2: *Differences and limitations of the toolbar*

.style	Conditions	Limitations and Differences
all		*.autoalign* is not supported.
all	*.sizeraster = true* and *.docking <> dock_window*	Toolbar area is not adjusted to grid size.
all	*.dock_offset > 0* and *.docking <> dock_window*	Toolbars cannot be positioned at a distance (whitespace).
notepage		*.max_height*, *.max_width*, *.min_height*, and *.min_width* have no effect.
toolbar	*.sizeable = true* and *.docking <> dock_window*	No size bar; toolbar size can only be influenced in one direction (depending on the docking position). The size in the other direction is determined by the maximum width or height. Behavior corresponds to *.autosize = true*.
toolbar	*.sizeable = true* and *.docking = dock_window*	Toolbar size cannot be changed.
toolbar	*.autosize = true* and *.docking <> dock_window*	Maximum possible width or height.
toolbar	*.docking <> dock_window* and set minimum or maximum width or height	Minimum and maximum sizes do not affect the overall size of the toolbar, but cut content or create empty areas.
toolbar		No *activate* and *deactivate* events.
toolbar	Move while preserving the docking state with *.docking = dock_window*	No *move*events.

42.10 Miscellaneous

The child objects of toolbars have to be arranged manually, even in the case of changing the docking position. It is recommended to hide a toolbar before changing its docking position from the Rule Language. Then the child objects can be rearranged before the toolbar is made visible in the new docking position.

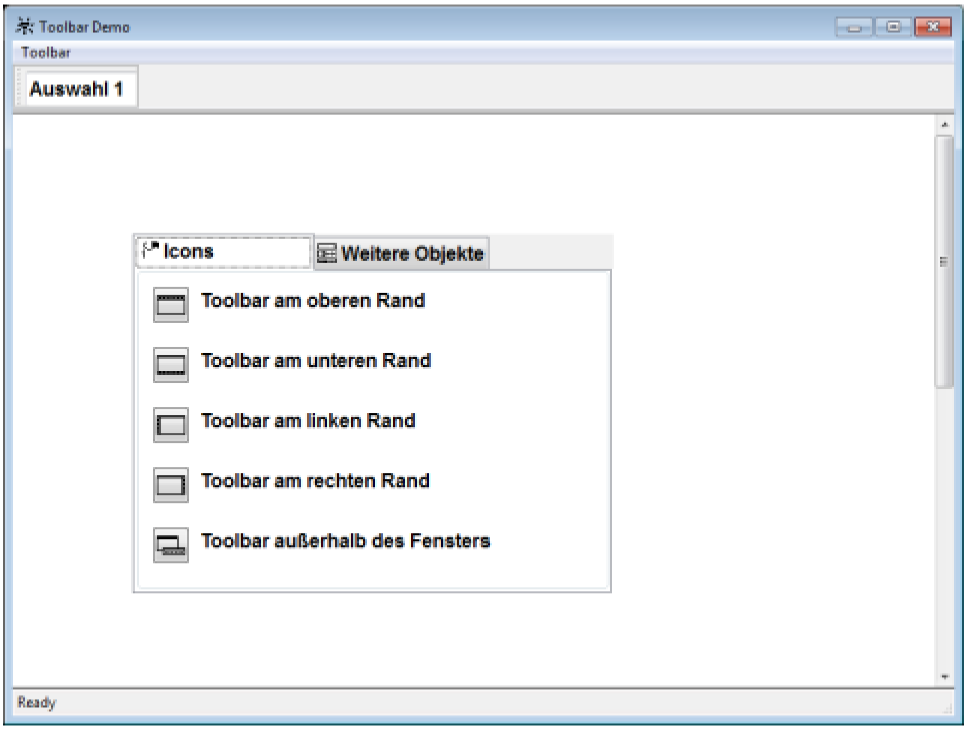
Accelerators for the child objects of toolbars also work in the parent window. If the same accelerators are defined in the parent window they take priority over those of the toolbar children.

42.11 Example

```
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] "Heading";
  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] "Option 1";
  .text[2] "Option 2";
  .text[3] "Option 3";
  .text[4] "Option 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 Window ---------------------------------
!! ------------------------------------------------------------------
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 "Setup";
      .font Fn;
      on select
      {
        Nb.visible := true;
      }
    }
    child menuitem
    {
      .text "New Toolbar";
      .font Fn;
      on select
      {
        create(MTb,W);
      }
    }
  }
  child MTb Tb1
  {
    .title "Toolbar 1";
    .dock_line 1;
    .font Fn;
    child MPt Pt
    {
      .font Fn;
    }
  }

  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 at top edge";
        .font Fn;
        .sensitive false;
      }
      child MImUp ImUp
      {
        .ytop 10;
        .xleft 10;
        .source SrcCopy;
      }
      child statictext
      {
        .xleft 50;
        .ytop 63;
        .text "Toolbar at bottom edge";
        .font Fn;
        .sensitive false;
      }
      child MImDown
      {
        .ytop 60;
        .xleft 10;
        .source SrcCopy;
      }
      child statictext
      {
        .xleft 50;
        .ytop 113;
        .text "Toolbar at left edge";
        .font Fn;
        .sensitive false;
      }
      child MImLeft
      {
        .ytop 110;
        .xleft 10;
        .source SrcCopy;
      }
      child statictext
      {
        .xleft 50;
        .ytop 163;
        .text "Toolbar at right edge";
        .font Fn;
        .sensitive false;
      }
      child MImRight
      {
        .ytop 160;
        .xleft 10;
        .source SrcCopy;
      }
      child statictext
      {
        .xleft 50;
        .ytop 213;
        .text "Toolbar outside the window";
        .font Fn;
        .sensitive false;
      }
      child MImWindow
      {
        .ytop 210;
        .xleft 10;
        .source SrcCopy;
      }
    }
!!
!! ------------------- Notepage 'Objects' ---------------
!!
    child notepage Np2
    {
      .target Tar;
      .title "More Objects";
      .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; }
}
!!
!! -------------------- Notebook ----------------
!!

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
}
```

This example produces the image below.



43 transformer

The transformer object allows for traversing an XML Document or an IDM hierarchy, whereby during the traversing semantic actions can be carried out on particular nodes. Through this it is easy to implement a transformation of data. For example, XML data can be transferred into IDM objects or vice versa, XML Documents can be generated from the data contained in IDM objects. Because semantic actions are described through user-defined code, the transformations can have many different forms.

Definition

```
{ export | reexport } { model } transformer { <Identifier> }
{
  [ <atribute definition> ]
  [ <method definition> ]
}
```

The following means are available to the IDM programmer regarding the definition of transformations.

- A transformation is started by invocation of the apply method of a transformer object. The source and the target of the transformation are passed as parameters, (i.e. a doccursor object as source and an IDM object as target, which either collects the data or delegates it elsewhere).
- It is normally desired that during each transformation a certain amount of nodes, regardless if these are IDM objects or nodes of an XML tree, are run through in a certain sequence. The apply method implements such a sequence by beginning with a start node (source) and calling the transformer's select_next method, which determines the successor of the actual node, in a loop. The sequence ends when the select_next method returns 0. The default implementation of the select_next method defines a pre-order sequence. The IDM programmer is able to intervene at two different points in this process. First of all, the apply method can be overwritten in order to set the start node or to change the abort condition. Secondly, the select_next method can be overwritten and therefore the entire sequence in which the nodes are visited can be redefined.
- After it has been guaranteed that all nodes are visited at least once, then a possibility must exist for setting the semantic action(s) to be carried out on certain nodes. For this purpose mapping objects are defined as children of the transformer object. Each of these objects defines a pattern in its name attribute, which is used to check if the currently visited node should trigger an action or not. The action is set through the action method of the mapping object that has to be overwritten by the IDM programmer. This method receives the currently visited node as first parameter, and the target object coming from the apply method of the transformer object as second parameter.
- In order to complete the picture: The transformer object also has an action method. This method is called for each visited node within the loop of the apply method mentioned above. It is then checked if the node matches the patterns of the mapping children. The sequence in which this happens is the same sequence as the mappings have been defined in the parent transformer. The inherited mapping objects are examined at the very end. If a matching mapping object is found

during the examination, the action method of the transformer calls the action method of the mapping object. This method can transfer the data from the current node to the target object. As a return value, this method can return *true*. In this case it is assumed that the node has been processed completely and no further mapping objects should be examined. Otherwise, the remaining mappings are examined and their action methods are called until no more mapping objects are left or the action method of one of the mappings returns *true*.

Events

None

Children

document

mapping

record

transformer

Parents

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

Menu

None

Methods

:action()

:apply()

:select_next()

43.1 Attributes

document[**I**]

external

external[**I**]

firstrecord

label

lastrecord

mapping[**I**]

model

module

parent

record[I]

recordcount

root

scope

userdata

43.2 Object-specific Attributes

mapping[I]

The mapping children can be accessed through the .mapping attribute. The attribute is indexed with the object index (similar to child). The sequence of the mappings within this vector determines the sequence in which nodes are compared to particular mappings during a transformation. The inherited mappings are not taken into this vector. During a transformation, the comparison to these mappings is done at the very end, i.e. the direct instances have priority. This is different in comparison to other inherited child objects within IDM, which are inserted at the beginning of the parent instance's child vector.

root

After calling the **:apply()** method the starting point of the transformation will be stored in this attribute. This makes it possible to decide during a transformation, if the starting point has been reached again, and if the transformation can be stopped.

The following cases should be distinguished:

- When the Src parameter of the apply method is a document or a doccursor, a string is saved in .root, which describes the corresponding position in the XML tree. The syntax of the string follows the convention, which is used for the .path attribute of the doccursor object (see doccursor object description). Because of this, comparisons to the .path attributes of doccursors are very easy.
- If the Src parameter of the apply method is an IDM object, this object will be stored in .root. Consequently, in this case, the data type of the attribute is object.

On the basis of the value in the .root attribute, the action and select_next methods of the transformer decide what they have to do (see description of these methods).

At the end of the apply method *.root* is set to *void* again.

The default value for the attribute is *void*.

43.3 Object-specific Methods

:apply()

The transformation is initiated with this method. The default implementation of the algorithm is as follows:

- If the Src parameter is a document or a doccursor object, it is assumed that data from the XML tree should be transferred to the IDM. In the case of a document object, a temporary doccursor object will be created which is used for the navigation within the XML tree. In the pseudo-code below, for the purpose of simplicity it is assumed that a doccursor is passed in Src.

```
: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;
}
```

- If the Src parameter is an IDM object (except for document and doccursor objects), it is assumed that data from IDM should be transferred to XML or somewhere else. The underlying code is identical to the code above, except that Src is stored in .root instead of Src.path. For example:

```
this.root ::= Src;
```

The apply method can be overwritten (similar to init).

:action()

This method is used by the apply method of the transformer (see above) to determine if the current node matches one of the mapping children. If it does, the action method of the mapping object is called.

The action method can be overwritten (similar to init).

:select_next()

This method is used by the apply method of the transformer for traversing all nodes of an XML tree or an IDM object hierarchy (see above). The default implementation of this is, that a pre-order sequence is processed by the repeated calls of the method.

The select_next method can be can be overwritten (similar to init).

43.4 Example

This example can be found in the *examples/xml* directory.

As XML Document the following file with the name “CD-Katalog.xml” is used:

```
<?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>
```

The data from this file can, for example, be read out as follows:

```
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 is for taking over data from an XML Document
  !! therefore a doccursor can be expected in Src

  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;
}
```

44 treeview

The ***treeview*** object offers the possibility to represent hierarchical information in a tree structure. The branches of a tree (also called subtrees) can be clicked shut to increase clarity. The ***treeview*** object is programmed almost in the same way as a ***listbox***; only a single selection is allowed though.

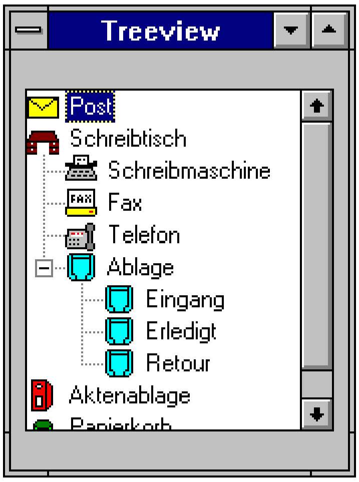


Figure 51: *Example of treeview object*

Definition

```
{ export | reexport } { model } treeview { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <hierarchy attributes>
  <layout attributes>
  <text attributes>
  <object-specific attributes>
}
```

Two different operations can be carried out interactively:

- **Opening/Closing of subtree**
  This can be carried out either by switching the +/- button or by a double click on the text.
- **Selection**
  Single or double click on an entry or changing the entry via keyboard control such as the cursor keys Left, Right, Up, Down, Page Up, Page Down, Home and End. Left/Right has the effect to close/open the subtree when used for the first time. Only after this state has been achieved, Up/Down will work.

The textual "emulation" has other differences apart from the visible ones:

- The +/- button cannot be activated; a subtree is opened/closed with a double click.
- The state of the subtree is not changed by using cursor keys Left and Right.

For the complete setting of treeview content including .open[], .level[] and .picture[] from C or COBOL side, no new interface functions have been introduced. Please use the existing functions **DM_SetVectorValue()** and **DM_GetVectorValue()**.

The following events are triggered by interactive user operations:

event(s)	allocated attributes at thisevent	trigger
open, close	.index	opening or closing of subtree
activate	.index	another entry is selected
select	.index	single click on entry (selection)
dbselect	.index	double click on entry

Further standard events, which are listed below, are available. Their meaning is described in manual "Rule Language".

Events

activate

close

cut

dbselect

extevent

focus

help

hscroll

key

open

paste

scroll

select

vscroll

Children

document

record

transformer

Parent

groupbox

layoutbox

module

notepage

splitbox

toolbar

window

Menu

Pop-up menu

Methods

:childcount()

:childindex()

:delete()

:exchange()

:find()

:insert()

:parent()

:reparent()

44.1 Attributes

Attribute	RSD	PSD	Properties	Short description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for Microsoft UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for Microsoft UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
activeitem	integer	integer	S,G/D/C	active treeview element

Attribute	RSD	PSD	Properties	Short description
bgc	identifier	color	S,G/D/C	background color of object
bordercolor	identifier	color	S,G/D/C	bordercolor of object
borderraster	boolean	boolean	S,G/D/-	controls computation of geometry with active grid
borderwidth	integer	integer	S,G/D/C	width of object border
class	class	class	-,G/-/-	object class
content[I]	string	string	S,G/D/C	content of treeview object Please refer also to chapter “.content and Content-specific Attributes”
control	identifier	instance	-,G/-/-	control the object currently belongs to
cursor	identifier	cursor	S,G/D/C	cursor belonging to the object
cut_pending	boolean	boolean	S,G/-/-	cut operation has not been carried out yet
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during cut operation
dialog	identifier	instance	-,G/-/-	object dialog
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class
fgc	identifier	color	S,G/D/C	foreground color of object
firstchar	integer	integer	S,G/D/C	number of the first displayed character (horizontal scroll position)
firstrecord	object	record	S,G/-/C	accesses the first record of an object

Attribute	RSD	PSD	Properties	Short description
focus	boolean	boolean	S,G/-/-	keyboard focus of object
focusitem	integer	integer	S,G/-/C	focus is positioned on a certain item
font	identifier	font	S,G/D/C	font of object
function	identifier	func	S,G/D/C	function of object
groupbox	identifier	instance	-,G/-/-	groupbox the object currently belongs to
height	integer	integer	S,G/D/C	indicates object height
help	string identifier	string text	S,G/D/C	helptext of object
index	integer	integer	-,G/-/-	current index of object in the child list of its parent
itemcount	integer	integer	S,G/D/C	number of elements in a treeview object
label	string	string	S,G/D/C	name (ID) of object
lastrecord	object	record	S,G/-/C	accesses the last record of an object
level[I]	integer	integer	S,G/D/C	hierarchy level of each item Please refer also to chapter ".content and Content-specific Attributes"
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
member[I]	attribute	attribute	-,G/-/-	i-th user-defined attribute of object
membercount	integer	integer	-,G/-/-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	menu of object
model	identifier	instance	S,G/D/C	model belonging to object

Attribute	RSD	PSD	Properties	Short description
notepage	identifier	instance	-,G/-/-	notepage the object currently belongs to
open[I]	boolean	boolean	S,G/D/C	defines whether a node is open or not Please refer also to chapter ".content and Content-specific Attributes"
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
picture[I]	object	tile	S,G/D/C	picture for each item (IDM for Windows and IDM for Qt) Please refer also to chapter ".content and Content-specific Attributes"
posraster	boolean	boolean	S,G/D/C	indication of position refers to raster
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectability of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from the left (in pixel)
real_y	integer	integer	-,G/-/-	real distance from the top (in pixel)
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)

Attribute	RSD	PSD	Properties	Short description
sizeraster	boolean	boolean	S,G/D/C	size referring to the raster of parent object
statushelp	string identifier	string text	S,G/D/C	text to be displayed in the statusbar
style[enum]	boolean	boolean	S,G/D/C	display properties
toolbar	object	object	-,G/-/-	toolbar of object
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
topitem	integer	integer	S,G/D/C	first treeview element to be displayed
userdata	anyvalue	anyvalue	S,G/D/C	userdata of object
userdata[I]	anyvalue	anyvalue	S,G/D/C	userdata of treeview elements
visible	boolean	boolean	S,G/D/C	visibility of object
width	integer may be 0	integer	S,G/D/C	current width of object
window	identifier	instance	-,G/-/-	window the object currently belongs to
xauto	integer (-1, 0, 1)	integer	S,G/D/C	type of x-coordinate definition
xleft	integer	integer	S,G/D/C	x-coordinate, distance to the left
xright	integer	integer	S,G/D/C	x-coordinate, distance to the right
yauto	integer (-1, 0, 1)	integer	S,G/D/C	type of y-coordinate definition
ybottom	integer	integer	S,G/D/C	y-coordinate, distance to the bottom
ytop	integer	integer	S,G/D/C	y-coordinate, distance to the top

Further information is provided in the following chapters and in the “Attribute Reference”.

44.1.1 .content and Content-specific Attributes

The contents of the ***treeview*** are set in the content[**I**] attribute. The individual items (and therefore index **I**) have to be consecutive (1, 2, 3, …).

The level[**I**] attribute then determines the hierarchy level (indentation) of the respective item. If an image shall be displayed, this is defined in the attribute picture[**I**].

The attribute open[**I**] determines whether a node is expanded or collapsed.

Remark

Please note that only by setting *.content[I]* further access to the content-specific attributes of this item is allowed. Therefore, the content must first be set first, followed by the content-specific attributes.

44.1.2 Note on Obsolete Attributes

The obsolete attributes *.focusitem* and *.focusable* are no longer supported and generate an “ignoring” warning when setting or resetting (**setinherit**) them and a `fail` when querying them.

44.2 Types of Information

The hierarchical information is stored in attribute fields. For each entry (= node of hierarchical tree) a static text and a tile is stored in the attributes .content[] and .picture[]; in the attribute .level[] the hierarchical relation is defined with a numeric value. The boolean attribute .open[] is used to make subtrees visible/invisible.

These attributes are used to store dynamical information and are thus not inherited! For attribute .content[] the index range is from 1 ... n; n represents the value of .itemcount here. For the attributes .level [], .picture[] and .open[] the index range is defined from 0 ... n. The element 0 serves as default value for unset values in the index range 1 ... n. The field size can be defined in the attribute .itemcount or can dynamically be maximized by one item by setting the .content[] field to n+1.

44.3 Tree Concept

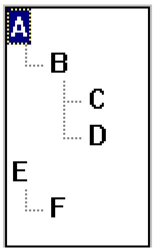
The representation of the items 1 ... n in a hierarchical structure, a tree with parent-child relations, is defined with the attribute .level[]. Its integer value indicates indention. Here the absolute value is not relevant, but the difference to the previous one.

If the value is greater than the previous value, then the entry will be a child of the previous one. If the value is the same, the entry will be a sibling. If the value is smaller than the previous one, then the entry will be a child of a previous one whose indention is even smaller. We talk about a root, if no parent is found in the index range 1 ... n.

Example

Index I	.content[I]	.level[I]
1	A	1
2	B	2
3	C	3
4	D	3
5	E	1
6	F	2

results in the following tree structure:



Please refer to ref: attribute .level[] for a more specific definition. All methods known by the listbox can be applied to the treeview. However these methods also affect the attributes .level[], .open[] and .picture[] in the same way. In addition special methods for the manipulation and query of the tree structure are available.

As with listbox the attribute .activeitem allows to activate an entry within range 1 ... n of a tree, or to query the selection state. The selectivity depends on the visibility of the tree entry, and thus also on the attributes .level[] and .open[]. The following is valid for the Rule Language as well as for the interactive operation: On setting .activeitem all parents are set to .open[] := true. If a subtree is set invisible with .open[I] := false, although it contains a selected child, the entry I will be selected and .activeitem will be adjusted correspondingly.

44.4 Representation

The representation differs from the listbox as follows:

1. The hierarchical relation of entries is represented by indention and optionally with additional lines (Ref. attributes .style[] and attributes .level[]). If the root of subtrees is set invisible (.open[] is false), only the root will be displayed.
2. Optional picture for a text entry (ref: attribute .picture[]).

3. Optional +/- button, which will become visible for entries with children and which is used to open (+)/close(-) a subtree (ref: attribute .open[]).

Moreover all geometry and layout attributes relevant for listbox are valid.

44.5 Specific Attributes

In this chapter the specific attributes are described in more detail. The standard and geometry attributes as well as the attribute content will not be described here as they behave exactly in the same way as those for the listbox. Please note that the content-specific attributes of the entry can be accessed only after having set .content[]. This is why you have to set content before setting the content-specific attributes.

44.5.1 open[I]

This attribute is used to define the visibility of a subtree within a treeview object. If .open[I] is set to false, the direct children of subtree I won't be displayed, if not, the direct children will be visible. The hierarchical division into subtrees is defined by the attribute .level[I].

The index range 0 ... n is valid, n represents the value of the attribute .itemcount for the treeview. The value of .open[0] represents the default value for unset values within the range 1 ... n.

This attribute is only statically available and is not inherited.

44.5.2 level[I]

This attribute of the treeview object defines the hierarchical order of the entries. The resulting tree is characterized by a parent-child hierarchy.

The valid index range is 0 ... n, n represents the value of attribute .itemcount. The default value for unset attribute values of the range 1 ... n is indicated in .level[0].

The hierarchy is defined by the difference between subsequent .level[I] values and **not** by its absolute value. This, however, can be achieved by assigning the .level[I] values in a disciplined manner.

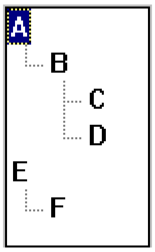
A single hierarchy can thus be achieved by any number of different .level[] definitions.

For the tree hierarchy only the range 1 ... n is considered, as only this range actually has a content (.content[I]) to be displayed.

The following rules are valid:

this.level[I+1] > this.level[I]	entry I+1 is a child of I
this.level[I+1] = this.level[I]	entry I+1 is a sibling of I
this.level[I+1] < this.level[I]	Entry I+1 is a child of entry J. The parent J is within 1 ... I-1 and this.level[J] is < this.level[I+1]. For all children in J+1 .. I+1 applies: this.level[J] < this.level[K]. If no J complies with the above conditions, then J=0 will be the parent.

Example: Let's have a look at the following treeview.



This hierarchy can be equally defined with e.g. the following .level[] values:

index I	.content[I]	.level[I] (absolute values)	.level[I] (other possibilities)
1	A	1	5
2	B	2	7
3	C	3	900
4	D	3	88
5	E	1	-42
6	F	2	0

Note: This attribute is available only statically and is not inherited.

44.5.3 picture[I]

This attribute defines the ***tile*** to be displayed for an entry.

The index range is *0 … .itemcount* with the value of *.picture[0]* representing the default value for unset values within *1 … .itemcount*.

This attribute is available only statically and is not inherited.

Notes

The tile will be displayed on the left-hand side of the text. Please note that the size of the displayed tiles cannot vary in the treeview due to technical restrictions. The size of the tiles is predefined by the tile size of *.picture[0]*. If this size is not specified the size of Smallicons (16x16) will be used as default. Bigger tiles of the field *.picture[]* will be clipped; smaller ones will be enlarged. There will be space for the tile on the left of the text as soon as at least one tile has been set. When *.picture[0]* is not set, a substitute icon (white cross on red background) will be displayed for items without a tile.

This attribute is not supported by the IDM FOR MOTIF.

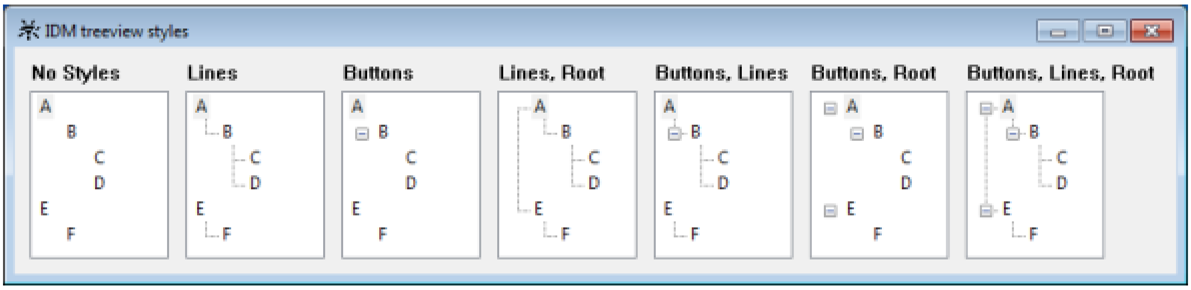
44.5.4 style[E]

This treeview attribute is used to activate specific representation properties. These properties are switched off by default.

This attribute is indexed with an enumeration. Its meaning is described in the following table:

dialog script	C/COBOL	meaning, if set on true
style_lines	OPT_style_lines	lines are drawn between parent and children
style_buttons	OPT_style_buttons	all entries (apart from the top parent) begin with a +/- button to clip open and shut subtrees
style_root	OPT_style_root	If style_lines are active, too, then the top parents will be connected with lines. If style_buttons is active, the top parents will also get a +/- button.

The different styles have the following representation:



Note

The three styles are supported by IDM FOR WINDOWS only. The textual simulation of the treeview on other platforms offers the option .style[style_buttons] only. This option can be compared to the "Buttons, Root" representation of the above picture.

Here as well "+"/"-" are used to represent an open/closed subtree with children; "." is used to represent an item without children.

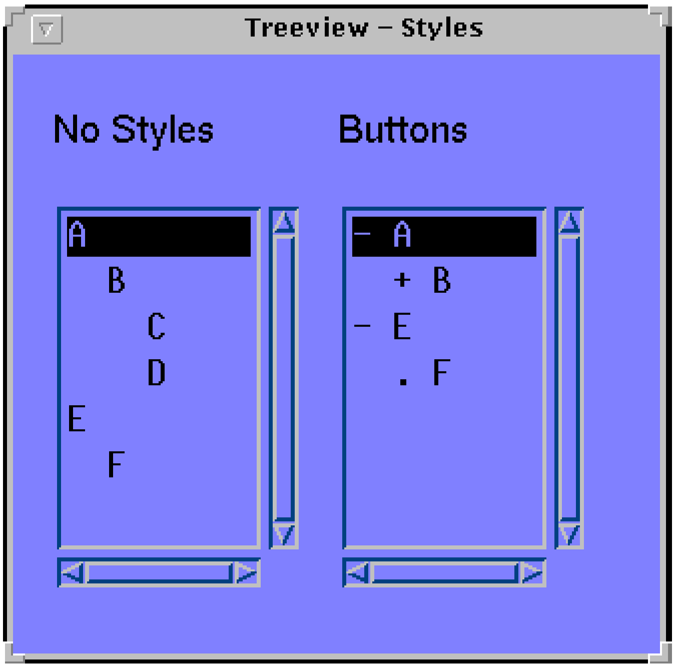


Figure 52: Treeview on Motif

44.6 Example

```
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(); }
  }
}
```

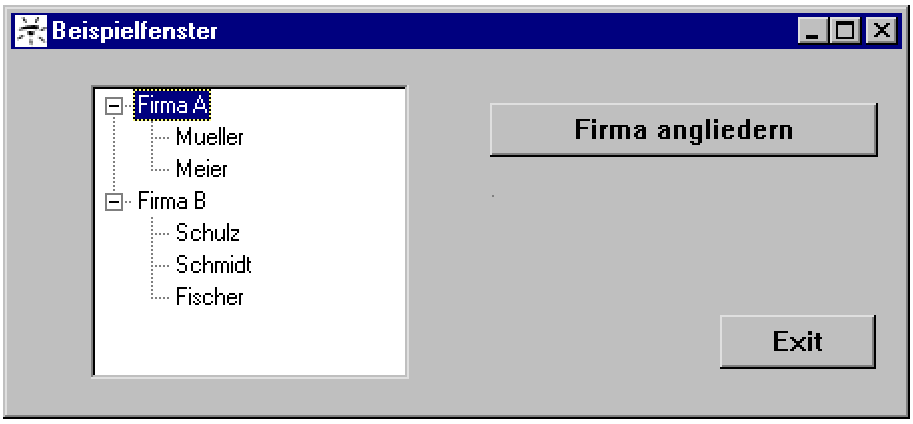


Figure 53: window with treeview object

45 window

The basic object of a window system is – as already implied by the term – the window. All other objects have to be generated within a window.

The declaration of a window starts with the keyword **window**, followed by the window identifier and the window definition.

The window definition specifies the appearance, size and various other features of the window.

Definition

```
{ export | reexport } { model } window { <Identifier> }
{
  <standard attributes>
  <plain attributes>
  <geometry attributes>
  <grid attributes>
  <hierarchy attributes>
  <layout attributes>
  <scrollbar attributes>
  <object-specific attributes>
}
```

45.1 Definition of Child Objects

All other objects supported by the DM are located within a window. Thus they must be declared as children of a window either directly or indirectly. The keyword for this is **child**. It is followed by a keyword describing the object type, and an optional identifier. This identifier only has to be specified if the object is to be addressed out of the rules or out of the application afterward. It is followed by the object definitions in braces.

For further information, see chapter “Inheritance of Attributes” in manual “Programming Techniques”.

Definition

```
{export} child  object class  {<Identifier>}
{
  object definition
}
```

Note

Integrating a menu into the window must be done with the keyword **child**. The menus integrated with **child** are located in the menu bar of the window. Their number is unlimited.

Events

activate

close

deactivate

deiconify

extevent

help

hscroll

iconify

key

move

resize

scroll

select

vscroll

Children

canvas

checkbox

document

edittext

groupbox

image

listbox

menubox

menuitem

menusep

notebook

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

statictext

tablefield

toolbar

transformer

treeview

window

Parent

dialog

module

window

Menu

Pop-up menu

menuitem

45.2 Attributes

Attribute	RLD	PID	Properties	Short Description
acc_label	string object	string text	S.G/D/C	overwrites the Automation Identifier for MICROSOFT UI Automation
acc_text	object string	text string	S.G/D/C	overwrites the Automation Name for MICROSOFT UI Automation
accelerator	identifier	accel	S,G/D/C	accelerator of object
bgc	identifier	color	S,G/D/C	background color
bordercolor	identifier	color	S,G/D/C	border color

Attribute	RLD	PID	Properties	Short Description
borderstyle	enum	enum	S,G/D/C	defines the style, i.e. representation and characteristics of the borders (since IDM version A.06.01.a)
borderwidth	integer	integer	S,G/D/C	border width
child[I]	object	object	S,G/-/C	accesses the I-th child object
childcount	integer	integer	-,G/-/-	queries the number of child objects
class	class	class	-,G/-/-	class/id of object
closeable	boolean	boolean	S,G/D/C	window can be closed by closebox
cursor	identifier	cursor	S,G/D/C	cursor belonging to object
cut_pending	boolean	boolean	S,G/-/-	cut operation has not been executed yet
cut_pending_changed	boolean	boolean	-,G/-/-	changing state during a cut operation
dialog	identifier	instance	-,G/-/-	dialog to which the object belongs
dialogbox	boolean	boolean	S,G/D/C	window becomes dialogbox
display	identifier	display	S,G/D/C	screen on which the window is shown
document[I]	object	document	S,G/-/-	accesses the I-th XML Document
external	boolean	boolean	-,G/-/-	returns if the object class is an USW class
external[I]	class	class	-,G/-/-	returns the I-th registered USW class

Attribute	RLD	PID	Properties	Short Description
fgc	identifier	color	S,G/D/C	foreground color
firstchild	object	object	S,G/-/C	accesses the first child object
firstmenu	identifier	instance	S,G/-/-	first menu of window
firstrecord	object	record	S,G/-/C	accesses the first record of an object
focus	boolean	boolean	S,G/-/C	input focus on object
font	identifier	font	S,G/D/C	object font
function	identifier	func	S,G/D/C	function belonging to object
height	integer	integer	S,G/D/C	object height
help	string identifier	string text	S,G/D/C	help text of object
hsb_arrows	boolean	boolean	S,G/-/-	defines whether horizontal scrollbar has arrows at its end
hsb_linemotion	integer	integer	S,G/D/C	horizontal scroll value for scrolling line by line
hsb_optional	boolean	boolean	S,G/D/C	horizontal scrollbar is only displayed if necessary
hsb_pagemotion	integer	integer	S,G/D/C	horizontal scroll value for scrolling page by page
hsb_visible	boolean	boolean	S,G/D/C	visibility of horizontal scrollbar
icon	object	tile	S,G/D/C	assigns object to icon
iconic	boolean	boolean	S,G/D/C	window can be turned into icon
iconifyable	boolean	boolean	S,G/D/C	mechanism for iconifying in window title

Attribute	RLD	PID	Properties	Short Description
ignorecursor	boolean	boolean	S,G/D/C	ignoring the cursor
label	string	string	S,G/D/C	name/identifier of object
lastchild	object	object	S,G/-/C	accesses the last child object
lastmenu	identifier	instance	S,G/-/C	last menu of window
lastrecord	object	record	S,G/-/C	accesses the last record of an object
mapped	boolean	boolean	S,G/D/-	defers the display of a visibly created object
maxheight	integer	integer	S,G/D/C	maximal window height
maximized	boolean	boolean	S,G/-/-	maximize state of window
maxwidth	integer	integer	S,G/D/C	maximal window width
member[I]	attribute	attribute	-,G/-,-	user-defined attribute [I] of object
membercount	integer	integer	-,G/-,-	number of user-defined attributes
menu	identifier	instance	S,G/D/C	object menu
menu[I]	identifier	instance	S,G/-/C	menu child [I] of window
menubgc	identifier	color	S,G/D/C	background color of menu
menucount	integer	integer	-,G/-/C	number of menu elements in window
menufgc	identifier	color	S,G/D/C	foreground color of menu
minheight	integer	integer	S,G/D/C	minimal window height
minwidth	integer	integer	S,G/D/C	minimal window width
model	identifier	instance	S,G/D/C	model belonging to object

Attribute	RLD	PID	Properties	Short Description
moveable	boolean	boolean	S,G/D/C	window can be moved interactively
options[enum]	boolean	boolean	S,G/D/-	special options of object
parent	identifier	instance	S,G/-/-	parent of object
posraster	boolean	boolean	S,G/D/C	positions refer to grid
.preedit	enum	enum	S,G/D/C	controls the display and selection of the input mode for the ***edittexts*** within the window
real_height	integer	integer	-,G/-/-	real height of object
real_sensitive	boolean	boolean	-,G/-/-	real selectivity of object
real_visible	boolean	boolean	-,G/-/-	real visibility of object
real_width	integer	integer	-,G/-/-	real width of object
real_x	integer	integer	-,G/-/-	real distance from left (in pixel)
real_xraster	integer	integer	-,G/-/-	width of grid internally used
real_y	integer	integer	-,G/-/-	real distance from top (in pixel)
real_yraster	integer	integer	-,G/-/-	height of grid internally used
record[I]	object	record	S,G/-/C	accesses the I-th record of an object
recordcount	integer	integer	-,G/-/-	queries the number of child records
reffont	identifier	font	S,G/D/C	reference font of object
scope	integer (1, 2, 3)	scope	-,G/-/-	queries the object type (Default, Model or instance)
sensitive	boolean	boolean	S,G/D/C	selectivity of object

Attribute	RLD	PID	Properties	Short Description
sizeable	boolean	boolean	S,G/D/C	window size can be changed interactively
sizeraster	boolean	boolean	S,G/D/C	size refers to grid of parent object
statushelp	string identifier	string text	S,G/D/C	text to be displayed in the statusbar
sysmodal	boolean	boolean	S,G/D/C	specifies whether ***window*** defined as "dialogbox" (*.dialogbox* = *true*) is displayed in front of all other windows on the desktop
tile	object	tile	S,G/D/C	tile resource used as background image
tilestyle	enum	enum	S,G/D/C	controls how the background image set in *.tile* is arranged
title	string identifier	string text	S,G/D/C	text displayed in titlebar
titlebar	boolean	boolean	S,G/D/C	window has title bar
titlebgc	identifier	color	S,G/D/C	background color of title bar
titlefgc	identifier	color	S,G/D/C	foreground color of title bar
toolbar	object	object	-,G/-/-	toolbar of object
toolbarcount	integer	integer	-,G/-/-	returns number of toolbars
toolhelp	text	text	S,G/-/-	specified text to be displayed in pop-up
userdata	anyvalue	anyvalue	S,G/D/C	object userdata (any DM data type)

Attribute	RLD	PID	Properties	Short Description
userplaced	boolean	boolean	S,G/D/C	window is positioned when opened
vheight	integer	integer	S,G/D/C	internal (virtual) height of object
vsb_arrows	boolean	boolean	S,G/-/-	defines whether vertical scrollbar has arrows at its end
visible	boolean	boolean	S,G/D/C	visibility of object
vsb_linemotion	integer	integer	S,G/-/C	vertical scroll value for scrolling line by line
vsb_optional	boolean	boolean	S,G/D/C	vertical scrollbar will be displayed, if necessary
vsb_pagemotion	integer	integer	S,G/D/C	vertical scroll value for scrolling page by page
vsb_visible	boolean	boolean	S,G/D/C	visibility of vertical scroll-bar
vwidth	integer	integer	S,G/D/C	internal (virtual) width of object
width	integer (may be 0)	integer	S,G/-/C	width of the object
xauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of x-coordinates
xleft	integer	integer	S,G/D/C	x-coordinate, distance from left
xorigin	integer	integer	S,G/D/C	shift of the origin along the x-axis in objects with scrollbars
xraster	integer	integer	S,G/D/C	units in x-direction
xright	integer	integer	S,G/D/C	x-coordinate, distance from right

Attribute	RLD	PID	Properties	Short Description
yauto	integer (-1, 0, 1)	integer	S,G/D/C	definition type of y-coordinates
ybottom	integer	integer	S,G/D/C	y-coordinate, distance from bottom
yorigin	integer	integer	S,G/D/C	shift of the origin along the y-axis in objects with scrollbars
yraster	integer	integer	S,G/D/C	units in y-direction
ytop	integer	integer	S,G/D/C	y-coordinate, distance from top

45.3 Specific Attributes

Standard Attributes

The attribute *.visible* defines whether the window is visible immediately on starting the dialog.

Plain Attributes

In the object window, these attributes always refer to the screen on which the object is to be displayed, i.e. the coordinates are always specified in relation to the screen.

Of course it is possible to specify the coordinates and the dimension of a window in relation to the grid of the affiliated dialog. To do this, the attributes *.posraster* and *.sizeraster* have to be set to true.

Geometry Attributes

The additional geometry attributes *.minheight*, *.minwidth*, *.maxheight*, *.maxwidth* define the borders within which the window size can be changed interactively.

The attributes *.vheight* and *.vwidth* define which internal size the window is to have. It may be that only part of the internal size is visible at one time. To make the rest of the window visible you have to scroll the window.

The visible window part is controlled by the attributes *.xorigin* and *.yorigin*. They specify the relative shift of the window origin, which is normally located in the top left corner of the window.

The attributes *.xraster* and *.yraster* define the unit by which the position and dimension of the children are to be specified. The actual pixel coordinates then result from multiplying the grid factor *.xraster* or *.yraster* by the specified coordinate. The attribute *.reffont* can link the given grid units to the used font; the DM then calculates the values for *.xraster* and *.yraster*.

Layout Attributes

.borderstyle

Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.

45.3.1 MDI Window

The MS-Windows multi-document interface (**MDI**) is supported by the Dialog Manager. To use this feature, you have to ensure the following:

To create a MDI frame window:

- define the window as toplevel window
- the window has to have a titlebar
- the window may not have children other than windows

To create a MDI child window:

- define the window as immediate child of a MDI frame window

All other windows are normal windows that can be toplevel windows or child windows. Keep in mind that a normal child window cannot get an active border and does have the same system accelerator as a toplevel window.

Moreover, you have to pay attention to the following:

MDI frame window:

- cannot have children other than windows, therefore adding other children is an error
- setting titlebar to FALSE is an error

MDI child window:

- always has a titlebar
- cannot have a menubar
- can only be a child of a MDI frame window

A normal window:

- cannot be a child of a MDI frame window
- if created as a child, always have a titlebar
- if created as a child, cannot have a menubar

Warning

Child windows not using MDI are not supported by MS Windows! This might cause a non-expected behavior of many attributes and events.

45.3.2 Scrollbar Attributes

The scrollbar attributes *.hsb_...* and *.vsb_...* define whether the window is to be provided with scrollbars.

The window can have the complete set of scollbar attributes:

- .hsb_linemotion
- .hsb_optional
- .hsb_pagemotion
- .hsb_visible
- .vsb_linemotion
- .vsb_optional
- .vsb_pagemotion
- .vsb_visible

Operation Mode of Scrollbars in Windows and Groupboxes

The visibility of scrollbars is influenced by the attributes

- .hsb/vsb_optional
- .hsb/vsb_visible

and

- .width/height
- .vwidth/vheight

The following rules are valid:

- A groupbox and a window can have horizontal scrollbars only if the virtual size *.vheight* is set.
- A groupbox and a window can have vertical scrollbars only if the virtual size *.vwidth* is set.

If no scrollbars are set, the virtual size will be ignored.

The attributes *.hsb/vsb_visible* and *.hsb/vsb_optional* control whether the scrollbars are to be visible at all times or only if necessary.

As mentioned above, scrollbars require the setting of the virtual size, i.e. if no virtual size is set, there are no scrollbars available.

If the scrollbars are visible

- and *.hsb/vsb_optional* is *true*,
  then the scrollbar is visible only if necessary.
- and *.hsb/vsb_optional* is *false*
  then the scrollbar is always visible.

Note

.hsb/vsb_optional is ignored by the Motif version, i.e. internally *.hsb/vsb_optional* is always implicitly set at *true*.

To query or to change the scrollbar slider position, the attributes *.xorigin* and *.yorigin* have to be used. These attributes specify the object origin shift.

The attributes *.pagemotion* and *.linemotion* define the shift in pixels of object contents during the relevant scrollbar action. The value *0* means that the system default value is to be used; in case of *.pagemotion* scrolling is then carried out by pages.

45.4 Particularities of the Window on Qt

The ***window*** object has four additional options for supporting toolbars and "tabbed docks", as well as sizing and positioning a window to be Qt-compliant.

Table 3: *Options of the window object*

.options[…] Value	Default Value	Meaning
opt_window_size	***false***	*true*: Size specifications (*.width*, *.height*, minimum and maximum values) refer to the entire window, including menu bar, toolbars, tabbed widgets and status bar, but without decoration (title bar, margins). ***false***: Size specifications refer to the interior ("client area") of the window.
opt_animated	***false***	*true*: Animated, interactive moving of ***toolbars***. However, this leads to significantly more *resize* and *move* events. ***false***: No animations when moving ***toolbars***.
opt_nested_docks	***false***	The option only affects ***toolbars*** with the *notepage* style. *true*: Adjacent, multi-row docks are possible. However, this leads to a not so clear operation with interactions and shifts. ***false***: Only single-row docks are possible.
opt_tabbed_docks	***false***	The option only affects ***toolbars*** with the *notepage* style. *true*: "Tabbed docks" that share the space are possible. "Tabbed docks" can be operated similar to notebooks and notepages (see chapter "Particularities of the Toolbar on Qt" at the ***toolbar*** object). ***false***: "Tabbed docks" are not possible.

As on Motif (and unlike Microsoft Windows), on Qt the *.sensitive* attribute does not affect the title bar of the ***window***. Even if *.sensitive* = *false* (and the window content is no longer operable), the title bar functions (close, move, minimize, maximize…) can still be used.

The attribute *.moveable*has no effect on Qt. Windows can always be moved.

In general, the IDM for Qt generates significantly more *move* and *resize* events than the other IDM versions, because for each pixel to be increased, decreased, or moved a corresponding event is triggered. There is no reliable way to determine the state and end of the actions.

Some window states can only be changed by internally clobbering, which triggers *activate* and *deactivate* events, however.

It may still happen that size specifications in certain constellations are not correctly put into effect (e.g. when changing them in the minimized state). Currently, the behavior may vary depending on the Window Manager (for example, when minimizing).

45.5 Example

Creating a Window

```
window Window_3
{
  .xleft      10;
  .ytop       10;
  .xauto      1;
  .yauto      1;
  .height     100;
  .width      200;
  .title      "vacation planning";
  .sensitive  true;
  .visible    true;
  .sizeable   true;
  .titlebar   true;
  .closeable  true;
}
```

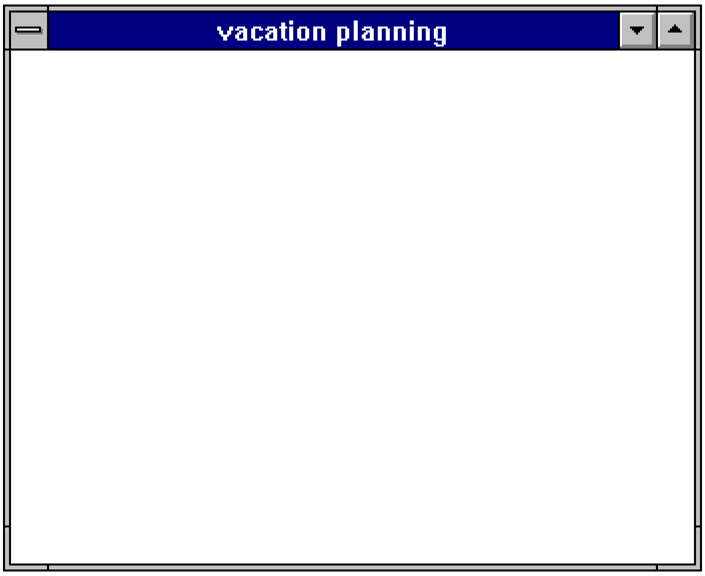


Figure 54: Window

Generation of a Window with Scrollbars on the Right and at the Bottom

```
window Window_3
{
  .title       "vacation planning";
  .xleft       10;
  .ytop        10;
  .xauto       1;
  .yauto       1;
  .height      100;
  .width       200;
  .sensitive   true;
  .visible     true;
  .sizeable    true;
  .titlebar    true;
  .closeable   true;
  .vwidth      400;
  .vheight     300;

  /* horizontal scrollbar */
  .hsb_visible  true;

  /* vertical scrollbar */
  .vsb_visible  true;
}
```

Result of the preceding example of a window with scrollbars on the right and at the bottom:

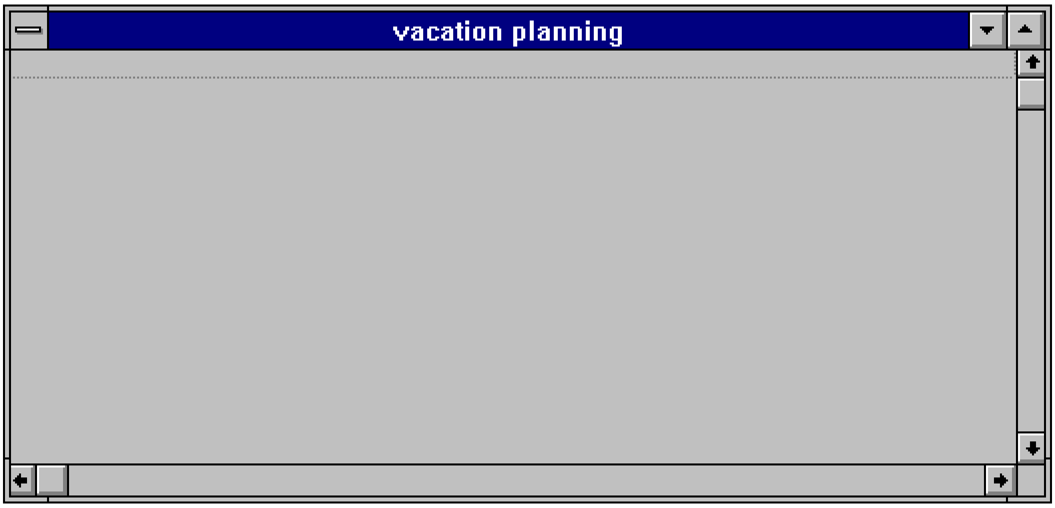


Figure 55: *Window with Scrollbars*

Index