ISA Dialog Manager

Attribute Reference

A.06.03.b

In this manual all predefined attributes of the ISA Dialog Manager objects are described. It contains the definitions and data types of the attributes in the Rule Language and the programming languages C and COBOL.

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 Correlations between attributes

1.1 Standard attributes

The *.visible* and *.sensitive* attributes specify the visibility and accessibility of an object from the user's point of view. Since the actual visibility/reachability of an object also depends on the respective parent object, the attributes *.real_visible* and *.real_sensitive* can be used to query the current, actual visibility/reachability of an object. This can be particularly useful when setting the focus via *.focus*, since setting the focus on invisible or unreachable objects leads to error messages.

As a special feature the attribute *.mapped* is available. It determines whether an object is already drawn visibly or not, when the *.visible* attribute of this object is set to *true*. If *.mapped* is *false* and *.visible* is *true*, operations with this object are already permissible in the invisible condition, which otherwise could only be executed at a visible object.

In order to display a help text in a status bar of a window when the mouse cursor hovers over an object, the text is specified in the attribute *.statushelp* at the respective object. The same applies to the popping up help texts (tooltip) which are specified in the attribute *.toolhelp*. In addition, any help texts can be specified in the attribute *.help*, whereby the programmer himself is responsible for the display.

To use keyboard shortcuts that are intended to trigger a *select* event on an object, the respective key combination is first defined as a resource and then assigned to the object via the *.accelerator* attribute.

1.2 Hierarchical attributes

Hierarchical attributes are used to indirectly access objects within the parent-child hierarchy.

While the *.groupbox*, *.notepage*, *.control*, *.window*, *.toolbar*, and *.dialog* attributes reference the next object of the corresponding class upward from the current object, *.parent* refers to the direct parent object.

Also, within child objects, *.child[integer]* can be used to reference a direct child of an object, where *.childcount* specifies the total number of direct children.

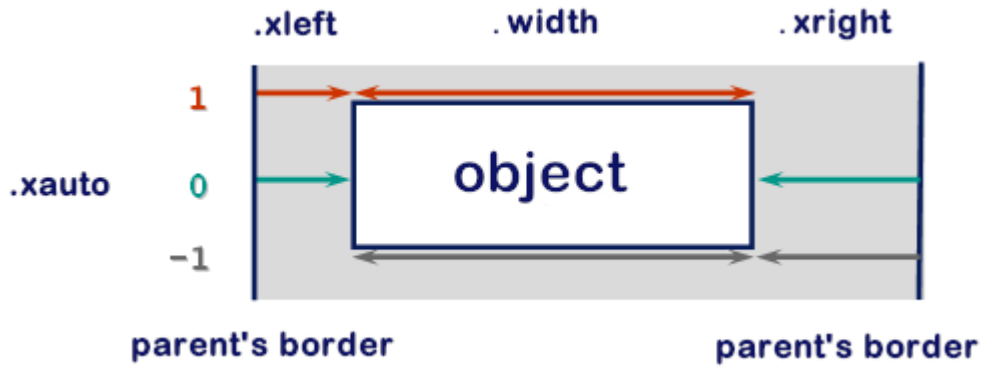
Menus, records and subcontrols take a special position and can be referenced with the attributes *.menucount* and *.menu[integer]*, *.recordcount* and *.record[integer]*, *.subcontrolcount* and *.subcontrol [integer]*, according to the direct children. These object classes are not referenced via *.child[integer]*!

1.3 Geometric attributes

Calculating the position of an object

To calculate the position of an object, the attributes *.xauto*, *.xleft*, *.width* and *.xright* or *.yauto*, *.ytop*, *.height* and *.ybottom* are necessary. The alignment is always defined relative to the parent object. In each case, two positional values are specified and the third is calculated. The attributes *.xauto* and *.yauto* control which values are specified and which value is calculated.

The graphic illustrates this as an example for the X direction, the same applies for the Y direction.



The attributes *.posraster* and *.sizeraster* indicate whether the set values correspond to raster or pixel values. If one or both attributes are set to *true*, the underlying raster is determined via the attributes *.xraster*, *.yraster* and *.reffont*.

Additional geometric attributes

Windows and toolbars have the additional geometric attributes *.minheight*, *.minwidth*, *.maxheight* and *.maxwidth*. They can be used to define the limits within which the size of the object can be changed interactively (by dragging it with the mouse).

Objects, whose content can be larger than the available display area, have additional geometric attributes, that define their virtual size and the visible section of the content. The *.vheight* and *.vwidth* attributes define the virtual height and width of the object. To display the non-visible parts, a user must scroll the object. Thereby the visible section is controlled by the attributes *.xorigin* and *.yorigin*. They define the relative translation of the object origin, i.e. the position of the visible area on the larger virtual area of the object. Normally, the object origin is located in the upper left corner of the object.

The *.xraster* and *.yraster* attributes define the horizontal and vertical units in which the position and dimension of the children are specified. The coordinates of the children in pixels are obtained by multiplying the specified coordinates (in raster units) by the raster factors *.xraster* and *.yraster*.

The *.reffont* attribute can be used to link the specification of the raster units to the character set used. The DM then calculates the values *.xraster* and *.yraster* itself.

Borderless geometry

To make the geometry behavior of objects with borders configurable, there is the attribute *.borderraster*. It can be used to specify that the geometry of the object is calculated as if it had no border.

Geometric attributes for high resolution screens

Pixel based values of geometric attributes are transformed by the IDM to match the scaling factor of the system in order to ensure a representation consistent with the system settings. The geometric units are set in **IDM pixel** values or raster as before. The **IDM pixel** values are internally extrapolated and converted into **real pixel values** according to the scaling factor. This conversion happens in both directions. Raster values are either linked to the used (HighDPI capable) character set or are set in **IDM pixel values** and then scaled. Raster units can also be resolved to **IDM pixel values** based on the underlying font.

The *.propscale* attribute controls whether the horizontal and vertical raster should be set proportionally to the maximum value, which is determined by the value calculation of xraster and yraster of the font raster. The exact calculation of the raster is described in the chapter „Berechnung der Rastergröße aus einem ReferenzfontCalculating the Grid Size from a Reference Font“ of the font resource.

See alsochapter in manual „ProgrammingTechniques“

1.4 Scrollbar attributes

Some objects, e.g. window, groupbox and notepage, can have scrollbars (sliding bars) that can be used to move the displayed contents of the object horizontally and vertically. The scrollbar attributes define the display and behavior of the scrollbars.

There are the following scrollbar attributes:

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

Attributes starting with *.hsb_* refer to the horizontal scrollbar and attributes starting with *.vsb_* refer to the vertical scrollbar.

The visibility of scrollbars is influenced by the following attributes:

- .hsb/vsb_optional
- .hsb/vsb_visible

and

- .width/height
- .vwidth/vheight

The following rules apply:

- An object can have a horizontal scrollbar only if its virtual width *.vwidth* is set and is >0.
- An object can have a vertical scrollbar only if its virtual height *.vheight* is set and is >0.

If no scrollbars are set, the virtual size is ignored.

The attributes *.hsb/vsb_visible* and *.hsb/vsb_optional* control whether the scrollbars should be visible all the time or only when they are needed.

As mentioned above, scrollbars require the virtual size to be set, i.e. if no virtual size is set, scrollbars are not available.

When the scrollbars are visible, and

- *.hsb/vsb_optional = true*
  => the scrollbar will only be visible when it is really needed.
- *.hsb/vsb_optional = false*
  => the scrollbar is always visible.

Remark

In the Motif version *.hsb/vsb_optional* is ignored, i.e. internally *.hsb/vsb_optional* is implicitly always set to true.

To query or change the position of the scrollbar slider, the *.xorigin* and *.yorigin* attributes must be used. These two attributes specify the displacement of the actual object origin.

The *.hsb/vsb_linemotion* and *.hsb/vsb_pagemotion* attributes define by how many units *.xorigin* or *.yorigin* - and thus the object content - should be moved during the respective scroll action. Units are either raster units or pixels, depending on whether .sizeraster is set for the object or not. The value 0 means that the default value of the system is taken. For *.hsb/vsb_pagemotion*, scrolling is then done page by page.

There are objects, e.g. the listbox, which display scrollbars if their content does not fit into the existing display area, but which do not have scrollbar attributes. For these objects, the scrollbars are controlled by the window system and cannot be influenced by the ISA Dialog Manager.

1.5 Layout attributes

The background color of an object is determined by assigning a color resource to the *.bgc* attribute. For the foreground color, i.e. usually the color of the text within an object, the color resource is

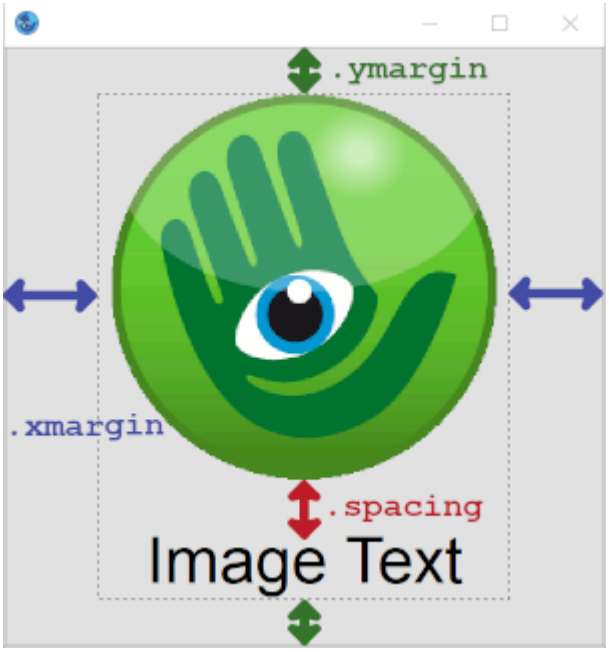
specified in the *.fgc* attribute.

The type of font representation is done by assigning a font resource to the *.font* attribute to the respective object.

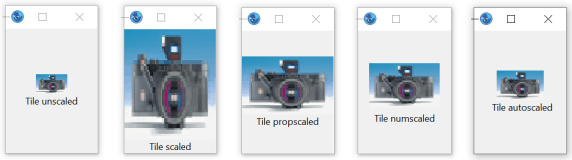
Objects, that can display images get these assigned to the *.picture* attribute as a tile resource. This attribute is indexed with integer values if it is a list object (e.g. treeview).

Layout attributes of the image object

For the image object, the *.xmargin* and *.ymargin* attributes define the spacing between the image border and the display area. The attribute *.spacing*, on the other hand, defines the distance between tile and text within the display area of the image. The following graphic illustrates the combination of said attributes:



The *.scalestyle* attribute can be used to specify the type of scaling of the tile within the display area. The following graphic illustrates the available scaling styles:



Layout attributes for high resolution screens

The *.tiledpi* attribute of the setup object can be used to specify the DPI resolution for which the application's tiles are designed. The size of an image or pattern is then converted to the currently valid DPI value based on the*.tiledpi*.

See also chaper „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

1.6 Text attributes

The title bar of a window and the title of a menu box are specified in the *.title* attribute. For all other texts to be displayed, the *.text* and *.content* attributes are used. An object usually uses only one of these attributes. Thus, the *.content* attribute applies only to edit text and list objects, indexed for the latter.

The ***Poptext*** / Combobox object has a **special position**. Here, the list content is accessed via *.text [integer]*, at the same time, read access to the currently displayed text is also supported with .content. Since write access to *.content* depends on the value in the *.style* attribute (poptext, edittext, listbox) of the poptext, the attribute and object description must be observed here.

2 Attributes in Alphabetical Order

2.1 .acc_label

With this attribute, the Automation Identifier that is queried for a user interface object from the IDM via the MICROSOFT UIA Interface can be overwritten. With an empty string, a meaningful Automation Identifier is usually predefined by the Windows Control or the UIA support in the IDM.

Definition

Data type
string, *object* [***text***]

Access
get, set

changed event
yes

C
Identifier: AT_acc_label
Data type: DT_string, DT_text

COBOL
Identifier: AT-acc-label
Data type: DT-string, DT-text

Inheritance
yes

Default value
""

Classification
standard attribute

Support of attribute by objects

Object	Support of the Attribute
filereq, ***messagebox***	Attribute has **no** effect
menubox, ***menuitem***, ***menusep***	

Object	Support of the Attribute
canvas, ***spinbox***, ***statusbar***, ***tablefield***	Attribute is supported
groupbox, ***notebook***, ***notepage***, ***splitbox***	
image, ***layoutbox***, ***window***	
rectangle, ***scrollbar***	
checkbox, ***pushbutton***, ***radiobutton***	
edittext, ***poptext***, ***statictext***	
control, ***listbox***, ***treeview***	
other object classes	Attribute is **not** supported

Remark

This attribute is only relevant for automated external control with active MICROSOFT UIA support. The attribute is without function on QT and MOTIF.

When overwriting, the rules given for the AutomationId in the MICROSOFT UI Automation documentation should be followed.

Availability

Since IDM version A.06.01.e

See also

Chapter “UIA Object Identification” in manual “Automatic Testing and Accessibility”

2.2 .acc_text

With this attribute, the Automation Name that is queried for a user interface object from the IDM via the Microsoft UIA Interface can be overwritten. When the value is *null*, then a meaningful name is usually predefined by the Windows Control or the UIA support in the IDM.

Definition

Data type
object [***text***], *string*

Access
get, set

changed event
yes

C
Identifier: AT_acc_text
Data type: DT_text, DT_string

COBOL
Identifier: AT-acc-text
Data type: DT-text, DT-string

Inheritance
yes

Default value
null

Classification
standard attribute

Support of attribute by objects

Object	Support of the Attribute
filereq, ***messagebox***	Attribute has **no** effect
menubox, ***menuitem***, ***menusep***	
canvas, ***spinbox***, ***statusbar***, ***tablefield***	Attribute is supported
groupbox, ***notebook***, ***notepage***, ***splitbox***	
image, ***layoutbox***, ***window***	
rectangle, ***scrollbar***	
checkbox, ***pushbutton***, ***radiobutton***	
edittext, ***poptext***, ***statictext***	
control, ***listbox***, ***treeview***	
other object classes	Attribute is **not** supported

Remark

This attribute is only relevant for automated external control with active MICROSOFT UIA support. The attribute is without function on QT and MOTIF.

Availability

Since IDM version A.06.01.e

See also

Chapter “UIA Object Identification” in manual “Automatic Testing and Accessibility”

2.3 .accelerator

This attribute defines a keyboard equivalent by means of which the object can be selected with the keyboard. The accelerator has first to be defined as a resource.

Definition

Data type	*Access*	*changed event*
object [***accelerator***]	get, set get (***thisevent***)	yes

C	*COBOL*
Identifier: AT_accelerator Data type: DT_accel	Identifier: AT-accelerator Data type: DT-accel

Classification
standard attribute

In ***notepage*** this attribute can be used to activate a certain notepage, i.e. to bring it on top. In doing so, a select-activate event will be triggered.

In ***thisevent***, this attribute can be used to request the accelerator identifier (valid for *key* event).

2.4 .action

This attribute determines the possible Drag&Drop actions of the resources ***source*** and ***target***.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_action	Identifier: AT-action
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	source, target

Value range

action_cut
: “Cut”, deletes the item at its current location.

action_copy
: “Copy”, keeps the item at its current place.

action_paste
: “Paste” of the cut or copied item (***target*** only).

2.5 .active

Defines whether the object is active, i.e. ready to accept input.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_active
Data type: DT_boolean

COBOL
Identifier: AT-active
Data type: DT-boolean

Classification
standard attribute

For the object ***application***, .active can be used to define and to query whether the application is currently running. Changing this attribute from *false* to *true* starts the application. Changing the attribute from *true* to *false* terminates the application.

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

In ***notepage***, *.active = true* refers to the page which is actually on top. For all other notepages *.active* is *false*, meaning that this attribute can only have the value *true* for one single notepage.

.active can be used to define and request whether the ***timer*** is currently active. Changing this attribute from false to true starts the timer. Changing the attribute from true to false resets the timer and makes it inactive.

To activate the input mode through the application, the attribute .active in the object ***tablefield*** is available both indexed (see below .active [I,J]) and non-indexed (.active). The behavior for the non-indexed attribute .active is as follows:

Action	Current State	Reaction
active := true	Focus on sensitive field	Input will be activated, if it is not yet active. If the input mode is already active, nothing happens.
active := false	Focus on sensitive field	Input mode will be deacti-vated, if it is active. Other-wise nothing happens.
active := true / false	Focus on non-sensitive field	No reaction from the object. The Dialog Manager sets .active back at false.

Action	Current State	Reaction
active := true / false	Focus is not on a field in the table-field, but on a corner or on one of the scrollbars.	No reaction from the object. The Dialog Manager sets .active back at false.
active := true	Tablefield does not have the focus.	The Dialog Manager sets .active back at false. Other-wise no reac-tion.
User activates input		.active is set at true.

See Also

Attribute .activeitem

2.6 .active[integer]

With this attribute each single entry in a ***listbox*** with *.multisel* = *true* can be set at active or non-active.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_active	Identifier: AT-active
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	listbox

See also

Attribute *.activeitem*

2.7 .active[index]

With this attribute, every text field in the ***tablefield*** ([row,column]) including titles and row headers can be set at active and not active.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_active Data type: DT_boolean	Identifier: AT-active Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

See also

Attributes *.activeitem*, *.nextactive[index]*

2.8 .activeitem

The index of the currently displayed text can be requested and set with the attribute *.activeitem*.

Definition

Data type	*Access*	*changed event*
integer (***listbox***, ***poptext***, ***spinbox***) *index* (***tablefield***)	get, set	yes

C	*COBOL*
Identifier: AT_activeitem Data type: DT_integer, DT_index	Identifier: AT-activeitem Data type: DT-integer, DT-index

Classification	*Objects*
object-specific attribute	listbox, poptext, spinbox, tablefield

With list objects and poptexts an error occurs, when it is tried to set *.activeitem* to a non-existent text entry. Therefore within static object declarations the definition of the text entry must be placed before the assignment to *.activeitem* in dialog scripts.

Meaning of .active and .activeitem for Listbox and Tablefield

.activeitem accesses the selected item in objects with single selection (multisel = false).

1. Setting of values: `Object.attrib := value`

	SingleSelection	MultiSelection
boolean active[I]	not allowed	item i
integer activeitem	number of active item	not allowed

2. Request of values: `? := Object.attrib`

	SingleSelection	MultiSelection
boolean active[I]	selection state of	selection state of
	item i	item i
integer activeitem	number of active item	not allowed

Note

You have to set *.activeitem = 0*, if no item shall be selected in a ***listbox*** or ***tablefield***.

See Also

Attributes .active, .multisel

2.9 .activeobject

The active, i.e. the current *notepage* on top.

Possible values are the notepage labels. The default value *0* is valid only so long as the notebook has no notepage. As soon as there is a notepage, this attribute refers to the active one.

Definition

Data type	*Access*	*changed event*
object	get	yes

C	*COBOL*
Identifier: AT_activeobject	Identifier: AT-activeobject
Data type: DT_instance	Data type: DT-instance

Classification	*Objects*
object-specific attribute	notebook

See also

Object notebook

2.10 .alignment

With this attribute you can define the alignment of a text.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_alignment Data type: DT_integer	Identifier: AT-alignment Data type: DT-integer

Classification	*Objects*
object-specific attribute	edittext, image, notebook, statictext

Value range

-1
: right-justified

0
: centered

1
: left-justified

image

The attribute determines how the text of an *image* object is horizontally aligned in the space available for it (default value *0*).

The effect of *.alignment* in dependency of the attributes *.spacing* and *.tilestyle* is explained at the *.tilestyle* attribute.

notebook

At a ***notebook*** this attribute defines the alignment of the status text that every ***notepage*** may have (default value *1*).

edittext

The attribute is only supported on MICROSOFT WINDOWS and QT. Please note the following:

» Lines whose length exceeds the edittext width are automatically wrapped when *.hsb_visible* = *false*.

WINDOWS-specific:

» If both attributes *.multiline* and *.hsb_visible* have the value *true*, *.alignment* is not supported.

» For RTF edittexts (*.options[opt_rtf] = true*), the RTF formatting and not the *.alignment* attribute determines the text alignment.

QT-specific:

» For HTML edittexts (*.options[opt_rtf] = true*), the HTML formatting and not the *.alignment* attribute determines the text alignment.

statictext

The text is top-aligned to be compatible with an insensitive ***statictext***.

2.11 .allowundefined

This attribute determines whether the user can set the value of the object to “indefinite”.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_allowundefined Data type: DT_boolean	Identifier: AT-allowundefined Data type: DT-boolean

Default value	*Inheritance*
false	yes

Classification	*Objects*
object-specific attribute	datetime

If the attribute has the value *true*, a checkbox is displayed in the ***datetime***. Deactivating the checkbox sets the value to “indefinite”. The attribute *.value* then has the value *""*.

With *.allowundefined* = *true*, invalid values of *.value* cause the checkbox to be activated.

Value range

true
: The content of *.value* can be set to “indefinite” by the user. To do so, a checkbox is displayed in the ***datetime***.

false
: The user can **not** set *.value* to “indefinite” (*""*).

Note

Changing *.allowundefined* in the visible state may lead to a reset of the object.

2.12 .application

This attribute of the object ***import*** defines the application to which the functions shall be linked.

Definition

Data type	*Access*	*changed event*
object [application]	get, set	yes

C	*COBOL*
Identifier: AT_application	Identifier: AT-application
Data type: DT_application	Data type: DT-application

Classification	*Objects*
object-specific attribute	import

See also

Attribute *.masterapplication[enum]*

Chapter “Modularization” in manual “Programming Techniques”

2.13 .arrows

.arrows decides whether the ***scrollbar*** shall have scroll arrows or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_arrows Data type: DT_boolean	Identifier: AT-arrows Data type: DT-boolean

Classification	*Objects*
object-specific attribute	scrollbar

Note

The attribute *.arrows* defines whether the two scroll arrows exist (true) or not (false).

With *.arrows = false*

- scale widget is used instead of the ***scrollbar*** with Motif
- *.curvalue* is displayed (digital display),
- *.linemotion* is ignored by Motif

Note for IDM on Microsoft Windows

.arrows is ignored! The scrollbars in principle already have arrows.

See Also

Attributes .curvalue, .linemotion, .pagemotion

2.14 .attribute

This attribute can be used for the event object ***thisevent*** to query a changed attribute (valid only for *changed* event).

Definition

Data type
attribute

Access
get

C
Identifier: AT_attribute
Data type: DT_attribute

COBOL
Identifier: AT-attribute
Data type: DT-attribute

Classification
object-specific attribute

Objects
thisevent

See also

Chapter “Event Object thisevent” in manual “Rule Language”

2.15 .attribute[integer]

The attribute serves, according to the indexing, for calling up the name or the value of an attribute within the DOM nodes.

If the index is a number, then the name of the corresponding attribute of the DOM node will be delivered. Please note, attributes of a DOM node are primarily unsorted. If the index is a string, then the attribute will display the name of the attribute and it will receive the value of the attribute in return.

Definition

Data type	*Access*	*changed event*
integer	get	no
string, *object* [***text***]	set (only with string index)	

C	*COBOL*
Identifier: AT_attribute	Identifier: AT-attribute
Data type: DT_integer, DT_string	Data type: DT-integer, DT-string

Inheritance
no

Classification	*Objects*
object-specific attribute	doccursor

An assignment to the attribute, which is indexed with a string, will result in the allocation of a corresponding attribute to the DOM nodes. The assignment of an empty string deletes the corresponding attribute of the DOM nodes. It is not allowed to assign an attribute that is indexed with a number.

The attribute is available for the XML Cursor, but it cannot be passed on. Please note, an XML Cursors, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM document.

Example

The attribute “OrderNumber” of the XML element, upon which the XML Cursor points, will be set with the following instruction to the value *0815*:

```
this.attribute["OrderNumber"] := "0815";
```

All attributes of the XML elements, upon which the XML Cursor points, can be given as follows:

```
variable integer I;
variable string  Name;

for I := 1 to this.count[.attribute] do
  Name := this.attribute[I];
  print Name + " = " + this.attribute[Name];
endfor
```

2.16 .autoalign

This attribute of the ***toolbar*** controls whether the ***toolbar*** is arranged automatically in its row or column.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_autoalign	Identifier: AT-autoalign
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	toolbar

Value range

true
: The ***toolbar*** is automatically placed after the last ***toolbar*** in a row or column respectively at the beginning of empty rows and columns.

false
: The ***toolbar*** is not automatically positioned.

2.17 .autosize

This attribute of the ***toolbar*** defines whether the ***toolbar*** should automatically be enlarged in the respective direction to cover the entire toolbar row or column.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_autosize	Identifier: AT-autosize
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	toolbar

Beginning with the last ***toolbar*** in a row or column, ***toolbars*** with *.autosize* = *true* – regardless of *.sizeable* – will be enlarged if empty space is available,

Expansion is limited to the docking direction: width is increased for horizontal direction and height for vertical direction.

Value range

true
: The ***toolbar*** is automatically enlarged and covers the available space in the row or column.

false
: The ***toolbar*** is **not** automatically enlarged.

2.18 .backpage

.backpage defines the corner in which the page borders visible on the side shall meet.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_binding Data type: DT_enum	Identifier: AT-bindinge Data type: DT-enum

Default value
bp_bottomright

Classification	*Objects*
object-specific attribute	notebook

Value range

bp_bottomright (default)
Page borders are visible at the bottom and on the right.
bp_bottomleft
Page borders are visible at the bottom and on the left.
bp_topright
Page borders are visible on the top and on the right.
bp_topleft
Page borders are visible on the top and on the left.

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

2.19 .barwidth

This attribute defines the width of the bars for interactive resizing.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_barwidth	Identifier: AT-barwidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	splitbox, window

splitbox

The width of the splitbars can be indicated in pixel with this attribute.

Microsoft Windows

The splitbars have a 3-D look to them. Only when the width is set to one will this effect disappear. The 1-pixel thin line will then assume the color of *.bordercolor*, if this is set.

Motif

Here, the splitbars have a grip (sash) for the user to pull on. The size of this grip is controlled by the attribute *.barwidth*. The width of the separating line is predetermined by the system and remains the same even if this attribute is changed.

window

This attribute defines the width of the bars for interactive resizing of toolbars.

See also

Object toolbar

2.20 .bgc

The attribute *.bgc* defines the object background color.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C
Identifier: AT_bgc
Data type: DT_color

COBOL
Identifier: AT-bgc
Data type: DT-color

Classification
layout attribute

Notes on the IDM FOR WINDOWS

- With ***poptext*** *.bgc* is supported as follows:
  - The closed poptext, i.e. the displayed item has foreground color and background color.
  - The popped-up box uses a default color.
- ***menubox***, ***menuitem***, ***menusep***, and ***messagebox*** ignore *.bgc*.
- The ***pushbutton*** always uses the system-wide default color for its background.
- Depiction of the **scrollbars** in IDM objects (e.g. ***edittext***, ***groupbox***, ***listbox***, ***treeview***…):
  In Microsoft Windows scrollbars are no separate objects but specific border styles. Therefore the ISA Dialog Manager generally has no influence on the color of the scollbar elements (rectangular area, arrows, thumb). As an example this may lead to the scrollbar elements being displayed in a system color (grey in most cases) although a different background color (*.bgc*) has been defined for the object.
- With “Visual Styles” turned on, the Windows objects ***checkbox*** and ***radiobutton*** use a wrong background color. The IDM tries to correct this. If the correction can be successful is affected by many “Visual Styles” properties.
  To be on the safe side, a background color should be set for the father object, especially if it is a ***notepage***. The drawback is that with the ***notepage*** an aditional border will be visible then.
- ***scrollbar*** objects with *.arrows* = *false* use the background color of their parent object and not their own.

2.21 .bgc[index]

This attribute defines the background color of one single field in a ***tablefield***.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_bgc	Identifier: AT-bgc
Data type: DT_color	Data type: DT-color

Classification	*Objects*
object-specific attribute	tablefield

2.22 .binding

Defines the way a ***notebook*** shall be bound.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_binding Data type: DT_enum	Identifier: AT-binding Data type: DT-enum

Default value
bind_solid

Classification	*Objects*
object-specific attribute	notebook

Value range

bind_solid (default)
The bound edge looks like with a perfect binding.

bind_spiral
The bound edge looks like with a spiral binding (IDM FOR MOTIF only).

bind_organizer
The bound edge looks like that of an organizer (IDM FOR WINDOWS only).

bind_none
The bound edge shows no binding (IDM FOR MOTIF only).
The look is more similar to the appearance on MICROSOFT WINDOWS.

2.23 .bordercolor

.bordercolor defines the border color of an object.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C
Identifier: AT_bordercolor
Data type: DT_color

COBOL
Identifier: AT-bordercolor
Data type: DT-color

Classification
layout attribute

See also

Attributes *.borderstyle*, *.borderwidth*

2.24 .borderraster

This attribute allows for objects with borders to be configured so that the geometry is calculated in a way as if the object did not possess any borders at all.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_borderraster
Data type: DT_boolean

COBOL
Identifier: AT-borderraster
Data type: DT-boolean

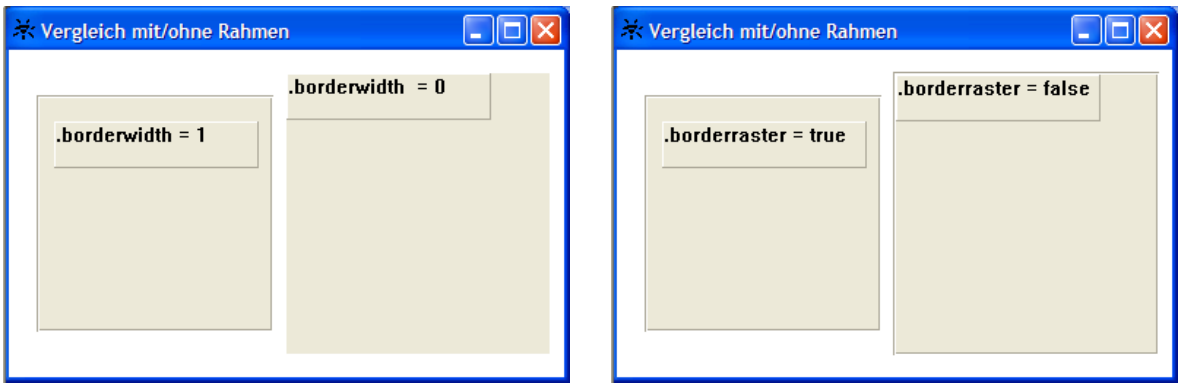
Default value
true

Classification
geometry attribute

Normally, the following adjustments are made for objects with borders:

- Shifting by one raster unit when calculating the position of an object that contains raster coordinates.
- Downscaling by one raster unit when calculating the size of an object that possesses a raster size.
- Shifting by one raster unit when calculating the position of the child object that possesses a raster, provided that the object is able to possess children.

The above mentioned adjustments can be disabled via this attribute. The effect is the same as if the borders of a groupbox object are disabled (*.borderwidth := 0*).



The difference is that the borders still appear, when *.borderraster = false*. The attribute can only make an appropriate adjustment if the object possesses a border and if raster coordinates are used. The position (*.posraster = true*), the size (*.sizeraster = true*) and where applicable also the position of the child object can be influenced only when their positions are specified in raster units.

The following table provides an overview of the attribute *.borderraster* along with the Dialog Manager objects that support it:

canvas	supported, when *.borderwidth > 0*
checkbox	**not** supported
control	**not** supported
edittext	supported, when *.multiline = true*
groupbox	supported, when *.borderwidth > 0*
image	**not** supported
layoutbox	supported, when *.borderwidth > 0*
listbox	supported
notebook	supported
notepage	supported, when *.borderwidth > 0*
poptext	supported, when *.style = listbox*
progressbar	**not** supported
pushbutton	**not** supported
radiobutton	**not** supported
rectangle	supported, when *.borderwidth > 0*
scrollbar	**not** supported
spinbox	**not** supported
splitbox	supported, when *.borderwidth > 0*
statictext	**not** supported
statusbar	**not** supported
tablefield	supported
toolbar	supported, when *.docking = dock_window* OR *.borderwidth > 0*
treeview	supported
window	supported

Value range

true

The calculation of raster coordinates takes place in the usual way (analog to earlier versions of the ISA Dialog Manager).
For objects with borders this means:

- The object will be shifted by one half of a raster unit when the position is specified in raster units (*.posraster = true*).
- The object will be reduced in size by one raster unit when the size is specified in raster units (*.sizeraster = true*).
- The child objects will be shifted by one half of a raster unit when their position is specified in raster units (*.posraster = true*).
  Note, the object must allow for and possess a child object.

false

The calculation of raster coordinates of an object possessing borders takes place as if the object did not possess borders. The adjustments mentioned at the value “true” do not take place here.

Note

Only those adjustments mentioned at the value “true” are not carried out; even when *.borderraster = false* is set, it can happen that an object with borders will not appear in the exact same line as another object without borders. This is caused by other factors. These objects would also not be positioned in the exact same line even if they would be positioned in the pixel coordinates.

References and Dependencies

The distinction between objects with and without borders is a ISA Dialog Manager classification and affects the way in which raster coordinates are converted into pixel values. This classification has nothing to do with distinguishing whether a particular window system displays a border around the object or not.

Currently, the classification is as follows:

Objects Without Borders

canvas	when *.borderwidth = 0*
checkbox	
control	
edittext	when *.multiline = false*
groupbox	when *.borderwidth = 0*

image

layoutbox when *.borderwidth = 0*

notepage when *.borderwidth = 0*

poptext when *.style <> listbox*

progressbar

pushbutton

radiobutton

rectangle when *.borderwidth = 0*

scrollbar

spinbox

splitbox when *.borderwidth = 0*

statictext

toolbar when *.docking <> dock_window* AND *.borderwidth = 0*

Objects With Borders

canvas when *.borderwidth > 0*

edittext when *.multiline = true*

groupbox when *.borderwidth > 0*

layoutbox when *.borderwidth > 0*

listbox

notebook

notepage when *.borderwidth > 0*

poptext when *.style = listbox*

rectangle when *.borderwidth > 0*

splitbox when *.borderwidth > 0*

statusbar when *.borderwidth > 0*

tablefield

toolbar when *.docking = dock_window* OR *.borderwidth > 0*

treeview

window

The *.borderraster* attribute only works on the ***toolbar*** object when it is undocked (*.docking = dock_window*). In a docked state the attribute only works on a child object of a ***toolbar*** object (given that: *.borderwidth > 0*).

If an object does not possess a border, then a change made to *.borderraster* will have no effect.

When *.borderraster* is set to *false*, then objects will not appear in the exact pixel position as before. On the contrary, the effect will be as if the objects in question would have been positioned to the exact same position by setting the pixel coordinates.

Please be aware that objects without borders can also be single-line. These types of objects are centered in the specified raster lines when raster coordinates are used (*.posraster = true*) and when they have no height (*.height = 0*).

Single-line Objects

checkbox	
edittext	when *.multiline = false*
poptext	when *.style <> listbox*
progressbar	when *.direction = 2*
pushbutton	
radiobutton	
scrollbar	when *.direction = 2*
spinbox	
statictext	

Other adjustments are carried out even when the attribute *.borderraster* has the value *false*. Especially the adjustments with respect to the compatibility to other versions such as *.options[opt_wntsizebug_compat]* and *.options[opt_w2kprefsize_compat]* will continue to be carried out.

2.25 .borderstyle

This attribute defines the style, i.e. the representation and characteristics of the borders of an object.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_borderstyle Data type: DT_enum	Identifier: AT-borderstyle Data type: DT-enum

Default value	*Inheritance*
border_compat	yes

Classification
layout attribute

Value range

border_compat
: Compatibility mode: border rendering like in IDM 5 determined by the attributes *.bodercolor* and *.borderwidth*.
This value cannot be actively set.

border_none
: No borders are drawn around the object.

border_plain
: The border is drawn as a simple, flat line around the object, can be influenced in width and color (if supported by the respective WSI).

border_raised
: The border is drawn as a 3D border around the object, the border is heightened outwards and can be influenced in its width (if supported by the respective WSI).

border_sunken
: The border is drawn as a 3D border around the object, the border is deepened inwards and can be influenced in its width (if supported by the respective WSI).

border_toolkit
: The toolkit border is drawn as designated by the WSI for the particular object.

Table 1: *Support of the .borderstyle attribute*

Object	Support of .borderstyle
canvas	Attribute is supported.
groupbox	Attribute is supported.
layoutbox	Attribute is supported.

Object	Support of .borderstyle
rectangle	Attribute is supported.
splitbox	Attribute is supported.
edittext	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
image	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
progressbar	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
spinbox	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
statusbar	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
toolbar	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
window	Attribute is supported, but only *border_none* and *border_toolkit* are permitted. *border_plain*, *border_raised* and *border_sunken* are mapped to *border_toolkit*.
checkbox	Attribute is **not** supported.
control	Attribute is **not** supported.
listbox	Attribute is **not** supported.
notebook	Attribute is **not** supported.
notepage	Attribute is **not** supported.
poptext	Attribute is **not** supported.
pushbutton	Attribute is **not** supported.
radiobutton	Attribute is **not** supported.
scrollbar	Attribute is **not** supported.
statictext	Attribute is **not** supported.
tablefield	Attribute is **not** supported.
treeview	Attribute is **not** supported.

Availability

Since IDM version A.06.01.a

2.26 .borderwidth

This attribute defines the border width of an object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_borderwidth
Data type: DT_integer

COBOL
Identifier: AT-borderwidth
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.bordercolor*, *.borderstyle*

2.27 .button[integer]

A ***messagebox*** can display a maximum of three buttons. The function allocation of each of these buttons can be made with *.button[integer]*.

Definition

Data type
enum

Access
get, set

changed event
yes

C
Identifier: AT_button
Data type: DT_enum

COBOL
Identifier: AT-button
Data type: DT-enum

Classification
object-specific attribute

Objects
messagebox

Value range

button_abort
: Displays the “Abort” button.

button_cancel
: Displays the “Cancel” button.

button_ignore
: Displays the “Ignore” button.

button_no
: Displays the “No” button.

button_ok
: Displays the “OK” button.

button_retry
: Displays the “Retry” button.

button_yes
: Displays the “Yes” button.

The following combinations are allowed for *.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[1]	.button[2]	.button[3]
button_yes	*button_no*	*nobutton*
button_yes	*button_no*	*button_cancel*

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

If an ineligible button combination is set, the combination will be treated as an error by the window system, i.e. the ***messagebox*** does not appear and the return value is *nobutton*.

Remark on the IDM FOR WINDOWS

The combinations *button_cancel – nobutton – nobutton* and *button_retry – button_abort – nobutton* are not supported.

Attention

Combinations differing from the table above and valid for Motif are not possible for Microsoft Windows.

Remark on the IDM FOR MOTIF

The `Escape` key usually activates a *"Cancel"* button.

Example

```
messagebox Messagebox1
{
  .text      "This text notifies you of an error.";
  .title     "Error Message";
  .icon      icon_exclamation;
  .button[1] button_ok;
  .button[2] button_cancel;
}
```

2.28 .bw

This attribute determines how the ***color*** resource is displayed on a black and white monitor (*setup.color_type = coltype_bw*).

Definition

Data type	*Access*	*changed event*
enum	get, set	no

C
Identifier: AT_bw
Data type: DT_enum

COBOL
Identifier: AT-bw
Data type: DT-enum

Classification
object-specific attribute

Objects
color

Value range

color_black
: Appears as black.

color_white
: Appears as white.

2.29 .calendaralignment

This attribute defines the position of the fold-out calendar relative to the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_calendaralignment Data type: DT_integer	Identifier: AT-calendaralignment Data type: DT-integer

Default value	*Inheritance*
1	yes

Classification	*Objects*
object-specific attribute	datetime

Value range

1
The calendar is **left** aligned (default).

0
The calendar is **centered**.
This value is not supported; the default value is used.

-1
The calendar is **right** aligned.

2.30 .canvasfunc

The attribute *.canvasfunc* specifies the function to be called if an event occurs. This function has to be provided by the application and has to be of the type `canvasfunc`.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C	*COBOL*
Identifier: AT_canvasfunc	Identifier: AT-canvasfunc
Data type: DT_func	Data type: DT-func

Classification	*Objects*
object-specific attribute	canvas

2.31 .certificatefile

This attribute defines the certificate file used for an SSL connection.

By default, the file **cert.pem** from the installation directory of the application is used.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_certificatefile Data type: DT_string	Identifier: AT-certificatefile Data type: DT-string

Inheritance
yes

Classification	*Objects*
object-specific attribute	application

Availability

Since IDM version A.06.02.g

2.32 .changedir

This attribute controls the adoption of the directory in which the user found himself by a successful selection in a file selection window (***filereq***) into the attribute *.directory*. If it is set to *true*, the directory path will be adopted. Thus, the next time the file selection is opened, one can continue where one left off.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_changedir	Identifier: AT-changedir
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	filereq

See also

Attribute *.directory*

2.33 .child[integer]

This attribute is used to define child objects. It puts the object at position "i" in the child list and can be queried correspondingly.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_child
Data type: DT_object

COBOL
Identifier: AT-child
Data type: DT-object

Classification
hierarchy attribute

2.34 .childcount

This attribute queries the number of children of an object.

Definition

Data type
integer

Access
get

C
Identifier: AT_childcount
Data type: DT_integer

COBOL
Identifier: AT-childcount
Data type: DT-integer

Classification
hierarchy attribute

2.35 .class

.class queries the class of an object or of a resource, e.g. *pushbutton*, *color*, *function*.

Definition

Data type
class

Access
get

C
Identifier: AT_class
Data type: DT_class

COBOL
Identifier: AT-class
Data type: DT-class

Classification
standard attribute

2.36 .closeable

.closeable defines whether a window shall be closeable, or if it shall have a close mechanism (e.g. a closebox or a closebutton).

The window contains a close mechanism if the value of this attribute is *true*. If the value is *false*, no close mechanism is added.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_closeable	Identifier: AT-closeable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

Particularity of Motif

Depending on the display or desktop manager in use, the attribute cannot be changed in the visible state; under certain conditions, it may not be possible to set it at runtime. In some cases, it may help to toggle the visibility of the window.

Since the ability of setting this attribute on MOTIF directly depends on the display or desktop manager used, it is recommended to set the attribute only statically or immediately after creating an instance with *:create(…, true)* in the invisible state.

2.37 .codepage

This attribute defines the code page used to call application functions of the ***application*** object.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_codepage Data type: DT_enum	Identifier: AT-codepage Data type: DT-enum

Inheritance
yes

Classification	*Objects*
object-specific attribute	application

By setting this attribute, the IDM application code page is redefined for the time of the call, which otherwise can only be achieved by using the IDM interface function **DM_Control** with the action *DMF_SetCodePage*.

As an alternative to using this attribute, **DM_Control** with the application object specified may also be used.

By using an application-specific code page, it is thus quite simple to achieve the division of application functions that require strings in different encodings.

Dynamic transition of this attribute, not only while the application is activated, is strongly discouraged. If application functions with ***record*** parameters are called using dynamic binding, the record definition (C header or COBOL copy file) should match.

Value range

cp_acp
: Currently used ANSI code page of an application on MICROSOFT WINDOWS.

  Availability
  Only on MICROSOFT WINDOWS.

cp_ascii
: ASCII character encoding.

cp_cp1252
: Western European character encoding according to MICROSOFT WINDOWS code page 1252.

cp_cp437
: English character encoding according IBM code page 437 (MS-DOS).

cp_cp850
: Western European character encoding according to IBM code page 850 (MS-DOS).

cp_dec169
: Character encoding according to DEC code page 169.

cp_euc
Character encoding according to “Extended UNIX Code” (EUC).
cp_hp15
Western European 16-bit character encoding used by HP systems.
cp_iso6937
Western European character encoding with variable length according to ISO 6937.
cp_iso8859
Western European Latin-1 encoding according to ISO 8859-1.
cp_jap15
Japanese 16-bit character encoding used by HP systems.
cp_kor15
Korean 16-bit character encoding used by HP systems.
cp_prc15
Traditional Chinese 16-bit character encoding used by HP systems.
cp_roc15
Simplified Chinese 16-bit character encoding used by HP systems.
cp_roman8
8-bit character encoding according to HP code page Roman-8.
cp_ucp
“User Code Page”; conversion to an arbitrary user-defined code page with **iconv()** by DM_ControlEx with the action *DMF_SetUserCodePage*.

Availability
Only on UNIX/LINUX systems. On MICROSOFT WINDOWS, non-displayable characters are converted to “?”
cp_utf16
16-bit Unicode encoding with character widths from 2 up to 4 bytes.
There are two variants:

- BE – big-endian, bytes with higher numerical significance first.
- LE – little-endian, bytes with lower numerical significance first.

UTF-16 without a specified byte order corresponds to the LE variant on MICROSOFT WINDOWS and to the BE variant on UNIX/LINUX systems.
cp_utf16b
16-bit Unicode encoding with character widths from 2 up to 4 bytes in the BE variant (big-endian, bytes with higher numerical significance first).
This is the default for UTF-16 on UNIX/LINUX systems.
cp_utf16l
16-bit Unicode encoding with character widths from 2 up to 4 bytes in the LE variant (little-endian, bytes with lower numerical significance first).
This is the default for UTF-16 on MICROSOFT WINDOWS.
cp_utf8
8-bit Unicode encoding with variable length, corresponds to ASCII encoding in the range *0 – 127*.

cp_utfwin
: 16-bit Unicode encoding like *cp_utf16l* with conversion of line breaks \r\n → \n.

cp_wcs
: “Wide Character String” (data type *wchar_t**) character encoding, depending on the system and the locale used.

cp_winansi
: MICROSOFT WINDOWS character encoding.

Remark

It should be noted that the code page is not forwarded to a DDM server application. On the DDM server side, it has already been possible to set the code page using **DM_Control**.

Availability

Since IDM version A.06.01.d

See also

Chapter “Dynamic Binding of Record Functions” in manual “C Interface - Basics”

Chapter “Dynamic Binding of Record Functions” in manual “COBOL Interface”

Command line option +writeheader

Setting the Codepage by **DM_Control** im Handbuch „C Interface - Functions“

Setting an userdefined Codepage DM_ControlEx im Handbuch „C Interface - Functions“

2.38 .colalignment[integer]

In a ***tablefield***, this attribute describes the alignment of the contents of a field relative to that field (left, centered, right). The default value is *.colalignment[0]* unless it is locally reset.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_colalignment Data type: DT_integer	Identifier: AT-colalignment Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

Value range

-1
: right-justified

0
: horizontally centered

1
: left-justified

2.39 .colcount

The attribute defines the number of columns.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_colcount	Identifier: AT-colcount
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listview, tablefield

tablefield

This attribute defines the total number of columns in a ***tablefield***. The value range is 0 - 65535.

This number does not necessarily have to be used in the actual tablefield, but is used to ensure the correctness of the horizontal scrollbar.

listview

The attribute defines the number of columns in the detail view.

Value range

0	Clears the ***listview*** object.
> 0	Number of columns in the detail view.

The value of *.colcount* can be implicitly increased by adding one new column to *.content* at a time.

Column 1 is shown in all views.

For invalid values, the default value *1* is used. The attribute value is not changed, however.

Note

Changing the attribute in the visible state may cause the object to flicker.

See also

Attribute *.rowcount*

2.40 .colfirst

In a ***tablefield***, this attribute defines the hierarchical number of the column visible next to the row headers.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_colfirst	Identifier: AT-colfirst
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.41 .colheader

In a ***tablefield***, *.colheader* defines the number of columns (of a tablefield, i.e. not of characters) which serve as column labels and therefore are not to be horizontally scrolled. Value range 0 - 255.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_colheader	Identifier: AT-colheader
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.42 .colheadfgc

This attribute defines the color of the characters in the column headers.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_colheadfgc	Identifier: AT-colheadfgc
Data type: DT_color	Data type: DT-color

Classification	*Objects*
object-specific attribute	tablefield

2.43 .colheadfont

This attribute defines the font for the characters in the column headers.

Definition

Data type	*Access*	*changed event*
object [***font***]	get, set	yes

C	*COBOL*
Identifier: AT_colheadfont	Identifier: AT-colheadfont
Data type: DT_font	Data type: DT-font

Classification	*Objects*
object-specific attribute	tablefield

2.44 .colheadshadow

In a ***tablefield***, *.colheadshadow* defines the shape of the column headers. If the attribute is *true*, the display has a shadow similar to a button.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_colheadshadow	Identifier: AT-colheadshadow
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.45 .colheadvisible

In a ***tablefield***, this attribute defines if column heads are displayed or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_colheadvisible Data type: DT_boolean	Identifier: AT-colheadvisible Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.46 .collinewidth[integer]

This attribute defines the width of the horizontal lines drawn in the ***tablefield***. The default value is *.collinewidth[0]* unless it is indicated differently.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_collinewidth	Identifier: AT-collinewidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.47 .color

In the ***setup*** object, this attribute can query the color variant.

Definition

Data type
integer

Access
get

C
Identifier: AT_color
Data type: DT_integer

COBOL
Identifier: AT-color
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.48 .color_type

With this attribute of the ***setup*** object the type of screen can be queried.

In multiscreen systems (IDM FOR MOTIF only) the attribute returns the value for the default screen.

Definition

Data type
enum

Access
get

C
Identifier: AT_color_type
Data type: DT_enum

COBOL
Identifier: AT-color-type
Data type: DT-enum

Classification
object-specific attribute

Objects
setup

Value range

coltype_bw
: The screen used is a black and white monitor.

coltype_color
: The screen used is a color monitor.

coltype_grey
: The screen used is a grayscale monitor.

2.49 .color_type[integer]

With this attribute of the ***setup*** object the type of screen I can be queried.

The indexed attribute is only available with multiscreen dialogs. The index range is *1 … setup.screen-count*.

Definition

Data type	*Access*
enum	get

C	*COBOL*
Identifier: AT_color_type	Identifier: AT-color-type
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	setup

Value range

coltype_bw
: The screen used is a black and white monitor.

coltype_color
: The screen used is a color monitor.

coltype_grey
: The screen used is a grayscale monitor.

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

See also

Resource display

2.50 .colorcount

With this attribute of the ***setup*** object the number of colors supported by the screen can be queried.

In multiscreen systems (IDM for Motif only) the attribute returns the value for the default screen.

Definition

Data type
integer

Access
get

C
Identifier: AT_colorcount
Data type: DT_integer

COBOL
Identifier: AT-colorcount
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark

On MICROSOFT WINDOWS *.colorcount* may return *-1* for screens supporting more than *256* colors.

2.51 .colorcount[integer]

With this attribute of the ***setup*** object the number of colors supported by screen I can be queried.

The indexed attribute is only available with multiscreen dialogs. The index range is *1 … setup.screen-count*.

Definition

Data type	*Access*
integer	get

C	*COBOL*
Identifier: AT_colorcount	Identifier: AT-colorcount
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

See also

Resource display

2.52 .colorname[integer]

Attribute of the ***setup*** object that contains a list with the names of the colors available in the WSI.

Definition

Data type
string

Access
get

C
Identifier: AT_colorname
Data type: DT_string

COBOL
Identifier: AT-colorname
Data type: DT-string

Classification
object-specific attribute

Objects
setup

Availability

IDM FOR QT only, since IDM version A.06.01.a.

IDM FOR MOTIF & WINDOWS, Since IDM version A.06.03.a

2.53 .colsizeable[integer]

This single-indexed attribute controls the interactive maximization of columns.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_colsizeable Data type: DT_boolean	Identifier: AT-colsizeable Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

If the mouse is moved beyond the margin of a manipulable field, the cursor will change displaying a symbol which indicates change of size. You cannot specify this cursor symbol. With columns the size can be changed at the right margin of each column.

You can start the maximization by using the left mouse button. A grey hatched line indicating the mouse position will be displayed (the width of the hatched line depends on the attribute *.collinewidth [integer]*). If the mouse is moved (by pressing the left mouse button), the hatched line will move along. The maximization stops when the mouse button is released.

This attribute is used as other single-indexed attributes for tablefield, i.e. *.colsizeable[0]* provides the default to be used if no value is specified for the column. The default of *.colsizeable[0]* is *false*.

2.54 .coltitle[integer]

This attribute is used to set the column headings in the detail view.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C	*COBOL*
Identifier: AT_coltitle Data type: DT_string, DT_text	Identifier: AT-coltitle Data type: DT-string, DT-text

Default value	*Inheritance*
""	yes

Classification	*Objects*
object-specific attribute	listview

For write access (“set”), values of the data type *string* and ***text*** resources can be specified. ***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *string*.

The value range of the index is *0 … .colcount*t, where the value with index *0* is used as default value for not set values in the range *1 … .colcount*.

If no heading is defined for a column and no default value *.coltitle[0]* is defined either, then this column is displayed without a column heading.

Note

Changing the attribute in the visible state may cause the object to flicker.

2.55 .colvisible[integer]

This single-indexed attribute controls the visibility of columns so that they can be displayed when needed.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_colvisible	Identifier: AT-colvisible
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

In some applications, information which is meant exclusively for the application and not for the user is held in the tablefield. These columns and rows are to be displayed only in specific situations.

This attribute is treated as the other ones in the ***tablefield***, i.e. in *.colvisible[0]* the default is used (the default value is *true*) if no value has been defined for the column.

2.56 .colwidth[integer]

This attribute is used to define the width of the individual columns.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_colwidth Data type: DT_integer	Identifier: AT-colwidth Data type: DT-integer

Classification	*Objects*
object-specific attribute	listview, tablefield

tablefield

In a ***tablefield***, *.colwidth* defines the width of individual columns in coordinate units (pixels, if *.sizeraster* is not set; grid units, if *.sizeraster* is set.). Unless specified differently, the default value for all columns is *.colwidth[0]*.

listview

This attribute defines the column widths in the detail view.

The value range of the index is *0 … .colcount*t, where the value with index *0* is used as default value for not set values in the range *1 … .colcount*.

Value range

0	The column width is calculated by the ***listview***. The set value *0* is retained until the user changes the column width interactively.
> 0	Width of the respective column in the detail view.

For invalid values, the default value *0* is used. However, the attribute value is not changed.

Note

Changing the attribute in the visible state may cause the object to flicker.

2.57 .configurable

With this attribute you can query if a ***record*** or a **global variable** is configurable, i.e. if it can be set in the configuration file loaded with **DM_LoadProfile()**.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_configurable	Identifier: AT-configurable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	record, ***variable***

Value range

true
: The value may be loaded from a configuration file.

false
: The attribute or global variable is **not** configurable.

For **global variables**, the configurability is marked by the preceding keyword **config**.

See also

Object record

Chapter “Configurable Variables” in manual “Rule Language”

Chapter “Configuration File” in manual “Development Environment”

C function DM_LoadProfile

2.58 .connect

For the object ***application***, this attribute defines the state and the properties of the connection to a server process. For the objects ***control*** and ***subcontrol***, it defines the state of the OLE connection.

Definition

Data type	*Access*	*changed event*
string (***application***) *boolean* (***control***, ***subcontrol***)	get, set	yes

C	*COBOL*
Identifier: AT_connect Data type: DT_string (***application***) Data type: DT_boolean (***control***, ***subcontrol***)	Identifier: AT-connect Data type: DT-string (***application***) Data type: DT-boolean (***control***, ***subcontrol***)

Classification	*Objects*
object-specific attribute	application, control, subcontrol

This attribute defines that the application shall connect to a running server process which was started on the host (defined by the host name or IP address and the port number).

To establish a connection via **SSL**, the specification of the connection with *.connect* can be prefixed by the scheme *"ssl://"* (example `.connect "ssl://myserver:0815";`).

Example

Without SSL	With SSL

`application Appl {`
`  .connect "localhost:4711";`
`}`
	`application Appl {`
	`  .connect "ssl://localhost:4711";`
	`}`

Remarks

- The attributes *.transport*, *.connect*, and *.exec* can only be changed if *.active* is set at *false*.
- The attributes *.connect* and *.exec* depend on the transport mechanism used, i.e. future versions of the transport layer may have different types of connection establishment.
- The scheme *"ssl://"* can also be specified at the *.transport* attribute. If a scheme is given at both attributes, these must be identical. A once specified scheme *"ssl://"* cannot be turned off again.
- As of IDM version A.05.02.i, the DISTRIBUTED DIALOG MANAGER (DDM) supports the IPv6 protocol on all architectures that **natively** support IPv6.

See Also

Manual “Distributed Dialog Manager (DDM)”

2.59 .constant

If this attribute is set to *true* a variable can become a “read-only” variable and as a result becomes a constant variable. After that it is impossible to release the write-protection. However if the variable contains an object reference then the variable value will become *null* when the object is destroyed (e.g. by the built-in function **destroy()**).

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_constant	Identifier: AT-constant
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	***variable***

This attribute is only available for the variable object (see also manual “Rule Language”).

More elegant than using this attribute is the use of the keyword constant instead of variable to directly define a constant variable in the static definition section of a dialog or module.

Example

```
dialog D
variable integer V := 123;
constant integer C := 456;
on dialog start
{
  C := V;    // evaluation error because C cannot be changed
  V := 234;  // OK
  V.constant := true;
  V := 345;  // error - variable is now write-protected
}
```

2.60 .content

Contains a string entered by the user in an ***edittext***.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C
Identifier: AT_content
Data type: DT_string, DT_text

COBOL
Identifier: AT-content
Data type: DT-string, DT-text

Classification
text attribute

2.61 .content[integer]

.content[integer] can be used in the rule base to fill a ***listbox***. A string of text can be put at a specific place in the ***listbox***; this place is specified by the index.

Note

This attribute is not inherited from models or defaults!

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C
Identifier: AT_content
Data type: DT_string, DT_text

COBOL
Identifier: AT-content
Data type: DT-string, DT-text

Classification
text attribute

See also

Object listbox

Attribute *.content*

2.62 .content[index]

In a ***tablefield***, *.content[I,J]* is used to change the contents of any cell indicated by [row, column].

For the ***listview***, the attribute defines the list items.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C
Identifier: AT_content
Data type: DT_string, DT_text

COBOL
Identifier: AT-content
Data type: DT-string, DT-text

Classification
text attribute

tablefield

In a ***tablefield***, *.content[I,J]* is used to change the contents of any cell indicated by [row, column]. The attribute may also be used to change or query row and column headers. Thus *.content[I,J]* also comprises the attribute .field[index].

This attribute is not passed on from the model to its instances.

Note on the IDM for Motif

The Motif tablefield ignores leading line breaks in table cells (one ore more \n at the beginning of texts). To display empty lines before a text however, a space character may be inserted into the text before the line breaks.

Example: `" \nXYZ"` instead of `"\nXYZ"`.

listview

The attribute defines the list items of the ***listview***.

For write access (“set”), values of the data type *string* and ***text*** resources can be specified. ***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *string*.

The value range of the index is *[0,1] … [.rowcount,.colcount]*, where the value with index *[0,C]* is used as default value for not set values in the range *[1,C] … [.rowcount,C]*.

Column 1 is the caption of the list item that will be shown in all views. The other columns are only displayed in the detail view.

If no content is defined for *.content[R,C]* and no default value *.content[0,C]* is set, then nothing is displayed, but the item still exists.

Note

Changing the attribute in the visible state may cause the object to flicker.

See also

Objects listview, tablefield

2.63 .contentfunc

This attribute defines the function that is used to dynamically reload the ***tablefield***.

The function is called by the IDM if new rows or columns are to be displayed and if these have not yet been loaded into the IDM.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C	*COBOL*
Identifier: AT_contentfunc	Identifier: AT-contentfunc
Data type: DT_func	Data type: DT-func

Classification	*Objects*
object-specific attribute	tablefield

See also

Chapter “Reloading Functions” in manual “C Interface - Basics”

2.64 .control

This attribute is used to query the ***control*** object which is superordinate in the hierarchy.

Definition

Data type
object

Access
get

C
Identifier: AT_control
Data type: DT_object

COBOL
Identifier: AT-control
Data type: DT-object

Classification
hierarchy attribute

2.65 .count

Generally, this attribute indicates the number of elements in vectors.

With the ***timer*** the attribute defines how often it should be repeated. For the event object ***thisevent***, it returns the number of *select* events triggered by the timer.

Definition

Data type	*Access*	*changed event*
integer	get, set get (***function***, ***rule***, ***thisevent***)	yes

C	*COBOL*
Identifier: AT_count Data type: DT_integer	Identifier: AT-count Data type: DT-integer

Classification	*Objects*
object-specific attribute	***function***, ***rule***, ***thisevent***, timer

Defines how often the ***timer*** is to be repeated. If the attribute is set at *.shadow*, the timer is repeated endlessly.

In the event object ***thisevent***, *.count* indicates the number of timer events (only valid for *select* event of timer).

With **user-defined attributes**, this attribute contains at indexed attributes the actual size of the given attribute (= *0* with non-indexed attributes and *.shadow* attributes).

For user-defined attributes the query is made by indicating the attribute to be requested as index, e.g. *.count[<user-defined attribute>]*.

Furthermore, you can ask the **length of vectors** by .count in general, for example *.count[.record]*, *.count[.child]*.

For **functions** and **rules** this attribute is used to query the number of parameters.

See also

Attributes *.shadowattr*, *.shadowindex*, *.shadowobject*, *.type*

Object timer

Chapters “Event Object thisevent”, “Functions” and “Named Rules (Subprograms)” in manual “Rule Language”

2.66 .cursor

This attribute allocates a cursor to an object.

With the *.cursor* attribute of the ***setup*** object the cursor variant (number) can be queried and set.

Definition

Data type	*Access*	*changed event*
object [***cursor***] *integer* (***setup***)	get, set	yes no (***setup***)

C
Identifier: AT_cursor
Data type: DT_cursor
Data type: DT_integer (***setup***)

COBOL
Identifier: AT-cursor
Data type: DT-cursor
Data type: DT-integer (***setup***)

Classification
standard attribute

Note

In the MOTIF version *.cursor* is not available for a ***rectangle***.

2.67 .cursorname[integer]

Attribute of the ***setup*** object that contains a list with the names of the cursors available at the WSI.

Definition

Data type
string

Access
get

C
Identifier: AT_cursorname
Data type: DT_string

COBOL
Identifier: AT-cursorname
Data type: DT-string

Classification
object-specific attribute

Objects
setup

Availability

IDM FOR QT only, since IDM version A.06.01.a.

IDM FOR MOTIF & WINDOWS, Since IDM version A.06.03.a

2.68 .curvalue

This attribute defines the current position respectively the currently displayed value.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_curvalue Data type: DT_integer	Identifier: AT-curvalue Data type: DT-integer

Classification	*Objects*
object-specific attribute	progressbar, scrollbar, spinbox

progressbar

The current position of the progress.

The *.curvalue* attribute indicates the position of the progress bar. This value is also used to calculate the percentage, which is displayed as an optional label.

scrollbar

The position of the scrollbar slider can be queried and specified with this attribute.

spinbox

The attribute defines the current value, which is displayed in the child object of the ***spinbox***.

2.69 .cut_pending

This attribute indicates that a Cut operation has not been finished yet.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C
Identifier: AT_cut_pending
Data type: DT_boolean

COBOL
Identifier: AT-cut-pending
Data type: DT-boolean

Classification
standard attribute

The value is set to *true* between executing the operation and processing the Cut events. As long as *.cut_pending* = *true*, changes at the object will be rejected. The manipulation of the relevant object during a Cut operation is so prevented.

If a blocked object is to be changed, you may set the attribute to *false*. As a consequence the attribute *.cut_pending_changed* is set to *true* in order to make the following rules recognize that attributes might be in a different state than on starting the Cut operation.

2.70 .cut_pending_changed

This attribute is set to *false* when triggering the Cut operation, indicating that the relevant object has not been manipulated during the Cut operation. This attribute will be *true*, as soon as *.cut_pending := false* is set in a rule.

Definition

Data type
boolean

Access
get

C
Identifier: AT_cut_pending_changed
Data type: DT_boolean

COBOL
Identifier: AT-cut-pending-changed
Data type: DT-boolean

Classification
standard attribute

2.71 .data

This attribute returns or sets data of a DOM node. The attribute is only available when the node type is either *nodetype_cdata_section* or *nodetype_processing_instruction*.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	no

C	*COBOL*
Identifier: AT_data Data type: DT_string, DT_text	Identifier: AT-data Data type: DT-string, DT-text

Inheritance
no

Classification	*Objects*
object-specific attribute	doccursor

The attribute is available for the XML Cursor, but it is not passed down. Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.72 .dataget[attribute]

This attribute defines the linking of a View attribute, specified as index, to a Model attribute, specified as value, for fetching the value from the Model component and representing it by the View component.

Definition

Data type	*Access*	*changed event*
attribute	get, set	no

C
Identifier: AT_dataget
Data type: DT_attribute

COBOL
Identifier: AT-dataget
Data type: DT-attribute

Inheritance
yes

Classification
Datamodel attribute

Without an index, the class-specific standard attribute is used. The attribute has to be defined at the View component.

Data is exchanged according to the synchronization rules of the involved Model and View components and is only active if there is a Data Model set in the *.datamodel* attribute.

Availability

Since IDM version A.06.01.a

See also

Chapter “Linkage Between Model and View” in manual “Programming Techniques”

2.73 .dataindex[attribute]

This attribute defines, in addition to the linkage of attributes between Model and View components, an index that further refines access to the attribute in order to enable the different kinds of relations.

Any View or Model attribute may be used as index. This attribute has to be set at the View component.

Definition

Data type	*Access*	*changed event*
attribute	get, set	no

C
Identifier: AT_dataindex
Data type: DT_attribute

COBOL
Identifier: AT-dataindex
Data type: DT-attribute

Inheritance
yes

Classification
Datamodel attribute

Availability

Since IDM version A.06.01.a

See also

Chapter “Sequence and Value Aggregation” in manual “Programming Techniques”

2.74 .datamap[attribute]

This attribute defines an additional mapping of attributes when linking attributes between View and Model components. This definition may become necessary if several Data Models shall write their data values into a single attribute of the View component in order to allow the relation kind “merging”.

Any View or Model attribute may be used as index. This attribute has to be set at the View component.

Definition

Data type	*Access*	*changed event*
attribute	get, set	no

C
Identifier: AT_datamap
Data type: DT_attribute

COBOL
Identifier: AT-datamap
Data type: DT-attribute

Inheritance
yes

Classification
Datamodel attribute

Availability

Since IDM version A.06.01.a

See also

Chapter “Sequence and Value Aggregation” in manual “Programming Techniques”

2.75 .datamodel[attribute]

With this attribute, the linkage of the object (View component) to a Data Model (Model component) is defined. The attribute has to be defined at the View component.

The index can be an attribute that specifies the View attribute for which the Data Model shall be used. Without an index, the Data Model applies to all View attributes.

An active linkage requires at least the definition of a Data Model without index and a linking of the View attribute with a Model attribute through *.dataget* or *.dataset*.

Precedence over the usual inheritance of the attribute value takes the superseding by a set value at the closest parent object (without dialog or module).

Definition

Data type	*Access*	*changed event*
object	get, set	no

C
Identifier: AT_datamodel
Data type: DT_object

COBOL
Identifier: AT-datamodel
Data type: DT-object

Inheritance
yes

Classification
Datamodel attribute

Availability

Since IDM version A.06.01.a

2.76 .dataoptions[enum]

This attribute defines the automatic synchronization between the Model and View components.

Definition

Data type
boolean

Access
get, set

changed event
no

C
Identifier: AT_dataoptions
Data type: DT_boolean

COBOL
Identifier: AT-dataoptions
Data type: DT-boolean

Inheritance
yes

Classification
Datamodel attribute

Attribute Index	Default	Component	Meaning
dopt_represent_on_map	***true***	View	Immediately before the View is made visible, the data values are fetched from the Model components and set on the View object.
dopt_represent_on_init	***false***	View	During object initialization (**:init** method), the data values are retrieved and set on the View object.
dopt_apply_on_unmap	***false***	View	Immediately before the View is made invisible, the data values are fetched from the View object and assigned to the linked Model components.
dopt_apply_on_event	***false***	View	If a user interaction triggers a dialog event which indicates a possible change of a View attribute, this is assigned to the linked Model components.
dopt_propagate_on_start	***true***	Model	When a dialog or module is started, the data from the Model objects is forwarded to the linked View components.

Attribute Index	Default	Component	Meaning
dopt_propagate_on_changed	***true***	Model	Modifications to a Model attribute are forwarded to the linked View components.
dopt_cache_data	***true***	Model	This index value is only available for the ***doccursor***. ***true*** The data selected by the ***doccursor*** is buffered for further accesses (“caching”). *false* With each access, the ***doccursor*** selects the data anew from the XML Document.

This attribute exists on all object classes that allow user-defined attributes, but possibly not with all indexes. The indexes *dopt_apply_on_unmap*, *dopt_represent_on_map* and *dopt_apply_on_event* are available on visual objects only.

Availability

Since IDM version A.06.01.a

See also

Chapter “Synchronization Between Model and View” in manual “Programming Techniques”

2.77 .dataselect[attribute]

This attribute at the same time defines the Data Model attribute specified as index and a selection pattern assigned as value for nodes of an XML Document.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_dataselect	Identifier: AT-dataselect
Data type: DT_string	Data type: DT-string

Inheritance
yes

Classification	*Objects*
Datamodel attribute	doccursor

The defined Data Model attribute may then be used for linking to a View. Data changes are forwarded through the ***doccursor*** attribute specified in the index. The selection pattern describes the nodes of the XML Document where the data is retrieved or stored.

The syntax of the selection pattern is the same as for the pattern definition for the **:select** method of the ***doccursor***.

The selection pattern of the non-indexed *.dataselect* attribute defines nodes as starting points for the indexed *.dataselect* attributes. The selection patterns of the indexed *.dataselect* attributes are then treated as relative IDM paths. They reference nodes within subtrees whose roots are selected by the non-indexed *.dataselect* attribute. In this case, the indexed *.dataselect*attributes do not access the entire XML Document, but only the subnodes of those nodes selected by the non-indexed *.dataselect* attribute.

For consistent handling of optional XML elements and attributes, the selection of a Data Model attribute returns an empty string if a value does not exist or is not contained within the preselected subtrees.

The attribute *.dataselectattr* can be used to determine whether the Data Model attribute is linked to the content or an attribute of a node. Data type and cardinality of the Data Model attribute can be controlled with the attributes *.dataselecttype* and *.dataselectcount*.

Example

In this example, the data for a list of Nobel laureates comes from an XML Document in which the information for each laureate is contained in a “prize” node.

```
<?xml version="1.0"?>
<nobelprizes>
  <category id="p">Physics</category>
```

```
  <category id="c">Chemistry</category>
  <prize year="1" category="p">
    <merit>discovery of x-rays</merit>
    <winner>Wilhelm Conrad Röntgen</winner>
  </prize>
  <prize year="11" category="c">
    <merit>discovery of radioactivity</merit>
    <winner>Marie Curie</winner>
  </prize>
  <prize year="18" category="p">
    <merit>development of the concept of quanta</merit>
    <winner>Max Planck</winner>
  </prize>
  <prize year="70" category="c">
    <merit>discovery of sugar nucleotides and their role
      in the biosynthesis of carbohydrates</merit>
    <winner>Luis Leloir</winner>
  </prize>
</nobelprizes>
```

The XML nodes then for instance can be linked through Data Model attributes as follows:

```
...
document Doc
{
  doccursor DocCur
  {
    .dataselect "..prize";
    .dataselect[.Winner]    ".winner";
    .dataselect[.Year]      ".";
    .dataselect[.Discovery] "merit";
    .dataselectattr[.Year]  "year";
    .dataselecttype[.Year]  integer;
    .dataselectcount[.Year] integer;
  }
}
...
```

Availability

Since IDM version A.06.01.b

See also

Attributes *.dataselectattr[attribute]*, *.dataselectcount[attribute]* and *.dataselecttype[attribute]*

2.78 .dataselectattr[attribute]

This attribute defines to which node attribute the Data Model attribute given as index is linked.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C
Identifier: AT_dataselectattr
Data type: DT_string

COBOL
Identifier: AT-dataselectattr
Data type: DT-string

Inheritance
yes

Classification	*Objects*
Datamodel attribute	doccursor

If *.dataselectattr* is not set, the Data Model attribute is linked to the node content. The data is retrieved and assigned using the *.text* attribute of the ***doccursor***.

If *.dataselectattr* is set to the name of a node attribute, the Data Model attribute is linked to the attribute values of that node attribute. The data is retrieved and assigned using the attribute *.attribute* of the ***doccursor***, with the name of the node attribute as index.

Availability

Since IDM version A.06.01.b

See also

Attribute *.dataselect[attribute]*

2.79 .dataselectcount[attribute]

This attribute defines the cardinality of the Data Model attribute given as index.

Definition

Data type	*Access*	*changed event*
datatype	get, set	no

C	*COBOL*
Identifier: AT_dataselectcount Data type: DT_type	Identifier: AT-dataselectcount Data type: DT-type

Default value	*Inheritance*
integer	yes

Classification	*Objects*
Datamodel attribute	doccursor

Value range

integer (default)
: The Data Model attribute is a *vector* and contains the contents or attribute values of all nodes that match the selection pattern of the *.dataselect* attribute.

void
: The Data Model attribute is a scalar and contains only the content or attribute value of the first node that matches the selection pattern of the *.dataselect* attribute.

Availability

Since IDM version A.06.01.b

See also

Attribute *.dataselecttype[attribute]*

2.80 .dataselecttype[attribute]

This attribute defines the data type to which the values of the Data Model attribute specified as index are converted.

Definition

Data type	*Access*	*changed event*
datatype	get, set	no

C
Identifier: AT_dataselecttype
Data type: DT_type

COBOL
Identifier: AT-dataselecttype
Data type: DT-type

Inheritance
yes

Classification	*Objects*
Datamodel attribute	doccursor

If *.dataselecttype* is not specified, the Data Model attribute will contain a vector of strings (data type *vector[string]*).

If the conversion fails, retrieval of the values is canceled with an error.

Example

The following dialog part defines the three Data Model attributes ".Name", ".Female" and ".Name3".

```
dialog D
...
document Doc
{
  doccursor DocCur
  {
    .dataselect[.Name] "..person";

    .dataselect[.Name3]     "..person[.birthyear=\"1978\"][3]";
    .dataselecttype[.Name3] string;

    .dataselect[.Female]      "..person";
    .dataselectattr[.Female]  "female";
    .dataselecttype[.Female]  boolean;
    .dataselectcount[.Female] integer;
  }
}
...
```

The attribute ".Name" collects the texts of all "person" nodes in a string vector, while the attribute ".Name3" only contains the name of the 3rd person born in *1978* as a scalar of type *string*.

With the ".Female" attribute, the respective node attributes of all "person" nodes are retrieved through the *DocCur.attribute["female"]*] access, converted to *boolean* and stored in a *vector*.

Availability

Since IDM version A.06.01.b

See also

Attribute *.dataselectcount[attribute]*

2.81 .dataset[attribute]

This attribute defines the linking of a View attribute, specified as index, to a Model attribute, specified as value, for fetching the value from the View component and assigning it at the Model component.

Definition

Data type	*Access*	*changed event*
attribute	get, set	no

C
Identifier: AT_dataset
Data type: DT_attribute

COBOL
Identifier: AT-dataset
Data type: DT-attribute

Inheritance
yes

Classification
Datamodel attribute

Without an index, the class-specific standard attribute is used. The attribute has to be defined at the View component.

Data is exchanged according to the synchronization rules of the involved Model and View components and is only active if there is a Data Model set in the *.datamodel* attribute.

Availability

Since IDM version A.06.01.a

See also

Chapter “Linkage Between Model and View” in manual “Programming Techniques”

2.82 .defbutton

This attribute defines a ***pushbutton*** as default button.

Definition

Data type	*Access*	*changed event*
boolean (***pushbutton***) *integer* (***messagebox***)	get, set	yes

C	*COBOL*
Identifier: AT_defbutton	Identifier: AT-defbutton
Data type: DT_boolean (***pushbutton***)	Data type: DT-boolean (***pushbutton***)
Data type: DT_integer (***messagebox***)	Data type: DT-integer (***messagebox***)

Classification	*Objects*
object-specific attribute	messagebox, pushbutton

The task of the default pushbutton is to select the corresponding object if the user presses the `Return` key and if the focus is not on a multilined edittext.

Note for Motif

The ***pushbutton*** must be in a dialogbox!

messagebox

.defbutton defines which of the three buttons is to be the default button. If the value is not valid because the button is not visible, the window system selects a default button.

If *.defbutton* is allocated to more than one visible pushbutton, the DM decides which is the default pushbutton. Therefore, only **one** pushbutton should be defined as default.

See also

Attribute *.dialogbox*

2.83 .deltavalue

This attribute defines the difference value, by which *.curvalue* of a ***spinbox*** is to be increased or decreased on each step.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_deltavalue Data type: DT_integer	Identifier: AT-deltavalue Data type: DT-integer

Classification	*Objects*
object-specific attribute	spinbox

See also

Attribute *.curvalue*

2.84 .depth

The appearance of the statictext within a statusbar is controlled with this attribute.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_depth	Identifier: AT-depth
Data type: DT_integer	Data type: DT-integer

Default value
0

Classification	*Objects*
object-specific attribute	statictext

Negative values determine the statictext to be displayed in a flattened position in contrast to the parent object. Positive values are used to display the statictext in a highlighted way. The scale of the value corresponds to the stress size. Accordingly this applies to the negative values.

This attribute will only be effective if the statictext is the child of a statusbar.

2.85 .dialog

.dialog defines the dialog belonging to an object.

Definition

Data type
object

Access
get

C
Identifier: AT_dialog
Data type: DT_object

COBOL
Identifier: AT-dialog
Data type: DT-object

Classification
standard attribute

2.86 .dialogbox

This attribute changes a window into a so-called **dialogbox**. If a dialogbox is visible, input by the keyboard or the mouse to other windows is deactivated. Input to other windows becomes possible only after the dialogbox has been closed.

If the argument is *true*, the affiliated window is a dialogbox. This means that all other windows are deactivated as soon as the dialogbox appears on the screen. Other active windows can be executed again only after the dialogbox has been closed.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_dialogbox	Identifier: AT-dialogbox
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

Note on IDM for Motif

MOTIF generates an application-modal dialogbox.

Note on IDM for Windows

Only toplevel window can be a dialogbox!

As usual for MICROSOFT WINDOWS the dialogboxes are displayed with a thick frame, if they are not sizeable (.*sizeable=false*).

See also

Attribute .*defbutton*

2.87 .direction

This attribute determines the orientation of the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_direction Data type: DT_integer	Identifier: AT-direction Data type: DT-integer

Classification	*Objects*
object-specific attribute	layoutbox, notebook, progressbar, scrollbar, spinbox, splitbox, tablefield

Value range

1
: Vertical or column-wise orientation.

2
: Horizontal or row-wise orientation

layoutbox

The *.direction* attribute controls how the child objects are arranged in the ***layoutbox***.

Value range

1
: The children of the ***layoutbox*** are arranged **column by column** in the order they appear in the child vector. If there is no more space in a column, i.e. the object would not be visible if it were placed in the same column, the next child is placed in the next column. In each column the maximum width is determined. The children of the ***layoutbox*** are then aligned according to this maximum width.

2 (default)
: The children of the ***layoutbox*** are arranged **row by row** in the order they appear in the child vector. If there is no more space in a row, i.e. the object would not be visible if it were placed in the same row, the next child is positioned in the next row. In each row the maximum height is determined. The children of the ***layoutbox*** are then aligned according to this maximum height.

notebook (Motif only)

This attribute defines the orientation of the ***notebook***, i.e. the direction of the binding.

Value range

1 (default)
The binding is **vertical**. Depending on the *.backpage* attribute, the ***notebook*** is bound on the left or right.

2
The binding is **horizontal**. Depending on the *.backpage* attribute, the ***notebook*** is bound at the top or bottom.

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

progressbar

The *.direction* attribute defines the orientation of the progress indicator.

Value range

1
The progress indicator runs vertically **from bottom to top**.

2 (default)
The progress indicator runs horizontally **from left to right**.

scrollbar

The *.direction* attribute determines the orientation of the ***scrollbar***.

Value range

1 (default)
Vertical ***scrollbar*** with *.minvalue* at the top and *.maxvalue* at the bottom.

2
: Horizontal ***scrollbar*** with *.minvalue* on the left and *.maxvalue* on the right

spinbox

With the ***spinbox***, the *.direction* attribute determines whether the arrows of the buttons point up and down or left and right.

Value range

1 (default)
: Vertical arrangement of the buttons with arrows pointing up and down.

2
: Horizontal arrangement of the buttons with arrows pointing left and right.

Note on the IDM FOR QT

The *.direction* attribute and the horizontal alignment of the arrows are not supported.

splitbox

The *.direction* attribute determines whether the ***splitbox*** is divided horizontally or vertically into split areas.

Value range

1 (default)
: The splitbars are vertical, i.e. the splitbox is divided horizontally into split areas and by moving the splitbars the width of the split areas can be changed.

2
: The splitbars are horizontal, i.e. the splitbox is divided vertically into split areas and by moving the splitbars the height of the split areas can be changed.

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*.

2.88 .directory

This attribute contains the path of the initial directory for the file or directory selection (***filereq***), when opening it via the built-in function **querybox()**.

When the attribute has the value *""* or an invalid path name, then the working directory of the application appears as the initial directory when called via **querybox()**.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_directory	Identifier: AT-directory
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	filereq

Please note that the path indication must correspond to the notation of the system. For MICROSOFT WINDOWS this is *<drive>:\<dir 1>\…\<dir n>* and for UNIX */<dir 1>/…/<dir n>*. Be aware that "\" within strings must be written "\\" in IDM files.

With a set *.changedir* attribute, and after a successful selection, this attribute contains the directory in which a file or directory was selected.

Particularities of Microsoft Windows

If the attribute *.style* is set to *fr_directory*on the ***filereq*** object, the *.directory* attribute defines the topmost path. However, the attribute *.directory* limits the selection with *.style = fr_directory* only if the attribute *.changedir* has the value *false*. Otherwise (*.changedir = true*) the directory selection is unlimited and the directory specified in the *.directory* attribute is preselected.

The attribute value *""* is interpreted as "workplace" and not as "working directory".

See also

Attribute *.changedir*

Built-in function querybox()

2.89 .display

This attribute of the ***window*** object links a ***display*** resource to the window. The display resource determines the screen on which the window shall be located.

In IDM versions without multiscreen support, or if the ***display*** resource is *null* or contains an “invalid” screen number, the window will be displayed on the default screen.

Definition

Data type	*Access*	*changed event*
*object (**display**)*	get, set	yes

C	*COBOL*
Identifier: AT_display	Identifier: AT-display
Data type: DT_display	Data type: DT-display

Classification	*Objects*
object-specific attribute	window

messageboxes and ***filereqs*** opened by the querybox() function with a window as parent are displayed on the same screen as the parent window. Without a parent window they are displayed on the default screen.

Remark Motif:

The IDM FOR MOTIF provides multi-screen support.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.90 .doccursor[integer]

The *.doccursor* attribute accesses the ***XML Cursors*** of an ***XML Document***. The attribute is indexed with the object index. The first child cursor has the index *1*.

Definition

Data type	*Access*	*changed event*
*object (**doccursor**)*	get, set	no

C	*COBOL*
Identifier: AT_doccursor	Identifier: AT-doccursor
Data type: DT_doccursor	Data type: DT-doccursor

Classification	*Objects*
object-specific attribute	document

2.91 .dock_line

This attribute of the ***toolbar*** defines the order of toolbars within the same toolbar area of a window. The toolbars are arranged in increasing order of *.dock_line*.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_dock_line	Identifier: AT-dock-line
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	toolbar

2.92 .dock_offset

This attribute of the ***toolbar*** controls the distance of the toolbar from the edge of the toolbar area. For horizontally docked toolbars the attribute defines the distance from the left edge and for vertically docked toolbars it defines the distance from the top edge.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_dock_offset Data type: DT_integer	Identifier: AT-dock-offset Data type: DT-integer

Classification	*Objects*
object-specific attribute	toolbar

The attribute can be set to negative values, which for example is required when position grid is turned on and the toolbar shall be placed without any distance to the edge of the toolbar area (*.dock_offset = -1*).

The attribute is ignored for undocked toolbars (tool windows).

2.93 .dockable[enum]

This attribute of the ***toolbar*** controls in which toolbar areas the toolbar can be docked.

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_dockable	Identifier: AT-dockable
Data type: DT_boolean	Data type: DT-boolean

Default value
true

Classification	*Objects*
object-specific attribute	toolbar

Value range

true (default)
The ***toolbar*** may take the docking position given as index.
false
The ***toolbar*** cannot take the docking position given as index.

Index Range

dock_window
Undocking the ***toolbar*** as free floating tool window.
dock_up
Docking the ***toolbar*** in the toolbar area above the window's client area.
dock_down
Docking the ***toolbar*** in the toolbar area below the window's client area.
dock_left
Docking the ***toolbar*** in the toolbar area left of the window's client area.
dock_right
Docking the ***toolbar*** in the toolbar area right of the window's client area.

The attribute has to be set to *true* for at least one of the indexes. Additionally with instances the attribute for the current docking state cannot be set to *false*. The following assignment would produce an error message in the trace file:

```
this.dockable[this.docking] := false;
```

Assignments can be done without an index:

```
this.dockable := true;
```

This sets the attribute to *true* for all indexes.

```
this.dockable := false;
```

This sets the attribute to *false* for all indexes except *this.dockable[this.docking]*.

See also

Attribute *.docking*

2.94 .docking

This attribute of the toolbar sets or returns the docking position of the toolbar.

Definition

Data type	*Access*	*changed event*
enum	get, set	no

C	*COBOL*
Identifier: AT_docking Data type: DT_enum	Identifier: AT-docking Data type: DT-enum

Default value
dock_window

Classification	*Objects*
object-specific attribute	toolbar

Value range

dock_window
: Undocking the ***toolbar*** as free floating tool window.

dock_up
: Docking the ***toolbar*** in the toolbar area above the window's client area.

dock_down
: Docking the ***toolbar*** in the toolbar area below the window's client area.

dock_left
: Docking the ***toolbar*** in the toolbar area left of the window's client area.

dock_right
: Docking the ***toolbar*** in the toolbar area right of the window's client area.

With instances it is only allowed to set values for which the attribute *.dockable[enum]* is *true*.

If the parent window of a toolbar instance is visible but not sizeable, the attribute can only be changed to values that do not influence the size attributes of the parent window. This means, that docking may be toggled between *dock_up* and *dock_down* or between *dock_left* and *dock_right* only.

For invisible windows this limitation does not apply as size changes can be handled programmatically without annoying visual effects.

See also

Attribute *.dockable[enum]*

2.95 .document[integer]

The *.document* attribute accesses an object's ***XML Documents***. The attribute is indexed with the object index. The first child document has the index *1*.

The attribute is present at all objects where the *.record* attribute is available.

Definition

Data type	*Access*	*changed event*
*object (**document**)*	get, set	no

C
Identifier: AT_document
Data type: DT_document

COBOL
Identifier: AT-document
Data type: DT-document

Classification
standard attribute

2.96 .editable

For the objects ***tablefield*** and ***edittext*** there are attributes which enable the dialog designer to influence the design and input behavior. This refers to the display of non-selectabel edittexts as well as to the definition of whether the user shall be able interact with these objects or not.

» **Usability or selectability**
Attribute.*sensitive*
Describes the quality of whether a user can select an object or not. Only if an object is selectable, the user can carry out the actions usual for the object, e.g. inputting data and scrolling.

» **Editability**
Attribute.*editable*
Describes the user's possibility to change and edit the contents of objects.

» **Focusability**
Attribute.*navigable*
Describes the possibility of including the object in the keyboard control. In this way the object receives the input focus.

Definition

Data type
boolean

Access
get, set

changed event
yes

C
Identifier: AT_editable
Data type: DT_boolean

COBOL
Identifier: AT-editable
Data type: DT-boolean

Classification
object-specific attribute

Objects
edittext, tablefield

These three attributes influence the object behavior as described below:

Attribute	Value	Effects
.sensitive	*true*	The user can select the object. Only if .sensitive is true, the attributes .navigable and .editable are effective, otherwise they are ignored.
.sensitive	*false*	The user cannot select the object. The object cannot be edited or changed. The object contents is displayed in grey as is usual for window systems.
.editable	*true*	The user can change the object contents, if he can select the object.
.editable	*false*	The user cannot change the object contents.
.navigable	*true*	The object is contained in the normal keyboard control, i.e. the user can get the object by navigating in the corresponding window.

Attribute	Value	Effects
.navigable	*false*	The object cannot be reached by keyboard control. It can, however, get the focus by the mouse.

2.97 .editable[index]

For the ***tablefield*** object, the *.editable* attribute is available indexed for each cell. This allows to control the editability of each cell in the table individually. The attribute at the cell overwrites the value in the entire table.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_editable	Identifier: AT-editable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.98 .editpos

.editpos specifies where and in what mode the ***tablefield*** can be edited.

If the attribute is set at *true*, the current active field position shall always be be edited. The width of the associated edittext is automatically adapted to the column width.

If the attribute is set at *false*, the position defined in the associated edittext is to be kept.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_editpos	Identifier: AT-editpos
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.99 .edittext

In a ***tablefield***, *.edittext* defines the identifier of the edittext associated with the tablefield. The edittext must have the same parent as the tablefield.

Definition

Data type	*Access*	*changed event*
object [***edittext***]	get, set	yes

C	*COBOL*
Identifier: AT_edittext	Identifier: AT-edittext
Data type: DT_object	Data type: DT-object

Classification	*Objects*
object-specific attribute	tablefield

2.100 .edittype

In a ***tablefield***, this attribute describes the cooperation between the associated edittext and the contents of the tablefield.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_edittype Data type: DT_enum	Identifier: AT-edittype Data type: DT-enum

Classification	*Objects*
object-specific attribute	tablefield

Value range

edit_locking
: The entry into the ***edittext*** is taken over into the tablefield only after `Return` has been pressed. Any further selection in the tablefield is not allowed while an element is being edited until the `Return` key has been pressed. The locking is effective on the first change of the contents in the edittext. By using the `Escape` key the locking is deactivated without any change to the tablefield contents.

edit_offline
: The entry into the edittext is used in the tablefield only after the edittext has been deselected. The tablefield will not be locked, however. If another field is selected in the tablefield, the edittext is deselected.

edit_online
: Any entry into the ***edittext*** is immediately displayed in the associated tablefield element.

2.101 .endsel

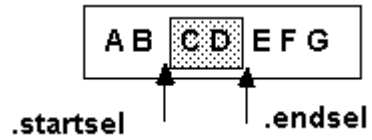
The attribute *.endsel* defines the end of the selection in the input field.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_endsel	Identifier: AT-endsel
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	edittext, poptext



When modifying *.endsel* or *.startsel*, *.focus* must be *true*, so that the new selection can be displayed. *.endsel* may be smaller than *.startsel* there.

Special Values

The following values, which are particularly useful when formats are set, have a special meaning:

-1
: Selects all decimal places after the decimal point, including zeros inserted by a format that are not contained in the content string

-2
: Selects all decimal places of the integer part up to the right end (decimal point)

-3
: Selects all decimal places of the integer part, inclusive of leading zeros inserted by a format

Selection of the leading sign is possible in no case.

Remark on the IDM for Windows

When querying *.startsel* and *.endsel* of an ***edittext***, since IDM version A.05.02.l *.endsel* may be smaller than *.startsel*. In this case the cursor is positioned left of the selection. However, if the user selects text with the mouse, *.startsel* is always smaller than *.endsel* and it cannot be recognized where the cursor is. In previous IDM versions *.startsel* was always less than or equal to *.endsel*, except it was set differently from the Rule Language.

See also

Attribute *.startsel*

2.102 .env[string]

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.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_env	Identifier: AT-env
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	setup

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.

2.103 .envvar[string]

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.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_envvar	Identifier: AT-envvar
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	setup

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.

2.104 .errfile

With this attribute of the ***setup*** object, the absolute path of the **error file** (file to which IDM error messages are redirected) can be queried at runtime.

Definition

Data type
string

Access
get

C
Identifier: AT_errfile
Data type: DT_string

COBOL
Identifier: AT-errfile
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.105 .errorcode

The attribute indicates the error type that occurred during activation/deactivation or when using the application functionality. This attribute is reset internally when activating an application via the *.active* attribute.

Definition

Data type
enum

Access
get

C
Identifier: AT_errorcode
Data type: DT_enum

COBOL
Identifier: AT-errorcode
Data type: DT-enum

Classification
object-specific attribute

Objects
application

Value range

error_file
: File error that e.g. occurs when loading a DYNLIB library.

error_network
: Error occurred when calling the network system routines. This therefore enables a clear distinction between DDM protocol errors.

error_none
: Activation or deactivation of an application took place without any errors.

error_protocol
: A non-reparable error has occurred in the DDM protocol. We advise that customers contact IDM Support in such cases. This error is not necessarily signalized by both sides, but only by the side that notices it first. As the network connection is closed in the case of a protocol error, the other side general receives report of a network error.

error_unavail
: Application functionality is not available – i.e.: NDX- or DYNLIB extension has not been integrated.

error_version
: DDM protocol version between client and server is incompatible.

The *.systemerror* attribute should also be taken into consideration. Basically *.systemerror* is only assigned for the error types *error_network* and *error_file*.

See also

Attribute *.systemerror*

2.106 .event[integer]

A single rule can be triggered by several events. All these events are contained in the indexed attribute *.event[integer]*.

The number of events can be queried by *.eventcount*.

If *.event* has an *integer* index, an event that has occurred at the same time is returned.

Definition

Data type	*Access*
anyvalue	get

C	*COBOL*
Identifier: AT_event	Identifier: AT-event
Data type: DT_event	Data type: DT-event

Classification	*Objects*
object-specific attribute	***thisevent***

.event[event]

If *.event* has an index value of the type *event* the result is of the type *boolean* and *true*, if the index event is one of the events which has occurred.

You can enter an arbitrary event as index; you can e.g. query if there is a special event: *.event[select]*.

See also

Chapter “Event Object thisevent” in manual “Rule Language”

2.107 .event_code

In the event object ***thisevent***, *.event_code* requests the code of the external event (valid only for the *extevent* event).

Definition

Data type
anyvalue (determined by the identifier of the external event)

Access
get

C
Identifier: AT_event_code
Data type: DT_anyvalue

COBOL
Identifier: AT-event-code
Data type: DT-anyvalue

Classification
object-specific attribute

Objects
thisevent

See also

Chapters “External Events” and “Event Object thisevent” in manual “Rule Language”

2.108 .eventcount

A single rule can be triggered by several events. All such events are contained in the indexed attribute *.event[integer]*.

The number of events can be queried by *.eventcount*.

Definition

Data type
integer

Access
get

C
Identifier: AT_eventcount
Data type: DT_integer

COBOL
Identifier: AT-eventcount
Data type: DT-integer

Classification
object-specific attribute

Objects
thisevent

See also

Chapter “Event Object thisevent” in manual “Rule Language”

2.109 .exec

.exec defines that the application should start a process specified by the path on the host (defined by the host name or IP address). It contains the program name, path, host name, and other information.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_exec Data type: DT_string	Identifier: AT-exec Data type: DT-string

Classification	*Objects*
object-specific attribute	application

For the TCP/IP protocol, the attribute value has the syntax

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

where either the name of the host or its IP address can be specified.

To start the application side using **SSH** instead of RSH, the command specification can be prefixed with the scheme *"ssh://"* (example `.exec "ssh://myserver:list";`).

By default, the RSH protocol is used if no security scheme is specified with the *.transport* attribute. Otherwise, the SSH protocol is used. To deviate from this default, the appropriate scheme has to be specified with the *.exec* attribute. The SSH protocol is selected by *"ssh://"* and the RSH protocol by *"rsh://"*.

The *"ssh://"* scheme also supports the OpenSSH command. First it is searched whether the `libssh` DLL is available. If it is not available, it checks whether an `ssh` command can be called. Additionally there is the scheme *"sshlib://"* to use only `libssh` and *"sshcmd://"* to use only OpenSSH. The disadvantage of the command is that no password can be used. OpenSSH must be configured to allow a connection without a password (see the man pages of `ssh`).

Example

Without SSH	With SSH

`application Appl1 {`
`  .exec"host%account%passwd:list";`
`}`
	`application Appl1 {`
	`  .exec "ssl://host%account%passwd:list";`
	`}`

Without SSH	With SSH and SSL

`application Appl2 {`
 	
`  .exec "host%account%passwd:list";`
`}`
	`application Appl2 {`
	`  .transport "ssl";`
	`  .exec "host%account%passwd:list";`
	`}`

Remarks

- The attributes *.transport*, *.connect*, and *.exec* can only be changed if *.active* is set at *false*.
- The attributes *.connect* and *.exec* depend on the transport mechanism used, i.e. future versions of the transport layer may have different types of connection establishment.
- To use a colon before the syntactically required colon (in the name or password), this needs to be doubled. In the command part of the *.exec* attribute (after the mandatory colon) colons are taken over directly.
- The additional command options required for SSL need to be specified with the command. Only the security scheme “ssl”, if used, is automatically included in the command as additional command line option `-IDMtransport ssl`.
- The command line automatically gets the option `-IDMtellport`. If instead of an IDM server application a script or something similar is specified as command, then from the output of the IDM server application at least the line beginning with `-IDMport` has to be forwarded completely (including leading and trailing line breaks “\n”).
- As of IDM version A.05.02.i, the DISTRIBUTED DIALOG MANAGER (DDM) supports the IPv6 protocol on all architectures that **natively** support IPv6.

See also

Object application

Manual “Distributed Dialog Manager (DDM)”

2.110 export

With *export*, objects and named rules of a module are made known externally so that they can be accessed in an importing module or dialog.

In contrast to the other attributes, the *export* property is placed at the beginning of the actual object definition.

Definition

```
export <object class> <Identifier>
{
  <further object definitions like attributes, children…>
}
```

Remark

export is ignored on child objects that have inherited their export property through *reexport*.

See also

Attribute *reexport*

Chapter “Modularization” in manual “Programming Techniques”

2.111 .extension

This attribute of the file dialog (***filereq***) defines a file extension that is appended to selected files without an extension.

When files selected within the **querybox()** call have no extension, a dot (“.”) and the attribute value are appended to the file name. When the attribute value is an empty string (*""*) no extension is appended.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_extension	Identifier: AT-extension
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	filereq

Special Features of Microsoft Windows

Extensions are only appended to files.

See also

Attributes .*style*, **value**

Built-in function querybox()

2.112 .external

Returns *true* when the class of the object is an “external” USW class, *false* otherwise.

The attribute is available on all object classes.

Definition

Data type
boolean

Access
get

C
Identifier: AT_external
Data type: DT_boolean

COBOL
Identifier: AT-external
Data type: DT-boolean

Classification
standard attribute

2.113 .external[integer]

Returns the registered “external” USW class at position *I*. By iterating over the indexes from *1* to *.count[.external]*, all external classes can be determined.

The attribute is available on all object classes.

Definition

Data type
class, *void*

Access
get

C
Identifier: AT_external
Data type: DT_class

COBOL
Identifier: AT-external
Data type: DT-class

Classification
standard attribute

2.114 .extevent

This attribute is used to set an application object which is to send asynchronous events to the client in the network, also on MICROSOFT WINDOWS. If set to *true*, a second communication canal will be established between server and client side. This canal is used to transport asynchronous external events via network.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_extevent	Identifier: AT-extevent
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	application

Please note that establishing a second communication canal takes up a lot of time and system resources. This is why the default of the attribute *.extevent* is set to false.

See also

Manual “Distributed Dialog Manager (DDM)”

2.115 .face

This attribute can be used to define the typeface / slant of a font.

Remark:

It should be noted that it depends on the selected character set whether and to what extent the selected modifiers are applied.

Definition

Data type	*Access*	*changed event*
enum	get, set	no

C
Identifier: AT_face
Data type: DT_enum

COBOL
Identifier: AT-face
Data type: DT-enum

Classification
layout attribute

Value range

face_default
: Regular, unchanged character representation.

face_italic
: Italic character representation.
In contrast to *face_oblique*, usually special, italic characters are used.

face_roman
: Upright, straight character representation (ignored on MICROSOFT WINDOWS).

face_oblique
: Inclined, slanted character representation.
In contrast to *face_italic* the oblique character representations are usually derived from the regular characters.
face_oblique is equivalent to *face_italic* on MICROSOFT WINDOWS.

Availability

Since IDM version A.06.03.a

2.116 .fgc

Defines the foreground color of the object.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C
Identifier: AT_fgc
Data type: DT_color

COBOL
Identifier: AT-fgc
Data type: DT-color

Classification
layout attribute

Remarks

- *.fgc* is supported in a ***poptext*** for Microsoft Windows as follows:
  - the closed poptext, i.e. the displayed item has foreground color and background color
  - the poptext which is popped up, i.e. the popped up box has a default color.
- *.fgc* is ignored at a ***menubox*** in the version for Microsoft Windows.
- *.fgc* is ignored at a ***menuitem***, ***menusep***, ***messagebox*** and ***scrollbar*** in the version for Microsoft Windows.
- At a ***pushbutton*** in the Microsoft Windows version, the system default color is always used for *.fgc*.

2.117 .fgc[index]

Defines foreground color of a single field in a ***tablefield***.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_fgc Data type: DT_color	Identifier: AT-fgc Data type: DT-color

Classification	*Objects*
object-specific attribute	tablefield

Remark

The attribute *.fgc[index]* is ignored in the header if the attributes *.colheadfgc* and/or *.rowheadfgc* are set.

2.118 .field[index]

With this attribute the contents of every single cell [row, column] in the interior of a ***tablefield*** can be queried and set. The attribute *.field[I,J]* does not include column and row headers.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C	*COBOL*
Identifier: AT_field Data type: DT_string, DT_text	Identifier: AT-field Data type: DT-string, DT-text

Classification	*Objects*
object-specific attribute	tablefield

Note on the IDM for Motif

The Motif tablefield ignores leading line breaks in table cells (one ore more \n at the beginning of texts). To display empty lines before a text however, a space character may be inserted into the text before the line breaks.

Example

" \nXYZ" instead of "\nXYZ".

See also

Attribute *.content[index]*

2.119 .fieldactive[index]

With this attribute any text field in a ***tablefield*** indicated by [row, column] can be set at active or non-active.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_fieldactive
Data type: DT_boolean

COBOL
Identifier: AT-fieldactive
Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.120 .fieldfocus

This attribute defines or queries the field in a ***tablefield*** (without header) that currently has the focus, indicated by *[row,column]*. (In contrast to *.focus [I,J]* where also the headers are considered.)

Definition

Data type	*Access*	*changed event*
index	get, set	yes

C	*COBOL*
Identifier: AT_fieldfocus	Identifier: AT-fieldfocus
Data type: DT_index	Data type: DT-index

Classification	*Objects*
object-specific attribute	tablefield

Example

```
Var := Tf.fieldfocus;
// Var => index
```

2.121 .fieldfocus[index]

This attribute defines or queries the field in a ***tablefield*** (without header) that currently has the focus, indicated by *[row,column]*. In contrast to *.focus[index]* where also the headers are considered.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_fieldfocus Data type: DT_boolean	Identifier: AT-fieldfocus Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

Example

```
Var := Tf.fieldfocus[2,2];
// Var => boolean
```

2.122 .fieldfocusable

.fieldfocusable defines whether the cursor control in a ***tablefield*** shall also consider non-active elements of the tablefield. This attribute (as the attribute .fieldfocus) can be used only if the attribute *.sensitive* of the field is set at *false*.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_fieldfocusable	Identifier: AT-fieldfocusable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.123 .fieldshadow

With this attribute, you can define whether sensitive fields of a ***tablefield*** shall have a shadow or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_fieldshadow Data type: DT_boolean	Identifier: AT-fieldshadow Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

Remark

Please note that this attribute influences the internal size of a tablefield!

2.124 .filled

This attribute defines whether a *rectangle* is filled with the object background color.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_filled Data type: DT_boolean	Identifier: AT-filled Data type: DT-boolean

Classification	*Objects*
object-specific attribute	rectangle

Remark for the IDM for Windows

Rectangles with the attribute *.filled = false* are not supported. They are filled with the parent's background color and are only selectable on the border.

2.125 .firstchar

The *.firstchar* attribute determines the first displayed character and thus the horizontal scroll position.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_firstchar Data type: DT_integer	Identifier: AT-firstchar Data type: DT-integer

Default value	*Inheritance*
1	yes

Classification	*Objects*
object-specific attribute	listbox, treeview

The *.firstchar* attribute returns or sets the number of the first displayed character. This means that the attribute can be used to query and define the horizontal scroll position of the ***listbox*** or ***treeview***.

The value range includes integer values > *0*.

With the ***treeview***, *.firstchar* refers to entries of the first hierarchy level.

Note

The character actually displayed first depends on several factors, among others whether a proportional or mono-spaced font is used, whether and in what size images are displayed at the entries (*.picture[integer]* attribute) and at the ***treeview*** additionally on the ***.style[enum]*** attribute.

2.126 .firstchild

.firstchild accesses the first child of an object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_firstchild
Data type: DT_object

COBOL
Identifier: AT-firstchild
Data type: DT-object

Classification
hierarchy attribute

2.127 .firstmenu

.firstmenu accesses the first child in the menu hierarchy.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_firstmenu
Data type: DT_object

COBOL
Identifier: AT-firstmenu
Data type: DT-object

Classification
hierarchy attribute

2.128 .firstrecord

This attribute returns or sets the first ***record*** of an object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_firstrecord
Data type: DT_record

COBOL
Identifier: AT-firstrecord
Data type: DT-record

Classification
hierarchy attribute

2.129 .firstsubcontrol

This attribute returns or sets the first ***subcontrol*** of a ***control*** or ***subcontrol*** object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_firstsubcontrol
Data type: DT_object

COBOL
Identifier: AT-firstsubcontrol
Data type: DT-object

Classification
hierarchy attribute

2.130 .firsttoolbar

This attribute of the ***window*** object accesses the first toolbar of the window.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_firsttoolbar
Data type: DT_object

COBOL
Identifier: AT-firsttoolbar
Data type: DT-object

Classification
hierarchy attribute

2.131 .focus

In a dialog, normally there is exactly one object which can have the keyboard focus. It is displayed in the usual manner of the underlying window system.

.focus is an attribute with a few special characteristics:

- It cannot be inherited from a model or a default.
- It cannot be indicated when defining the object.
- The data type of the attribute depends on the type of the object and the type of the access
- SetVal: only setting of the focus is possible, the reset is done automatically by the IDM.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C
Identifier: AT_focus
Data type: DT_boolean

COBOL
Identifier: AT-focus
Data type: DT-boolean

Inheritance
no

Classification
standard attribute

SetValue

The object has to be *real_visible* and *real_sensitive*.

If the relevant window is the active window of the application, the focus is set and displayed.

If the relevant window is not the active window of the application, setting the focus is done only internally and stored as “savedfocus”.

If a window is deactivated or reactivated, the focus is set again on the object which last had the focus, i.e. the focus is always stored with reference to the window as “savedfocus”. However, this is only valid as long as the window exists in the window system.

The focus can only be set, resetting is done automatically by the DM.

The data type for normal objects is *boolean*. Only *true* is accepted, *false* is rejected as an error.

The following expressions are therefore permitted in the rule syntax:

```
Lb.focus := true;
Pb.focus := true;
Wn.focus := true;
```

Object	Behavior for .focus
Canvas	Focus is set at the relevant canvas and the associated canvas function CCR_focus is called. If the canvas is not in the visible area of the parent, the necessary scroll operations are carried out. **See Also** Chapter “Specific Attributes of Canvas Object” in the “Object Reference”
Groupbox	The focus is set at first object (in groupbox) that is real_sensitive and real_visible. The search begins with the first child of the window, goes on to the first child's children and finally to the first child's siblings. If an object is found which can have the focus, the focus is set at this object. If no object is found, the focus is not taken from the actual focus-object, i.e. the focus remains unchanged.
Checkbox	The focus is set at the indicated object and displayed in the object specific manner. If the object is currently not visible in its parents' area, the scroll operations necessary for making the object visible are carried out.
Edittext	
Listbox	
Poptext	
Pushbutton	
Radiobutton	
Rectangle*	
Scrollbar	
Statictext	
Menubox	not valid
Menuitem	not valid
Tablefield	Focus is set at the corresponding tablefield. To set the focus to a specific field, a double index which determines the field coordinates has to be indicated for the attribute focus: *.focus[I,J]*
Window	The focus is set at the first object (in the window) that is real_sensitive and real_visible. The search begins with the first child of the window, goes on to the first child's children and finally to the first child's brothers. If an object is found which can have the focus, the focus is set at this object. If no object is found, the focus is set at the window.

*** Note**

The DM object rectangle cannot obtain the focus under Motif, thus this object cannot get the focus. On querying the focus of a rectangle, false is returned.

GetValue

.focus can be queried for any object. However, the type of return value depends on the object class.

Object	Data Type	Return Value
Groupbox	DM_ID	child (of given object) which has the focus
Window		
Canvas	boolean	*true*, if the focus is on the queried object *false*, if the queried object does not have the focus.
Checkbox		
Edittext		
Listbox		
Poptext		
Pushbutton		
Radiobutton*		
Rectangle		
Scrollbar		
Statictext		
Tablefield	index	Index (see above) if the tablefield has the focus. In this case, the field specified by the index is accessed.

*** Note for the IDM for Windows**

A ***radiobutton*** is activated when it obtains the focus!

See Also

Attribute .focusitem

2.132 .focus_on_click

This attribute defines if a mouse click into an object's client area activates the object, i.e. sets the focus on it.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_focus_on_click	Identifier: AT-focus-on-click
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	groupbox, image, rectangle, statictext, toolbar

Value range

true
: A mouse click activates the object and focuses the object or one of its children.

false
: A mouse click does not activate the object and the focus remains unchanged.

The main purpose of the attribute is to define ***toolbars*** that do not pull the focus out of the related window when the toolbar is clicked with the mouse. 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.

2.133 .focusable

This attribute controls whether an object may obtain the focus or not.

Attention

Deprecated, is no longer supported. Access (get or set) causes a `fail`.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_focusable
Data type: DT_boolean

COBOL
Identifier: AT-focusable
Data type: DT-boolean

Classification
object-specific attribute

2.134 .focusitem

In addition to *.focus*, the special element *.focusitem* exists for the object ***listbox***. *.focusitem* sets the focus on a particular entry (e.g. the fourth entry). The availability depends especially on the underlying window system.

Attention

Deprecated, is no longer supported. Access (get or set) causes a `fail`.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_focusitem Data type: DT_integer	Identifier: AT-focusitem Data type: DT-integer

Classification	*Objects*
object-specific attribute	listbox

SetValue

Object	Behavior for .focusitem
listbox	Focus is set at the specified listbox entry, if the listbox has the focus. If a non-existing index is given, the focus is set at the last listbox entry.

GetValue

Object	Data Type	Return value
listbox	*integer*	If object has the focus, the actual focusitem is returned.; otherwise 0 is returned.

See also

Attribute *.focus*

2.135 .font

This attribute defines the font related to an object.

With the *.font* attribute of the ***setup*** object the font variant (number) can be queried and set.

Definition

Data type	*Access*	*changed event*
object [***font***]	get, set	yes
integer (***setup***)		no (***setup***)

C
Identifier: AT_font
Data type: DT_font
Data type: DT_integer (***setup***)

COBOL
Identifier: AT-font
Data type: DT-font
Data type: DT-integer (***setup***)

Classification
standard attribute

Note

.font at ***menubox***, ***menuitem*** and ***messagebox*** is ignored in the version for Microsoft Windows.

2.136 .font[index]

Defines the font of a single field in a tablefield.

Definition

Data type	*Access*	*changed event*
object [***font***]	get, set	yes

C	*COBOL*
Identifier: AT_font Data type: DT_font	Identifier: AT-font Data type: DT-font

Classification	*Objects*
object-specific attribute	tablefield

Remark

The attribute *.font[index]* is ignored in the header if the attributes *.colheadfont* and/or *.rowheadfont* are set.

2.137 .fontname[integer]

Attribute of the ***setup*** object that contains a list with the names of the fonts available in the WSI.

Definition

Data type
string

Access
get

C
Identifier: AT_fontname
Data type: DT_string

COBOL
Identifier: AT-fontname
Data type: DT-string

Classification
object-specific attribute

Objects
setup

Availability

IDM FOR QT only, since IDM version A.06.01.a.

IDM FOR MOTIF & WINDOWS, Since IDM version A.06.03.a

2.138 .format

This attribute allocates a format to an object.

With the ***datetime*** object this attribute defines the display format.

With the *.format* attribute of the ***setup*** object the format variant (number) can be queried.

Definition

Data type	*Access*	*changed event*
string, *object* [***format***]	get, set	yes
string, *object* [***text***] (***datetime***)	get, set (***setup***)	
integer (***setup***)		

C
Identifier: AT_format
Data type: DT_string, DT_format
Data type: DT_string , DT_text (***datetime***)
Data type: DT_integer (***setup***)

COBOL
Identifier: AT-format
Data type: DT-string, DT-format
Data type: DT-string, Dt-text (***datetime***)
Data type: DT-integer (***setup***)

Classification
text attribute

As formats either string or identifier is permitted:

Bei String ist es die Formatbeschreibung. Details hierzu finden Sie in der „Ressourcenreferenz“ bei der Layoutressource format.

string	format description
identifier	must be a format resource

For details please refer to chapter “Format” in the “Resource Reference”.

A format resource may only be specified if the attribute *.formatfunc* has not already been set (see also chapter “Format Functions” in manual “C Interface - Basics”).

datetime

This attribute defines the display format of the ***datetime*** object.

For write access (“set”), values of the data type *string* and ***text*** resources can be specified. ***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *string*.

Value range

""	short date format (like *"widget:d"*)
widget:d	system format: short date format

widget:dd	system format: short date format with 4-digit year
widget:ddd	system format: long date format
widget:t	system format: time format
<format_string>	format string for defining the display format

System formats

The **system formats** depend on the language and region settings of the operating system. They completely determine the display format and cannot be combined with each other or with format strings. For example, no caption can be added to a system format.

Format strings

Format strings consist of formatting characters that define which parts of a date are displayed and how those parts are displayed.

Table 2: *Formatting characters of the datetime object*

Formatting Character	Display
d	day with as many digits as necessary
dd	day with 2 digits, possibly with leading *"0"*
ddd	name of the weekday (language-dependent), abbreviated to 3 characters
dddd	full name of the weekday (language-dependent)
h	hour in 12-hour format with as many digits as necessary
hh	hour in 12-hour format with 2 digits, possibly with leading *"0"*
H	hour in 24-hour format with as many digits as necessary
HH	hour in 24-hour format with 2 digits, possibly with leading *"0"*
m	minute with as many digits as necessary
mm	minute with 2 digits, possibly with leading *"0"*
M	month with as many digits as necessary
MM	month with 2 digits, possibly with leading *"0"*
MMM	month name (language-dependent), abbreviated to 3 characters
MMMM	full month name (language-dependent)

Formatting Character	Display
s	second with as many digits as necessary
ss	second with 2 digits, possibly with leading *“0”*
t	morning or afternoon indicator (language-dependent, in English AM and PM) with one character; use is not recommended.
tt	morning or afternoon indicator (language-dependent, in English AM and PM) with 2 characters
yy	year with 2 digits
yyyy	year with 4 digits

Any characters other than the formatting characters mentioned are included in the display. Characters to be displayed may optionally be enclosed in single quotes ('). They must be enclosed in single quotes if they are equivalent to a formatting character. To display a single quotation mark, it must be doubled in the format string ('').

Example

The format string *'Today is 'dddd', 'MMM dd', 'yyyyy* produces the following output: *“Today is Friday, Jul 04, 2014”*.

Notes

- If the format string is incorrect, the system-dependent standard format is used (corresponds to *.format =""*).
- The labels and symbols displayed for *ddd*, *dddd*, *MMM*, *MMMM*, *t*, and *tt* are determined by the language and region settings of the system.
- With long display texts, it may happen that no calendar symbol is displayed on the button for opening the calendar.

setup

With the *.format* attribute of the ***setup*** object the format variant (number) can be queried.

2.139 .format[index]

In a ***tablefield***, this attribute defines the format for an individual field indicated by [row, column].

.format[0,0] is the default value which is used for all remaining fields unless otherwise specified.

Definition

Data type	*Access*	*changed event*
string, *object* [***format***]	get, set	yes

C	*COBOL*
Identifier: AT_format	Identifier: AT-format
Data type: DT_string, DT_format	Data type: DT-string, DT-format

Classification	*Objects*
object-specific attribute	tablefield

See also

Attribute *.format*

2.140 .formatfunc

The format function interprets the format string in the attribute *.format*. *.formatfunc* cannot be specified anymore if the attribute *.format* is set at a format resource in an ***edittext*** or in a ***tablefield***.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C	*COBOL*
Identifier: AT_formatfunc	Identifier: AT-formatfunc
Data type: DT_func	Data type: DT-func

Classification	*Objects*
object-specific attribute	edittext, tablefield

Note

.formatfunc can only be used in a single-line edittext; it will be ignored in all other edittexts!

See also

Chapter “Format Functions” in manual “C Interface - Basics”

function DM_FmtDefaultProc

2.141 .function

Defines the function related to the object. This function must be defined by the application programmer. It is then called by the DM at the time specified in the function definition.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C	*COBOL*
Identifier: AT_function	Identifier: AT-function
Data type: DT_func	Data type: DT-func

Classification
standard attribute

See also

Chapter “Callback Function” in manual “Rule Language”

Chapter “Object Callback Functions” in manual “C Interface - Basics”

2.142 .gradient

This attribute determines the kind and parameters of a gradient for a ***color*** resource.

Availability

IDM FOR QT only

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_gradient	Identifier: AT-gradient
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	color

The value of the attribute is a string in the form *"<kind> [, <arg>]"*.

The parameter <*kind*> defines the gradient type. Only one gradient type may be specified.

The other parameters may include:

» Supplementary parameters for the gradient type
» Stop point definitions
» Color definitions

These **gradient types** are available:

Table 3: *Types of gradients*

Gradient	Definition	Explanation
linear	`"Linear, ..."` `"LinearV, ..."`	vertical gradient
	`"LinearH, ..."`	horizontal gradient
radial	`"Radial, ..."` `"Radial, <R>, ..."`	radial gradient with default radius 50% radial gradient with **supplementary parameter** radius R% radius is given as a percentage of the available space
conical	`"Conical, ..."` `"Conical, <S>, ..."`	conical gradient with start at 90° conical gradient with **supplementary parameter** S°, which indicates the starting angle

The **color definitions** are always appended after the gradient type and any supplementary parameters. You can use color names, HTML notation, and the *rgb(…)*, *hls(…)*, and *grey(…)* notations known for color resources.

In addition, a **stop point** can be specified for each color definition, which determines the weighting of the color. A stop point is a percentage with an optional percentage sign that always precedes each color and affects how much space that color occupies in the gradient. A gradient starts at *0%* and ends at *100%*, based on the area it fills. The specification *… , 20%, green, ...* means, for example, that after *20%* of the area to be filled, the color green is set. If no other color is set for the *0–20%* range, the first *20%* of the range is colored green. If another color is already set before *20%*, then a transition between this color and green is displayed.

Stop points should always be set in ascending order. If several colors are defined with the same stop point, the color that is furthest at the end of the parameter list applies. For example, the definition *.... , 20%, green, 40%, blue, 20%, red, ...* produces a gradient with a shade of red at *20%*, which changes to a blue tone that is shown as saturated at *40%*.

Examples

```
"Linear, green, yellow, red"    named colors, evenly distributed
"Linear, #00FFFF, #00FF00, #FF0000");    HTML notation,
                                         colors evenly distributed
"Linear,red, #00FF00, rgb(0,0,255)"      mixed notations,
```

```
                                          colors evenly distributed
"Linear, 20%, green, 60%, yellow, 80%, red");    named colors with
                                                 percentage stop points
"Linear, 20, green, 60, yellow, 80 ,red"    the % sign at the stop points
                                            is optional
```

Notes

Qt allows the setting of gradients in most places, but does not necessarily apply them. Whether a gradient is displayed depends very much on the object and the UI style. In this case, areas (e.g. backgrounds) are usually unproblematic, with delicate structures (e.g. texts) the gradient is often replaced by a single color. For grouping objects, gradients as a background are usually displayed.

2.143 .grey

This attribute defines the grayscale (*0 … 255*) of a ***color*** resource that is used for display on a gray-scale monitor (*setup.color_type = coltype_grey*).

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_grey	Identifier: AT-grey
Data type: DT_integer	Data type: DT-integer

Value range
0 … 255

Classification	*Objects*
object-specific attribute	color

2.144 .groupbox

The hierarchically superordinate groupbox can be queried by this attribute.

Definition

Data type
object

Access
get

C
Identifier: AT_groupbox
Data type: DT_object

COBOL
Identifier: AT-groupbox
Data type: DT-object

Classification
hierarchy attribute

2.145 .height

This attribute defines the current height of the object.

Definition

Data type	*Access*	*changed event*
integer (may be *0*)	get, set	yes

C	*COBOL*
Identifier: AT_height	Identifier: AT-height
Data type: DT_integer	Data type: DT-integer

Classification
geometry attribute

.height defines the height of the entire ***notebook*** object with all ist elements. If the height is defined too small, a minimum value depending on the window system will be used. The height must not be *0* since the required height cannot be calculated.

In ***tablefield*** the attribute can have the value *0*. This means that the Dialog Manager shall calculate the corresponding height so that all elements can be displayed in the respective directions without a scrollbar. If the ***tablefield*** height is set at *0*, it will get so wide that all lines can be displayed completely in the object.

With the ***toolbar***, *.height* (without index) defines the default value for the values not set by *.height [class]*.

Remark for the IDM for Windows

The height of a ***poptext*** cannot be specified; the window system chooses a value depending on the font.

See also

Attributes *.real_height*, *.width*, *.yauto*, *.ybottom*, *.ytop*

2.146 .height[class]

This attribute of the toolbar defines the height of the toolbar in the docking state given by the index.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_height	Identifier: AT-height
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	toolbar

Index Range

toolbar
: Height when the toolbar is docked

window
: Height when the toolbar is undocked (tool window)

Without an index the attribute returns or sets the default value for both docking states.

See also

Attribute *.width[class]*

2.147 .height[enum]

This attribute can be used to define correction factors for calculating the grid height from the ***font***.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_height	Identifier: AT-height
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	font

Index Range

scale_factor
With this index value, the attribute defines a percentage, which is multiplied with the base value as a scaling factor.
The base value is the height of a single-line ***edittext*** with this ***font***.

scale_offset
With this index value, the attribute defines a pixel value that is added as a constant to the scaled base value.

Thus the grid height is calculated by the following formula:

```
grid height =
  ( <height of edittext with this font> * .height[scale_factor] ) / 100 +
    .height[scale_offset]
```

See also

Attributes *.real_yraster*, *.width[enum]*, *.yraster*

Chapter “Calculating the Grid Size from a Reference Font” in manual “Resource Reference”

2.148 .help

Texts which are related to an object can be filed with this attribute. They can be used for a help system.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C
Identifier: AT_help
Data type: DT_string, DT_text

COBOL
Identifier: AT-help
Data type: DT-string, DT-text

Classification
standard attribute

2.149 .helpmenu

This attribute defines a right-aligned menubox in the menubar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_helpmenu Data type: DT_boolean	Identifier: AT-helpmenu Data type: DT-boolean

Classification	*Objects*
object-specific attribute	menubox

Note

If *.helpmenu* is specified for more than one visible menubox in the menubar, the DM decides which one is used as helpmenu. Therefore, define only one menubox which shall be right-aligned.

Remark for the IDM für Windows

If the attribute is set, all menuboxes defined thereafter are attached on the right side.

2.150 .helppos

This attribute defines the position of the helptext in a statusbar. It indicates the position in which the helptext *.statushelp* is to be displayed. The indication of *0* (default value) determines the help information to be displayed in one line. Value *n* indicates that the help text is to be displayed via all children up to the *n*-th. 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*, however, may always be set and will be displayed according to the description above.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_helppos	Identifier: AT-helppos
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	statusbar

2-

.-

151 .hls[enum]

This attribute defines the hue, lightness and saturation of an HLS color at a ***color*** resource.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_hls	Identifier: AT-hls
Data type: DT_integer	Data type: DT-integer

Value range
0 … 255

Classification	*Objects*
object-specific attribute	color

Index Range

color_hue
Determines the hue.

color_light
Determines the lightness (luminance).
color_sat
Determines the saturation.

In the HLS color model, colors are defined by hue **H**, lightness or luminance **L**, and saturation **S**. The range for all three values is *0* to *255*.

The colors are arranged within a cylinder whose top plane is white and whose bottom plane is black. Lightness L increases from *0* (dark) at the bottom to *255* (bright) at the top. For colors within the circle intersecting the center point of the cylinder, L is *127*. These colors are neither brightened (by mixture with white) nor darkened (by mixture with black).

Hue H is determined by the angle on the circular area. The range for H from *0* to *255* and the angle A in the color circle (actually ranging from *0* to *359* degrees) relate as given below:

» H = A / 2.

» A = (H modulo 180) * 2; that is the values from *180* to *255* match the color shades from *0* to *75*.

Red has a value of *H = 0* (or *180*), *H = 60* (or *240*) yields green and *H = 120* yields blue.

The shades of gray lie on the vertical line through the center of the cylinder, all having a saturation of *S = 0*. The colors on the lateral area of the cylinder have the highest saturation *S = 255*. The colors on the perimeter of the middle circle with *L = 127* and *S = 255* sometimes are referred as pure colors.

Note

The attributes *.rgb[enum]*, *.hls[enum]* and *.name* are mutually exclusive, so that “get” may lead to a *“can't get value”* error message.

2.152 .hsb_arrows

The attribute *.hsb_arrows* defines, whether the horizontal scrollbar has arrows at its ends.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_hsb_arrows
Data type: DT_boolean

COBOL
Identifier: AT-hsb-arrows
Data type: DT-boolean

Classification
scrollbar attribute

Value range

true
: Scrollbar with arrows

false
: Scrollbar without arrows

Availability

The attribute is only supported on Motif and since IDM version A.05.02.h.

See also

Attribute *.vsb_arrows*

2.153 .hsb_linemotion

This is the pixel value at which the horizontal scrollbar position changes during scrolling by lines.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_hsb_linemotion
Data type: DT_integer

COBOL
Identifier: AT-hsb-linemotion
Data type: DT-integer

Classification
scrollbar attribute

2.154 .hsb_optional

The horizontal scrollbar will only be displayed if necessary.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_hsb_optional
Data type: DT_boolean

COBOL
Identifier: AT-hsb-optional
Data type: DT-boolean

Classification
scrollbar attribute

tablefield

If the attribute is set at *true*, the Dialog Manager decides whether the horizontal scrollbar is actually needed or not. If for example all columns in a tablefield can be displayed completely in the available space and if the horizontal scrollbar has been set at optional, the scrollbar will not be displayed.

Note

.hsb_optional is ignored by the Motif version, i.e. *.hsb_optional* is always implicitly set at *true*.

2.155 .hsb_pagemotion

This is the pixel value at which the horizontal scrollbar position changes when scrolling by pages.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_hsb_pagemotion
Data type: DT_integer

COBOL
Identifier: AT-hsb-pagemotion
Data type: DT-integer

Classification
scrollbar attribute

2.156 .hsb_visible

.hsb_visible defines the visibility of the horizontal scrollbar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_hsb_visible
Data type: DT_boolean

COBOL
Identifier: AT-hsb-visible
Data type: DT-boolean

Classification
scrollbar attribute

2.157 .icon

.icon assigns an icon to an object.

Data type	*Access*	*changed event*
object [***tile***] (***window***) *enum* (***messagebox***)	get, set	yes

C	*COBOL*
Identifier: AT_icon Data type: DT_tile (***window***) Data type: DT_enum (***messagebox***)	Identifier: AT-icon Data type: DT-tile (***window***) Data type: DT-enum (***messagebox***)

Classification	*Objects*
object-specific attribute	messagebox, window

window

For the ***window*** object, .icon is used to assign a ***tile*** resource that will appear as symbol in the title bar of the window. If *.icon* is not set, the IDM uses the graphic resources “IDM_AppIcon” or “IDM_DefIcon”, which are predefined in the IDM libraries.

The resources “IDM_AppIcon” and “IDM_DefIcon” are provided by the IDM for the icon in the title bar of windows and, on MICROSOFT WINDOWS, also as application icon. This icon can be changed by defining a custom graphic resource with the identifier “IDM_AppIcon” in the application. The “IDM_DefIcon” provided by IDM is used if no “IDM_AppIcon” is available. This depends among other things on how the application is linked. Information on defining and linking graphic resources can be found in the developer documentation of the system.

The attribute is not supported by the IDM FOR MOTIF.

messagebox

For a***messagebox*** the *.icon* attribute has the data type *enum*.

Value range

icon_asterisk
: Displays the window system specific information icon.

icon_error
: Displays the window system specific error icon.

icon_exclamation
: Displays the window system specific warning icon.

icon_hand
: Displays the window system specific error icon.

icon_information
: Displays the window system specific information icon.

icon_query
: Displays the window system specific question icon.

icon_question
: Displays the window system specific question icon.

icon_warning
: Displays the window system specific warning icon.

noicon
: Does not display an icon.

The representation of *.icon* depends on the window system, its version and – on Qt – the UI style. There may also be multiple assignment.

Value	Windows	Motif	Qt
icon_asterisk	like *icon_information*		
icon_error	stop sign	error sign (crossed out circle)	lowercase “x”
icon_exclamation	exclamation mark “!”		
icon_hand	like *icon_error*		
icon_information	lowercase “i” in a circle	lowercase “i”	
icon_query	like *icon_question*		
icon_question	question mark “?”	question mark “?” (in a head)	question mark “?”
icon_warning	like *icon_exclamation*		
noicon	none		

2.158 .iconic

This attribute specifies whether the window is iconic.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_iconic Data type: DT_boolean	Identifier: AT-iconic Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

2.159 .iconifyable

This attribute defines whether a window is iconifyable and whether an iconifying mechanism is included in the ***window*** title.

The corresponding window will get an iconify button if the attribute is set at true. If it is set at false, there will be no iconifying mechanism.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_iconifyable	Identifier: AT-iconifyable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

2.160 .idispatch

The Idispatch COM interface pointer of the object can be accessed through this attribute under Microsoft Windows. When a new value is set to an object, the saved COM object will be deleted.

Definition

Data type	*Access*	*changed event*
pointer	get set (***document***, ***subcontrol***)	no

C	*COBOL*
Identifier: AT_idispatch Data type: DT_pointer	Identifier: AT-idispatch Data type: DT-pointer

Inheritance
no

Classification	*Objects*
object-specific attribute	control, doccursor, document, subcontrol

In the Rule Language the attribute can only be assigned to the same attribute of a different Dialog Manager object. Be aware, that in the programming interface the COM object is only valid during the time when it is in use by the 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 Dialog Manager cannot recognize this situation and this will lead to a system crash. The Dialog Manager will also crash if the given pointer does not point to a COM interface.

The attribute is available for the control in mode_client, the XML Cursor, the XML Document and the subcontrol. It, however, cannot be passed down. The XML Document tests to see if the IXMLDOMDocument2 COM interface is implemented, while the XML Cursor tests to see if the IXMLDOMNode COM interface is implemented. Please note, an XML Cursors, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM document.

2.161 .ignorecursor

This attribute decides whether ***windows*** and ***dialogs*** shall ignore the cursor (see also *.overridecursor*); this happens if the value is set at true.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_ignorecursor Data type: DT_boolean	Identifier: AT-ignorecursor Data type: DT-boolean

Classification	*Objects*
object-specific attribute	dialog, window

Remarks

- If *.ignorecursor* is set true at a ***dialog*** or ***window***, the temporary override-cursor is ignored by this relevant object and all its child objects.
- Usually the cursor is inherited from parent to child (if the child has no own defined cursor). The only exception is the object window which does not inherit its parents' cursor.
  If you specify an object other than a window in the attribute *.ignorecursor*, this object has to have an own cursor, otherwise it inherits the cursor specified at the parent, i.e. the cursor defined by Dialog.cursor.

Example

Except for one window, a temporary wait cursor is set for the entire dialog.

```
window Wi
{
  .ignorecursor true;
  .cursor CursorCross;
}
on Pb select
{
  setup.overridecursor := CursorWait;
}
```

See also

Attribute *.overridecursor*

2.162 .imagebgc

This attribute defines the background color of an ***image***.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_imagebgc Data type: DT_color	Identifier: AT-imagebgc Data type: DT-color

Classification	*Objects*
object-specific attribute	image

2.163 .imagefgc

.imagefgc defines the foreground color of an ***image***.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_imagefgc	Identifier: AT-imagefgc
Data type: DT_color	Data type: DT-color

Classification	*Objects*
object-specific attribute	image

2.164 .incrtime

.incrtime defines the time increment in which the ***timer*** is activated.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_incrtime	Identifier: AT-incrtime
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	timer

2.165 .index

With the help of this attribute, the selection of a particular listbox, poptext or tablefield entry can be queried for the event object ***thisevent*** (only valid for the *select* event).

For objects that can be displayed on the screen, the *.index* attribute can be used to find out what number of children the specified object is.

Definition

Data type
integer, *index*

Access
get

C
Identifier: AT_index
Data type: DT_integer, DT_index

COBOL
Identifier: AT-index
Data type: DT-integer, DT-index

Classification
standard attribute

In the object ***thisevent***, this attribute queries the selection of a listbox or poptext item (valid only for *select* events in a listbox or poptext; data type *integer*).

In the object ***thisevent***, this attribute queries the selection of a tablefield field (valid only for *select* events in a tablefield; data type *index*).

See Also

Chapter “Event Object thisevent” in manual “Rule Language”

2.166 .indexscope[attribute]

This attribute queries the validity range for the index of user-defined associative attributes (associative arrays).

To access a particular user-defined attribute, its attribute identifier with the data type *attribute* is used as index.

This attribute is available for all objects that may have user-defined attributes.

Definition

Data type
anyvalue

Access
get

C
Identifier: AT_indexscope
Data type: DT_anyvalue

COBOL
Identifier: AT-indexscope
Data type: DT-anyvalue

Classification
plain attribute

See also

Attributes .*scope[attribute]*, .*typescope*, .*typescope[integer]*

Chapter “Validity Range for Better Type Checking” in manual “Rule Language”

2.167 .input[integer]

This attribute is used for functions and rules to query whether the parameter is an input parameter.

Definition

Data type
boolean

Access
get

C
Identifier: AT_input
Data type: DT_boolean

COBOL
Identifier: AT-input
Data type: DT-boolean

Classification
object-specific attribute

Objects
function, ***rule***

Example

```
function Test (integer P input, integer Q output,
               string R input output)

for I := 1 to Test.count do
  print Test.input[I];
endfor
```

Output

```
true
false
true
```

See also

Chapters “Functions” and “Named Rules (Subprograms)” in manual “Rule Language”

2.168 .instance[integer]

For a Model, this attribute enables accessing the instances that are derived from this Model. The index is an *integer* value.

The total number of instances derived from a Model can be queried with *<model_id>.count[.instance]*.

Definition

Data type
object

Access
get

C
Identifier: AT_instance
Data type: DT_object

COBOL
Identifier: AT-instance
Data type: DT-object

Classification
object-specific attribute

2.169 .interfaceid

This attribute is used to specify an unambiguous ID for a Control object. This ID is generated by using the program **guidgen.exe**.

Definition

Data type
string

Access
get

C
Identifier: AT_interfaceid
Data type: DT_string

COBOL
Identifier: AT-interfaceid
Data type: DT-string

Classification
object-specific attribute

Objects
control

See also

Chapter “The control Object” in manual “OLE Interface”

2.170 .is_applet

This attribute can be used to query the setup object at runtime to determine whether the IDM Java Client is executed as an applet or as an application. For example, the attribute can be used to display a dialog embedded in an HTML page or in a window, depending on where the client is executed.

Definition

Data type
boolean

Access
get

C
Identifier: AT_is_applet
Data type: DT_boolean

COBOL
Identifier: AT-is-applet
Data type: DT-boolean

Classification
object-specific attribute

Objects
setup

2.171 .itemcount

This attribute specifies the number of items in a listbox or poptext. It can be set as well as read in the rules.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_itemcount	Identifier: AT-itemcount
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listbox, poptext

The number of texts to be displayed in the ***listbox*** can be specified with *.itemcount*. In order to be more efficient, the DM allows the itemcount to be set at a higher value than the number of texts already existing in the listbox. The new texts can be set afterwards. If a listbox is to be filled with several texts by the Rule Language or by DM_SetValue, *.itemcount* should first be set at the expected value, then the contents can be specified.

In a ***poptext***, however, the attribute *.itemcount* can only be set at a lower value than, or equal to, the current number of texts in the poptext. By setting this attribute, only a reduction in the number of texts in the poptext can be achieved.

2.172 .itemorder

This attribute is used to query the definition order for attributes and records within records. For each attribute there will be an “A”, for each child record there will be an “R” at the corresponding position in the string.

Definition

Data type
string

Access
get

C
Identifier: AT_itemordewr
Data type: DT_string

COBOL
Identifier: AT-itemorder
Data type: DT-string

Classification
object-specific attribute

Objects
record

Example

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

Output

```
"ARA"
```

Note

The *.itemorder* attribute is intended to provide the static definition order of the record children. It is not guaranteed that it will always and for any dynamic change of attributes and sub-records reflect the correct order of the children of a ***record***.

2.173 .ixmldomdocument2

The IXMLDOMDocument2 COM interface pointer of the XML Document can be accessed through this attribute under Microsoft Windows. When a new value is set in the XML Document, the saved DOM tree will be deleted. All existing XML Cursors will become invalid. The attribute *.mapped* has the value *false* by invalid XML Cursors.

Definition

Data type	*Access*	*changed event*
pointer	get, set	no

C
Identifier: AT_ixmldomdocument2
Data type: DT_pointer

COBOL
Identifier: AT-ixmldomdocument2
Data type: DT-pointer

Inheritance
no

Classification	*Objects*
object-specific attribute	document

In the Rule Language the attribute can only be assigned to the same attribute of a different Dialog Manager object. Be aware, that in the programming interface the COM object is only valid during the time when it is in use by the 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 Dialog Manager cannot recognize this situation and this will lead to a system crash. The Dialog Manager will also crash if the given pointer does not point to a COM interface. The attribute is available for the XML Document, but it cannot be passed down.

2.174 .ixmldomnode

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

Definition

Data type	*Access*
pointer	get

C	*COBOL*
Identifier: AT_ixmldomnode	Identifier: AT-ixmldomnode
Data type: DT_pointer	Data type: DT-pointer

Classification	*Objects*
object-specific attribute	doccursor

In the Rule Language the attribute can only be assigned to the same attribute of a different Dialog Manager object. Be aware, that in the programming interface the COM object is only valid during the time when it is in use by the 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 Dialog Manager cannot recognize this situation and this will lead to a system crash.

The attribute is available for the XML Cursor. Please note, an XML Cursors, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM document.

2.175 .ixmldomnodelist

The IXMLDOMNodeList COM interface pointer of the XML Cursor can be accessed through this attribute under Microsoft Windows. Via this interface pointer it is possible to access the direct children of the XML Cursor.

Definition

Data type	*Access*
pointer	get

C	*COBOL*
Identifier: AT_ixmldomnodelist	Identifier: AT-ixmldomnodelist
Data type: DT_pointer	Data type: DT-pointer

Classification	*Objects*
object-specific attribute	doccursor

In the Rule Language the attribute can only be assigned to the same attribute of a different Dialog Manager object. Be aware, that in the programming interface the COM object is only valid during the time when it is in use by the 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 Dialog Manager cannot recognize this situation and this will lead to a system crash.

The attribute is available for the XML Cursor. Please note, an XML Cursors, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM document.

2.176 .keyboard

In the ***setup*** object, this attribute requests the accelerator variant.

Definition

Data type
integer

Access
get

C
Identifier: AT_keyboard
Data type: DT_integer

COBOL
Identifier: AT-keyboard
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.177 .label

This attribute can be used to request and manipulate identifiers, i.e. internal object names.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C
Identifier: AT_label
Data type: DT_string

COBOL
Identifier: AT-label
Data type: DT-string

Classification
standard attribute

2.178 .label[anyvalue]

This attribute can be used to query identifiers of certain IDM elements such as methods or user-defined attributes of objects or arguments of functions. With the help of this attribute, type conversions can also be performed..

Definition

Data type	*Access*	*changed event*
string, attribute, method, object [***text, instance***], enum, class, event, type, boolean	get	yes

C	*COBOL*
Identifier: AT_label Data type: DT_string, DT_text, DT_instance, DT_attribute, DT_method, DT_enum, DT_event, DT_class, DT_type, DT_boolean	Identifier: AT-label Data type: DT-string, DT-text, DT-instance, DT-attribute, DT-method, DT-enum, DT-event, DT-class, DT-type, DT-boolean

Classification
standard attribute

Determining identifiers

Arguments of functions

In functions, this attribute can be used to query the identifiers of the arguments at a specific position. The position of the argument at the function call is specified as index.
A call is made according to the following pattern:

```
string  := function.label[integer]
```

Custom attributes and methods

User-defined attributes and methods can be requested from an object by passing the name as an index string.
A call is made according to the following pattern:

```
attribute := object.label[string]
method := object.label[string]
```

The following distinction applies:

- Index string starts with a ".": the user-defined attribute of the object is obtained
- Index string starts with a ":": the user-defined method of the object is obtained

Type conversions

This attribute can also be used for type conversions of the following data types:

- attribute
- method
- enum
- class
- event
- datatype
- boolean

A conversion can be done in both directions:

```
type := object.label["type-string"]
"type-string" := object.label[type]
```

Example

```
dialog D

function boolean Func(string Path, object Id) {
  return false;
}

window Wi {
    integer MySum := 512;

    rule void MyRule (string S)
    {
       print "Method called with: "+S;
    }
}

on dialog start {
   variable method M;
   variable attribute A;

   !! query parameters
   print Func.label[1];                          // result: "Path"
   print Func.label[2];                          // result: "Id"

   !! retrieve userdefined attributes and methods
   M := Wi.label[":MyRule"];
   Wi:call(M, "Hello");                          // result: "Method called
with: Hello"

   A := Wi.label[".MySum"];
   Wi.type[A] := string;
```

```
   !! type conversions
   print this.label[button_ok];                    // result: "button_ok"
   print this.label["button_ok"];                  // result: button_ok
(enum)

   print Wi.label[".MySum"];                       // result: .MySum
(attribute)
   print Wi.label[.MySum];                         // result: "MySum"

   print Wi.label[:MyRule];                        // result: "MyRule"

   print this.label[window];                       // result: "window"
   print this.label["window"];                     // result: window (class)

   print this.label[select];                       // result: "select"
   print this.label["select"];                     // result: select (event)
}
```

2.179 .language

At the ***setup*** object, this attribute queries or sets the current language variant of texts.

For a function, this attribute can be used to query the programming language in which the function is implemented.

Data type	*Access*	*changed event*
integer (***setup***) *enum* (***function***)	get, set (***setup***) get (***function***)	no

C	*COBOL*
Identifier: AT_language Data type: DT_integer (***setup***) Data type: DT_enum (***function***)	Identifier: AT-language Data type: DT-integer (***setup***) Data type: DT-enum (***function***)

Classification	*Objects*
object-specific attribute	***function***, setup

function

Value range

lang_default
: No programming language is specified in the function definition. The function is implemented in the default programming language C.

lang_c
: The function is implemented in the programming language C.

lang_cobol
: The function is implemented in the programming language COBOL.

lang_cobol_cancel
: The function is implemented in the COBOL programming language and is called using a CANCEL statement.

2.180 .language[integer]

This attribute queries or sets the text of variant *I* for a ***text*** resource.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_language	Identifier: AT-language
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	text

2.181 .lastchild

This attribute accesses the last child of an object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_lastchild
Data type: DT_object

COBOL
Identifier: AT-lastchild
Data type: DT-object

Classification
hierarchy attribute

2.182 .lastmenu

.lastmenu addresses the last child in the menu hierarchy.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_lastmenu
Data type: DT_object

COBOL
Identifier: AT-lastmenu
Data type: DT-object

Classification
hierarchy attribute

2.183 .lastrecord

This attribute returns or sets the last ***record*** of an object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_lastrecord
Data type: DT_record

COBOL
Identifier: AT-lastrecord
Data type: DT-record

Classification
hierarchy attribute

2.184 .lastsubcontrol

This attribute returns or sets the last ***subcontrol*** of a ***control*** or ***subcontrol*** object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_lastsubcontrol
Data type: DT_object

COBOL
Identifier: AT-lastsubcontrol
Data type: DT-object

Classification
hierarchy attribute

2.185 .lasttoolbar

This attribute of the ***window*** object accesses the last toolbar of the window.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_lasttoolbar
Data type: DT_object

COBOL
Identifier: AT-lasttoolbar
Data type: DT-object

Classification
hierarchy attribute

2.186 .layoutbox

This attribute returns the layoutbox an object belongs to.

When the object is a direct or indirect child of a layoutbox, this layoutbox is returned; otherwise *null* is returned. With nested layoutboxes, the layoutbox is returned, which comes first when you move up the hierarchy starting from the object.

Definition

Data type
object

Access
get

C
Identifier: AT_layoutbox
Data type: DT_object

COBOL
Identifier: AT-layoutbox
Data type: DT-object

Classification
hierarchy attribute

2.187 .level[integer]

This attribute is used to define the indention of each item for treeview object, i.e. it defines the child-parent-relation and indicates the indention of content field. The default value is stored in index *0*. Changing the default value will thus result in changes of unchanged entries.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_level	Identifier: AT-level
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	treeview

2.188 .license_key

In this attribute of the ***control*** object, the license key for an ActiveX control can be indicated; currently applicable in client mode only.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_license_key	Identifier: AT-license-key
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	control

2.189 .linemotion

This attribute defines the pixel number at which the scrollbar value changes if the user wants to scroll by lines.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_linemotion	Identifier: AT-linemotion
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	scrollbar

Remark

.linemotion is ignored by Motif if *.arrows* is set at *false*.

2.190 .load

This attribute of the ***import*** object provides information about the load status of the associated module.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_load Data type: DT_boolean	Identifier: AT-load Data type: DT-boolean

Classification	*Objects*
object-specific attribute	import

See also

Chapter “Modularization” in manual “Programming Techniques”

2.191 .local

With this attribute you can define whether an application runs locally or via a network.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_local
Data type: DT_boolean

COBOL
Identifier: AT-local
Data type: DT-boolean

Classification	*Objects*
object-specific attribute	application

2.192 .logfile

With this attribute of the ***setup*** object, the absolute path of the **log file** can be queried at runtime.

Definition

Data type
string

Access
get

C
Identifier: AT_logfile
Data type: DT_string

COBOL
Identifier: AT-logfile
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.193 .majortabheight

This attribute defines the height of all majortabs (main index) of a ***notebook***.

With the default value *0*, the height is calculated automatically so that the majortabs of all visible notepages (*.real_visible* = *true*) fit.

If the attribute *.sizeraster* is set at *true*, the value is to be indicated in grid coordinates.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_majortabheight	Identifier: AT-majortabheight
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	notebook

Note

When a notepage is set at visible or non-visible, a recalculation of several tabs is necessary. This recalculation can also change the size of notepages and of all child objects. Therefore we suggest you to use *0* only for notebooks having a static layout.

Note for the IDM for Windows

On MICROSOFT WINDOWS, the value *0* means that WINDOWS determines the height itself. However, if the attribute .majortabwidth is not *0* or the attribute .tabalignment is set to *1*, the height is calculated so that text and image are visible in the tabs.

See also

Attributes *.majortabwidth*, *.sizeraster*, *.tabalignment*

2.194 .majortabwidth

Defines the width of all majortabs (main index) of a ***notebook***.

With the default value *0*, the width is calculated automatically so that the majortabs of all visible notepages (*.real_visible* = *true*) fit.

If the attribute *.sizeraster* is set at *true*, the value is to be indicated in grid coordinates.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_majortabwidth	Identifier: AT-majortabwidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	notebook

Note

When a notepage is set at visible or non-visible, a recalculation of several tabs is necessary. This recalculation can also change the size of notepages and of all child objects. Therefore we suggest you to use *0* only for notebooks having a static layout.

Note for the IDM for Windows

On MICROSOFT WINDOWS, the value *0* means that WINDOWS determines the width of each tab individually. However, if the attribute .majortabheight is not *0* or the attribute .tabalignment is set to *1*, the width is calculated so that the text and image of the widest, visible tab can be displayed. All tabs get this uniform width.

See also

Attributes *.majortabheight*, *.sizeraster*, *.tabalignment*

2.195 .mapped

An object with this attribute set to *false* is created when *.visible* is *true*, but it is not displayed yet. Not until *.mapped* is set to *true*, the object is made visible.

The default value is *true*.

Attention

This attribute should only be set to *false* in exceptional cases.

doccursor

This attribute returns whether the doccursor points to a node in the DOM tree (*true*) or not (*false*). It should be kept in mind that accessing a different attribute normally positions the doccursor at the root of the DOM tree.

Definition

Data type	*Access*	*changed event*
boolean	get, set get (***doccursor***)	no

C	*COBOL*
Identifier: AT_mapped Data type: DT_boolean	Identifier: AT-mapped Data type: DT-boolean

Classification	*Objects*
object-specific attribute	doccursor

2.196 .mapping[integer]

The mapping objects of a transformer can be accessed through this attribute. The attribute is indexed with the object index (similar to child).

Definition

Data type	*Access*	*changed event*
*object (**mapping**)*	get, set	no

C	*COBOL*
Identifier: AT_mapping Data type: DT_mapping	Identifier: AT-mapping Data type: DT-mapping

Classification	*Objects*
object-specific attribute	transformer

2.197 .masterapplication[enum]

This attribute defines the assignment of functions that are located directly under a module/dialog to an ***application*** object.

Values from the enum range lang_default...lang_java and func_normal...func_data can be used as optional index value.

Definition

Data type	*Access*	*changed event*
object [application]	get, set	yes

C	*COBOL*
Identifier: AT_masterapplication	Identifier: AT-masterapplication
Data type: DT_application	Data type: DT-application

Classification	*Objects*
object-specific attribute	dialog, module

Index Range

lang_default
: No programming language is specified in the function definition. The function is implemented in the default programming language C.

lang_c
: The function is implemented in the programming language C.

lang_cobol
: The function is implemented in the programming language COBOL.

lang_cobol_cancel
: The function is implemented in the COBOL programming language and is called using a CANCEL statement.

lang_fortran (obsolete)
: The function is implemented in the programming language FORTRAN.

lang_java (obsolete)
: The function is implemented in the programming language JAVA.

lang_pascal (obsolete)
: The function is implemented in the programming language PASCAL.

func_callback (internal)
: The function is an object callback function that can be assigned to an object in its *.function* attribute and is called for defined events.
Keyword in the function definition: **callback**.

func_canvas (internal)
: The function is a canvas function that can be assigned to a ***canvas*** object in its *.canvasfunc* attribute.
Keyword in the function definition: **canvasfunc**.

func_content (internal)
The function is a reloading function that can be assigned to a ***tablefield*** in its *.contentfunc* attribute.
Keyword in the function definition: **contentfunc**.

func_data (internal)
The function is a data function that can be assigned to objects in their *.datamodel* attribute and is called by the IDM when synchronizing the model and view components.
Keyword in the function definition: **datafunc**.

func_format (internal)
The function is a format function that can be assigned to the objects ***edittext*** and ***tablefield*** in their *.formatfunc* attribute.
Keyword in the function definition: **formatfunc**.

func_normal (internal)
The function is a "normal" application function that can be called from the Rule Language.

func_spinbox (internal)
The function is used to control the displayed value of a ***spinbox*** with *.style = void* from the application and can be assigned to the ***spinbox*** in its *.spinfunc* attribute.
Keyword in the function definition: **spinfunc**.

When calling a module function, the assignment to an application is searched for in the .masterapplication[] attribute of the module. This is done according to the specified language definition, the function type or even without any typing at all (without index specification). This application assignment is then used to determine the function connection and thus call the function accordingly. If no assignment has been defined at the module, the search is performed at the dialog. A definition to **null** prevents the search at the dialog.

The .masterapplication attribute was introduced for IDM version A.06.02.m to allow assignment of module functions in a similar way to the .application attribute on the import object.

Example

Modularized dialog with the functions in the module "Functions.mod", where COBOL functions are attached to the server application "ApplServer" and all other functions to the "dynlib" application "ApplLocal".

```
// Dialog.dlg
dialog Dlg
{
  .masterapplication ApplLocal;
}

application ApplLocal {
  .transport "dynlib";
  .exec "libusercheck.so";
  .active true;
}
```

```
use Functions;

on dialog start
{
  variable string Username := setup.env["USER"];
  if  ValidateUser(Username) then
    print "Age = "+GetAge(Username);
  endif
  exit();
}

// Functions.mod
module Functions
{
   .masterapplication[lang_cobol] ApplServer;
}

use Server;

record RecUser
{
  string[50] FirstName;
  string[50] LastName;
  integer Age;
}

export function boolean ValidateUser(string Username);
function cobol integer GetUserProfile(string[50] Username,
                                      record RecUser output);

export rule integer GetAge(string Username)
{
  if GetUserProfile(Username, RecUser)>0 then
    return RecUser.Age;
  endif
  return 0;
}

// Server.mod
module Server

export application ApplServer {
  .connect "server:4712";
  .active true;
}
```

Availability

Since IDM version A.06.02.m

See also

Attribute *.application*

Chapter “Modularization” in manual “Programming Techniques”

2.198 .maxchars

This attribute specifies the maximum number of input characters possible in an edittext.

If *.maxchars = 0*, the number of characters is unlimited.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_maxchars	Identifier: AT-maxchars
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	edittext

Remark

If in a single-line edittext the attribute *.width* is set at a value smaller than the value of *.maxchars*, input characters are shifted horizontally.

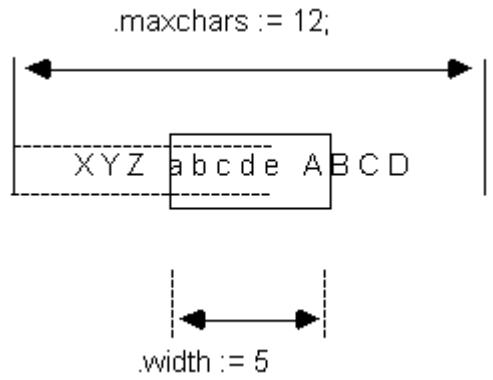


Figure 1: *Visible part of an edittext*

If the edittext is multi-lined, lines can be added as long as the number of characters defined in *.maxchars* has not been reached. If more than the displayable number of lines has been generated, the top lines move out of the top border of the edittext object (scrolling). Any position in the input lines can be reached by positioning the text cursor correspondingly.

2.199 .maxchars[index]

In a ***tablefield***, this attribute defines the maximum number of characters to be input into a specific field indicated by [row, column].

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_maxchars	Identifier: AT-maxchars
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.200 .maxheight

This attribute specifies the maximum height of an object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_maxheight
Data type: DT_integer

COBOL
Identifier: AT-maxheight
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.maxwidth*, *.minheight*

2.201 .maximized

This attribute defines the maximal state of the object ***window***.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_maximized Data type: DT_boolean	Identifier: AT-maximized Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

2.202 .maxsize[integer]

This attribute allows for the setting of the largest allowed size that each separate split area can have. If the split bar is moved beyond this maximal value, it will automatically spring back to the last possible value allowed. Pixel and raster values are both allowed.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_maxsize	Identifier: AT-maxsize
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	splitbox

The index is zero-based. The valid values for this are:

maxsize[I] – valid, if *0<=I* and *I<=childcount*

maxsize[0] is the so-called zero-element, that no split area is assigned to. The value of the zero-element is transferred to the other elements of the maxsize[I] vector for which no explicit setting exists.

See also

Attributes *.minsize[integer]*, *.size[integer]*

2.203 .maxvalue

This attribute defines the maximum value.

Definition

Data type	*Access*	*changed event*
integer *string*, *object* [***text***] (***datetime***)	get, set	yes

C	*COBOL*
Identifier: AT_maxvalue Data type: DT_integer Data type: DT_string, DT_text (***datetime***)	Identifier: AT-maxvalue Data type: DT-integer Data type: DT-string, DT-text (***datetime***)

Classification	*Objects*
object-specific attribute	datetime, progressbar, scrollbar

scrollbar

.maxvalue specifies the maximal scrollbar value.

The value range of the queried slider position is defined by *.minvalue* and *.maxvalue*.

progressbar

Maximum value of the progress display.

Defines the 100% value. This is the value that *.curvalue* reaches when the action whose progress is displayed has been completed.

datetime

This attribute determines the maximum value.

For write access (“set”), values of the data type *string* and ***text*** resources can be specified. ***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *string*.

Value range

""	No maximum value set.
<*value_string*>	Sets the contained date or time value as the maximum value. The syntax and evaluation of the string are described at the attribute **.value**.

The attribute defines the largest value that can be entered or selected by the user. An existing content of *.value* is usually not corrected to comply with *.maxvalue*. It is also not defined whether the displayed value is corrected when the ***datetime*** is made visible.

See also

Attribute *.minvalue*

2.204 .maxwidth

This attribute defines the maximal width of an object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_maxwidth
Data type: DT_integer

COBOL
Identifier: AT-maxwidth
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.maxheight*, *.minwidth*

2.205 .member[integer]

This attribute provides the [I]-user-defined attribute.

Definition

Data type	*Access*	*changed event*
attribute	get, set	yes

C
Identifier: AT_member
Data type: DT_attribute

COBOL
Identifier: AT-member
Data type: DT-attribute

Classification
hierarchy attribute

See also

Chapter “User-defined Attributes” in manual “User-defined Attributes and Methods”

2.206 .membercount

This attribute defines the number of user-defined attributes for an object.

Definition

Data type
integer

Access
get

C
Identifier: AT_membercount
Data type: DT_integer

COBOL
Identifier: AT-membercount
Data type: DT-integer

Classification
hierarchy attribute

See also

Chapter “User-defined Attributes” in manual “User-defined Attributes and Methods”

2.207 .menu

.menu is the identifier of the menu related to an object. It appears as **pop-up menu**.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_menu
Data type: DT_object

COBOL
Identifier: AT-menu
Data type: DT-object

Classification
plain attribute

Note for IDM with Windows

Popup menus may also be opened via the system menu key (for Windows this is the `Alt` key). If you want to open the window menu directly, you should use the `F10` key.

Note for DM with Motif

As of Motif 2.1, popup menus can be opened here by default for all object classes with the key combination `Shift + F10` .

It is not possible to attach a pop-up menu to the object ***rectangle***. The attribute *.menu* is ignored for that object in the Motif version.

See also

Attribute *.menu[integer]*

Method openpopup

2.208 .menu[integer]

Addresses menu child number "i" of a ***window*** (=> pull-down menu).

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_menu
Data type: DT_object

COBOL
Identifier: AT-menu
Data type: DT-object

Classification
hierarchy attribute

If the menu was created by a hierarchical model it is read only, i.e. the attribute can be defined but not changed

See also

Attribute .menu

2.209 .menubgc

This attribute defines the background color for a **menubar**.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C
Identifier: AT_menubgc
Data type: DT_color

COBOL
Identifier: AT-menubgc
Data type: DT-color

Classification
layout attribute

2.210 .menucount

.menucount queries the number of menu elements in a window.

Definition

Data type
integer

Access
get

C
Identifier: AT_menucount
Data type: DT_integer

COBOL
Identifier: AT-menucount
Data type: DT-integer

Classification
hierarchy attribute

2.211 .menufgc

This attribute defines the foreground color of the **menubar**.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C
Identifier: AT_menufgc
Data type: DT_color

COBOL
Identifier: AT-menufgc
Data type: DT-color

Classification
layout attribute

2.212 .message[integer]

Message to be sent to the client at the ***control*** object (OLE).

Definition

Data type
*object (**message**)*

Access
get

C
Identifier: AT_message
Data type: DT_object

COBOL
Identifier: AT-message
Data type: DT-object

Classification
object-specific attribute

Objects
control

See also

Chapter “Events” in manual “OLE Interface”

2.213 .mincolwidth

This attribute sets the minimum column width for a ***layoutbox*** with column-by-column arrangement (*.direction = 1*). It is ignored with row-by-row arrangement (*.direction = 2*). The attribute defines the minimum width for objects with *.xauto = 0*, if no other objects or only objects with *.xauto = 0* are in the same column.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_mincolwidth	Identifier: AT-mincolwidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	layoutbox

See also

Attribute *.minrowheight*

2.214 .mincolwidth[integer]

The attribute specifies the minimum column widths in the detail view of the ***listview*** object. Users cannot interactively make the columns narrower than defined in this attribute.

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_mincolwidth Data type: DT_integer	Identifier: AT-mincolwidth Data type: DT-integer

Default value	*Inheritance*
0	yes

Classification	*Objects*
object-specific attribute	listview

The value range of the index is *0 … .colcount*t, where the value with index *0* is used as default value for not set values in the range *1 … .colcount*.

Value range

0 (default)
No minimum column width.
The user can hide the column by dragging the right edge of the column header to the left edge of the column with the mouse button pressed. The column can be shown again by pressing the mouse button in the column header of the next column slightly to the right of the left column edge and dragging the mouse to the right while holding down the button. Pay attention to the different shape of the mouse pointer directly above the left column edge and slightly to the right of it.

> 0
Minimum column width that the user can interactively set in the detail view.
For invalid values, the default value *0* is used. However, the attribute value is not changed.

Note

Changing the attribute in the visible state may cause the object to flicker.

2.215 .minheight

This attribute specifies the minimum height of an object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_minheight
Data type: DT_integer

COBOL
Identifier: AT-minheight
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.maxheight*, *.minwidth*

2.216 .minortabheight

This attribute defines the height of all minortabs (side index).

Default value *0*. The height will be calculated automatically so that the minortabs of all visible notepages (*.real_visible* = *true*) fit.

If the attribute *.sizeraster* is set at *true*, the value will be indicated in coordinates.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_minortabheight Data type: DT_integer	Identifier: AT-minortabheight Data type: DT-integer

Classification	*Objects*
object-specific attribute	notebook

Note

When a notepage is set at visible or non-visible, a recalculation of several tabs is necessary. This recalculation can also change the size of notepages and of all child objects. Therefore we suggest you to use *0* only for notebooks having a static layout.

See also

Attributes *.majortabheight*, *.minortabwidth*, *.sizeraster*

2.217 .minortabwidth

This attribute defines the width of all minortabs (side index).

Default value *0*. The width will be calculated automatically so that the minortabs of all visible note-pages (*.real_visible* = *true*) fit.

If the attribute *.sizeraster* is set at *true*, the value will be indicated in coordinates.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_minortabwidth	Identifier: AT-minortabwidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	notebook

Note

When a notepage is set at visible or non-visible, a recalculation of several tabs is necessary. This recalculation can also change the size of notepages and of all child objects. Therefore we suggest you to use *0* only for notebooks having a static layout.

See also

Attributes *.majortabwidth*, *.minortabheight*, *.sizeraster*

2.218 .minrowheight

This attribute sets the minimum row height for a ***layoutbox*** with row-by-row arrangement (*.direction = 2*). It is ignored with column-by-column arrangement (*.direction = 1*). The attribute defines the minimum height for objects with *.yauto = 0*, if no other objects or only objects with *.yauto = 0* are in the same row.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_minrowheight Data type: DT_integer	Identifier: AT-minrowheight Data type: DT-integer

Classification	*Objects*
object-specific attribute	layoutbox

See also

Attribute *.mincolwidth*

2.219 .minsize[integer]

This attribute allows for the setting of the smallest allowed size that each separate split area can have. If the split bar is moved beyond this minimal value, it will automatically spring back to the last possible value allowed. Pixel and raster values are both allowed.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_minsize	Identifier: AT-minsize
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	splitbox

The index is zero-based. The valid values for this are:

minsize[I] – valid, if *0<=I* and *I<=childcount*

minsize [0] is the so-called zero-element, that no split area is assigned to. The value of the zero-element is transferred to the other elements of the minsize[I] vector for which no explicit setting exists.

See also

Attributes *.maxsize[integer]*, *.size[integer]*

2.220 .minvalue

This attribute defines the minimum value.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes
string, *object* [***text***] (***datetime***)		

C	*COBOL*
Identifier: AT_minvalue	Identifier: AT-minvalue
Data type: DT_integer	Data type: DT-integer
Data type: DT_string, DT_text (***datetime***)	Data type: DT-string, DT-text (***datetime***)

Classification	*Objects*
object-specific attribute	datetime, progressbar, scrollbar

scrollbar

.minvalue contains the minimum value of the scrollbar.

The value range of the queried slider position is defined by *.minvalue* and *.maxvalue*.

progressbar

Minimum value of the progress display.

Defines the 0% value. This is the value that *.curvalue* has when the action is started whose progress will be displayed.

datetime

This attribute determines the minimum value.

For write access (“set”), values of the data type *string* and ***text*** resources can be specified. ***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *string*.

Value range

""	No minimum value set.
<*value_string*>	Sets the contained date or time value as the minimum value. The syntax and evaluation of the string are described at the attribute **.value**.

The attribute defines the smallest value that can be entered or selected by the user. An existing content of *.value* is usually not corrected to comply with *.minvalue*. It is also not defined whether the displayed value is corrected when the ***datetime*** is made visible.

See also

Attribute *.maxvalue*

2.221 .minwidth

This attribute specifies the minimum width of an object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_minwidth
Data type: DT_integer

COBOL
Identifier: AT-minwidth
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.maxwidth*, *.minheight*

2.222 .mode

This attribute controls whether the control of ***control*** object can be used as OLE Server.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_mode	Identifier: AT-mode
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	control

Value range

mode_client

The ***control*** object is treated as an OLE Client, i.e. an OLE Server is accessed via this object.

mode_server

The ***control*** object acts as an OLE Server and can be accessed by other clients.

mode_none

Type of use is undefined, i.e. the ***control*** is used neither as client nor as server. This value can be set, for example, if you have actually implemented an OLE Server that for some reason is currently not able to act as a server, e.g. because the program was started as a normal program.

2.223 .model

The value of this attribute is the identifier of an object which has been defined as a Model. The new object contains all attributes of the model object unless they have been overwritten by a local object definition.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_model
Data type: DT_object

COBOL
Identifier: AT-model
Data type: DT-object

Classification
standard attribute

See also

Method :instance_of()

Chapter “Use of Models” in manual “Programming Techniques”

2.224 .module

This attribute returns the containing module of an object.

Definition

Data type
object

Access
get

C
Identifier: AT_module
Data type: DT_object

COBOL
Identifier: AT-module
Data type: DT-object

Classification
standard attribute

2.225 .mouse_buttons

In the setup object, this attribute queries the number of logical mouse buttons (*0* = no mouse).

Definition

Data type
integer

Access
get

C
Identifier: AT_mouse_buttons
Data type: DT_integer

COBOL
Identifier: AT-mouse-buttons
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark

The physical number of mouse buttons is not necessarily the same as the logical number. Logical mouse buttons may exist; this depends on the mouse driver software used; very often, a third button is simulated by pressing two buttons at the same time.

2.226 .mouseover

The *.mouseover* attribute enables to configure an ***image*** object to respond to mouseover events.

By default, the attribute is set to *false* and the picture displayed by the ***image*** object does not change when the mouse pointer is above the object.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_mouseover	Identifier: AT-mouseover
Data type: DT_boolean	Data type: DT-boolean

Default value
false

Classification	*Objects*
object-specific attribute	image

If *.mouseover* is set to *true*, the ***tile*** resources given in *.picture[tile_mouse_over]* and *.picture[tile_active_mouse_over]* are displayed when the mouse pointer is moved over the ***image*** object or the left mouse button is pressed over it.

See also

Attribute *.picture[enum]*

2.227 .moveable

This attribute declares whether a ***window*** is interactively moveable by the user.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_moveable Data type: DT_boolean	Identifier: AT-moveable Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

Particularity of Motif

Depending on the display or desktop manager in use, the attribute cannot be changed in the visible state; under certain conditions, it may not be possible to set it at runtime. In some cases, it may help to toggle the visibility of the window.

Since the ability of setting this attribute on MOTIF directly depends on the display or desktop manager used, it is recommended to set the attribute only statically or immediately after creating an instance with *:create(…, true)* in the invisible state.

Remark Qt

The attribute .moveable has no effect in Qt. Windows can always be moved.

2.228 .msgboxtext[enum]

This attribute defines the text for buttons in a ***messagebox***. If a button text is not supported by the window system used, *.msgboxtext* is ignored.

Data type	*Access*	*changed event*
object [***text***], *string*	get, set	yes

C	*COBOL*
Identifier: AT_msgboxtext	Identifier: AT-msgboxtext
Data type: DT_text, DT_string	Data type: DT-text, DT-string

Classification	*Objects*
object-specific attribute	dialog

Some window systems (MICROSOFT WINDOWS) have fixed button texts and do not allow labeling by the user. Other systems (MOTIF) do not define texts and allow the application to define them (internationalization).

.msgboxtext has an index. The index can assume values of type *enum*, with the form “button_…”. The values are texts (***text*** resources or strings) which contain the strings of the seven available buttons (see objects messagebox and dialog). Usually, the buttons strings are *OK*, *Cancel*, *Retry*, *Abort*, *Ignore*, *Yes*, *No*.

Index Range

button_abort
: Defines the caption for the “Abort” button.

button_cancel
: Defines the caption for the “Cancel” button.

button_ignore
: Defines the caption for the “Ignore” button.

button_no
: Defines the caption for the “No” button.

button_ok
: Defines the caption for the “OK” button.

button_retry
: Defines the caption for the “Retry” button.

button_yes
: Defines the caption for the “Yes” button.

Example with German Texts

```
dialog Test
{
  .msgboxtext[button_ok]      "OK";
  .msgboxtext[button_cancel]  "Abbrechen";
  .msgboxtext[button_retry]   "Wiederholen";
```

```
  .msgboxtext[button_abort]   "Abbrechen";
  .msgboxtext[button_ignore]  "Ignorieren";
  .msgboxtext[button_yes]     "Ja";
  .msgboxtext[button_no]      "Nein";
}
```

See also

Object messagebox

2.229 .multiline

This attribute defines whether an ***edittext*** is single- or multi-line respectively whether the tabs of a notebook should be displayed multi-line or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_multiline	Identifier: AT-multiline
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	edittext, notebook

2.230 .multisel

This attribute controls whether the user can select multiple entries in an object or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_multisel	Identifier: AT-multisel
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	filereq, listbox, tablefield

filereq (File Requester, File Dialogs)

This attribute controls the choices of the user in a ***filereq*** object. If its value is *true*, the user can select multiple files. The selected files are then found in the indexed attribute *.value[integer]*. Normally, the selected file or directory is in the scalar attribute *.value*.

Particularities

This attribute is **only** supported on MICROSOFT WINDOWS for the modes *fr_load* and *fr_save*, which is the selection of files.

listbox

This defines the selection type in a ***listbox***, i.e. multiple selection or single selection.

If *.multisel*; is set at *false*, only one of the listbox items can be selected with a mouse click. If *.multisel* is set at *true*, several listbox items can be selected with a mouse click. If an item which has already been selected is selected again with a mouse click, the selection is reversed (deselected).

Remark on the IDM for Windows

If the cursor is moved up and down with the cursor keys in a ***listbox*** that has *.multisel* set at *false*, the item that has the focus is also selected.

tablefield

At the ***tablefield***, *.multisel* controls whether several fields/columns/rows can be selected, depending on the selection types activated (details see object tablefield).

Multiple Selection in Motif

Multiselection is the selection of several different objects or a series of objects (e.g. items in a ***listbox***) with only one single action.

1. To select **several listbox items** together which are **not direct neighbors**, select one item with the left mouse button and release the button. Then move on to the next item to be selected and press the `Ctrl` key and afterward the left mouse button to select the second item. This action can be repeated for any further number of items.
2. To select **several following listbox items**, select an item by the left mouse button; this item becomes the first in the series of items. The selected state of the item appears inverted. All previously selected listbox items are now deselected!
   If you keep the mouse button is pressed and move the mouse over further items, all items located between the first item and the current cursor position appear inverted, which shows that they are selected. The selection ends with the item on which the mouse button is released. All items between the first and last selected items are selected.

   The same effect is achieved if you move the cursor to the item which shall be the last in the row, beginning from the first selected item, and then press the `Shift` key, keep it pressed and then also press the left mouse button.
   Entries can be **added** or **subtracted** from a series of items already selected by the `Ctrl` key.

   Add
   After selecting the “first” item row, move the cursor to the first item in the “new” item row and press the `Ctrl` key and the left mouse button at the same time. The item in which the cursor is located appears inverted. Previous selections remain unchanged.
   Thus, if another row is to be selected after selecting the first one, the cursor must be moved to the corresponding positions: press the `Ctrl` key and execute the further action as described above in 1. and 2..

   Subtract
   Subtraction is carried out by reversing above actions. This means that the same procedures are used on already selected items.
   Set the cursor on the item to be deselected first, press the `Ctrl` key and afterward the left mouse button. The item on which the cursor is located “de-inverts”, i.e. it appears in normal state. If the mouse button is kept pressed and the cursor moved on, all items touching the cursor are deselected. The same effect is reached by moving the cursor to the relevant position, pressing the `Shift` key and then the left mouse button.
   The described selection procedures can also be used with the **keyboard**. The “keyboard cursor” (`Tab`), displayed in a frame, has to be moved into the listbox. In this mode, the left mouse button is replaced by the selection key `Space`. Otherwise the procedure is carried out as described above.

Example

```
Listbox.multisel  := true;
Listbox.active[I] := false;
```

See also

Attributes *.active[integer]*, *.activeitem*, *.selstyle*

2.231 .mustexist

This attribute of the ***filereq*** object ensures that only existing files and indexes can be selected (value *true*). Otherwise, there is no check whether the file or the entered path is correct.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_mustexist Data type: DT_boolean	Identifier: AT-mustexist Data type: DT-boolean

Classification	*Objects*
object-specific attribute	filereq

Special Features of Microsoft Windows

This attribute only works in the *fr_load* mode.

2.232 .name

With the layout resources ***color***, ***font*** and ***tile***, this attribute specifies the name of the resource. For ***color*** resources, *.name* can be one of the system's predefined color names. With ***fonts***, it can be the system identifier of one of the fonts available on the system. With ***tiles***, *.name* can be the path of an image file.

This attribute defines the program ID of the OLE Server for a ***control*** object.

For the ***doccursor***, the attribute returns the tag name of the current DOM node.

With the ***mapping*** object this attribute defines a pattern for nodes of an XML tree or an IDM object hierarchy.

Definition

Data type	*Access*	*changed event*
string	get, set get (***doccursor***)	yes

C	*COBOL*
Identifier: AT_name Data type: DT_string	Identifier: AT-name Data type: DT-string

Classification	*Objects*
object-specific attribute	color, control, doccursor, font, mapping, tile

control

For an OLE Client, this attribute has to be set to the name or ProgID of the corresponding server.

For an OLE Server the content of this attribute is written as server name into the registry.

Example

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

mapping (XML)

This attribute defines a pattern for nodes of an XML tree or an IDM object hierarchy. When the pattern matches a node during a transformation, the :action() method of the ***mapping*** object is called for this node.

Please refer to the description of the mapping object in the manual “XML Interface” for the syntax of the pattern.

doccursor (XML Cursor)

Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.233 .navigable

For the objects ***tablefield*** and ***edittext*** there are attributes which enable the dialog designer to influence the design and input behavior. This refers to the display of non-selectabel edittexts as well as to the definition of whether the user shall be able interact with these objects or not.

» **Usability or selectability**
Attribute.*sensitive*
Describes the quality of whether a user can select an object or not. Only if an object is selectable, the user can carry out the actions usual for the object, e.g. inputting data and scrolling.

» **Editability**
Attribute.*editable*
Describes the user's possibility to change and edit the contents of objects.

» **Focusability**
Attribute.*navigable*
Describes the possibility of including the object in the keyboard control. In this way the object receives the input focus.

Definition

Data type
boolean

Access
get, set

changed event
yes

C
Identifier: AT_navigable
Data type: DT_boolean

COBOL
Identifier: AT-navigable
Data type: DT-boolean

Classification
object-specific attribute

Objects
edittext, tablefield

These three attributes influence the object behavior as described below:

Attribute	Value	Effects
.sensitive	*true*	The user can select the object. Only if .sensitive is true, the attributes .navigable and .editable are effective, otherwise they are ignored.
.sensitive	*false*	The user cannot select the object. The object cannot be edited or changed. The object contents is displayed in grey as is usual for window systems.
.editable	*true*	The user can change the object contents, if he can select the object.
.editable	*false*	The user cannot change the object contents.
.navigable	*true*	The object is contained in the normal keyboard control, i.e. the user can get the object by navigating in the corresponding window.

Attribute	Value	Effects
.navigable	*false*	The object cannot be reached by keyboard control. It can, however, get the focus by the mouse.

Remarks on the IDM for Motif

» Modification of *.content* or *.navigable* on an ***edittext*** may influence whether the cursor (caret) is drawn solid in another edittext. It should be noted that in an edittext with *.navigable = false* no input is possible on Motif.

» The solid cursor does not allow any conclusions about navigation order or focusing of an ***edittext***.

See also

Attributes *.editable*, *.sensitive*

2.234 .navigation

This attribute defines the keyboard navigation type at the ***checkbox***. The types differ between objects that are navigated using the `Tab` key and object groups (e.g. several radiobuttons or checkboxes) where navigation happens with the cursor keys.

Definition

Data type	*Access*	*changed event*
enum	get, set	no

C	*COBOL*
Identifier: AT_navigation Data type: DT_enum	Identifier: AT-navigation Data type: DT-enum

Default value
navi_default

Classification	*Objects*
object-specific attribute	checkbox

Value range

navi_default (default)
: Objects with the same parent object and on the same level of the object hierarchy form a group. Within a group, you can navigate using the cursor keys.

navi_grouped
: The object forms a group with other objects that have *.navigation* set to *navi_grouped*. Within this group, keyboard navigation happens with the cursor keys.

navi_tab
: The object stands alone and can be accessed with the `Tab` key.

It should be ensured that all objects of a group, that means all objects within one grouping object (e.g. ***groupbox***, ***window***), have the same value.

Availability

This attribute is available since IDM versions A.04.04.o and A.05.01.e respectively.

2.235 .nextactive[integer]

This attribute defines the next active ***listbox*** item. It is necessary to specify an index for the definition of “next active listbox item”: *.nextactive[I]*. Using this index, the application can get all active items or find out whether no items, one item or several items are active.

Definition

Data type
integer

Access
get

C
Identifier: AT_nextactive
Data type: DT_integer

COBOL
Identifier: AT-nextactive
Data type: DT-integer

Classification
object-specific attribute

Objects
listbox

Example

```
dialog D

listbox Lb
{
  .multisel   true;
  .content[1] "Hello";
  .content[2] "World";
  .content[3] "!!";
  .active[1]  true;
  .active[3]  true;
}

on dialog start
{
  variable integer I := Lb.nextactive[0];

  while I <> 0 do
    print I;
    I := Lb.nextactive[I];
  endwhile

  exit();
}
```

2.236 .nextactive[index]

With *.nextactive[index]* the application is able to get all active fields or to find out how many fields are active and if there are any active fields at all.

To define the next active field, it is necessary to specify an index from where an active entry should be searched for. If no further entry is found, *[0,0]* is returned as index.

A search with the indices *[I,0]* or *[0,J]* is not possible. Since these indices do not represent real fields of the table, the query is treated as if the index *[0,0]* had been specified.

The search direction is determined by the .direction attribute. If *.direction = 1*, the next selected cell is searched for row by row and if *.direction = 2*, it is searched for column by column. 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*.

Definition

Data type
integer

Access
get

C
Identifier: AT_nextactive
Data type: DT_integer

COBOL
Identifier: AT-nextactive
Data type: DT-integer

Classification
object-specific attribute

Objects
tablefield

See also

Attributes *.active[index]*, *.direction*, *.selstyle*

2.237 .nodetype

This attribute of the ***doccursor*** queries the type of the current DOM node.

Definition

Data type	*Access*
enum	get

C	*COBOL*
Identifier: AT_nodetype Data type: DT_enum	Identifier: AT-nodetype Data type: DT-enum

Classification	*Objects*
object-specific attribute	doccursor

Value range

nodetype_attribute
: The node is an attribute of an element.

nodetype_cdata_section
: The node is a section with unparsed character data (CDATA).

nodetype_comment
: The node is a comment.

nodetype_document
: The node is a complete document.

nodetype_document_fragment
: The node is a section of a document.

nodetype_document_type
: The node is a document type declaration.

nodetype_element
: The node is an element.

nodetype_entity
: The node is an entity declaration.

nodetype_entity_reference
: The node is a reference to a declared entity.

nodetype_notation
: The node is a notation declared in the DTD.

nodetype_processing_instruction
: The node is a processing instruction.

nodetype_text
: The node is the text content of an element or an attribute value.

Note

Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.238 .notepage

This attribute queries the notepage which the object belongs to. If the object does not belong to a notepage the *null* object will be returned as result.

Definition

Data type
object

Access
get

C
Identifier: AT_notepage
Data type: DT_object

COBOL
Identifier: AT-notepage
Data type: DT-object

Classification
hierarchy attribute

2.239 .open[integer]

This attribute is used for the ***treeview*** object to define whether a subtree is to be displayed open or closed. The subtree will be opened or closed on selecting.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_open	Identifier: AT-open
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	treeview

In index *0* a new default value is to be defined. It is thus possible to indicate in the ***treeview*** definition if the subtree is completely open or completely closed. The default value is inherited when new entry is generated.

If there are still active items in the subtree on closing this subtree, they will be set to inactive and the root of the subtree will be set to active.

2.240 .opsys_string

In the ***setup*** object, this attribute requests the operation system identification.

Definition

Data type
string

Access
get

C
Identifier: AT_opsys_string
Data type: DT_string

COBOL
Identifier: AT-opsys-string
Data type: DT-string

Classification
object-specific attribute

Objects
setup

See also

Attribute *.opsys_type*

2.241 .opsys_type

In the ***setup*** object, this attribute queries the type of operating system.

Definition

Data type	*Access*
enum	get

C	*COBOL*
Identifier: AT_opsys_type Data type: DT_enum	Identifier: AT-opsys-type Data type: DT-enum

Classification	*Objects*
object-specific attribute	setup

Value range

os_nt
: The Operating System is a 32-bit version of MICROSOFT WINDOWS, based on the technology of WINDOWS NT.

os_unix
: The Operating System is a variant of UNIX or LINUX.

os_w64
: The Operating System is a 32-bit version of MICROSOFT WINDOWS.

See also

Attribute *.opsys_string*

2.242 .options[enum]

This attribute provides special settings for the respective object classes.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C
Identifier: AT_options
Data type: DT_boolean

COBOL
Identifier: AT-options
Data type: DT-boolean

Classification
object-specific attribute

.options can be addressed with IDM enums. The *enum* values are as follows:

Object	option_index	Meaning
application	*opt_cert_required* (since IDM version A.06.02.h)	Determines whether a certificate is required for an SSL connection.
	opt_no_ssl_v2 (since IDM version A.06.02.h)	Specifies that version 2 of the SSL protocol is not used for an SSL connection.
	opt_verify_peer (since IDM version A.06.02.h)	Indicates whether for an SSL connection the opposite side should be verified.
canvas	*opt_accept_child* (IDM FOR MOTIF only)	Canvas accepts child widgets (i.e. no focus).
	opt_addevents (IDM FOR QT only)	Invokes the canvas function for all *QEvents*. In this case *CCR_event* is passed as *reason*. For instance, this enables responding to mouse moves. ***false*** (default) Usual invocation of the canvas function. *true* The canvas function is called for all *QEvents*, besides the usual invocations.

Object	option_index	Meaning
	opt_center_toolhelp (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_focus_frame (IDM FOR MOTIF only)	No focus frame is drawn.
	opt_graphicsview (IDM FOR QT only)	Determiines whether the ***Canvas*** is implemented by a ***QFrame*** or a ***QGraphicsView***. Depending on the option value, there are differences in the **DM_CanvasUserArgs**. ***false*** (default) Implementation by a ***QFrame***. During the *expose* event, it can be drawn directly into the QWidget. *true* Implementation by a ***QGraphicsView***. No *expose* messages are passed to the canvas function. This mode, where the ***QGraphicsView*** is a container for a ***QGraphicsScene***, is suitable, for example, to add objects to the QGraphicsScene, which it then draws independently.
	opt_motif_shadow (IDM FOR MOTIF only)	Canvas draws Motif shadow border.
checkbox	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.

Object	option_index	Meaning
	opt_push_like (IDM FOR WINDOWS only)	The Checkbox looks like a Pushbutton.
	opt_use_widget (IDM FOR MOTIF only)	Does not use gadget, but always widget.
dialog	*opt_et_margin* (IDM FOR WINDOWS only)	This option controls whether the inner border of edittexts is drawn (default) or not.
edittext	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_html (IDM FOR QT only)	Controls whether the ***edittext*** renders HTML and displays contents formatted accordingly. ***false*** (default) No rendering of HTML and no formatting of contents. *true* HTML rendering with formatted display of contents.
	opt_rtf (IDM FOR WINDOWS only)	Controls whether the ***edittext*** accepts “Rich Text Format” (RTF) and displays formatted texts. ***false*** (default) No processing of RTF and no formatted display. *true* RTF processing with formatted display of text.
filereq	*fro_createprompt* (IDM FOR WINDOWS only, only in mode *fr_load* with *.mustexist* = *true*)	When selecting an existing file a dialog pops up for safeguarding the creation of a file.

Object	option_index	Meaning
	fro_overwriteprompt (IDM FOR WINDOWS and IDM FOR QT only, only in mode *fr_save*)	When selecting an existing file a dialog pops up to confirm overwriting the file. **Note on the IDM FOR QT** For the combination *.mustexist = true* with *.options[opt_overwriteprompt] = true*, no dialog will be displayed to confirm overwriting.
	fro_relativepath (IDM FOR MOTIF ONLY, since Motif version 2.1)	The pattern and the current path are displayed separately and the selected file is only shown as file name without directory.
groupbox	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_scroll_on_focus (IDM FOR MOTIF only, since IDM version A.05.02.i)	Controls the behavior with focusing and may improve accessibility through keyboard navigation (see chapter “Motif Option .options[opt_scroll_on_focus]”). ***true*** (default) The IDM tries to scroll the child object that shall receive the focus into the visible area. *false* The IDM does **not** try to scroll the child object that shall receive the focus into the visible area.

Object	option_index	Meaning
image	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_sim_insensitive (IDM FOR WINDOWS and IDM FOR QT only, IDM FOR QT since IDM version A.06.02.g)	Controls if a picture for the insensitive state is generated. ***false*** (default) No picture is generated. *true* A faded picture is generated for the insensitive state.
layoutbox	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_scroll_on_focus (IDM FOR MOTIF only, since IDM version A.05.02.i)	Controls the behavior with focusing and may improve accessibility through keyboard navigation (see chapter “Motif Option .options[opt_scroll_on_focus]”). ***true*** (default) The IDM tries to scroll the child object that shall receive the focus into the visible area. *false* The IDM does **not** try to scroll the child object that shall receive the focus into the visible area.

Object	option_index	Meaning
listbox	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_quick_navigate (IDM FOR MOTIF only)	Turns off the “Quick Navigate Feature”. This is a work-around for a Motif bug (since Motif 2.1) that causes beeps when the keyboard is pressed with enabled “Quick Navigate Feature”. *false* “Quick Navigate Feature” is deactivated. ***true*** (default) “Quick Navigate Feature” is activated.
	opt_scroll_pixels (IDM FOR QT only, since IDM version A.06.01.c)	Activates the “Pixel Scroll Mode”, which does not scroll by item but moves the next undisplayed area into the viewport. Thus, long entries exceeding the viewport can be scrolled stepwise. A page step equates the height of the viewport here, while a single step corresponds to the height of the first item (both measured in pixels). ***false*** (default) “Pixel Scroll Mode” is deactivated. Scrolling happens by item. *true* “Pixel Scroll Mode” is activated. Scrolling moves the next undisplayed area into the viewport.
menubox	*opt_enable_tearoff* (IDM FOR MOTIF only)	Determines if the user can detach the ***menubox*** from the menubar and freely position it.
menuitem	*opt_use_widget* (IDM FOR MOTIF only)	Does not use gadget, but always widget.

Object	option_index	Meaning
notebook	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
notepage	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_scroll_on_focus (IDM FOR MOTIF only, since IDM version A.05.02.i)	Controls the behavior with focusing and may improve accessibility through keyboard navigation (see chapter “Motif Option .options[opt_scroll_on_focus]”). ***true*** (default) The IDM tries to scroll the child object that shall receive the focus into the visible area. *false* The IDM does **not** try to scroll the child object that shall receive the focus into the visible area.
poptext	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.

Object	option_index	Meaning
	opt_hscroll (IDM FOR WINDOWS only, since IDM version A.06.01.h)	Activates a horizontal scrollbar in the ***poptext*** list. ***false*** (default) No horizontal scrollbar is activated. *true* A horizontal scrollbar is activated when the list contains items that cannot be displayed completely. Note 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.
	opt_mnemonic	Defines if Mnemonics are processed. *false* Mnemonics are ignored. A “&” is displayed as character. ***true*** (default) Mnemonics are processed. A “&” character is not shown, but interpreted as mark for a Mnemonic.
	opt_old_select	Controls the triggering of *select* and *activate* events.
	opt_quick_navigate (IDM FOR MOTIF only)	Turns off the “Quick Navigate Feature”. This is a work-around for a Motif bug (since Motif 2.1) that causes beeps when the keyboard is pressed with enabled “Quick Navigate Feature”. *false* “Quick Navigate Feature” is deactivated. ***true*** (default) “Quick Navigate Feature” is activated. **Note** “Quick Navigate” only works with *.style* = *poptext*.

Object	option_index	Meaning
	opt_sort (IDM FOR WINDOWS only)	Determines if the list is sorted. The sorting only affects the display, the order in the attribute *.text [integer]* and the indexes of the entries remain unchanged. ***false*** (default) No automatic sorting of the entries. *true* The list entries are sorted automatically. Supports the default behavior of MICROSOFT WINDOWS which assumes a sorted list.
progressbar	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
pushbutton	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_use_widget (IDM FOR MOTIF only)	Does not use gadget, but always widget.
radiobutton	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.

Object	option_index	Meaning
	opt_push_like (IDM FOR WINDOWS only)	The Radiobutton looks like a Pushbutton.
	opt_use_widget (IDM FOR MOTIF only)	Does not use gadget, but always widget.
rectangle	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
scrollbar	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
setup	*opt_balloon_toolhelp* (IDM FOR WINDOWS only, since IDM version A.05.01.c)	Defines the appearance of the toolhelp: ***true*** (default) The toolhelp is displayed as speech bubble. *false* The toolhelp is displayed as rectangle.
	opt_yi_monitoring	If this option is set to *false*, monitor functions installed through **YiRegisterUserEventMonitor ()** are not invoked. This is important for debug-ging, if there is a suspicion that a monitor func-tion is implemented faultily. When monitor functions are disabled through the command line option or environment variable, they cannot be enabled by setting this option to *true*.

Object	option_index	Meaning
	opt_fontraster_compat	This option can be used to specify that the old calculation of the raster size should be used. **Attention:** with version A.06.03.a the calculation of the raster size has changed. If no reference string is specified, ***M*** is used as reference string now. The option *opt_fontraster_compat* can be used to temporarily use the old calculation method. Using this option the size calculation is partly based on the system font, which is not High DPI capable, so High DPI capable applications created with IDM FOR WINDOWS 11 should not use this option. See also “Options for raster calculation under Windows”
spinbox	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
splitbox	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.

Object	option_index	Meaning
statictext	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_use_widget (IDM FOR MOTIF only)	Does not use gadget, but always widget.
tablefield	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_new_align (IDM FOR MOTIF and IDM FOR WINDOWS)	Defines, which attributes determine the alignment of text. ***false*** (default) Text alignment is determined by the attributes *.colalignment[integer]* and *.rowalignment[integer]*. *true* Text alignment is determined by the attributes *.xalignment[index]* and *.yalignment [index]*.

Object	option_index	Meaning
	opt_new_colwidth (IDM FOR MOTIF only)	In older versions of the IDM for Motif (before A.03.10.f) for tables with *.reffont* defined, the column widths had been calculated incorrectly. For this there was a **second** patch in version A.03.10.f where the option *opt_new_colwidth* was introduced. ***true*** (default) Corrected column widths (calculated on the basis of grid units as with the layout of other objects). *false* Column widths are calculated according to the option *opt_old_colwidth*.
	opt_old_colwidth (IDM FOR MOTIF only)	In older versions of the IDM for Motif (before A.03.04.a) for tables with *.reffont* defined the columns had been too wide. For this there was a **first** patch in version A.03.04.a where the option *opt_old_colwidth* was introduced. ***false*** (default) Corrected, smaller column widths. *true* Column widths as before the correction. **Note** In version A.03.10.f the column widths were corrected again and the option *opt_new_colwidth* had been introduced. If *opt_new_colwidth* is set to *true*, this setting takes precedence over *opt_old_colwidth*.

Object	option_index	Meaning
	opt_old_select (IDM FOR MOTIF only, since IDM version A.05.02.h)	With this option a selection- or action-oriented triggering of *select* events can be set. This enables consistent behavior of the Motif and Windows versions of the IDM. *true* *select* events are triggered when the activation state changes (selection-based). This corresponds to the behavior up to and including IDM version A.05.02.g. ***false*** (default) *select* events are triggered when a mouse button or the selection key is pressed (action-based). This corresponds to the behavior on Microsoft Windows. Consequently *select* events do not indicate a change of the activation state anymore. In order to respond to changes of the selection with *.options [opt_old_select] = false*, the *activate* and *deactivate* events can be used.
	opt_xmborder_compat (IDM FOR MOTIF only, since IDM version A.06.01.a)	Compatibility option to draw the ***tablefield*** again with a *1* pixel wide border. Since IDM version A.06.01.a the default width for the border is *0*. This could also be achieved by setting *.borderwidth = 1*, but then a warning message would appear because *.borderwidth* will no longer be supported for the ***tablefield*** in the future.
toolbar	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.

Object	option_index	Meaning
treeview	*opt_center_toolhelp* (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_rightclick_selects (IDM FOR WINDOWS only)	Determines if entries can be selected with the right mouse button: ***false*** (default) No selection of entries with the right mouse button. *true* Entries can be selected with the right mouse button.
	opt_scroll_on_focus (IDM FOR MOTIF only, since IDM version A.05.02.i)	Controls the behavior with focusing and may improve accessibility through keyboard navigation (see chapter “Motif Option .options[opt_scroll_on_focus]”). ***true*** (default) The IDM tries to scroll the child object that shall receive the focus into the visible area. *false* The IDM does **not** try to scroll the child object that shall receive the focus into the visible area. **Note** A ***treeview*** with *.options[opt_scroll_on_focus]* = *false* may become unreachable by keyboard navigation when the active entry is outside the visible area. However it remains focusable by mouse click.

Object	option_index	Meaning
window	*opt_animated* (IDM FOR QT only)	Toggles animations when ***toolbars*** are moved interactively. ***false*** (default) No animations when ***toolbars*** are moved. *true* Animated, interactive moving of ***toolbars***. However, this leads to significantly more *resize* and *move* events.
	opt_auto_close (IDM FOR WINDOWS and IDM FOR QT only, IDM FOR QT since IDM version A.06.01.e)	This option prevents the window from closing automatically when the user selects the *"Close"* command from the system menu or the *"Close"* button in the title bar. ***true*** (default) Window closes automatically when *"Close"* is selected. *false* Window is not closed when selecting *"Close"*, but must be explicitly closed by the application. A behavior similar to the value *true* can be simulated by the rule on `WINDOW close { this.visivle: = false; }`.
	opt_center_toolhelp (IDM FOR WINDOWS only)	Controls where the toolhelp is displayed: ***false*** (default) The toolhelp is displayed at the position of the mouse pointer. *true* The toolhelp is displayed centered below the object, provided there is enough space.
	opt_help (IDM FOR WINDOWS only)	When this option is set, the help button with the question mark is displayed in the titlebar. **Note** The help button is not shown when all three other buttons (minimize, maximize, close) are displayed.

Object	option_index	Meaning
	opt_nested_docks (IDM FOR QT only)	Enables “Nested Docks”, i.e. the arrangement of ***toolbars*** with *.style = notepage* in multi-row docks. ***false*** (default) Only single-row docks are possible. *true* Adjoined, multi-row docks are possible. However, this leads to a not so clear operation in interactions and shifts. The *.dock_line* attribute is observed. “Nesting” into the same dock is always relative to the previous toolbar in the same docking area. **Notes** » The option **only** affects ***toolbars*** with *.style = notepage*. » If both *.options[opt_nested_docks] = false* and *.options[opt_tabbed_docks] = false* (default), the *.dock_line* attribute of ***toolbars*** with *.style = notepage* is ignored and all those toolbars are arranged side by side (horizontal) or one on top of another (vertical).
	opt_scroll_on_focus (IDM FOR MOTIF only, since IDM version A.05.02.i)	Controls the behavior with focusing and may improve accessibility through keyboard navigation (see chapter “Motif Option .options[opt_scroll_on_focus]”). ***true*** (default) The IDM tries to scroll the child object that shall receive the focus into the visible area. *false* The IDM does **not** try to scroll the child object that shall receive the focus into the visible area.

Object	option_index	Meaning
	opt_tabbed_docks (IDM FOR QT only)	Enables “Tabbed Docks”, i.e. the arrangement of ***toolbars*** with *.style = notepage* as tabs in one dock. ***false*** (default) “Tabbed Docks” are not possible. *true* “Tabbed Docks” sharing space are possible. ***toolbars*** with the same *.dock_line* are displayed as stacked pages with tabs. “Tabbed Docks” are similar to notebooks and notepages. **Notes** » The option **only** affects ***toolbars*** with *.style = notepage*. » If both *.options[opt_nested_docks] = false* and *.options[opt_tabbed_docks] = false* (default), the *.dock_line* attribute of ***toolbars*** with *.style = notepage* is ignored and all those toolbars are arranged side by side (horizontal) or one on top of another (vertical).
	opt_window_size (IDM FOR QT only)	Determines how sizes are interpreted for the ***window***. ***false*** (default) Size specifications (*.width*, *.height*, minimum and maximum values) refer to the client area of the window. *true* Size specifications refer to the entire window, including menubar, toolbars, tabbed widgets and statusbar, but without decoration (titlebar, borders).

Example

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* has been made available.

```
Canvas1.options[opt_accept_child] := true;
```

See also

Object canvas

2.242.1 Motif Option .options[opt_scroll_on_focus]

Objects with virtual size (***groupbox***, ***layoutbox***, ***notepage***, ***window***) and ***treeview*** have the option *.options[opt_scroll_on_focus]* since IDM version A.05.02.i. This option is used to define whether a child object is scrolled into the visible area when it gets focused.

Purpose of the Option

The *option .options[opt_scroll_on_focus]* primarily serves to improve the accessibility of objects via keyboard navigation and to enable that they are focused in certain constellations via keyboard or mouse click. This achieves a more consistent behavior between Motif and Windows applications.

However, the option – due to the differences between platforms in focus handling – cannot achieve full consistency, nor that objects can be focused by key command in all situations.

Relation Between Object Position, Visibility, Keyboard Navigation and Focusability on Motif

On Motif, only objects that are visible or that can be scrolled into the visible area may obtain focus. This is the typical behavior of Motif applications, unlike MICROSOFT WINDOWS. Please note that a child object may be fully visible in a grouping object, but is still not accessible and reachable via keyboard, as the grouping object is not fully visible.

For compatibility reasons between the platforms, the ISA Dialog Manager also enables the positioning of child objects with negative positions for the x and y coordinates of the left upper corner in the non-visible areas on Motif. Motif does not actually allow this, so that the IDM must bypass check mechanisms. Consequentially, this usually rules out that the objects are reachable via keyboard navigation and that they can be focused.

Recommendations

- The best way to ensure the accessibility of objects via keyboard navigation and their ability to be focused with keyboard and mouse is to always position them fully in the visible area and to avoid negative values for their x and y coordinates.
- If this is not possible, you can attempt to improve key navigation and focusability with the help of *.options[opt_scroll_on_focus]*.

2.242.2 Options for Grouping Objects under Windows

Attention

The options *opt_wntsizebug_compat* and *opt_w2kprefsize_compat* are deprecated. Since IDM version A.06.01.a they are not evaluated anymore.

It is strongly recommended to adapt dialogs that still rely on these options. Only if adaptation is not feasible, the command line option **-IDMborder5_compat** or the environment variable `IDM_BORDER5_COMPAT` may be used to bring back evaluation of *opt_wntsizebug_compat* and *opt_w2kprefsize_compat* (with the downside of losing support for the *.borderstyle* attribute).

With the change to “Visual Styles” on Windows XP and higher, several bugs in size calculation (causing a different behavior) were fixed. These bugs had existed since 3D objects had been supported on Windows.

Especially in dialogs which require pixel-perfect representation these problems had major impacts. Therefore the version of the IDM for Windows XP and higher have been made “error compatible”. For all grouping objects (***groupbox***, ***layoutbox***, ***notebook***, ***notepage***, ***spinbox***, ***splitbox***, ***statusbar***, ***toolbar*** and ***window***) two additional options have been introduced:

.options[opt_wntsizebug_compat]

Default value: *false*

With the value *false* positions and sizes of all direct child objects are set in a correct manner. With the value *true* positions and sizes of all direct child objects are set in an error compatible mode. Since Windows versions dating back a long time there had been an error calculating positions and sizes of the objects ***edittext***, ***groupbox***, ***listbox***, ***poptext*** and ***treeview***. This error resulted in these objects being one pixel smaller than desired on each side.

This option is preset to the value *false*.

.options[opt_w2kprefsize_compat]

Default value: *false*

With the value *true* the preferred sizes of all direct child objects are calculated as if the application is running on Windows 2000. With the value *false* the preferred sizes of all direct child objects are calculated according to the “Visual Styles” in the Windows XP and higher versions of the IDM.

This option is preset to the value *false*. Therefore objects with no dimensions set appear in optimal dimensions with active “Visual Styles”.

If the appearance of a dialog is broken, this option can be turned on for the Default objects. Afterward the dialog can be adjusted successively.

2.242.3 Options for raster calculation under Windows

With IDM version A.06.03.a, the calculation of the raster width has been substantially changed. If no reference string is set, the raster width is now calculated from an internal reference string (“M”) to avoid excessive width growth due to excessively wide letters within a font.

For compatibility reasons, however, the *opt_fontraster_compat* option, the **-IDMfontraster_compat** startup option, or the `IDM_FONTRASTER_COMPAT` environment variable can temporarily reactivate the old raster width calculation (with the drawback that excessive width growth may occur again).
When using the *opt_fontraster_compat*, the size calculation is partly based on the system font, which is not High DPI capable, so High DPI capable applications created with IDM FOR WINDOWS 11 should not use this option.

2.243 .order

With the help of this attribute the order of menus and menu commands can be defined. The menus or menu commands are sorted in order of increasing *.order* values in menuboxes from top to bottom and in windows from left to right. For equal values, the definition order decides.

The attribute is only supported by the IDM FÜR WINDOWS.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_order	Identifier: AT-order
Data type: DT_integer	Data type: DT-integer

Default value
0

Classification	*Objects*
object-specific attribute	menubox, menuitem, menusep

When grouping menuitems of the radiobutton style, if an active item changes the group by changing the *.order* value, this item remains active and the item previously active in this group is deactivated. This is also to be considered when creating menus statically.

It's a good idea to stick to the MICROSOFT conventions for mixed menu bars when assigning *.order* values. With the **InPLace** representation of **ActiveX** controls, mixed menu bars are composed of the container and control menus. The conventions provide fixed values for certain groups of menuboxes, namely:

File	*0*
Edit	*1*
Container	*2*
Object	*3*
Window	*4*
Help	*5*

The groups *0*, *2* and *4* are taken over from the container and groups *1*, *3* and *5* from the **ActiveX** control into the common menu bar.

Although the IDM does not support mixed menu bars, the MICROSOFT conventions should be taken into account when using the **InPlace** representation of **ActiveX** controls .

Note

It is quite common for a menu group to consist of several menus, e.g. the menu group “Edit” may consist of “Edit” and “Search” (with TEXTPAD).

2.244 .output[integer]

This attribute is used with functions and rules to query whether the parameter is an output parameter.

Definition

Data type
boolean

Access
get

C
Identifier: AT_output
Data type: DT_boolean

COBOL
Identifier: AT-output
Data type: DT-boolean

Classification
object-specific attribute

Objects
function, ***rule***

Example

```
function Test (integer P input, integer Q output,
               string R input output)

for I := 1 to Test.count do
  print Test.output[I];
endfor
```

Output

```
false
true
true
```

See also

Chapters “Functions” and “Named Rules (Subprograms)” in manual “Rule Language”

2.245 .overridecursor

This attribute can set a temporary cursor globally for all loaded ***dialogs***.

Definition

Data type	*Access*	*changed event*
object [***cursor***]	get, set	no

C	*COBOL*
Identifier: AT_overridecursor Data type: DT_cursor	Identifier: AT-overridecursor Data type: DT-cursor

Classification	*Objects*
object-specific attribute	setup

Note

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

Example

When a button is pressed, the wait cursor should be set for all open windows of the current dialogs, except for the window in which the button is located. This is turned into a dialogbox to lock input into other windows.

```
on PbCompute select
{
  this.window.dialogbox    := true;
  this.window.ignorecursor := true;
  setup.overridecursor     := CursorBusy;
  Compute();
  setup.overridecursor     := null;
  this.window.ignorecursor := false;
  this.window.dialogbox    := false;
}
```

See also

Attribute *.ignorecursor*

Object messagebox

2.246 .pagemotion

.pagemotion specifies the pixel number at which the scrollbar value changes if the user wants to scroll by pages.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_pagemotion	Identifier: AT-pagemotion
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	scrollbar

2.247 .parent

This attribute queries and changes the parent of an object.

Definition

Data type
object

Access
get, set

C
Identifier: AT_parent
Data type: DT_object

COBOL
Identifier: AT-parent
Data type: DT-object

Classification
hierarchy attribute

See also

Method :parent_of()

2.248 .password

This attribute can be used to define the password for starting the application side with the RSH or SSH protocol.

The attribute is evaluated if no password is given with the command in the *.exec* attribute. This attribute provides an alternative when there are problems to properly construct the command in the *.exec* attribute.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C
Identifier: AT_password
Data type: DT_string

COBOL
Identifier: AT-password
Data type: DT-string

Inheritance
yes

Classification	*Objects*
object-specific attribute	application

Availability

Since IDM version A.06.02.g

See also

Attribute .username

2.249 .path

Returns 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 that node in the DOM tree.

Definition

Data type	*Access*
string	get

C	*COBOL*
Identifier: AT_path	Identifier: AT-path
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	doccursor

If the value of the *.path* attribute is stored elsewhere (e.g. in the *.userdata* attribute), it is important to know that the stored value will not be adjusted when the structure of the DOM tree changes. When the **:select()** method is called with the stored value after this, the ***XML Cursor*** will end up pointing to an incorrect DOM node.

2.250 .pattern

This attribute defines a pattern for the choices of a ***filereq*** object. Only directories and files matching this pattern are displayed. If "" is given, all files and directories (Motif *, Windows *.*) are listed.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_pattern	Identifier: AT-pattern
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	filereq

Particularities of Motif

On Motif the pattern is a regular expression that can be changed by the user. The functional range is determined by Motif, therefore just the most important expressions are given below.

Expression	Meaning
*	none or any number of arbitrary characters
?	one arbitrary character
[a-z]	one character from the range of a-z, e.g. a lowercase letter
[0123]	one character from the enumeration 0, 1, 2, 3

Particularities of Microsoft Windows

The pattern is only effective in the modes*fr_load* and *fr_save*. Microsoft Windows expects a list of type labels with file extensions. The user only sees the type labels as choices and cannot enter different file extensions. Type labels and file extensions are separated by tabulator characters `"\t"`. Multiple file extensions for one type label are separated by semicolons `";"`.

Example

```
.pattern "IDM files\t*.idm;*.dlg;*.mod\tDocuments          ...\
(*.doc)\t*.doc;\tAll files (*.*)\t*.*";
```

2.251 .picheight

For ***listbox*** and ***poptext***, this attribute defines the height of the images shown at each list item. For ***listview*** it defines the height of the large icons.

For the ***notebook***, the attribute defines the height of the pictures shown in the tabs of the individual ***notepages***.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_picheight	Identifier: AT-picheight
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listbox, listview, notebook, poptext

listbox, poptext

The attribute defines the height of the space to the left of the entries where the pictures (***tile*** resources) are displayed that have been assigned to the individual entries in the attributes *.picture [integer]* and *.picture_hilite[integer]*.

Higher or lower pictures are centered vertically within this space. For ***tile*** resources with *.scale = true* their height is scaled to *.picheight*.

listview

This attribute defines the height of the large icons.

Value range

0	The width *.picwidth* is used. If this is also *0*, the size of the icons is determined by the system.
> 0	Height of the large icons.

The large icons will be shown in the icon view (*.style = "icon"*, *.style = "picture"*) and in the tile view (*.style = "tile"*).

For invalid values, the default value *0* is used. However, the attribute value is not changed.

Note

Changing the attribute in the visible state may cause the object to flicker.

notebook

The attribute defines the height of the pictures displayed in the tabs. The ***tile*** resources to be displayed are assigned to the ***notepages*** in their *.picture* attribute.

The value range is *.picheight >= 0*. For *.picheight = 0* the height is determined by the system and is *16* pixels.

The height of the pictures is scaled to *.picheight* regardless of the *.scale* attribute on the ***tile*** resource.

The attribute is only available on MICROSOFT WINDOWS.

2.252 .picture

The *.picture* attribute defines the picture (***tile*** resource) that is displayed in an ***image*** object with *.style = 0* (*pushbutton*, default value).

At the ***notepage***, the attribute defines the picture that is displayed next to the label in the ***notebook*** tab belonging to the ***notepage***.

For the ***control*** object this attribute defines the picture that is displayed in the inactive state of the object.

Definition

Data type	*Access*	*changed event*
object [***tile***]	get, set	yes

C	*COBOL*
Identifier: AT_picture	Identifier: AT-picture
Data type: DT_tile	Data type: DT-tile

Classification	*Objects*
object-specific attribute	control, image, notepage

image

Setting the attribute *.mouseover* to *true* enables the ***image*** object to show other pictures when the mouse pointer is moved over the object or the left mouse button is pressed over it. These pictures are defined in the *.picture[enum]* attribute with the indexes *tile_mouse_over* and *tile_active_mouse_over*.

With *.style <> 0*, the pictures displayed in different situations are defined in the indexed *.picture[enum]* attribute as well.

Instead of a ***tile*** resource, the *.picture* attribute can also be assigned a string with the file path of a graphic file. However, this feature may be dropped in future IDM versions. Therefore it is recommended to use a ***tile*** resource with an external graphic file.

Note for the IDM for Qt

Specifying a file path (data type *string*) is **not** supported.

See also

Attributes *.mouseover*, *.picture[enum]*, **style**

Note for the notepage

The size of the picture is set with the attributes *.picheight* and *.picwidth* of the ***notebook*** for all contained notepages.

See also

Resource tile

2.253 .picture[enum]

This attribute can be used to define different images that are displayed in the object depending on its style, activation state and situation.

Definition

Data type	*Access*	*changed event*
object [***tile***]	get, set	yes

C	*COBOL*
Identifier: AT_picture Data type: DT_tile	Identifier: AT-picture Data type: DT-tile

Classification	*Objects*
object-specific attribute	image, menuitem

Index Range

tile_default
: ***tile*** resource that is displayed in the “normal” state of the object.

tile_active
: ***tile*** resource that is displayed when the object is activated.

tile_insensitive (***image*** only)
: ***tile*** resource that is displayed when the object is not operable (insensitive).

tile_mouse_over (***image*** only)
: ***tile*** resource that is displayed when the mouse pointer is over the object and the left mouse button is not pressed and the object is not activated.
The attribute *.mouseover* has to be *true* for this picture to be displayed.

tile_active_mouse_over (***image*** only)
: ***tile*** resource that is displayed when the mouse pointer is over the object and the left mouse button is pressed or the object is activated.
The attribute *.mouseover* has to be *true* for this picture to be displayed.

image

For the ***image*** object, the *.picture[enum]* attribute specifies the pictures (***tile*** resources) that are displayed in different situations. The picture to be displayed depends on several factors that are described in the table below.

Apart from the exceptions listed in the table, no other picture is displayed if the picture for a certain state is missing. Therefore, all required pictures (***tile*** resources) need to be defined.

Index	To Be Displayed	Required
tile_default	In normal state, when none of the following situations applies.	Always.

Index	To Be Displayed	Required
tile_active	With *.style = checkbox*, if » the image object is active. » the image object is not active and the left mouse button is pressed over the object. With *.style = pushbutton*, if the left mouse button is pressed over the object. **Exception:** In this case *tile_default* is displayed if *tile_active* is not set. With *.style = menubox* temporary and independent of *.mouseover*, if the context menu is open and the image object is not the actual menu object. *tile_active* will not be displayed if *.mouseover = true* and the mouse pointer is over the image object.	In the styles *checkbox* and *menubox*.
tile_insensitive	If the image object is insensitive (*.real_sensitive = false*). **Exception:** If *tile_insensitive* is not set, then *tile_default* or *tile_active* (with *.style = checkbox* and activated checkbox) will be displayed.	Optional in all styles.
tile_mouse_over	If *.mouseover = true* and the mouse pointer is over the image object and » the left mouse button is not pressed. » with *.style = checkbox* the checkbox is not activated. With *.style = menubox* independent of *.mouseover*, if the context menu is closed and the image object is the actual menu object. In this state, the border is highlighted.	With *.style = menubox*. Otherwise only if *.mouseover = true*.

Index	To Be Displayed	Required
tile_active_mouse_over	If *.mouseover* = *true* and the mouse pointer is over the image object and » the left mouse button is pressed. » with *.style* = *checkbox* the checkbox is activated. With *.style* = *menubox* independent of *.mouseover*, if the context menu is open and the image object is the actual menu object.	With *.style* = *menubox*. With *.style* = *checkbox* if *.mouseover* = *true*.

See also

Attributes *.mouseover*, *.picture*

menuitem

The attribute *.picture[enum]* of the ***menuitem*** can contain different pictures, which are displayed as symbols at a menu entry depending on its *.style* attribute and activation state. The attribute is indexed with an enumeration type, *tile_default* and *tile_active* are permitted as index values.

If the ***menuitem*** has the *radiobutton* or *checkbox* style, the picture from *tile_active* is displayed when the radiobutton or checkbox is activated. If they are not activated, the picture from *tile_default* is displayed. The IDM automatically switches between both pictures.

With a “usual” menu entry – i.e. with *.style* = *pushbutton* – always the picture from *tile_default* is displayed.

See also

Attribute *.style*

2.254 .picture[integer]

The attribute defines the images (***tile*** resources) that are displayed to the left of each entry.

At the ***listview***, the attribute defines the large icons for the list items.

Definition

Data type	*Access*	*changed event*
object [***tile***]	get, set	yes

C	*COBOL*
Identifier: AT_picture	Identifier: AT-picture
Data type: DT_tile	Data type: DT-tile

Classification	*Objects*
object-specific attribute	listbox, listview, poptext, treeview

listbox, poptext

The value range for the index is *1 … .itemcount*. The respective item *.content[l]* in the ***listbox*** or *.text [l]* in the ***poptext*** must already exist before the attribute *.picture[l]* with the corresponding index may be accessed.

For selected entries the image defined in *.picture_hilite[integer]* is displayed – when available (**not** supported by the IDM FOR MOTIF).

The size of the space provided for the images to the left of the entries is determined by the *.picwidth* and *.picheight* attributes.

At the ***listbox*** the attribute is **not** inherited.

With a ***poptext*** with *.style* = *edittext* or *.style* = *listbox*, **no** picture is displayed in the input field.

Note for the IDM for Qt

If in ***listbox*** and ***poptext*** a ***tile*** is only set for *.picture_hilite[l]*, but not for *.picture[l]*, then Qt will display the ***tile*** defined in *.picture_hilite[l]* (or a slight modification of it) even in the unselected state. Qt always tries to provide images for all states for the list entries. If the image is missing for a state, Qt generates it from the images provided.

listview

The attribute defines the large icon for each list item.

The value range of the index is *0 … .rowcount*t, where the value with index *0* is used as default value for not set values in the range *1 … .rowcount*.

Value range

null The displayed icon is derived as follows:

- If *.picture[0]* <> *null*, the icon defined there is used.
- If *.picture[0]* = *null*, then the small icon is shown enlarged (*.smallpicture[I]* or *.small-picture[0]*, depending on presence).
- If there is neither a large nor a small icon, the item is displayed without an icon.

tile Resource that contains the large icon.

The large icon will be shown in the icon view (*.style* = *"icon"*, *.style* = *"picture"*) and in the tile view (*.style* = *"tile"*).

Large and small icons belong together and should represent the same information. Both icons can only be referenced together. Therefore, different combinations should be avoided, as each combination will consume additional memory.

Note

Changing the attribute in the visible state may cause the object to flicker.

treeview

The value range for the index is *0 … .itemcount*. The ***tile*** resource in *.picture[0]* on the one hand defines the size of the images, on the other hand it is displayed for all items where *.picture[I]* with *1* <= *I* <= *.itemcount* is not set (*null*). All other images are scaled to the size of *.picture[0]* regardless of the *.scale* attribute on the ***tile*** resource.

If *.picture[0]* is not set (*null*), the size of the pictures is determined by the system and is *16x16* pixels. A substitute icon (white cross on red background) will then be displayed for items without a picture.

Before the attribute *.picture[I]* with *1* <= *I* <= *.itemcount* may be accessed, the item *.content[I]* with the respective index must already exist.

Space for pictures to the left of the items is only provided if *.picture[I]* is set for at least one index.

The attribute is **not** inherited.

The attribute is **not** supported by the IDM FOR MOTIF.

Note for the IDM for Windows

When using a “Windows Icon Resource” in *.picture[0]*, the pictures always have the default size of *16x16* pixels.

2.255 .picture_hilite[integer]

The attribute defines the pictures (***tile*** resources) that are shown next to selected entries.

The value range for the index is *1 … .itemcount*. The respective item *.content[l]* in the ***listbox*** or *.text [l]* in the ***poptext*** must already exist before the attribute *.picture_hilite[l]* with the corresponding index may be accessed.

For not selected entries the image defined in *.picture[integer]* is displayed – when available.

Definition

Data type	*Access*	*changed event*
object [***tile***]	get, set	yes

C	*COBOL*
Identifier: AT_picture_hilite	Identifier: AT-picture-hilite
Data type: DT_tile	Data type: DT-tile

Classification	*Objects*
object-specific attribute	listbox, poptext

The size of the space provided for the images to the left of the entries is determined by the *.picwidth* and *.picheight* attributes.

At the ***listbox*** the attribute is **not** inherited.

With a ***poptext*** with *.style* = *edittext* or *.style* = *listbox*, **no** picture is displayed in the input field.

The attribute is **not** supported by the IDM FOR MOTIF.

Note for the IDM for Qt

If in ***listbox*** and ***poptext*** a ***tile*** is only set for *.picture_hilite[l]*, but not for *.picture[l]*, then Qt will display the ***tile*** defined in *.picture_hilite[l]* (or a slight modification of it) even in the unselected state. Qt always tries to provide images for all states for the list entries. If the image is missing for a state, Qt generates it from the images provided.

2.256 .picwidth

For ***listbox*** and ***poptext***, this attribute defines the width of the images shown at each list item. For ***listview*** it defines the width of the large icons.

For the ***notebook***, the attribute defines the width of the pictures shown in the tabs of the individual ***notepages***.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_picwidth	Identifier: AT-picwidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listbox, listview, notebook, poptext

listbox, poptext

The attribute defines the width of the space to the left of the entries where the pictures (***tile*** resources) are displayed that have been assigned to the individual entries in the attributes *.picture[integer]* and *.picture_hilite[integer]*.

Wider or narrower pictures are centered horizontally within this space. For ***tile*** resources with *.scale = true* their width is scaled to *.picwidth*.

listview

This attribute defines the width of the large icons.

Value range

0	The width of the icons is determined by the system.
> 0	Width of the large icons.

The large icons will be shown in the icon view (*.style = "icon"*, *.style = "picture"*) and in the tile view (*.style = "tile"*).

For invalid values, the default value *0* is used. However, the attribute value is not changed.

If *.picheight = 0* then *.picwidth* also determines the **height** of the large icons.

Note

Changing the attribute in the visible state may cause the object to flicker.

notebook

The attribute defines the width of the pictures displayed in the tabs. The ***tile*** resources to be displayed are assigned to the ***notepages*** in their *.picture* attribute.

The value range is *.picwidth >= 0*. For *.picwidth = 0* the width is determined by the system and is *16* pixels.

The width of the pictures is scaled to *.picwidth* regardless of the *.scale* attribute on the ***tile*** resource.

The attribute is only available on MICROSOFT WINDOWS.

2.257 .pointer_height

With this attribute of the ***setup*** object the maximum height of the mouse cursor in pixels can be queried.

In multiscreen systems (IDM for Motif only) the attribute returns the value for the default screen.

Definition

Data type
integer

Access
get

C
Identifier: AT_pointer_height
Data type: DT_integer

COBOL
Identifier: AT-pointer-height
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.258 .pointer_height[integer]

With this attribute of the ***setup*** object the maximum height of the mouse cursor in pixels on screen I can be queried.

The indexed attribute is only available with multiscreen dialogs. The index range is *1 … setup.screen-count*.

Definition

Data type
integer

Access
get

C
Identifier: AT_pointer_height
Data type: DT_integer

COBOL
Identifier: AT-pointer-height
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.259 .pointer_width

With this attribute of the ***setup*** object the maximum width of the mouse cursor in pixels can be queried.

In multiscreen systems (IDM for Motif only) the attribute returns the value for the default screen.

Definition

Data type
integer

Access
get

C
Identifier: AT_pointer_width
Data type: DT_integer

COBOL
Identifier: AT-pointer-width
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.260 .pointer_width[integer]

With this attribute of the ***setup*** object the maximum width of the mouse cursor in pixels on screen I can be queried.

The indexed attribute is only available with multiscreen dialogs. The index range is *1 … setup.screen-count*.

Definition

Data type
integer

Access
get

C
Identifier: AT_pointer_width
Data type: DT_integer

COBOL
Identifier: AT-pointer-width
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.261 .posraster

This attribute determines the interpretation of the attribute values for the position of an object (*.xleft*, *.xright*, *.ytop*, *.ybottom*). It can be used to specify the position of an object depending on the used grid or exactly in pixels.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_posraster
Data type: DT_boolean

COBOL
Identifier: AT-posraster
Data type: DT-boolean

Classification
geometry attribute

Value range

true
: The values of the position attributes are interpreted as grid units. The underlying grid is the grid of the parent object.

false
: The values of the position attributes are interpreted as pixel units. The grid of the parent object is ignored.

See also

Attribute *.sizeraster*

2.262 .preedit

This attribute of the ***window*** object controls the display and selection of the input mode for the ***edit-texts*** within the window.

Input mode display and selection support the pre-editing of symbolic (ideographic) characters that are composed of several characters of a phonetic alphabet, as is the case with many Asian languages.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_preedit	Identifier: AT-preedit
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	window

Value range

preedit_root(default)
: Input mode display only when you are inside the edittext

preedit_overthespot
: Permanent input mode display at the bottom of the window

Note
: This will make the window larger.

preedit_offthespot (unsupported)
: Not supported by IDM.

preedit_onthespot (unsupported)
: Not supported by IDM.

Availability

Only IDM FOR MOTIF with Unicode support.

2.263 .preeditsel

The *.preeditsel* attribute controls the initial selection of the text, if the edittext of a ***tablefield*** is explicitly or implicitly activated. An implicit activation of the edittext takes place by entering a character, explicit activation for example takes place by double clicking.

With the aid of *.preeditsel* the same initial text selection can be set for both activation forms and the same behavior can be achieved. Entry behavior refers to the replacement of existing text through entry, as well as to inserting entered characters before or after the content of the edittext.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_preeditsel	Identifier: AT-preeditsel
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	tablefield

Value range

presel_default(default)
With explicit activation, an entered character replaces the content.
With implicit activation, an entered character is appended to the content.

presel_all
For explicit as well as implicit activation, the content is always entirely selected – from *.startsel = 0* to *.endsel = length(content)*.
A character entered replaces the content.

presel_begin
No selection, the cursor is set at the beginning for explicit as well as implicit activation.
A character entered is placed before the content.

presel_end
No selection, the cursor is set at the end for explicit as well as implicit activation.
A character entered is appended to the content.

Availability

From IDM version A.05.02.h the attribute is supported on UNIX platforms with MOTIF and on MICROSOFT WINDOWS.

2.264 .privatekeyfile

This attribute defines the file with the private key used for the SSH protocol.

By default, the file **private.pem** from the installation directory of the application is used.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C
Identifier: AT_privatekeyfile
Data type: DT_string

COBOL
Identifier: AT-privatekeyfile
Data type: DT-string

Inheritance
yes

Classification	*Objects*
object-specific attribute	application

Availability

Since IDM version A.06.02.h

See also

Attribute .publickeyfile

2.265 .propscale

The attribute controls whether the horizontal and vertical raster should be set proportionally to the maximum value, which is determined by the value calculation of xraster and yraster of the font raster.

The exact calculation of the raster is described in the chapter „Berechnung der Rastergröße aus einem ReferenzfontCalculating the Grid Size from a Reference Font“ of the font resource.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_propscale	Identifier: AT-propscale
Data type: DT_boolean	Data type: DT-boolean

Default value
false

Classification	*Objects*
object-specific attribute	font

Value range

true
: Width and height of the font raster are determined to the same value. The value results from the maximum of the previous value calculation of xraster and yraster.

false
: The width and height of the font grid are not adjusted and retain the values determined from the font.

Availability

Since IDM version A.06.03.a

See also

Chapter „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

2.266 .publicid

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

Definition

Data type
string

Access
get

C
Identifier: AT_publicid
Data type: DT_string

COBOL
Identifier: AT-publicid
Data type: DT-string

Classification
object-specific attribute

Objects
doccursor

The attribute is available for the XML Cursor. Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

See also

Attribute *.nodetype*

2.267 .publickeyfile

This attribute defines the file with the public key used for the SSH protocol.

By default, the file **public.pem** from the installation directory of the application is used.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_publickeyfile Data type: DT_string	Identifier: AT-publickeyfile Data type: DT-string

Inheritance
yes

Classification	*Objects*
object-specific attribute	application

Availability

Since IDM version A.06.02.h

See also

Attribute .privatekeyfile

2.268 .real_height

This attribute queries the real object height in pixel. The object has to be visible (*.real_visible* = *true*).

The height can be queried for objects without height definition.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_height
Data type: DT_integer

COBOL
Identifier: AT-real-height
Data type: DT-integer

Classification
geometry attribute

Remark

If there is no width and / or height given when defining objects, you can use the attributes *.real_width* and *.real_height* to query the width and the height of those objects.

These attributes can query the width of an object whose geometry has been defined depending on the right and left border of a parent (*.xauto* = *0*).

The dimensions returned are always in pixels!

See also

Attributes *.height*, *.real_width*

2.269 .real_modified

This attribute indicates whether the content (*.content*) of an ***edittext*** or a ***poptext*** has been modified since the object has received the focus.

With the ***datetime*** object, this attribute indicates whether the attribute *.value* has been modified since the object received the focus.

The attribute can be queried for visible objects (*.real_visible* = *true*) only.

With the ***edittext*** and the ***datetime*** object, this attribute is set to *false* when a value is assigned to the *.content* attribute.

The ***poptext*** supports this attribute with the styles *.style* = *edittext* and *.style* = *listbox* only.

Definition

Data type
boolean

Access
get

C
Identifier: AT_real_modified
Data type: DT_boolean

COBOL
Identifier: AT-real-modified
Data type: DT-boolean

Classification
object-specific attribute

Objects
datetime, edittext, poptext

Example

Check if the content has been changed through a *charinput*:

```
edittext Et
{
  on charinput
  {
    if this.real_modified then
      print "Content of " + this + " has changed";
    endif
  }
}
```

Note on the IDM for Motif

The attribute *.real_modified* is not supported and always returns *true*.

2.270 .real_screen

With this attribute of the ***display*** resource the actual screen number of the resource can be queried.

The value of *.real_screen* is always greater than or equal to *0* and a valid screen number, even if an invalid screen number has been specified for the *.screen* attribute of the ***display*** resource.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_screen
Data type: DT_integer

COBOL
Identifier: AT-real-screen
Data type: DT-integer

Classification
object-specific attribute

Objects
display

Note

Only the IDM for Motif provides multiscreen support.

See also

Attribute *.screen*

Chapter„Multiscreen Ssupport untder Motif“in manual„Programmiertechniken“.

2.271 .real_sensitive

This attribute queries the actual sensitivity of the object on the screen.

Definition

Data type
boolean

Access
get

C
Identifier: AT_real_sensitive
Data type: DT_boolean

COBOL
Identifier: AT-real-sensitive
Data type: DT-boolean

Classification
standard attribute

See also

Attribute *.sensitive*

2.272 .real_shadowobject

With this attribute you can access user-defined attributes. In this case the currently referenced object will be queried.

Definition

Data type
object

Access
get

C
Identifier: AT_real_shadowobject
Data type: DT_object

COBOL
Identifier: AT-real-shadowobject
Data type: DT-object

Classification
plain attribute

See also

Attributes *.count*, *.shadowattr*, *.shadowindex*, *.shadowobject*, *.type*

2.273 .real_size[integer]

This attribute returns the actual size of respective split area in pixels. The index is one-based and valid values for it are:

real_size[I] – valid, if *1<=I* and *I<=childcount*

This attribute, as all other real_… attributes, cannot be queried before the object is created and visible on the screen. If *.real_size[I]* is queried for an invisible child (whose split areas have not been created yet), the value *0* is returned.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_size
Data type: DT_integer

COBOL
Identifier: AT-real-size
Data type: DT-integer

Classification
object-specific attribute

Objects
splitbox

See also

Attribute *.size[integer]*

2.274 .real_version[enum]

With this attribute of the ***document*** object it can be queried if the desired version of the XML toolkit has been loaded. Querying this attribute forces the XML toolkit to be loaded. If the XML toolkit could not be loaded, *-1* is returned; otherwise the value of *.version[enum]* is returned.

The optional index for this attribute is the `toolkit` enumeration. If no index is given, the current toolkit is accessed implicitly.

Currently *toolkit_windows* is the only index value supported, because only Microsoft Windows allows to set the runtime version of the MSXML control. To indicate the version number, the major version has to be multiplied with *100* and the minor version has to be added if applicable. When *0* is given as version number, the system's default version of the MSXML control is loaded.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_version
Data type: DT_integer

COBOL
Identifier: AT-real-version
Data type: DT-integer

Classification
object-specific attribute

Objects
document

Example

MSXML 5.0 shall be used so that the XSD document type is supported.

```
this.version[toolkit_windows] := 500;
```

Afterward it is checked if MSXML 5.0 could be loaded.

```
if (this.real_version[toolkit_windows] = 500) then
  // OK
endif
```

See also

Attribute *.version[enum]*

2.275 .real_visible

This attribute queries the actual visibility of the object on the screen.

Whether this object is hidden for the user by other objects is not considered.

Definition

Data type
boolean

Access
get

C
Identifier: AT_real_visible
Data type: DT_boolean

COBOL
Identifier: AT-real-visible
Data type: DT-boolean

Classification
standard attribute

See also

Attribute *.visible*

2.276 .real_width

This attribute queries the real object width in pixel. The object has to be visible ()..*real_visible* = *true*

The width can be queried for objects without width definition.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_width
Data type: DT_integer

COBOL
Identifier: AT-real-width
Data type: DT-integer

Classification
geometry attribute

Remark

If there is no width and / or height given when defining objects, you can use the attributes *.real_width* and *.real_height* to query the width and the height of those objects.

These attributes can query the width of an object whose geometry has been defined depending on the right and left border of a parent (*.xauto* = *0*).

The dimensions returned are always in pixels!

See also

Attributes *.real_height*, *.width*

2.277 .real_x

The attribute *.real_x* has been made available to specify the real distance from the left (in pixel) for objects which were defined with virtual coordinates.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_x
Data type: DT_integer

COBOL
Identifier: AT-real-x
Data type: DT-integer

Classification
geometry attribute

Note

This attribute is available only for the Dialog Manager on MOTIF.

See also

Attribute *.real_y*

2.278 .real_xraster

This attribute specifies the width which is internally used; it is calculated on the basis of the indicated reference font.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_xraster
Data type: DT_integer

COBOL
Identifier: AT-real-xraster
Data type: DT-integer

Classification
object-specific attribute

See also

Attribute *.real_yraster*

2.279 .real_y

The attribute *.real_y* has been made available to determinate the real distance from the top (in pixel) with objects which were defined with virtual coordinates.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_y
Data type: DT_integer

COBOL
Identifier: AT-real-y
Data type: DT-integer

Classification
geometry attribute

Note

This attribute is available only for the Dialog Manager on MOTIF.

See also

Attribute *.real_x*

2.280 .real_yraster

This attribute specifies the height which is internally used; it is calculated on the basis of the indicated reference font.

Definition

Data type
integer

Access
get

C
Identifier: AT_real_yraster
Data type: DT_integer

COBOL
Identifier: AT-real-yraster
Data type: DT-integer

Classification
object-specific attribute

See also

Attribute *.real_xraster*

2.281 .record[integer]

This attribute returns or sets the record at index I. It contains – similar to *.child* – the record vector.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_record
Data type: DT_record

COBOL
Identifier: AT-record
Data type: DT-record

Classification
hierarchy attribute

See also

Object record

2.282 .recordcount

This attribute queries the number of ***records*** that an object has as children.

Definition

Data type
integer

Access
get

C
Identifier: AT_recordcount
Data type: DT_integer

COBOL
Identifier: AT-recordcount
Data type: DT-integer

Classification
hierarchy attribute

2.283 reexport

With *reexport*, objects and named rules of a module are made known externally so that they can be accessed in an importing module or dialog. Thus it provides the same function as *export* with the difference that the export property of the child objects of a model is inherited with *reexport*.

For this, *reexport* must be specified with the model on the child and father object. With an object that was inferred from the model, reexport only has to be specified on the father. Unlike with *export*, there is no *reexport* necessary for the inherited child of the inferred object.

Definition

```
reexport <object class> <Identifier>
{
  <further object definitions like attributes, children…>
}
```

When *reexport* is indicated on the father of an inferred object, all children that are exported with the model will automatically be exported with this object. Here it does not matter if their export results from the keywords export or reexport or if it is inherited already. When *export* is indicated on the father of an inferred object, this interrupts the inheritance of the export property for the child objcts. This holds true even if they are declared with reexport in the model or if they inherited the property themselves. A child object whose father is not exported in any kind will not be (re)exported.

Like export, *reexport* is placed at the beginning of the object definition. It can be used wherever export can also be used. This means that *reexport* can only be used in modules and not in dialogs. On child objects that have inherited their export properties, the keywords reexport and export are ignored. Therefore in existing modules, export can be replaced with reexport everywhere in order to use the changed inheritance behavior.

Availability

Since IDM version A.05.02.j

Examples

```
reexport model window MWiBase
{
  reexport child edittext Et {}
}
reexport model MWiBase MWiDerived
{
}
```

The inherited child *MWiDerived.Et* can be accessed from the outside as it inherits the export property of *MWiBase.Et*.

If export is used instead of reexport, *MWiDerived.Et* must be explicitly exported if you want to access it from the outside.

```
export model window MWiBase
{
  export child edittext Et {}
}
export model MWiBase MWiDerived
{
  export .Et {}  // inherited child
}
```

This also applies if export is used with the inferred object and the inheritance is interrupted by this. In the following example export on *MWiOne* prevents *MWiOne.Et* from being exported although with the model *MWiRe* reexport is indicated at the father als well as at the child. Conversely for an inferred object a reexport of the father causes all cildren that are exported in the model to be exported in the inferred object. In the example *MWiTwo.Et* inherits its export property through the reexport of *MWiTwo* although there is export indicated on *MWiEx* and *MWiEx.Et* and not reexport.

```
reexport model window MWiRe
{
  reexport child edittext Et {}
}
export model window MWiEx
{
  export child edittext Et {}
}
export model MWiRe MWiOne
{
  // No access from outside to the inherited child MWiOne.Et
  // as export on MWiOne interrupts the inheritance.
  // To make MWiOne.Et accessible from outside
  // an explicit export .ET {} is required.
}
reexport model MWiEx MWiTwo
{
  // The inherited child MWiTwo.Et can be accessed from outside
  // because MWiEx.Et has export and MWiTwo has reexport.
  // Therefore MWiTwo.Et inherits the export property from MWiEx.Et.
}
```

The next example illustrates that the inheritance of the export can only occur with inherited hierarchical children. As with export, with reexport there is no inheritance for the instantiable object at the root of a father-child-hierarchy.

```
export default pushbutton PUSHBUTTON { }
reexport default image IMAGE { }

model pushbutton MPbOne {}  // does not inherit export from PUSHBUTTON
model image MImOne {}       // does not inherit reexport from IMAGE either
export model pushbutton MPbTwo {}  // Explicit export or reexport are
```

```
reexport model image MImTwo {}     // necessary to enable access to MPbTwo
                                   // and MImTwo from outside the module.
// In the code below the pushbuttons and images are not
// inherited hierarchical children of the groupbox or layout models.
// Hence thy do not inherit export or reexport form their defaults
// and must be explicitly exported to publish them outside the module.
export model groupbox MGb {
  pushbutton PbOne {}         // not accessible from outside
  export pushbutton PbTwo {}  // export needed for access from outside
  image ImOne {}              // not accessible from outside
  reexport image ImTwo {}     // reexport needed for access from outside
}
reexport model layoutbox MLy {
  pushbutton PbOne {}           // not accessible from outside
  reexport pushbutton PbTwo {}  // reexport needed for access from outside
  image ImOne {}                // not accessible from outside
  reexport image ImTwo {}       // reexport needed for access from outside
}
```

Remark

» We recommend using either *export* or *reexport* consistently in order to prevent access restrictions that are difficult to trace.

» *reexport* is ignored on child objects that have inherited their export property.

See also

Attribute *export*

Chapter “Modularization” in manual “Programming Techniques”

2.284 .reffont

This attribute is the identifier of the font to which the x- and y-grid shall refer.

The ***dialog*** object also has the attribute *.reffont*. This allows to control the positioning of windows.

Definition

Data type	*Access*	*changed event*
object [***font***]	get, set	yes

C
Identifier: AT_reffont
Data type: DT_font

COBOL
Identifier: AT-reffont
Data type: DT-font

Classification
geometry attribute

Remarks

» The grid attributes *.reffont*, *.xraster* and *.yraster* are only significant if the application is to be realized on as many different hardware environments as possible. These attributes specifies a basic unit for the object size which is independent of the previously used pixel units. The letter size of the reference font selected or the input the user makes without specifying a font, are used as the new unit. All dimensions or positions then refer to this unit.

Identifier	Data Type	Meaning
.reffont	*object* (***font***)	Identifier of the font to which the units shall refer.
.xraster	*integer*	Basic unit on x-axis
.yraster	*integer*	Basic unit on y-axis

» If a reference font is given, the DM automatically calculates the values for *.xraster* and *.yraster*, i.e. a *.reffont* specification overwrites the specifications of *.xraster* and *.yraster*.

Example

```
.xraster    8;
.yraster    16;
...
.posraster  true;
.xleft      10;
.ytop       4;
```

The thus defined object has the position 80, 64 on pixel coordinates, because the position was given in relation to the bases 8 and 16.

See also

Attributes *.xraster*, *.yraster*

2.285 .refstring

This attribute holds the reference string of a ***font*** resource. This reference string is used to calculate the grid width.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_refstring	Identifier: AT-refstring
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	font

2-

.-

286 .rgb[enum]

This attribute defines the intensities (*0 … 255*) of the red, green and blue components of an RGB color at a ***color*** resource.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_rgb	Identifier: AT-rgb
Data type: DT_integer	Data type: DT-integer

Value range
0 … 255

Classification	*Objects*
object-specific attribute	color

Index Range

color_red
: Determines the intensity of the red part.

color_green
: Determines the intensity of the green part.

color_blue
: Determines the intensity of the blue part.

Note

The attributes *.rgb[enum]*, *.hls[enum]* and *.name* are mutually exclusive, so that “get” may lead to a *“can't get value”* error message.

2.287 .root

This attribute defines the position, where the transformation in a XML tree or in an IDM object hierarchy started. Therefore the value of the attribute is only meaningful during a transformation (i.e. during invocation of the **:apply()** method). Otherwise, the value should be *null*.

Definition

Data type	*Access*	*changed event*
string, *object*	get, set	no

C	*COBOL*
Identifier: AT_root Data type: DT_string, DT_object	Identifier: AT-root Data type: DT-string, DT-object

Classification	*Objects*
object-specific attribute	transformer

In its default implementation, the **:apply()** method of the ***transformer*** sets this attribute as follows:

- If a ***document*** object is transferred in the *Src* parameter, *.root* is assigned an empty string *""*, which represents the root of the document.
- If a ***doccursor*** object is transferred, *.root* is assigned a string that determines the position in the XML tree to which the ***doccursor*** points, i.e. *.root := Src.path*.
- If a different IDM object is transferred, *.root* is assigned this object itself.

When leaving the **:apply()** method, *.root* is set to *null*.

2.288 .rowalignment[integer]

In a ***tablefield***, this attribute describes the alignment of field contents (top, bottom, centered). The default value is *.rowalignment[0]* unless it is specified otherwise.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_rowalignment Data type: DT_integer	Identifier: AT-rowalignment Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

Value range

-1
: bottom-justified

0
: vertically centered

1
: top-justified

2.289 .rowcount

The attribute defines the number of rows.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_rowcount	Identifier: AT-rowcount
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listview, tablefield

tablefield

This attribute defines the total number of rows in a ***tablefield***. The value range is from 0 to 65535.

This number does not necessarily have to be used in the actual tablefield, but it is used to ensure the correctness of the vertical scrollbar.

listview

This attribute defines the number of list items.

The value range of *.rowcount* is *0 … n* (*n > 0*).

The value of *.rowcount* can be implicitly increased by adding one new row to *.content* at a time.

For invalid values, the default value *0* is used. The attribute value is not changed, however.

Note

Changing the attribute in the visible state may cause the object to flicker.

See also

Attribute *.colcount*

2.290 .rowfirst

In a ***tablefield***, this attribute defines the hierarchical number of the first row visible below the title.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_rowfirst	Identifier: AT-rowfirst
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.291 .rowheader

In a ***tablefield***, this attribute defines the number of rows which serve as row headers and thus not need to be scrolled. Value range 0 - 255.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_rowheader	Identifier: AT-rowheader
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.292 .rowheadfgc

In a ***tablefield***, this attribute defines the color of the row headers.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_rowheadfgc Data type: DT_color	Identifier: AT-rowheadfgc Data type: DT-color

Classification	*Objects*
object-specific attribute	tablefield

2.293 .rowheadfont

In a ***tablefield***, this attribute defines the font for the row headers.

Definition

Data type	*Access*	*changed event*
object [***font***]	get, set	yes

C	*COBOL*
Identifier: AT_rowheadfont	Identifier: AT-rowheadfont
Data type: DT_font	Data type: DT-font

Classification	*Objects*
object-specific attribute	tablefield

2.294 .rowheadshadow

In a ***tablefield***, this attribute defines the shape of the row headers. If the attribute is true, the display has a shadow (similar to a button).

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_rowheadshadow Data type: DT_boolean	Identifier: AT-rowheadshadow Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.295 .rowheadvisible

In a ***tablefield***, this attribute defines if row headers are to be displayed. Headers can be switched on and off dynamically.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_rowheadvisible Data type: DT_boolean	Identifier: AT-rowheadvisible Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.296 .rowheight[integer]

In a ***tablefield***, this attribute defines the height of individual rows in coordinate units (pixels, if *.sizeraster* is not set; grid units, if *.sizeraster* is set).

The default value for all rows is *.rowheight[0]* unless specified otherwise.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_rowheight	Identifier: AT-rowheight
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.297 .rowlinewidth[integer]

In a ***tablefield***, this attribute defines the width of the vertical lines drawn in the tablefield. The default value is *.rowlinewidth[0]* unless it is specified differently.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_rowlinewidth	Identifier: AT-rowlinewidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.298 .rowsizeable[integer]

This single-indexed attribute controls the interactive maximization of rows.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_rowsizeable	Identifier: AT-rowsizeable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

If the mouse is moved beyond the margin of a manipulable field, the cursor will change displaying a symbol which indicates change of size. You cannot specify this cursor symbol.

You can start the maximization by using the left mouse button. A gray hatched line indicating the mouse position will be displayed (the width of the hatched line depends on the attribute *.rowlinewidth [integer]*). If the mouse is moved (by pressing the left mouse button), the hatched line will move along. The maximization stops when the mouse button is released.

This attribute is used as other single-indexed attributes for tablefield, i.e. *.rowsizeable[0]* provides the default to be used if no value is specified for the column. The default of *.rowsizeable[0]* is *false*.

2.299 .rowvisible[integer]

This attribute controls the visibility of lines so that they can be hidden if requested.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_rowvisible Data type: DT_boolean	Identifier: AT-rowvisible Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

In some applications, information which is exclusively meant for the application and not for the user is held in the ***tablefield***. These columns or rows shall be displayed only in specific situations.

These attributes are treated as the other ones in the tablefield, i.e. in *.rowvisible[0]* the default is used (the default value is *.rowvisible[0]*) if no value has been defined for the column.

2.300 .scale

Meaning for the *tile* resource

Attention: This attribute is **deprecated** for the ***tile*** resource and is only present for compatibility reasons. It tells whether there is scaling on the image/ pattern or not. Which scaling exactly is active can be taken from the attribute *.scalestyle*. The attribute should not be used anymore. The scaling of a ***tile*** resource should only be controlled via the attribute *.scalestyle*.

Meaning for the *setup* object

On the ***setup*** object, this attribute returns the current system scaling (the one from the primary monitor). Setting *.scale* on the ***setup*** object is not possible.

Definition

Data type	*Access*	*changed event*
boolean (***tile***)	get, set (***tile***)	no
integer (***setup***)	get (***setup***)	

C
Identifier: AT_scale
Data type: DT_boolean, DT_integer

COBOL
Identifier: AT-scale
Data type: DT-boolean, DT-integer

Default value	*Inheritance*
false (***tile***)	no

Classification	*Objects*
object-specific attribute	tile, setup

Value range for Tile

false
: The tile retains its defined size. No scaling is applied.

true
: Compatibly says only that there is a scaling. Which scaling is applied can be taken from the attribute *.scalestyle*. In case of a *true* value the attribute *.scalestyle* is set to the value *scalestyle_any*.

See also Attribute *.scalestyle*

Value range for Setup

0
: Windows: DPI awareness is disabled.
Motif: Scaling is switched off. No XFT fonts are used and images/patterns are not scaled.
Qt: The scaling used by the system is used.

> 0
: Currently active scaling of the application in %.

Note: scaling should be done in reasonable steps of 50%, 100%, 200%, 300% etc.. Otherwise, unsightly scaling and possibly also performance effects may occur.

Availability

Since IDM version A.06.03.a.

This affects the availability at the ***setup*** object as well as the classification as deprecated for the ***tile*** resource.

See also

Chapter „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

2.301 .scalestyle

This attribute controls the display and scaling of a picture or tile pattern.

A scaling has an effect when the ***tile*** resource is used in the *.picture*, *.picture[enum]*, *.picture[integer]* and/ or *.picture_hilite[integer]* attributes.

Definition

Data type	*Access*	*changed event*
enum	get, set	no

C	*COBOL*
Identifier: AT_scalestyle Data type: DT_enum	Identifier: AT-scalestyle Data type: DT-enum

Default value	*Inheritance*
scalestyle_auto	yes

Classification	*Objects*
object-specific attribute	tile

Value range

scalestyle_none
: The pattern or image is not scaled. *.tiledpi* has no impact.

scalestyle_any
: The height and width of the pattern or image are fully enlarged to fit the available area.

scalestyle_prop
: Height and width of the pattern or image are enlarged to the available area, whereby height and width proportions of the pattern or image are maintained in any case. I.e. free spaces might appear at the top and bottom or left and right.

scalestyle_num
: The pattern or image is scaled according to a specified factor.

scalestyle_auto
: The pattern or image is scaled according to the set screen scaling. A scaling compatible to the previous version takes place for the window, treeview and notepage icon.

scalestyle_dpi
: The pattern or image is always scaled according to the set screen scaling.

See also the „SkalierungScaling“ section in the tile esource manual.

RemarkMOTIF & QT

The objects menuitem and notepage do not support this attribute or support it only to a limited extent.

Remark aggregate objects:

Aggregate objects that may have an image or tile pattern set via the *.tile* attribute do not support the *.scalestyle* attribute.

Availability

Since IDM version A.06.03.a

See also

Chapter „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

2.302 .scope

This attribute queries whether the object is a Default, a Model or an instance.

Definition

Data type
integer

Access
get

C
Identifier: AT_scope
Data type: DT_scope

COBOL
Identifier: AT-scope
Data type: DT-scope

Classification
plain attribute

Value range

1
Default

2
Model

3
instance

2.303 .scope[attribute]

This attribute returns the validity range of user-defined attributes.

To access the validity range of a particular user-defined attribute, the attribute identifier with the data type *attribute* is used as index.

This attribute is available for all objects that may have user-defined attributes.

Definition

Data type
anyvalue

Access
get

C
Identifier: AT_scope
Data type: DT_anyvalue

COBOL
Identifier: AT-scope
Data type: DT-anyvalue

Classification
plain attribute

See also

Attributes *.indexscope[attribute]*, *.typescope*, *.typescope[integer]*

Chapter “Validity Range for Better Type Checking” in manual “Rule Language”

2.304 .screen

At a ***display*** resource, the screen number can be queried and set with this attribute. Setting *.screen* moves all windows using the affected ***display*** resource to the specified screen.

With the attribute of the ***setup*** object, the screen number of the default screen can be queried in multiscreen systems.

Definition

Data type	*Access*	*changed event*
integer	get (***setup***) get, set (***display***)	no

C	*COBOL*
Identifier: AT_screen Data type: DT_integer	Identifier: AT-screen Data type: DT-integer

Classification	*Objects*
object-specific attribute	display, setup

display

The value range of the screen attribute goes from *-32767* to *32767*, but only values greater than zero are valid screen numbers. Invalid screen numbers will lead to windows being displayed on the default screen if they use the display resource in question.

The default value for *.screen* is *-1*.

Remarks

Only the IDM FOR MOTIF provides multiscreen support.

Valid screen numbers can be determined with the program **xdpyinfo** or the attribute *.screen[integer]* of the ***setup*** object. The screen numbers returned by the screen attribute correspond to those obtained with the program **xdpyinfo**.

An example dialog can be found at the display resource.

See also

Attribute *.real_screen*.

Chapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“.

2.305 .screen[integer]

With this attribute of the setup object the screen number of every available screen can be queried in multiscreen systems. The data type of the index is *integer*, with a valid range from *1* to *setup.screen-count*.

This attribute is available only for multiscreen dialogs (MOTIF only).

Definition

Data type	*Access*
integer	get

C	*COBOL*
Identifier: AT_screen Data type: DT_integer	Identifier: AT-screen Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

Remarks

Only the IDM FOR MOTIF provides multiscreen support.

The screen numbers returned by *.screen[integer]* correspond to those obtained with the program **xdpyinfo**. Note that for the screen numbers no order is guaranteed.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

2.306 .screen_height

With this attribute of the setup object the height of the screen in pixels can be queried.

In multiscreen systems (IDM for MOTIF only) the attribute returns the value for the default screen.

For multi-monitor dialogs (WINDOWS only) the value for the default monitor is returned.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_height
Data type: DT_integer

COBOL
Identifier: AT-screen-height
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.307 .screen_height[integer]

With this attribute of the setup object the height of screen I in pixels can be queried.

The indexed attribute is available only for multiscreen dialogs (MOTIF only) or in a multi-monitor environment (WINDOWS only), where the index can be in the valid range from *1* to *setup.screencount*.

The value is dynamic in a multi-monitor environment under WINDOWS, since you can add or remove monitors at any time.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_height
Data type: DT_integer

COBOL
Identifier: AT-screen-height
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.308 .screen_width

With this attribute of the setup object the width of the screen in pixels can be queried.

In multiscreen systems (IDM for MOTIF only) the attribute returns the value for the default screen.

For multi-monitor dialogs (WINDOWS only) the value for the default monitor is returned.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_width
Data type: DT_integer

COBOL
Identifier: AT-screen-width
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.309 .screen_width[integer]

With this attribute of the setup object the width of screen I in pixels can be queried.

The indexed attribute is available only for multiscreen dialogs (MOTIF only) or in a multi-monitor environment (WINDOWS only), where the index can be in the valid range from *1* to *setup.screencount*.

The value is dynamic in a multi-monitor environment under WINDOWS, since you can add or remove monitors at any time.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_width
Data type: DT_integer

COBOL
Identifier: AT-screen-width
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.310 .screen_x

With this attribute of the setup object the X coordinate of the origin of the primary screen resp. working area in pixels can be queried.

For multi-monitor dialogs (WINDOWS only) the value for the default monitor is returned.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_x
Data type: DT_integer

COBOL
Identifier: AT-screen-x
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.311 .screen_x[integer]

With this attribute of the setup object the x coordinate of the screen (in pixels) can be queried.

The indexed attribute is available only in a multi-monitor environment (WINDOWS only), where the index can be in the valid range from *1* to *setup.screencount*.

The value is dynamic in a multi-monitor environment under WINDOWS, since you can add or remove monitors at any time.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_x
Data type: DT_integer

COBOL
Identifier: AT-screen-x
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.312 .screen_y

With this attribute of the setup object the Y coordinate of the origin of the primary screen resp. working area in pixels can be queried.

For multi-monitor dialogs (WINDOWS only) the value for the default monitor is returned.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_y
Data type: DT_integer

COBOL
Identifier: AT-screen-y
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.313 .screen_y[integer]

With this attribute of the setup object the Y coordinate of the screen (in pixels) can be queried.

The indexed attribute is available only in a multi-monitor environment (WINDOWS only), where the index can be in the valid range from *1* to *setup.screencount*.

The value is dynamic in a multi-monitor environment under WINDOWS, since you can add or remove monitors at any time.

Definition

Data type
integer

Access
get

C
Identifier: AT_screen_y
Data type: DT_integer

COBOL
Identifier: AT-screen-y
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.314 .screencount

With this attribute of the ***setup*** object. For MOTIF the number of available screens can be queried. Available screens are the screens configured for the display in use. For WINDOWS the number of available monitors can be queried. The value is dynamic, since you can add or remove monitors at any time.

The value of *.screencount* is always greater than *0*. It is *1* on systems without multi-screen or multi-monitor support.

Definition

Data type
integer

Access
get

C
Identifier: AT_screencount
Data type: DT_integer

COBOL
Identifier: AT-screencount
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.315 .searchpath

With this attribute, the current search path used for dialog, module, interface and binary files for imports with **use** can be queried and modified.

Definition

Data type	*Access*	*changed event*
string	get, set	no

C	*COBOL*
Identifier: AT_searchpath	Identifier: AT-searchpath
Data type: DT_string	Data type: DT-string

Default value
"~;"

Classification	*Objects*
object-specific attribute	setup

Availability

Since IDM version A.06.02.g

See also

Chapter “Search Path for Interface, Module, Dialog, and Binary Files” in manual “Programming Techniques”

2.316 .selected[integer]

This attribute determines for each list item of the ***listview*** object whether it is selected.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_selected Data type: DT_boolean	Identifier: AT-selected Data type: DT-boolean

Default value	*Inheritance*
false	yes

Classification	*Objects*
object-specific attribute	listview

The value range of the index is *0 … .rowcount*t, where the value with index *0* is used as default value for not set values in the range *1 … .rowcount*.

Value range

false
: The list item is **not** selected.

true
: The list item is selected.
The attribute will be modified through user interaction (*activate* and *deactivate* events).

Note

The appearance of selected items depends on the system. This applies in particular to the visibility of the selection when the ***listview*** does not have the focus.

2.317 .selection[enum]

This attribute determines which kind of selection is permitted in the ***tablefield***.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_selection	Identifier: AT-selection
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

Index Range

sel_column (column selection)
: Complete columns can be selected.

sel_header (header selection)
: Column headers and row headers can be selected.

sel_row (row selection)
: Complete rows can be selected.

sel_single (single field selection; default)
: Single fields can be selected.

A selection type is activated by setting *.selection[<selection_type>]* to *true*.

See also

Attribute *.selstyle*

Chapter “Selection in a Tablefield” in the “Object Reference”

2.318 .self

With *.self* an object itself is addressed. In most cases this can be achieved by simply using the identifier of an object. Sometimes, however, an attribute is necessary. In these cases *.self* may be used.

For instance, using the identifier of a global variable in Rule Language accesses the value of the variable. An access to the variable as object, e.g. to query or change its data type, can be accomplished through *.self*.

Definition

Data type
object

Access
get

C
Identifier: AT_self
Data type: DT_object

COBOL
Identifier: AT-self
Data type: DT-object

Classification
plain attribute

Example

```
dialog Dialog
variable integer Answer := 42;
on dialog start
{
  print Answer;       // output is "42", that is the value
  print Answer.self;  // output is "variable Dialog.Answer",
                      // that is the variable itself
}
```

2.319 .selstyle

This attribute controls whether the user may select multiple entries in an object and how the selection is carried out.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_selstyle Data type: DT_enum	Identifier: AT-selstyle Data type: DT-enum

Classification	*Objects*
object-specific attribute	listbox, tablefield

Value range

single
: Only one item can be selected and active at a time.

multiple
: Several items can be selected and active, whereby items can be selected and deselected one by one with a mouse click.
Compatible to *.multisel = true*.

extended
: Several items can be selected and active, whereby an existing selection can be expanded and reduced by mouse clicks in combination with the `Ctrl` and `Shift` keys. Multiple items can be selected and deselected at once using the `Shift` key.
In detail, the behavior is as follows:

User Operation	Selection
“Click”	delete current selection, activate current element
`Shift` + “Click”	maintain current selection, activate area from last “Click” without `Shift` to current element
`Ctrl` + “Click”	maintain current selection, toggle activation of current element
`Shift` + `Shift` + “Click”	maintain current selection, area from last “Click” without `Shift` up to current element gets activation of the first one

The interaction of *.selstyle* with attribute *.selection[enum]*, which is used to select a selection type in the ***tablefield***, is described in the following table:

Selection Type	Current Element
sel_single	one field
sel_header	one field of header
sel_row	one row
sel_column	one column

Note for *.selstyle single* and *extended*: a ***tablefield*** with one column behaves as a ***listbox***.

Remark

The attribute *.multisel* has become superfluous for ***tablefield*** and ***listbox*** by the attribute *.selstyle*, but *.multisel* will still be supported.

See also

Attribute *.selection[enum]*

Chapter “Selection in a Tablefield” in the “Object Reference”

2.320 .sensitive

The attribute defines whether the object can be selected by the user.

At the ***tablefield***, the attribute defines whether the object is to be selectable or not. Thus, it can be used to make the complete tablefield insensitive at once.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_sensitive
Data type: DT_boolean

COBOL
Identifier: AT-sensitive
Data type: DT-boolean

Classification
standard attribute

Particularities of the Attributes .sensitive and .visible

Unlike the other object attributes, the attributes *.visible* and *.sensitive* get their final shape only in connection with the object hierarchy. This is because the corresponding attributes of the related parent object have a decisive influence on them. This means that an object can only be visible if itself **and** its parent are visible. The same applies to selectivity.

For the objects ***tablefield*** and ***edittext*** there are attributes which enable the dialog designer to influence the design and input behavior. This refers to the display of non-selectabel edittexts as well as to the definition of whether the user shall be able interact with these objects or not.

- **Usability or selectability**
  Attribute*.sensitive*
  Describes the quality of whether a user can select an object or not. Only if an object is selectable, the user can carry out the actions usual for the object, e.g. inputting data and scrolling.
- **Editability**
  Attribute*.editable*
  Describes the user's possibility to change and edit the contents of objects.
- **Focusability**
  Attribute*.navigable*
  Describes the possibility of including the object in the keyboard control. In this way the object receives the input focus.

These three attributes influence the object behavior as described below:

Attribute	Value	Effects
.sensitive	*true*	The user can select the object. Only if .sensitive is true, the attributes .navigable and .editable are effective, otherwise they are ignored.
.sensitive	*false*	The user cannot select the object. The object cannot be edited or changed. The object contents is displayed in grey as is usual for window systems.
.editable	*true*	The user can change the object contents, if he can select the object.
.editable	*false*	The user cannot change the object contents.
.navigable	*true*	The object is contained in the normal keyboard control, i.e. the user can get the object by navigating in the corresponding window.
.navigable	*false*	The object cannot be reached by keyboard control. It can, however, get the focus by the mouse.

Note for the IDM for Motif

If the attributes *.active* and *.sensitive* are not explicitly specified, the default for *.active* is *false*, and the default for *.sensitive* is *true*.

Motif 1.1

If *.sensitive* = *false* at a ***listbox***, the scrollbars stay sensitive, i.e. they can be selected and are operable.

Motif 1.2

If *.sensitive* = *false* at a ***listbox***, the scrollbars become insensitive, too, i.e. they are not operable.

See also

Attributes *.editable*, *.navigable*, *.visible*

2.321 .sensitive[integer]

This attribute defines for each entry in a ***listbox*** whether it can be selected (*true*) or not (*false*).

On Motif *.sensitive[integer]* is ignored.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_sensitive	Identifier: AT-sensitive
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	listbox

2.322 .sensitive[index]

This attribute defines whether single fields [row, column] shall be selectable or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_sensitive Data type: DT_boolean	Identifier: AT-sensitive Data type: DT-boolean

Classification	*Objects*
object-specific attribute	tablefield

2.323 .shadowattr

With this attribute, you can access and change user-defined attributes. In specific, this attribute provides the index of the referenced attribute of shadow attributes.

Definition

Data type	*Access*	*changed event*
attribute	get, set	yes

C
Identifier: AT_shadowattr
Data type: DT_attribute

COBOL
Identifier: AT-shadowattr
Data type: DT-attribute

Classification
plain attribute

See also

Attributes *.count*, *.real_shadowobject*, *.shadowindex*, *.shadowobject*, *.type*

2.324 .shadowindex

With this attribute, you can access and change user-defined attributes. In specific, this attribute provides the index of the referenced attribute of shadow attributes.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_shadowindex
Data type: DT_integer

COBOL
Identifier: AT-shadowindex
Data type: DT-integer

Classification
plain attribute

See also

Attributes *.count*, *.real_shadowobject*, *.shadowattr*, *.shadowobject*, *.type*

2.325 .shadowinstance

This attribute is used when a model is instantiated. It defines that the reference indicated in the user-defined attribute is to be changed to the newly generated instance.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_shadowinstance
Data type: DT_boolean

COBOL
Identifier: AT-shadowinstance
Data type: DT-boolean

Classification
plain attribute

Example

```
model record MRec
{
  integer I 123;
}
model window MWi
{
  string S shadows instance Et.content;
  integer I shadows instance MRec.I;
  child edittext Et
  {
    .content "Hello";
  }
}
MRec Rec
{
  .I := 456;
}

MWi Wi
{
  Et.content := "World";
}
on dialog start
{
  !! Wi.S shadows Wi.Et.content
  !! and not MWi.Et.content!
  print Wi.content;   // result: "World"
  !! shadow and shadowed object are not in the same model
  print Wi.I;         // result: 123
```

```
}
```

2.326 .shadowobject

With this attribute, you access and change user-defined attributes. In specific, this attribute provides the referenced object of shadow attributes.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_shadowobject
Data type: DT_object

COBOL
Identifier: AT-shadowobject
Data type: DT-object

Classification
plain attribute

See also

Attributes *.count*, *.real_shadowobject*, *.shadowattr*, *.shadowindex*,*.type*

2.327 .shortdaynames

This attribute determines whether short weekday names are displayed in the fold-out calendar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_shortdaynames Data type: DT_boolean	Identifier: AT-shortdaynames Data type: DT-boolean

Default value	*Inheritance*
false	yes

Classification	*Objects*
object-specific attribute	datetime

Value range

false
: Normal weekday names are displayed in the calendar.

true
: Short weekday names are displayed in the calendar.

Note

In German there is no difference between short and normal weekday names.

2.328 .showitem

This attribute defines how many entries should be displayed in the list of a ***poptext***. If the list contains more entries than specified here, a vertical scrollbar is displayed in the opened list.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_showitem	Identifier: AT-showitem
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	poptext

Notes on the IDM FOR QT

On QT, a maximum of *10* entries are displayed in an open ***poptext*** by default. If the ***poptext*** list contains more than *10* entries, then a scrollbar is additionally displayed in the list.

The *.showitem* attribute is ignored in the “mac” and “gtk+” UI styles. In UI style “cleanlooks”, *.showitem* is ignored with *.style = poptext*, here always the complete list is displayed.

With *.style = listbox*, *.showitem* is completely ignored.

2.329 .size

This attribute of the ***font*** resource defines the font size (in points).

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_size	Identifier: AT-size
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	font

2.330 .size[integer]

This attribute allows for the setting of the desired width of each separate split area. Both pixel and raster values are allowed. Index is zero-based. The valid values for this are:

size[I] – valid, if *0<=I* and *I<=childcount*

size[0] is the so-called zero-element, that no split area is assigned to. The value of the zero-element is transferred to the other elements of the size[I] vector for which no explicit setting exists.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_size Data type: DT_integer	Identifier: AT-size Data type: DT-integer

Classification	*Objects*
object-specific attribute	splitbox

See also

Attributes *.maxsize[integer]*, *.minsize[integer]*

2.331 .sizeable

This attribute controls whether the size of the object may be changed interactively by the user.

For the ***window***, this attribute also determines if it should provide a mechanism (button, box) for the user to enlarge or minimize the window.

With the ***toolbar***, *.sizeable* (without index) defines the default value for the values not set by *.sizeable [class]*.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_sizeable	Identifier: AT-sizeable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	toolbar, window

Particularity of Motif

Depending on the display or desktop manager in use, the attribute cannot be changed in the visible state; under certain conditions, it may not be possible to set it at runtime. In some cases, it may help to toggle the visibility of the window.

Since the ability of setting this attribute on MOTIF directly depends on the display or desktop manager used, it is recommended to set the attribute only statically or immediately after creating an instance with *:create(…, true)* in the invisible state.

2.332 .sizeable[class]

This attribute of the ***toolbar*** defines whether the toolbar can be resized interactively when it is in the docking state given by the index.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_sizeable	Identifier: AT-sizeable
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	toolbar

Value range

true
: The toolbar can be resized using the mouse (docked and undocked) or the system menu (undocked).

false
: The size of the ***toolbar*** cannot be changed interactively.

Index Range

toolbar
: Possibility of interactive resizing when the toolbar is docked

window
: Possibility of interactive resizing when the toolbar is undocked (tool window)

Without an index the attribute returns or sets the default value for both docking states.

2.333 .sizeraster

This attribute determines the interpretation of the attribute values for the size of an object (*.width*, *.height*). It can be used to specify the size of an object depending on the used grid or exactly in pixels.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_sizeraster
Data type: DT_boolean

COBOL
Identifier: AT-sizeraster
Data type: DT-boolean

Classification
geometry attribute

Value range

true
: The values of the size attributes are interpreted as grid units. The underlying grid is the grid of the parent object.

false
: The values of the size attributes are interpreted as pixel units. The grid of the parent object is ignored.

Note on IDM for Motif (from IDM A.05.02.e)

The resizing of windows with *.sizeraster* = *true* happens in increments of the grid, provided this is supported by the window manager.

See also

Attribute *.posraster*

2.334 .smallpicheight

This attribute defines the height of the small icons.

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_smallpicheight Data type: DT_integer	Identifier: AT-smallpicheight Data type: DT-integer

Default value	*Inheritance*
0	yes

Classification	*Objects*
object-specific attribute	listview

0
: The width *.smallpicwidth* is used. If this is also *0*, the size of the icons is determined by the system.

> 0
: Height of the small icons.

The small icons will be shown in the small icon view (*.style = "smallicon"*, *.style = "smallpicture"*), in the list view (*.style = "list"*) and in the detail view (*.style = "detail"*, *.style = "report"*).

For invalid values, the default value *0* is used. However, the attribute value is not changed.

Changing the attribute in the visible state may cause the object to flicker.

See also

Attributes *.smallpicture[integer]*, *.smallpicwidth*

2.335 .smallpicture[integer]

The attribute defines the small icon for each list item.

Data type	*Access*	*changed event*
object [***tile***]	get, set	yes

C	*COBOL*
Identifier: AT_smallpicture Data type: DT_tile	Identifier: AT-smallpicture Data type: DT-tile

Default value	*Inheritance*
null	yes

Classification	*Objects*
object-specific attribute	listview

The value range of the index is *0 … .rowcount*t, where the value with index *0* is used as default value for not set values in the range *1 … .rowcount*.

null

The displayed icon is derived as follows:

- If *.smallpicture[0]* <> *null*, the icon defined there is used.
- If *.smallpicture[0]* = *null*, then the small icon is shown downsized (*.picture[I]* or *.picture[0]*, depending on presence).
- If there is neither a small nor a large icon, the item is displayed without an icon.

tile

Resource that contains the small icon.

The small icons will be shown in the small icon view (*.style = "smallicon"*, *.style = "smallpicture"*), in the list view (*.style = "list"*) and in the detail view (*.style = "detail"*, *.style = "report"*).

Small and large icons belong together and should represent the same information. Both icons can only be referenced together. Therefore, different combinations should be avoided, as each combination will consume additional memory.

Changing the attribute in the visible state may cause the object to flicker.

See also

Attributes *.smallpicheight*, *.smallpicwidth*

2.336 .smallpicwidth

This attribute defines the width of the small icons.

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_smallpicwidth Data type: DT_integer	Identifier: AT-smallpicwidth Data type: DT-integer

Default value	*Inheritance*
0	yes

Classification	*Objects*
object-specific attribute	listview

0
: The width of the icons is determined by the system.

> 0
: Width of the small icons.

The small icons will be shown in the small icon view (*.style = "smallicon"*, *.style = "smallpicture"*), in the list view (*.style = "list"*) and in the detail view (*.style = "detail"*, *.style = "report"*).

For invalid values, the default value *0* is used. However, the attribute value is not changed.

If *.smallpicheight = 0* then *.smallpicwidth* also determines the **height** of the small icons.

Changing the attribute in the visible state may cause the object to flicker.

See also

Attributes *.smallpicheight*, *.smallpicture[integer]*

2.337 .source

This attribute determines the object behavior as source of a Drag & Drop or clipboard operation. Resources of the type ***source*** are used as values.

Definition

Data type	*Access*	*changed event*
object [***source***]	get, set	yes

C
Identifier: AT_source
Data type: DT_source

COBOL
Identifier: AT-source
Data type: DT-source

Classification
standard attribute

2.338 .spacing

This attribute of the ***image*** object defines the gap between the picture and the text. The gap is ignored if the image object contains either a picture only or a text only.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_spacing	Identifier: AT-spacing
Data type: DT_integer	Data type: DT-integer

Value range	*Default value*
-32767 … 32767	*0* (MICROSOFT WINDOWS) *2* (MOTIF)

Classification	*Objects*
object-specific attribute	image

The effect of *.spacing* in dependency of the attributes *.alignment* and *.tilestyle* is explained at the *.tilestyle* attribute.

2.339 .specified

This attribute returns whether an attribute of a DOM node was explicitly given or inherited from a standard value. The value of *.specified* is only meaningful when the node type is *nodetype_attribute*. For all other node types *.specified* always returns *true*.

Definition

Data type
boolean

Access
get

C
Identifier: AT_specified
Data type: DT_boolean

COBOL
Identifier: AT-specified
Data type: DT-boolean

Classification
object-specific attribute

Objects
doccursor

The attribute is available for the XML Cursor. Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

See also

Attribute *.nodetype*

2.340 .startsel

For ***edittext*** and ***poptext***, the *.startsel* attribute defines the beginning of the selection in the input field.

For the file and directory dialogs (***filereq***), this attribute can be used to specify an initial value. If *.startsel* is *true*, the content of the *.value* attribute is used as default value.

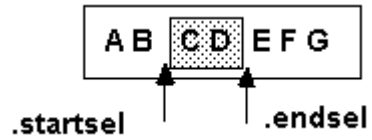
Definition

Data type	*Access*	*changed event*
integer *boolean* (***filereq***)	get, set	yes no (***filereq***)

C	*COBOL*
Identifier: AT_startsel Data type: DT_integer Data type: DT_boolean (***filereq***)	Identifier: AT-startsel Data type: DT-integer Data type: DT-boolean (***filereq***)

Classification	*Objects*
object-specific attribute	edittext, filereq, poptext

2.340.1 edittext, poptext



When modifying *.startsel* or *.endsel*, *.focus* must be *true*, so that the new selection can be displayed. *.startsel*may be greater than *.endsel* there.

Special Values

The following values, which are particularly useful when formats are set, have a special meaning:

-1
: Selects all decimal places after the decimal point, including zeros inserted by a format that are not contained in the content string

-2
: Selects all decimal places of the integer part up to the right end (decimal point)

-3
: Selects all decimal places of the integer part, inclusive of leading zeros inserted by a format

Selection of the leading sign is possible in no case.

Remark on the IDM for Windows

When querying *.startsel* and *.endsel* of an ***edittext***, since IDM version A.05.02.I *.endsel* may be smaller than *.startsel*. In this case the cursor is positioned left of the selection. However, if the user selects text with the mouse, *.startsel* is always smaller than *.endsel* and it cannot be recognized where the cursor is. In previous IDM versions *.startsel* was always less than or equal to *.endsel*, except it was set differently from the Rule Language.

See also

Attribute *.endsel*

2.340.2 filereq

Only one value can be specified, that is *.value* has to be scalar. The *.value* attribute must contain a full path where the directory matches the *.directoy* attribute.

The default value of *.startsel* is *false*.

Notes

- The default value is displayed in the input field of the file and folder dialogs but not highlighted in the select lists. This is a property of the WINDOWS and MOTIF system dialogs.
- On MICROSOFT WINDOWS, only defaults for loading and saving files are possible, but not for the choice of a directory.

See also

Attributes *.directory*, *.value*

2.341 .starttime

Defines the starting time of the ***timer*** indicated. The time specified may be either relative or absolute.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_starttime Data type: DT_string	Identifier: AT-starttime Data type: DT-string

Classification	*Objects*
object-specific attribute	timer

2.342 .state

The *.state* attribute defines the activation state of a tristate ***checkbox***.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_state	Identifier: AT-state
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	checkbox

Value range

state_unchecked
: Normal state – the object is not activated; corresponds to *.active := false*.

state_checked
: The object is activated.

state_indeterminate
: The object is in an indeterminate state; in this case *.active* is not defined (*false*).
Only possible when *.style = 3*.

The setting of the attribute *.state* is also possible when *.style = 2*; in this case *state_ indeterminate* is interpreted as *state_unchecked*. Thus *.active* is also switched.

If a tristatebutton is switched into a ***checkbox*** by allocating a value, the attribute *.active* is always *false*.

2.343 .static

This attribute provides the information of whether a ***module*** may be unloaded or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_static	Identifier: AT-static
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	import

2.344 .statusbar

This attribute of the ***window*** returns the window's statusbar.

Definition

Data type
object

Access
get

C
Identifier: AT_statusbar
Data type: DT_object

COBOL
Identifier: AT-statusbar
Data type: DT-object

Classification
object-specific attribute

Objects
window

2.345 .statushelp

The helptext to be displayed will be deposited in this attribute. The helptext is displayed in case that the object is focussed (if the mouse is positioned on the object) and if the parent window has a visible statusbar containing a valid *.helppos* definition. If the object is a menuitem, its helptext will be displayed.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C	*COBOL*
Identifier: AT_statushelp	Identifier: AT-statushelp
Data type: DT_string, DT_text	Data type: DT-string, DT-text

Classification
standard attribute

A blank string will not display helptexts. If the attribute has null ID, the helptext of its parent will be used.

Status help will be displayed, if a menuitem is focussed or if the mouse is positioned on the corresponding object (see also *.toolhelp*).

If a blank string "" is indicated in the attribute, statushelp will not be displayed.

If a null string (null) is indicated in the attribute, the status help of the parent will be used.

This hierarchical query will only be pursued up to the parent window - dialogs and modules have no statushelp.

Only the help for objects of the main window are displayed in statushelp. It is therefore not possible to have several main windows, but only one statusbar for the entire statushelp.

2.346 .style

The attribute *.style* determines the appearance and behavior of the object.

Definition

Data type	*Access*	*changed event*
integer (***menusep***) *integer, class* (***checkbox***, ***image***, ***menuitem***) *integer, string* (***listview***) *class* (***datetime***, ***poptext***, ***toolbar***) *enum* (***filereq***, ***font***) *datatype* (***spinbox***)	get, set	yes no (***font***)

C	*COBOL*
Identifier: AT_style	Identifier: AT-style
Data type: DT_integer (***menusep***)	Data type: DT-integer (***menusep***)
Data type: DT_integer, DT_class (***image***, ***menuitem***)	Data type: DT-integer, DT-class (***image***, ***menuitem***)
Data type: DT_integer, DT_class/DT_enum (***checkbox***)	Data type: DT-integer, DT-class/DT-enum (***checkbox***)
Data type: DT_integer, DT_string (***listview***)	Data type: DT-integer, DT-string (***listview***)
Data type: DT_class (***datetime***, ***poptext***, ***toolbar***)	Data type: DT-class (***datetime***, ***poptext***, ***toolbar***)
Data type: DT_enum (***filereq***, ***font***)	Data type: DT-enum (***filereq***, ***font***)
Data type: DT_datatype (***spinbox***)	Data type: DT-datatype (***spinbox***)

Classification	*Objects*
object-specific attribute	checkbox, datetime, filereq, font, image, listview, menuitem, menusep, poptext, spinbox, toolbar

2.346.1 checkbox, image, listview, menuitem and menusep

checkbox

The *.style* attribute determines whether the ***checkbox*** can take on two or three states.

For write access (“set”), values of the data types *integer* and *class*/*enum* can be specified. Read access (“get”) always returns a value of the data type *integer*.

Value range

2 | *checkbox*
: The checkbox can assume the two states “on” and “off”.
Its state can be queried and set with the *.active* attribute.
Interactive state changes by the user trigger *activate* or *deactivate* events.

3 | *tristate*
The checkbox can assume the three states “on”, “off” and “indefinite”.
Its state can be queried and set with the *.state* attribute.
Interactive state changes by the user do **not** trigger *activate* or *deactivate* events.

When setting the value *3*, the checkbox is displayed in the “indefinite” state.

image

With the attribute *.style*, the ***image*** object can be enabled to display two states or – on MICROSOFT WINDOWS – to open a context menu at a mouse click. Thus the object can be used as customized checkbox or as illustrated menu element.

For write access (“set”), values of the data types *integer* and *class* can be specified. Read access (“get”) always returns a value of the data type *integer*.

Value range

0 | *pushbutton*
The ***image*** object is selectable like a pushbutton.
2 | *checkbox*
The ***image*** object can be toggled between the states “active” and “inactive”.
Each state can be assigned its own image (***tile*** resource) with the *.picture[enum]* attribute.
3 | *menubox*
The ***image*** object has the behavior of a ***menubox***. A mouse click opens the context menu. In this style **no** *select* event is triggered.
When the context menu is open, the images *.picture[tile_active]* or *.picture[tile_active_mouse_over]* are displayed.
If a context menu is not available, no menu action is triggered. The state of the ***image*** object is nevertheless adjusted accordingly and the image belonging to the state is displayed from the *.picture[enum]* attribute.
The exact processing of the action depends on the respective window system. Also the focus is handled as it is usual for menus on the respective window system.

Availability
MICROSOFT WINDOWS only.

listview

This attribute determines the presentation mode of the ***listview***.

For write access (“set”), values of the data type *integer*, *string* and ***text*** resources can be specified.***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *integer*.

Value range

0 | *"icon"* | *"picture"*
Icon view with large icons.

Arrangement of list items first from left to right, then from top to bottom.
The icon is placed above the item's caption.

1 | *"smallicon"* | *"smallpicture"*
Icon view with small icons.
Arrangement of list items first from left to right, then from top to bottom.
The icon is placed to the left of the item's caption.

2 | *"list"*
List view with small icons.
Arrangement of list items first from top to bottom, then from left to right.
The icon is placed to the left of the item's caption.

3 | *"detail"* | *"report"*
List with small icons where the list items are arranged one below the other.
A small icon is shown to the left of the caption.
For each list item, detailed information is displayed in several columns.
The presentation resembles a table.

4 | *"tile"*
Display of the list items as tiles with large icons on the left and captions on the right.
Arrangement of the list items one below the other.

For invalid values, the default value *0* is used. However, the attribute value is not changed.

menuitem

The attribute defines the kind of menu item.

For write access (“set”), values of the data types *integer* and *class* can be specified. Read access (“get”) always returns a value of the data type *integer*.

For ***menuitems*** that are direct children of a ***window***, the *.style* attribute is ignored.

Value range

0 | *pushbutton*
Usual menu item, commonly used to trigger an action.

1 | *radiobutton*
Menu item with with two states and the behavior of a radiobutton.
In general, multiple menu items of this kind are used to select one from several mutually exclusive options via the menu.
A group of ***menuitems*** with *.style* = *radiobutton*, in which only one of the menu options can be activated, is delimited by the menu beginning, the menu ending or ***menu separators***.

2 | *checkbox*
Menu entry with two states and the behavior of a checkbox, which is usually used to make a setting via the menu.

menusep (Menu Separator)

The attribute defines the appearance of the menu separator.

Value range

0
Default separation line of the respective window system.

1
Single line.

2
Double line.

Note for the IDM for Windows

The *.style* attribute of the ***menu separator*** is ignored by the IDM FOR WINDOWS.

2.346.2 datetime, poptext and toolbar

datetime

The attribute determines the appearance and the method of operation at the ***datetime*** object.

Value range

poptext
To select a date, a calendar can be opened.

spinbox
Values can be set using a spinbox.

Note

Changing the attribute in the visible state leads to a reset of the object and should be avoided.

poptext (Combobox)

At the ***poptext***, the *.style* attribute determines how the list is displayed and whether the content of the input field can be edited by the user.

Value range

edittext
The list is closed and can be expanded and collapsed by the user. The content of the input field can be edited.
The user can select an existing item from the list or enter a different value.

listbox
The list is always open and the content of the input field can be edited.
The user can select an existing item from the list or enter a different value.

poptext
The list is closed and can be expanded and collapsed by the user. The content of the input field can **not** be edited.
The user can only select an item that already exists in the list.

toolbar

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.

Value range

notepage
: Toolbars use a dock area that lies between toolbars and inner area. “Tabbed toolbars” and “nested toolbars” are possible.

toolbar
: Conventional toolbar, corresponding to the well-known IDM toolbar.

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” at the window object in the “Object Reference” 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 2”).

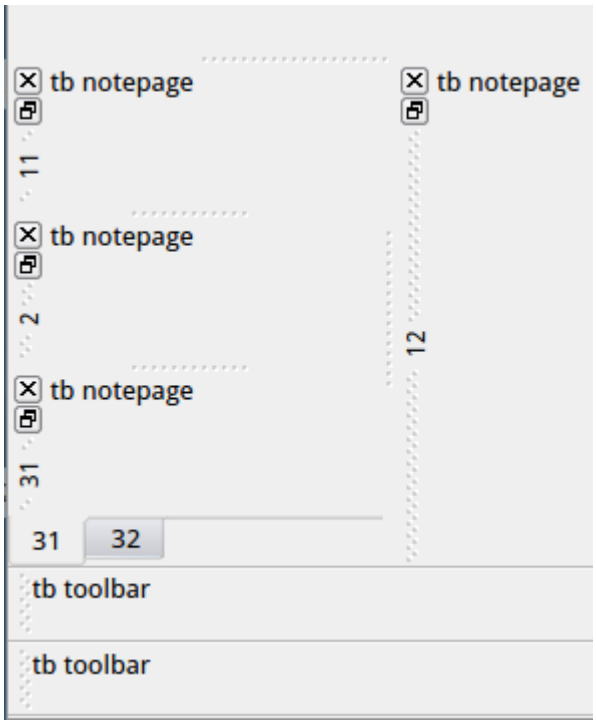


Figure 2: *Toolbars of different styles in the lower docking area (dock_down)*

Availability

The attribute *.style* at the ***toolbar*** is only supported by the IDM FOR QT.

See also

Objects toolbar, window

Attribute *.options[enum]*

2.346.3 filereq (File Requester)

The attribute determines the displayed system dialog and thus the purpose (mode) of the ***file requester***.

Value range

fr_directory
: System dialog for selecting a directory.

fr_load
: System dialog for opening files.

fr_save
: System dialog for saving files.

Particularities of the IDM for Windows

Different system dialogs are used for the different modes. Their characteristics such as labels (e.g. of the buttons), color and font are mainly predefined for the respective purpose and cannot be redefined.

The *fr_directory* mode is more restrictive in the handling of the *.directory* and *.pattern* attributes. Additionally, it does not permit the input of a non-existent directory.

2.346.4 font

For the ***font*** resource, the *.style* attribute determines which font style of the selected character set is used. To ensure better independence and combinations between face/slant and weight, the *.style* attribute has been changed to both *.face* and *.weight* attributes. However, the *.style* attribute can still be used.

Value range

face_default
: Regular, unchanged character representation.

face_light
: Light character representation.

face_normal
: Regular, unchanged character representation.

face_medium
: Font weight between *face_normal* and *face_demibold*.

face_demibold
: Font weight between *medium* and *bold*.

face_bold
Bold character representation.
face_black
Black character representation.
face_italic
Italic character representation.
In contrast to *face_oblique*, usually special, italic characters are used.
face_oblique
Inclined, slanted character representation.
In contrast to *face_italic* the oblique character representations are usually derived from the regular characters.
face_oblique is equivalent to *face_italic* on MICROSOFT WINDOWS.
face_roman
Upright, straight character representation (ignored on MICROSOFT WINDOWS).

2.346.5 spinbox

At the ***spinbox***, the attribute *.style* defines the type of the displayed values.

Value range

integer
The numerical values from the interval *.minvalue … .maxvalue* are displayed.
The current value can be queried and set with the *.curvalue* attribute.
string
The texts from the *.text[integer]* attribute are displayed.
The index of the current value can be queried and set with the *.activeitem* attribute.
void
The display in the child object is controlled by the application.
The values are switched as with *.style = integer*, except that the value displayed in the child object (***edittext*** or ***statictext***) is not changed automatically but must be set by the application.

2.347 .style[enum]

This attribute controls certain presentation chracteristics of the ***progressbar*** and ***treeview***.

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_style	Identifier: AT-style
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	progressbar, treeview

progressbar

The attributes determines the appearance of the progress indicator.

Index Range

style_continuous
: Defines whether the progress indicator is drawn continuously (*true*, default value) or as blocks (*false*).

Note on the IDM FOR QT
.style[continuous] is not supported. QT draws the progress bar depending on the used UI style (“cleanlooks”, “plastique” etc.). By default, a continuous bar is drawn in which a subdivision is indicated by color change. However, once you set *.fgc*, the bar will be drawn continuously in that color.

style_labeled
: Defines whether the progressbar is labeled with a integral percentage value (*true*, default) or not (*false*).
The label is only displayed with horizontal direction of the progressbar.

treeview

This attribute controls several characteristics of the ***treeview*** presentation. All these characteristics are turned off by default.

Index Range

style_buttons
: All nodes, except for the top-level nodes, are prefixed with a +/- button to expand and collapse the sub-tree.

style_lines
: Lines are drawn between parent and child nodes.

style_root
: Together with *style_lines*, lines are drawn between the top-level nodes.
Together with *style_buttons*, the top-level nodes are prefixed with +/- buttons too.

The screenshot below shows the appearances produced by the different styles and their combinations.

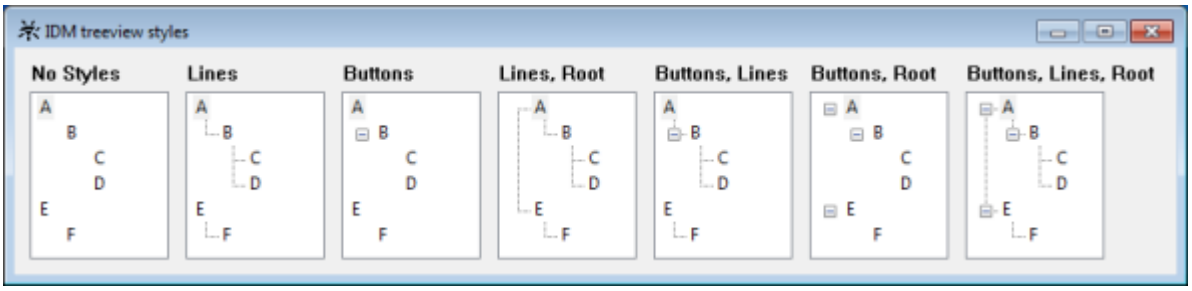


Figure 3: *Different treeview appearances*

Remark

The three styles are only supported by the IDM for Microsoft Windows. The textual simulation of the treeview on other platforms just offers the setting *.style[style_buttons]*. This is similar to the treeview labeled “Buttons, Root” in the screenshot above. “+” and “-“ here as well are used to display expanded and collapsed nodes with children; “.” is used to display nodes without children.

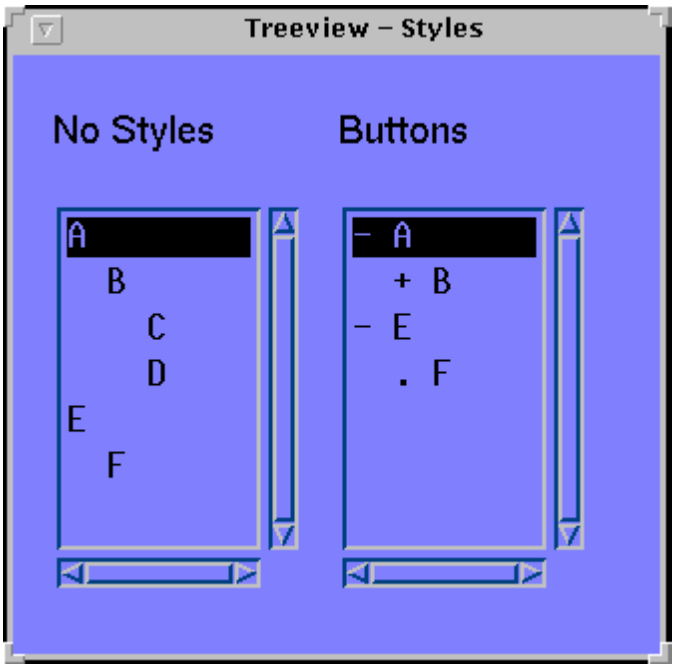


Figure 4: *treeview on Motif*

2.348 .subcontrol[integer]

This attribute returns or sets the I-th ***subcontrol*** of a ***control*** or ***subcontrol*** object.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_subcontrol
Data type: DT_object

COBOL
Identifier: AT-subcontrol
Data type: DT-object

Classification
hierarchy attribute

2.349 .subcontrolcount

This attribute queries the number of ***subcontrols*** in a ***control*** or ***subcontrol*** object.

Definition

Data type
integer

Access
get

C
Identifier: AT_subcontrolcount
Data type: DT_integer

COBOL
Identifier: AT-subcontrolcount
Data type: DT-integer

Classification
hierarchy attribute

2.350 .sysmodal

The attribute specifies whether a dialogbox or ***messagebox*** is displayed in front of all other windows on the desktop.

With *.sysmodal = true*, the respective IDM dialog is blocked until the dialogbox or ***messagebox*** is acknowledged by the user, i.e. the user cannot access a window of this application. Other applications, however, can still be operated, but the dialogbox or ***messagebox*** remains permanently in the foreground, so it cannot be covered.

With *.sysmodal = false* (default value), the modality is only restricted to the respective IDM dialog. The dialogbox or ***messagebox*** can therefore be hidden by other applications.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_sysmodal Data type: DT_boolean	Identifier: AT-sysmodal Data type: DT-boolean

Default value
false

Classification	*Objects*
object-specific attribute	messagebox, window

Note for the IDM for Motif

The IDM tries to keep the topmost modal window (**dialogbox**, ***filereq*** or ***messagebox***) in front of other windows of the respective IDM application. The IDM cannot guarantee an order with regard to other windows and dialogboxes, since the “stacking order” is determined by the Window Manager.

Note for the IDM for Qt

The attribute *.sysmodal* is **not** supported by the ***messagebox*** of the IDM FOR QT.

2.351 .systemerror

When an error occurs while using the application functionality, this attribute is set with the error string that the system provides and can be queried in the event of an error. It is only supported for the error types *error_network* and *error_file*, in all other cases an empty string is returned.

Definition

Data type	*Access*
string	get

C	*COBOL*
Identifier: AT_systemerror Data type: DT_string	Identifier: AT-systemerror Data type: DT-string

Classification	*Objects*
object-specific attribute	application

It should be noted that the error string cannot be influenced by IDM in terms of its format and language. Neither can IDM ensure that the system provides an adequate error string for each error condition.

See also

Attribute *.errorcode*

2.352 .systemid

This attribute returns the public identifier of the DOM node. The attribute is only available, when the attribute *.nodetype* has either the value *nodetype_entity* or *nodetype_notation*.

Definition

Data type
string

Access
get

C
Identifier: AT_systemid
Data type: DT_string

COBOL
Identifier: AT-systemid
Data type: DT-string

Classification
object-specific attribute

Objects
doccursor

The attribute is available for the XML Cursor. Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

See also

Attribute *.nodetype*

2.353 .tabalignment

With the help of this attribute, on MICROSOFT WINDOWS it is defined for the ***notebook*** object, how the text in its tabs is aligned.

The IDM FOR MOTIF does **not** evaluate this attribute.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_tabalignment	Identifier: AT-tabalignment
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	notebook

Value range

0
: centered
If *.majortabwidth* and *.majortabheight* are both set to *0*, the width and height will be set individually for each tab.

1 (default)
: left justified
All tabs get the same width and height.

Note

A text that is too long for the set tab width will always be left justified (never centered), regardless of the settings made.

See also

Attributes *.majortabheight*, *.majortabwidth*

2.354 .tabshape

This attribute defines the form of major and minor tabs.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_tabshape	Identifier: AT-tabshape
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	notebook

Value range

shape_square (default)
: The tabs are rectangular.

shape_rounded
: The tabs have rounded corners.

shape_polygon
: The tabs have beveled corners.

shape_chamfered
: The tabs are chamfered (IDM FOR WINDOWS only).

2.355 .tabtype

This attribute defines the type of tabs in the object ***notepage***.

Definition

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_tabtype Data type: DT_enum	Identifier: AT-tabtype Data type: DT-enum

Classification	*Objects*
object-specific attribute	notepage

Value range

tab_major (default)
: The tab is is a major tab or main index.

tab_minor
: The tab is is a minor tab or side index.

2.356 .target

This attribute determines the object behavior as target of a Drag & Drop or clipboard operation. Resources of the type ***target*** are used as values.

With the ***doccursor***, the attribute contains 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*.

Definition

Data type	*Access*	*changed event*
object [***target***]	get, set	yes
string (***doccursor***)	get (***doccursor***)	

C
Identifier: AT_target
Data type: DT_target
Data type: DT_string (***doccursor***)

COBOL
Identifier: AT-target
Data type: DT-target
Data type: DT-string (***doccursor***)

Classification
standard attribute

doccursor

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

Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.357 .terminal

In the ***setup*** object, if used with ALPHAWINDOWS, this attribute requests the terminal type, e.g. *vt200*.

Definition

Data type
string

Access
get

C
Identifier: AT_terminal
Data type: DT_string

COBOL
Identifier: AT-terminal
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.358 .terminaltype

In the ***setup*** object, if used with ALPHAWINDOWS, this attribute queries the type of terminal, e.g. */dev/tty*.

Definition

Data type
string

Access
get

C
Identifier: AT_terminaltype
Data type: DT_string

COBOL
Identifier: AT-terminaltype
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.359 .text

This attribute defines the static text of an object.

With an ***image***, the text to appear with the image is defined.

In ***notepage*** the text which appears in the status line is defined with *.text* (default 0) If no text is defined, the notepage contains no status line.

With the ***doccursor***, the attribute contains 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.

The *.text* attribute must not be used for a ***statictext*** that is the child of a ***spinbox***. Its value is set exclusively though the corresponding attributes of the ***spinbox*** object itself.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C
Identifier: AT_text
Data type: DT_string, DT_text

COBOL
Identifier: AT-text
Data type: DT-string, DT-text

Classification
text attribute

doccursor

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.

Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.360 .text[enum]

This attribute defines the labels for the dialog elements of the file dialogs (***filereq*** object).

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_text	Identifier: AT-text
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	filereq

Index Range

frt_cancel
: Caption of the “Cancel” button.

frt_directories
: Caption of the directory list.

frt_files
: Caption of the file list.

frt_help
: Caption of the “Help” button.

frt_nomatch
: Text that will be displayed in the file list if there is no item matching the name pattern.

frt_ok
: Caption of the “OK” button.

frt_path
: Caption of the input field for the directory path.

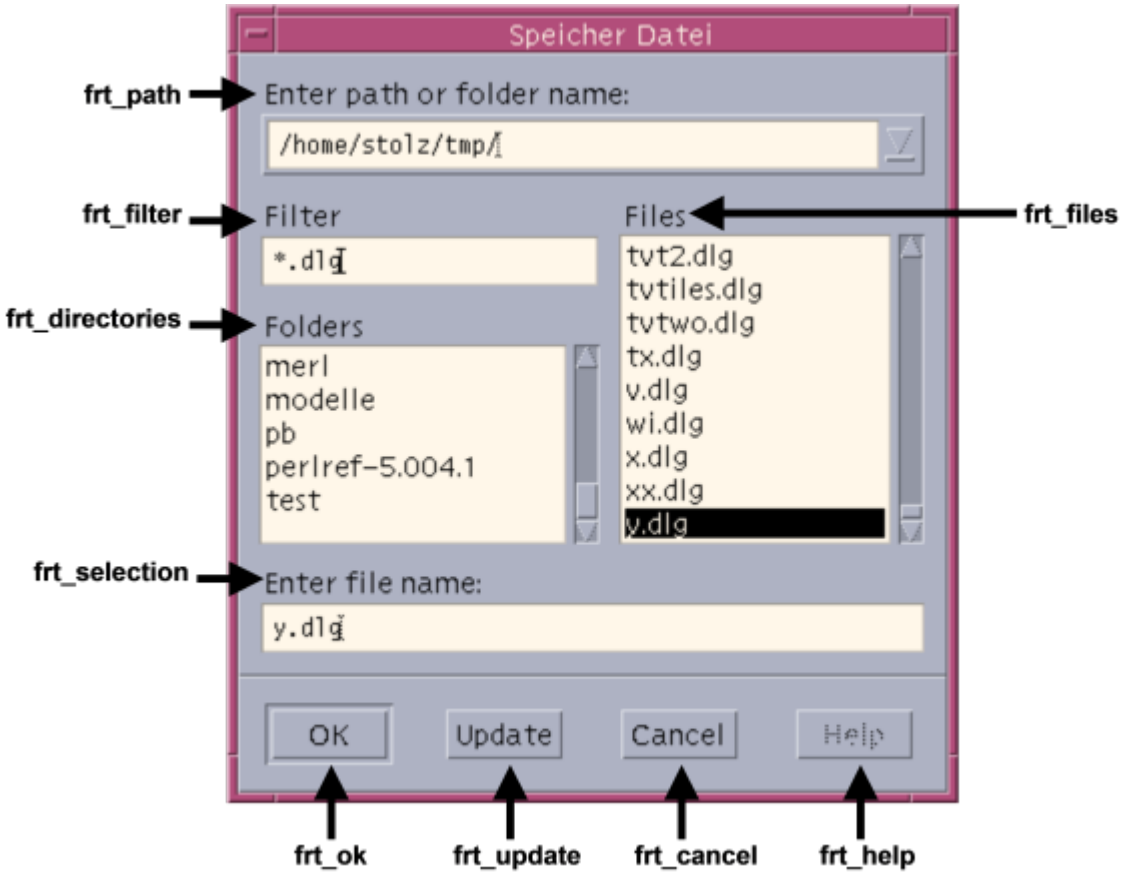
frt_pattern
: Caption of the input field for the name pattern.

frt_selection
: Caption of the input field for the selected item.

frt_update
: Caption of the “Update” button.

Particularities

This attribute is only supported on MOTIF. The image below shows the mapping of the texts to the several dialog elements. This mapping applies to file and directory dialogs.


Speicher Datei
frt_path
Enter path or folder name:
/home/stolz/tmp/
frt_filter
Filter
*.dlg
Files
frt_files
tvt2.dlg
tvtiles.dlg
tvtwo.dlg
tx.dlg
v.dlg
wi.dlg
x.dlg
xx.dlg
y.dlg
frt_directories
Folders
merl
modelle
pb
perlref-5.004.1
test
frt_selection
Enter file name:
y.dlg
OK
Update
Cancel
Help
frt_ok
frt_update
frt_cancel
frt_help

2.361 .text[integer]

This attribute can set single texts of a ***poptext***.

If the value of the index is >= *1*, and <= *.itemcount*, you will get the text of the entry. If the value is *0*, you will get the currently chosen text.

For a ***spinbox***, the attribute defines the values that are cycled through and displayed in the associated ***edittext*** or ***statictext*** when *.style* = *string*.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_text	Identifier: AT-text
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	poptext, spinbox

2.362 .textbgc

With this attribute the background color of an object's caption can be set, provided the object supports this. When the value is set to *null* (no value set), the default color of the system is used.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C
Identifier: AT_textbgc
Data type: DT_color

COBOL
Identifier: AT-textbgc
Data type: DT-color

Classification
object-specific attribute

2.363 .textfgc

With this attribute the color of an object's caption can be set, provided the object supports this. When the value is set to *null* (no value set), the default color of the system is used.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_textfgc Data type: DT_color	Identifier: AT-textfgc Data type: DT-color

Classification	*Objects*
object-specific attribute	progressbar

2.364 .textwidth

The *.textwidth* attribute can be used to set and query the maximum text width in pixels for an ***edittext*** with formatting (RTF edittext).

.textwidth is an attribute of the RTF edittext and like the RTF mode of the edittext is only available on MICROSOFT WINDOWS.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_textwidth	Identifier: AT-textwidth
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	edittext

If *.textwidth* is set to *0*, the maximum text width is determined like this:

- If *.hsb_visible* = *true* the text width is determined by the widest line in the formatted text.
- If *.hsb_visible* = *false* the text is automatically wrapped and the text width will be the width of the edittext without margins.

A value > *0* sets the maximum text width to this value. If required, a horizontal scrollbar is available whose display is controlled by the attribute *.hsb_visible*.

See also

Chapter “Editable Text with Formatting (RTF edittext)” in the “Object Reference”

2.365 .tile

With **grouping objects**, the *.tile* attribute defines a ***tile*** resource, which is drawn as background of the object. The *.tilestyle* attribute determines how the background image is drawn.

In the ***setup*** object, this attribute queries or sets the tile variant.

Definition

Data type	*Access*	*changed event*
object [***tile***] *integer* (***setup***)	get, set	yes no (***setup***)

C	*COBOL*
Identifier: AT_tile Data type: DT_tile Data type: DT_integer (***setup***)	Identifier: AT-tile Data type: DT-tile Data type: DT-integer (***setup***)

Classification	*Objects*
object-specific attribute	groupbox, layoutbox, notebook, notepage, setup, spinbox, splitbox, statusbar, toolbar, window

Note for the Window Object

A background image only affects the actual window pane but not the titlebar, menubar, statusbar and docked toolbars.

See also

Attribute *.tilestyle*

2.366 .tiledpi

This attribute allows the user to specify for which DPI resolution the application's tiles were designed. This is especially necessary if graphics were created for a specific (and different) resolution and you want to make sure that they are displayed correctly in the application. The size of an image/pattern is then converted to the currently valid DPI value based on this value.
Likewise, it is of course also possible to create the graphics for a high resolution and then have them scaled down by the IDM for lower DPIs.

Definition

Data type	*Access*	*changed event*
integer (may be *0*)	get, set	no

C	*COBOL*
Identifier: AT_tiledpi	Identifier: AT-tiledpi
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

The .tiledpi attribute was introduced for IDM version A.06.03.a in order to allow HighDPI support.

Note

This attribute can be set only after IDM initialization, but not in the running application.

Availability

Since IDM version A.06.03.a

See also

Chapter „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

2.367 .tilestyle

For **grouping objects**, the attribute *.tilestyle* controls the layout of the background image.

This attribute defines the position of the picture (*.picture*) inside the ***image*** object. In case there is no picture, this is handled like there was a picture whose width and height are *0*.

Data type	*Access*	*changed event*
enum	get, set	yes

C	*COBOL*
Identifier: AT_tilestyle	Identifier: AT-tilestyle
Data type: DT_enum	Data type: DT-enum

Classification
layout attribute

Grouping Objects

Value range

tilestyle_tiled
: The background image is repeated horizontally and vertically until the background is covered entirely.

tilestyle_centered (not on MOTIF)
: The background image is drawn in the center. If a virtual size is set, this will be used to calculate the position.

tilestyle_stretched (not on MOTIF)
: The background image is resized to cover the entire plain. If a virtual size is set, the image will cover the complete virtual plain.

tilestyle_parent_tile
: The setting of the parent object is taken on. The object appears to be transparent. This value is not applicable for ***statusbar***, ***toolbar*** and ***window***.

Remark

To avoid positioning according to virtual sizes with *tilestyle_centered* and filling the whole virtual plain with *tilestyle_stretched*, *tilestyle_parent_tile* can be used and the background image can be set on the parent object (possibly inserting a ***groupbox*** as an intermediate parent.

image

Value range

tilestyle_icon
: The picture is placed above the text with a distance defined by the attribute *.spacing*. The picture is centered horizontally; the text is positioned horizontally inside the ***image*** object according to the *.alignment* attribute (default: centered). Vertically picture and text are treated as a unit which gets placed in the middle. If the picture is scalable, the text is positioned at the

bottom of the ***image*** and the picture covers the remaining space.

tilestyle_left

The picture is placed on the left edge of the ***image*** object. The display area for the text starts to the right of the picture, with a distance defined by the attribute *.spacing*, and extends to the right edge of the ***image***. The text is positioned horizontally inside its display area according to the *.alignment* attribute. Both picture and text are vertically centered. If the picture is scalable, the text is placed on the right edge of the ***image***, regardless of the *.alignment* attribute. The picture then covers the remaining space.

tilestyle_right

The picture is placed on the right edge of the ***image*** object. The display area for the text starts to the left of the picture, with a distance defined by the attribute *.spacing*, and extends to the left edge of the ***image***. The text is positioned horizontally inside its display area according to the *.alignment* attribute. Both picture and text are vertically centered. If the picture is scalable, the text is placed on the left edge of the ***image***, regardless of the *.alignment* attribute. The picture then covers the remaining space.

tilestyle_top

The picture is placed on the top edge of the ***image*** object. The text is displayed below the picture with a distance defined by the attribute *.spacing*. The picture is centered horizontally; the text is positioned horizontally inside the ***image*** object according to the *.alignment* attribute. If the picture is scalable, the text is positioned at the bottom of the ***image*** and the picture covers the remaining space.

tilestyle_bottom

The picture is placed on the bottom edge of the ***image*** object. The text is displayed above the picture with a distance defined by the attribute *.spacing*. The picture is centered horizontally; the text is positioned horizontally inside the ***image*** object according to the *.alignment* attribute. If the picture is scalable, the text is positioned at the top of the ***image*** and the picture covers the remaining space.

tilestyle_background

The picture is displayed as background. The text is shown in front of the picture. The picture is centered horizontally; the text is positioned horizontally inside the ***image*** object according to the *.alignment* attribute. Both picture and text are vertically centered. If the picture is scalable, it covers the complete space provided by the ***image*** object.

See also

Attributes *.alignment*, *.spacing*

2.368 .timeout

The attribute defines a period of time (in seconds) after which a timeout is to come into effect. The time is specified according to the pattern when a time increment for the ***timer*** object is defined.

If .timeout of the dialog is not *0*, the IDM sets up the timer every time it reads non-blocking information from the window system. When the timer comes into effect, a *deactivate* event for the ***dialog*** is sent. Every event with the exception of another timer resets the timeout.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_timeout	Identifier: AT-timeout
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	dialog

Timeouts enable the IDM user to cancel critical operations after specified given period of time or to start other actions. A timeout always refers to user actions, not to internal operations. For example, a timeout may be defined when a critical window is opened or a connection to another host is established. The timeout comes into effect when the user has not worked with the program for a certain period of time. The application is then free to close the window or to release the connection.

Remark

Since several dialogs can be defined, you can obtain several timeouts that are defined for these dialogs. In this case, the shortest timeout (> *0*) will be taken into consideration for all timeouts defined for further dialogs.

See also

Object timer

2.369 .title

This is the text of an object's title.

In the object ***notepage*** this attribute defines the labeling of tabs (default = *0*). You will only get a text if the value is <> *0*. The labeling may be in form of a text or of a bitmap.

Definition

Data type	*Access*	*changed event*
string, *object* [***text***]	get, set	yes

C
Identifier: AT_title
Data type: DT_string, DT_text

COBOL
Identifier: AT-title
Data type: DT-string, DT-text

Classification
text attribute

Note for the notepage

Please note that you should usually indicate a text. Depending on the window system, it may be that only certain bitmaps can be used or that there always has to be a text.

Note for the filereq on Microsoft Windows

In the directory selection (mode *fr_directory*), the title appears only as heading and not as window title.

2.370 .titlebar

This attribute defines if the window shall have a title bar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_titlebar	Identifier: AT-titlebar
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

Note

On MOTIF, *.iconifyable* must also be set to *false* so that no title bar appears when using the MWM window manager.

2.371 .titlebgc

This attribute defines the background color of the window title bar.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_titlebgc	Identifier: AT-titlebgc
Data type: DT_color	Data type: DT-color

Classification	*Objects*
object-specific attribute	window

Remark

This attribute has no effect on current supported window systems. The appearance of the window frame cannot be influenced by an application.

2.372 .titlefgc

This attribute defines the foreground color of the window title bar.

Definition

Data type	*Access*	*changed event*
object [***color***]	get, set	yes

C	*COBOL*
Identifier: AT_titlefgc Data type: DT_color	Identifier: AT-titlefgc Data type: DT-color

Classification	*Objects*
object-specific attribute	window

Remark

This attribute has no effect on current supported window systems. The appearance of the window frame cannot be influenced by an application.

2.373 .today

The attribute determines whether the current date is displayed at the bottom of the fold-out calendar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_today Data type: DT_boolean	Identifier: AT-today Data type: DT-boolean

Default value	*Inheritance*
true	yes

Classification	*Objects*
object-specific attribute	datetime

Value range

false
: The current date is not displayed in the calendar.

true
: The current date is displayed at the bottom of the calendar.

2.374 .todaymarker

This attribute controls whether the current date is marked in the fold-out calendar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_todaymarker Data type: DT_boolean	Identifier: AT-todaymarker Data type: DT-boolean

Default value	*Inheritance*
true	yes

Classification	*Objects*
object-specific attribute	datetime

Value range

false
: The current date is not marked in the calendar.

true
: The current date is marked in the calendar.

2.375 .toolbar

This attribute returns the toolbar an object belongs to.

When the object is a direct or indirect child of a toolbar, this toolbar is returned; otherwise *null* is returned.

Definition

Data type
object

Access
get

C
Identifier: AT_toolbar
Data type: DT_object

COBOL
Identifier: AT-toolbar
Data type: DT-object

Classification
hierarchy attribute

2.376 .toolbar[integer]

The *.toolbar[]* vector contains all toolbars of a window.

A single toolbar can be accessed through its index. The value range of the index is *1* to *.toolbarcount*.

Definition

Data type	*Access*	*changed event*
object	get, set	yes

C
Identifier: AT_toolbar
Data type: DT_object

COBOL
Identifier: AT-toolbar
Data type: DT-object

Classification
hierarchy attribute

2.377 .toolbarcount

This attribute of the window returns the number of toolbars that the window has.

Definition

Data type
integer

Access
get

C
Identifier: AT_toolbarcount
Data type: DT_integer

COBOL
Identifier: AT-toolbarcount
Data type: DT-integer

Classification
hierarchy attribute

2.378 .toolhelp

When this attribute is set, a small pop-up with the specified text is displayed as soon as the mouse pointer rests on the respective object without motion. The pop-up is hidden automatically after a certain time interval or after the mouse pointer has been moved.

The pop-ups may be used to show a short explanation for an object.

Definition

Data type	*Access*	*changed event*
object [***text***]	get, set	yes

C
Identifier: AT_toolhelp
Data type: DT_text

COBOL
Identifier: AT-toolhelp
Data type: DT-text

Classification
standard attribute

Support of attribute by objects

- canvas
- checkbox
- edittext
- image
- listbox
- poptext
- pushbutton
- radiobutton
- rectangle (MICROSOFT WINDOWS only)
- scrollbar
- spinbox
- statictext
- tablefield
- treeview

Example

```
child pushbutton PbOK
{
  .text      "OK";
  .toolhelp  "accepts file";
}
```

Notes on the IDM for Motif

» On Motif background and foreground colors can be set through X resources, e.g.

```
IDM*ToolHelpLabel.Background: red
IDM*ToolHelpLabel.Foreground: blue
```

» As default colors the IDM uses “LightYellow” for the background and “Black” for the foreground.

» The X toolkit name of the toolhelp object is “ToolHelpLabel”.

Notes on the IDM for Windows

“WM_MOUSEMOVE” events that are provoked from another application can interrupt the showing and hiding of the toolhelp. For instance, *.toolhelp* may open after its change, although the mouse was not moved. It may also happen that a visible toolhelp may not be hidden again or a toolhelp may not be shown at all.

An application-driven display of toolhelps is not intended. If this is desired, the function **DM_GetToolkitData()** can be used to query the attribute *AT_toolhelp* of the ***setup*** object to get the Windows handle of the “tooltip control” that the ISA Dialog Manager uses for display.

The toolhelp can be opened with the following example code:

```
#include <windows.h>
#include <commctrl.h>
#include IDMuser.h

void DML_default DM_ENTRY OpenToolhelp __0() {
  DM_ID idSetup = DM_ParsePath(
    (DM_ID) 0, (DM_ID) 0, "setup", 0, 0);

  if (idSetup != (DM_ID) 0) {
    HWND hwndToolhelp = (HWND) DM_GetToolkitData(
      idSetup, AT_toolhelp);

    if (hwndToolhelp != (HWND) 0) {
      SendMessage(hwndToolhelp, TTM_POPUP,
        (WPARAM) 0, (LPARAM) 0);
    }
  }
}
```

The “OpenToolhelp()” function must be defined in the dialog respectively. The Windows message TTM_POPUP is available from version 6 of the Common Controls Library (**comctl32.dll**).

2.379 .toolkit

In the ***setup*** object, this attribute requests the type of toolkit.

Definition

Data type	*Access*
enum	get

C	*COBOL*
Identifier: AT_toolkit	Identifier: AT-toolkit
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	setup

Value range

toolkit_motif
: The toolkit is MOTIF.

toolkit_qt
: The toolkit is QT.

toolkit_windows
: The toolkit are the “Common Controls” of MICROSOFT WINDOWS.

2.380 .toolkit_string

In the ***setup*** object, this attribute queries the toolkit.

Definition

Data type
string

Access
get

C
Identifier: AT_toolkit_string
Data type: DT_string

COBOL
Identifier: AT-toolkit-string
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.381 .toolkit_version

In the ***setup*** object, this attribute queries the version of the toolkit.

Definition

Data type	*Access*
integer	get

C	*COBOL*
Identifier: AT_toolkit_version Data type: DT_integer	Identifier: AT-toolkit-version Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

Note on the IDM for Windows

As of IDM version A.06.01.a the attribute no longer returns the Windows version for which the IDM was compiled, but the version of the “Common Controls DLL” (**comctl32.dll**). This DLL determines the appearance of the user interface objects, thus now a value is returned that provides information about the appearance and layout (e.g. border widths) of the interface.

The version is coded as a decimal number:

major_version * 100 + minor_version

In addition, the following predefined values may occur:

0	No “Common Controls DLL” available.
400	Version could not be determined since the “Common Controls DLL” is older than version *4.71*.
471	“Common Controls DLL” has version *4.71* or version could not be determined because an error occurred when calling **DllGetVersion**.
582	“Common Controls DLL” has version *5.82* or version is *6.00* or higher and “Visual Styles” are disabled.
>= *600*	“Visual Styles” are enabled.

2.382 .top_most

This attribute defines that a ***window*** shall be displayed up front. Windows with *.top_most = true* are arranged as high as possible in the z-order of all top-level windows, e.g. only other windows with *.top_most = true* may be displayed further in front. The attribute has no effect for child windows.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_top_most	Identifier: AT-top-most
Data type: DT_boolean	Data type: DT-boolean

Default value
false

Classification	*Objects*
object-specific attribute	window

2.383 .topitem

The attribute *.topitem* specifies the item which shall be displayed at the top of a ***listbox*** or ***treeview***.

The value of *.topitem* is corrected if the entry set as *.topitem* cannot be scrolled to the top of the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_topitem	Identifier: AT-topitem
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listbox, treeview

Note on the treeview

Since the *.topitem* entry is always visible, the sub-tree containing this entry cannot be closed programmatically in a visible ***treeview***. This sub-tree may be closed interactively because a higher-level entry must be accessible for this purpose. This ensures that the current *.topitem* is always above the set *.topitem*.

2.384 .tracefile

With this attribute of the ***setup*** object, the absolute path of the **trace file** can be queried at runtime.

Definition

Data type
string

Access
get

C
Identifier: AT_tracefile
Data type: DT_string

COBOL
Identifier: AT-tracefile
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.385 .tracetime

By using this attribute it is possible to influence the time output into the tracefile via the setup object during runtime.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_tracetime	Identifier: AT-tracetime
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

Value range

0

No times are logged in the trace file.

1

This value indicates start time mode. In this mode all start and end times are logged. The time needed for a single structure may then be calculated with the difference. In this mode only the system and user time will be considered.
The times are given in format [hh:mm:ss:uuu] at the beginning of line:

- hh = hours
- mm = minutes
- ss = seconds
- uuu = milliseconds

2

This value indicates the trace time mode. In this mode the time difference to the last logged call is given. It is thus possible to easily recognize how much time is needed for individual actions. In this mode the time difference to the last trace output is given in the format [sss:uuu] at the beginning of line:

- ss = seconds
- uuu = milliseconds

3

This value specifies the real-time mode. In this case the real time is indicated for each action to be logged in the trace file.
In this mode the real time is given in format [hh:mm:ss] at the beginning of line:

- hh = hours
- mm = minutes
- ss = seconds

Remark

The option **-IDMtracetime <nr>** additionally allows to set the trace mode via the command line.

2.386 .tracing

This attribute defines whether the tracing shall be carried out completely or not.

.tracing is only active if the command line option **-IDMtracefile** has been given. Thus, you can only limit the tracing if a trace file is created at all.

Warning

This attribute should only be used temporarily and carefully!

.tracing is meant to ensure that for example passwords which are passed on by the Dialog Manager to the application functions, do not appear in the trace file.

Definition

Data type	*Access*	*changed event*
boolean	get, set	no

C	*COBOL*
Identifier: AT_tracing	Identifier: AT-tracing
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	setup

Example

```
setup.tracing := false;  // switch off tracing
// actions
setup.tracing := true;   // switch on tracing again
```

.tracing[string]

When the attribute is indexed with a string that represents a trace code, the tracing of particular operations (determined by the trace code) is turned on (*true*) and off (*false*).

See also

Chapter “Tracing” in manual “Development Environment”

2.387 .trailingdates

This attribute controls whether the fold-out calendar displays the days adjacent to the current month.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_trailingdates Data type: DT_boolean	Identifier: AT-trailingdates Data type: DT-boolean

Default value	*Inheritance*
true	yes

Classification	*Objects*
object-specific attribute	datetime

Value range

false
: The calendar only shows the days of the set month.

true
: The calendar also shows days of the previous and following month, so that the weeks are always displayed in full.

2.388 .transformer[integer]

The ***transformers*** of an object can be accessed through this attribute. The attribute is indexed with the object index (similar to *.child*).

This attribute is used in the same way as the *.record* attribute.

The attribute is available on the classes, which can also have ***records*** as children.

Definition

Data type	*Access*	*changed event*
object	get, set	no

C
Identifier: AT_transformer
Data type: DT_transformer

COBOL
Identifier: AT-transformer
Data type: DT-transformer

Classification
standard attribute

2.389 .transport

This attribute defines the internal transport mechanism which is used by the communication layer; possible are *"tcpip"*, *"dynlib"* etc.

To establish a connection via **SSL**, the specification of the protocol with *.transport* can be prefixed by the scheme *"ssl://"* (example `.transport "ssl://tcpip";`). If no protocol is preset, "://" may be omitted (i.e. `.transport "ssl";`).

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_transport Data type: DT_string	Identifier: AT-transport Data type: DT-string

Classification	*Objects*
object-specific attribute	application

Example

Without SSL

```
application Appl1 {

  .connect "localhost:4711";
}

application Appl2 {
  .transport "tcpip-winsock";
  .connect "localhost:4711";
}
```

With SSL

```
application Appl1 {
  .transport "ssl";
  .connect "localhost:4711";
}

application Appl2 {
  .transport "ssl://tcpip-winsock";
  .connect "localhost:4711";
}
```

Remarks

- The attributes *.transport*, *.connect*, and *.exec* can only be changed if *.active* is set at *false*.
- The scheme *"ssl://"* can also be specified at the *.connect* attribute. If a scheme is given at both attributes, these must be identical. A once specified scheme *"ssl://"* cannot be turned off again.

2.390 .type

This specifies the data type of a global variable or a user-defined attribute.

For **global variables** and **user-defined attributes**, the attribute defines the data type of the variable or the user-defined attribute is returned.

With user-defined attributes, the request is made by indicating the attribute to be queried as index, e.g. *.type[<user-defined attribute>]*.

For **rules** and **functions**, the attribute returns

- with index *0* or without index:
  the return value type
- with index >= *1*:
  type of the respective parameter

If no parameter exists for the index, the return value is *void*.

At the ***thisevent*** object, the attribute queries the type of event.

Definition

Data type	*Access*	*changed event*
datatype	get, set	yes
enum (***thisevent***)	get (***function***, ***rule***, ***thisevent***)	

C
Identifier: AT_type
Data type: DT_type
Data type: DT_enum (***thisevent***)

COBOL
Identifier: AT-type
Data type: DT-type
Data type: DT-enum (***thisevent***)

Classification
plain attribute

See also

Chapters “Functions” and “Named Rules (Subprograms)” in manual “Rule Language”

Chapter “Event Object thisevent” in manual “Rule Language”

2.391 .typescope

This attribute retrieves the validity range for return types of user-defined functions and rules.

It is available for the object classes ***function*** and ***rule***.

Definition

Data type	*Access*
anyvalue	get

C	*COBOL*
Identifier: AT_typescope	Identifier: AT-typescope
Data type: DT_anyvalue	Data type: DT-anyvalue

Classification	*Objects*
object-specific attribute	***function***, ***rule***

See also

Attributes *.indexscope[attribute]*, *.scope[attribute]*, *.typescope[integer]*

Chapter “Validity Range for Better Type Checking” in manual “Rule Language”

2.392 .typescope[integer]

This attribute queries the validity range for the parameters of user-defined functions and rules.

The access to the validity ranges for parameters of named rules, event rules and functions happens via indexing with the data type *integer* in the range of *1 … .count[.typescope]*.

Definition

Data type	*Access*
anyvalue	get

C	*COBOL*
Identifier: AT_typescope	Identifier: AT-typescope
Data type: DT_anyvalue	Data type: DT-anyvalue

Classification	*Objects*
object-specific attribute	***function***, ***rule***

See also

Attributes *.indexscope[attribute]*, *.scope[attribute]*, *.typescope*

Chapter “Validity Range for Better Type Checking” in manual “Rule Language”

2.393 .userdata

Information of any data type can be stored in this attribute. The application controls the process. The IDM merely provides the necessary memory and administration.

Definition

Data type	*Access*	*changed event*
anyvalue	get, set	yes

C
Identifier: AT_userdata
Data type: DT_anyvalue (determined by parameter type)

COBOL
Identifier: AT-userdata
Data type: DT-anyvalue (determined by parameter type)

Classification
standard attribute

Remark

The attribute *.userdata* can be used to store any values. This means that also other objects like colors, numbers and texts are stored. This attribute is interpreted as a *void* variable since it always accepts the data type of the corresponding contents.

2.394 .userdata[integer]

With this attribute the userdata for an entry of a ***listbox*** or a ***poptext*** may contain any kind of data.

Definition

Data type	*Access*	*changed event*
anyvalue	get, set	yes

C	*COBOL*
Identifier: AT_userdata	Identifier: AT-userdata
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	listbox, poptext

2.395 .userdata[index]

In a ***tablefield***, this attribute assigns any type of data to a field indicated by [row, column].

Definition

Data type	*Access*	*changed event*
anyvalue	get, set	yes

C	*COBOL*
Identifier: AT_userdata	Identifier: AT-userdata
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	tablefield

2.396 .username

This attribute can be used to define the user name for starting the application side with the RSH or SSH protocol.

The attribute is evaluated if no user name is given with the command in the *.exec* attribute. This attribute provides an alternative when there are problems to properly construct the command in the *.exec* attribute.

Definition

Data type
string

Access
get, set

changed event
yes

C
Identifier: AT_username
Data type: DT_string

COBOL
Identifier: AT-username
Data type: DT-string

Inheritance
yes

Classification
object-specific attribute

Objects
application

Availability

Since IDM version A.06.02.g

See also

Attribute .password

2.397 .userplaced

This attribute ensures that the window can be arbitrarily positioned on opening. It defines whether the window is to appear at a specified position or if the user has to interactively place it.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_userplaced	Identifier: AT-userplaced
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	window

2.398 .uuid

This attribute is used to give an unambiguous ID to a ***control*** object. This ID is generated using the program **guidgen.exe**.

If the ***control*** object is used as an OLE Client, the UUID of the server can be stored in this attribute. This attribute then replaces the *.name* attribute, in which the server name is normally stored.

Definition

Data type
string

Access
get

C
Identifier: AT_uuid
Data type: DT_string

COBOL
Identifier: AT-uuid
Data type: DT-string

Classification
object-specific attribute

Objects
control

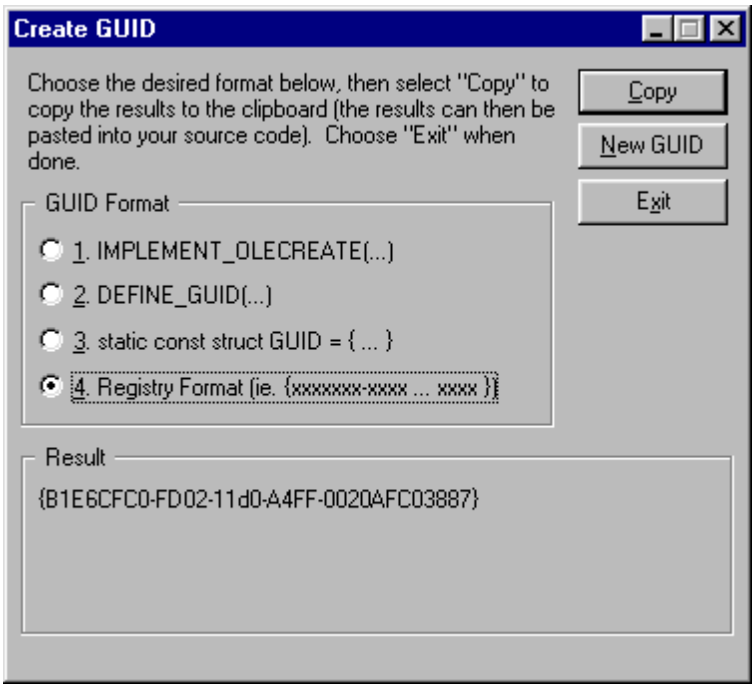


Figure 5: *Generation of GUID*

2.399 .value

This attribute queries or sets the current value of the object.

Definition

Data type	*Access*	*changed event*
*string, object (**text**)* data type of current content (***variable***)	get, set	yes no (***doccursor***)

C	*COBOL*
Identifier: AT_value Data type: DT_string, DT_text Data type: data type of current content (***variable***)	Identifier: AT-value Data type: DT-string, DT-text Data type: data type of current content (***variable***)

Inheritance
yes (***datetime***, ***filereq***)
no (***doccursor***, ***variable***)

Classification	*Objects*
object-specific attribute	datetime, doccursor, filereq, ***variable***

2.399.1 datetime

This attribute defines the value of the ***datetime*** object.

For write access (“set”), values of the data type *string* and ***text*** resources can be specified. ***text*** resources are automatically converted to *string*.

Read access (“get”) always returns a value of the data type *string*.

Value range

""
The current date or time is displayed.
<value_string>
Sets the contained date or time value.

Syntax and Evaluation of the Value String

The value is evaluated according to the “Definition of Time Spans” for the timer object, whereby the following absolute time definition is permitted:

```
<Value>     ::= {<Datevalue>}{ <TimeValue>}
<DateValue> ::= {<day>}.{<month>}.{<year>}
<TimeValue> ::= {<hour>}:{<minute>}{:{<second>}}
```

Deviating from the definition of time spans for the ***timer***, it is permitted to specify only a date.

The ***datetime*** uses the Gregorian calendar. The value range extends from *01/01/1601 00:00:00* to *12/31/9999 23:59:59*. Two-digit years are mapped to the time range *1970* to *2069*.

When the user selects, sets or enters a value in the ***datetime***, *.value* contains this value in full format *dd.MM.yyyyy HH:mm:ss* (see attribute *.format*). When a change occurs, a *select* event is triggered.

Parts not specified are completed in the display by the corresponding values of the current system date or the current system time. For example, for a missing year, the current year is inserted.

If the value string is syntactically incorrect, the current date or time is displayed (as with *""*).

Examples

The current date is “06/25/2014” and the current time is “11:53:07”.

Value String	Displayed Value
12.1.	*“01/12/2014 11:53:07”* Year and time are supplemented from the current values.
12.1	*“06/25/2014 11:53:07”* Value string is syntactically incorrect. Current date and time are displayed.

If the value string is syntactically correct but does not represent a valid date or time (for example “29.2.2013” or “33.33.33”), then it is undefined what is shown in the ***datetime***. If the system returns an error, the current date or time is displayed. Otherwise, the system displays what it has made of the value string.

If *.value =""* or is erroneous and the *.allowundefined* attribute has the value *true*, then the value is shown as indefinite (checkbox unchecked).

The additions and corrections made by the object for displaying the value do not change the attribute value. That is, *.value* keeps its set content until the user selects, sets or enters a value in the ***datetime***.

The current date and time are calculated on visualization. In addition, they are calculated when in the visible state *.value* is set to a different value that the object adds or corrects for display (for example *""*). Again, the attribute value is usually not adjusted to the shown value. Repeated setting of the same value (also of *""*) therefore does not represent a change in value, so in this case there is no update of the displayed value either.

2.399.2 doccursor (XML Cursor)

The attribute contains the value of the DOM node that the ***doccursor*** points to. It is only available if the *.nodetype* attribute has one of the following values:

- *nodetype_attribute*
- *nodetype_cdata_section*
- *nodetype_comment*

» *nodetype_processing_instruction*

» *nodetype_text*

Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.399.3 filereq (File Dialogs)

This attribute, after a successful selection (return value of **querybox()** has been *button_ok*), contains the full path of the selected file or directory.

It should be noted that this path contains system-dependent delimiters.

If multiple selection is activated for the ***filereq*** object (*.multisel* = *true*), then the list of selected files is contained in the indexed attribute *.value[integer]*.

In addition, the *.value* attribute defines the initially selected value in file and directory dialogs with *.startsel* = *true*.

See also

Attributes *.multisel*, *.startsel*, *.value[integer]*

2.399.4 Global Variables

Queries the current value of a global variable or assigns a new value to it.

For an assignment, the data type of the value or expression must correspond to the data type of the variable.

2.400 .value[integer]

This attribute of the file dialogs (***filereq***), after a successful multiple selection (attribute *.multisel = true*, return value of **querybox()** has been *button_ok*), contains a list with the complete paths of the files that the user has selected.

The number of selected files can be determined through *.count[.value]*.

Definition

Data type	*Access*	*changed event*
string	get, set	yes

C	*COBOL*
Identifier: AT_value	Identifier: AT-value
Data type: DT_string	Data type: DT-string

Classification	*Objects*
object-specific attribute	filereq

Please bear in mind that the paths contain system-specific path delimiters.

In case of single-selection (*.multisel = false*), the selected file is returned in the attribute **.value** (without index).

Example

```
rule void PrintValues(object Fr)
{
  variable integer I;
  for I:=1 to Fr.count[.value] do
    print Fr.value[I];
  endfor
}
```

Particularities Microsoft Windows

The memory for the complete list of the selected paths is limited to 64 kB.

See also

Attribute **value**

2.401 .version[enum]

This attribute of the ***document*** object sets the desired version of the XML toolkit.

When the attribute is changed, the stored DOM tree is deleted and all existing ***doccursors*** are invalidated (*.mapped = false*).

Setting the attribute does not automatically force the XML toolkit to be loaded. When the attribute is queried, it returns the value that it has been set to. The attribute *.real_version* can be used to check at runtime, whether the desired XML toolkit can be loaded.

The index for this attribute is the `toolkit` enumeration.

Currently *toolkit_windows* is the only index value supported, because only Microsoft Windows allows to set the runtime version of the MSXML control. To indicate the version number, the major version has to be multiplied with *100* and the minor version has to be added if applicable. When *0* is given as version number, the system's default version of the MSXML control is loaded.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_version	Identifier: AT-version
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	document

Example

MSXML 5.0 shall be used so that the XSD document type is supported.

```
this.version[toolkit_windows] := 500;
```

Afterward it is checked if MSXML 5.0 could be loaded.

```
if (this.real_version[toolkit_windows] = 500) then
  // OK
endif
```

Note

The versions of the MSXML control differ in the features they support. Only some of the main differences can be listed here. Comprehensive information can be found in the documentation of the MSXML control provided by Microsoft.

MSXML 6.0

- Some features that are considered insecure, like DTD's and embedded schemas, are turned off by default.
- Support for XDR Schema is removed.
- Support for XML Signatures is removed.

MSXML 5.0 for Microsoft Office applications

- Support for XML Signatures.
- Support for embedded XSD Schemas.

MSXML 4.0

- Support for XML Schemas (XSD).
- Support for the Schema Object Model (SOM).
- Older versions of the MSXML control are not replaced on installation.
- Version-independent ProgID removed.
- Legacy code removed:
- Non-standard XSL replaced by XSLT 1.0
- Non-standard XSLpattern language replaced XPath 1.0.

MSXML 3.0

- Compliance with XSLT 1.0 and XPath 1.0 specifications.
- Support for namespaces in XPath queries.

See also

Attribute *.real_version[enum]*

2.402 .version_string

In the ***setup*** object, this attribute requests the IDM version string.

Definition

Data type
string

Access
get

C
Identifier: AT_version_string
Data type: DT_string

COBOL
Identifier: AT-version-string
Data type: DT-string

Classification
object-specific attribute

Objects
setup

2.403 .vheight

This attribute specifies the internal (“virtual”) height of the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_vheight
Data type: DT_integer

COBOL
Identifier: AT-vheight
Data type: DT-integer

Classification
geometry attribute

2.404 .visible

.visible defines whether the object is displayed on the screen or not.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_visible
Data type: DT_boolean

COBOL
Identifier: AT-visible
Data type: DT-boolean

Classification
standard attribute

Particularities of the Attributes .sensitive and .visible

Unlike the other object attributes, the attributes *.visible* and *.sensitive* get their final shape only in connection with the object hierarchy. This is because the corresponding attributes of the related parent object have a decisive influence on them. This means that an object can only be visible if itself **and** its parent are visible. The same applies to selectivity.

Remark

The statement *Window.visible := true* does the following:

- If the window is visible, it will emerge to the foreground after the statement is executed, i.e. it will become the foremost window.
- If the window is invisible, it will appear on the screen after the statement is executed.

See also

Attribute *.sensitive*

2.405 .vsb_arrows

The attribute *.vsb_arrows* defines, whether the vertical scrollbar has arrows at its ends.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_vsb_arrows
Data type: DT_boolean

COBOL
Identifier: AT-vsb-arrows
Data type: DT-boolean

Classification
scrollbar attribute

Value range

true
: Scrollbar with arrows

false
: Scrollbar without arrows

Availability

The attribute is only supported on Motif and since IDM version A.05.02.h.

See also

Attribute *.hsb_arrows*

2.406 .vsb_linemotion

This attribute specifies pixel value at which the vertical scrollbar position changes during scrolling by lines.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_vsb_linemotion
Data type: DT_integer

COBOL
Identifier: AT-vsb-linemotion
Data type: DT-integer

Classification
scrollbar attribute

2.407 .vsb_optional

This attribute defines that the vertical scrollbar will only be displayed if actually necessary.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_vsb_optional
Data type: DT_boolean

COBOL
Identifier: AT-vsb-optional
Data type: DT-boolean

Classification
scrollbar attribute

tablefield

If the attribute is set at *true*, the Dialog Manager decides whether the vertical scrollbar is currently necessary or not. If, for example, all lines of a tablefield are displayed completely in the space available and if the vertical scrollbar is set optionally, the scrollbar will not be displayed.

2.408 .vsb_pagemotion

This attribute specifies the pixel value at which the vertical scrollbar position changes during scrolling by pages.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_vsb_pagemotion
Data type: DT_integer

COBOL
Identifier: AT-vsb-pagemotion
Data type: DT-integer

Classification
scrollbar attribute

2.409 .vsb_visible

This attribute defines the visibility of the vertical scrollbar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C
Identifier: AT_vsb_visible
Data type: DT_boolean

COBOL
Identifier: AT-vsb-visible
Data type: DT-boolean

Classification
scrollbar attribute

2.410 .vscreen_height

With this attribute of the setup object the height of the virtual screen resp. working area in pixels can be queried.

This attribute is only available on WINDOWS, and is intended for multi-monitor environments.

Definition

Data type
integer

Access
get

C
Identifier: AT_vscreen_height
Data type: DT_integer

COBOL
Identifier: AT-vscreen-height
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.411 .vscreen_width

With this attribute of the setup object the width of the virtual screen resp. working area in pixels can be queried.

This attribute is only available on WINDOWS, and is intended for multi-monitor environments.

Definition

Data type
integer

Access
get

C
Identifier: AT_vscreen_width
Data type: DT_integer

COBOL
Identifier: AT-vscreen-width
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.412 .vscreen_x

With this attribute of the setup object the X coordinate of the origin of the virtual screen resp. working area in pixels can be queried.

This attribute is only available on WINDOWS, and is intended for multi-monitor environments.

Definition

Data type
integer

Access
get

C
Identifier: AT_vscreen_x
Data type: DT_integer

COBOL
Identifier: AT-vscreen-x
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.413 .vscreen_y

With this attribute of the setup object the Y coordinate of the origin of the virtual screen resp. working area in pixels can be queried.

This attribute is only available on WINDOWS, and is intended for multi-monitor environments.

Definition

Data type
integer

Access
get

C
Identifier: AT_vscreen_y
Data type: DT_integer

COBOL
Identifier: AT-vscreen-y
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

Availability

Since IDM version A.06.03.a

2.414 .vwidth

This attribute specifies the internal (“virtual”) width of the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_vwidth
Data type: DT_integer

COBOL
Identifier: AT-vwidth
Data type: DT-integer

Classification
geometry attribute

2.415 .weeknumbers

This attribute controls whether the calendar week numbers are displayed in the fold-out calendar.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_weeknumbers Data type: DT_boolean	Identifier: AT-weeknumbers Data type: DT-boolean

Default value	*Inheritance*
false	yes

Classification	*Objects*
object-specific attribute	datetime

Value range

false
: No calendar weeks are indicated in the calendar..

true
: Calendar weeks are indicated in the calendar..

Note

The first calendar week of a year is the first week with at least 4 days in that year.

2.416 .weight

This attribute can be used to define the font weight / thickness of a font.

Remark:

It should be noted that it depends on the selected character set whether and to what extent the selected modifiers are applied.

Definition

Data type	*Access*	*changed event*
enum	get, set	no

C
Identifier: AT_weight
Data type: DT_enum

COBOL
Identifier: AT-weight
Data type: DT-enum

Classification
layout attribute

Value range

face_default
: Regular, unchanged character representation.

face_light
: Light character representation.

face_normal
: Regular, unchanged character representation.

face_medium
: Font weight between *face_normal* and *face_demibold*.

face_demibold
: Font weight between *medium* and *bold*.

face_bold
: Bold character representation.

face_black
: Black character representation.

Availability

Since IDM version A.06.03.a

2.417 .width

This attribute specifies the current width of the object.

Definition

Data type	*Access*	*changed event*
integer (may be *0*)	get, set	yes

C	*COBOL*
Identifier: AT_width	Identifier: AT-width
Data type: DT_integer	Data type: DT-integer

Classification
geometry attribute

.width defines the width of the entire ***notebook*** object with all its elements. If the width is defined too small, a minimum value depending on the window system will be used. The width must not be *0*, since the required width cannot be calculated.

In ***tablefield*** the attribute can also have the value *0*. This means that the Dialog Manager shall calculate the corresponding size in such a way that all elements can be displayed in the corresponding direction, without a scrollbar being needed. If, for example, the width of the tablefield is set at *0*, the ***tablefield*** will be wide enough that all columns can be displayed completely in the object.

With the ***toolbar***, *.width* (without index) defines the default value for the values not set by *.width [class]*.

See also

Attributes *.height*, *.real_width*, *.xauto*, *.xleft*, *.xright*

2.418 .width[class]

This attribute of the ***toolbar*** defines the width of the toolbar in the docking state given by the index.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_width	Identifier: AT-width
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	toolbar

Index Range

toolbar
: Width when the toolbar is docked

window
: Width when the toolbar is undocked (tool window)

Without an index the attribute returns or sets the default value for both docking states.

See also

Attribute *.height[class]*

2.419 .width[enum]

This attribute can be used to define correction factors for calculating the grid width from the ***font***.

Definition

Data type	*Access*	*changed event*
integer	get, set	no

C	*COBOL*
Identifier: AT_width	Identifier: AT-width
Data type: DT_integer	Data type: DT-integer

Classification
geometry attribute

Index Range

scale_factor

With this index value, the attribute defines a percentage, which is multiplied with the base value as a scaling factor.

The base value is the character width of the character set determined from the font metric or the reference string.

scale_offset

With this index value, the attribute defines a pixel value that is added as a constant to the scaled base value.

See also

Attributes *.height[enum]*, *.refstring*, *.real_xraster*, *.xraster*

Chapter “Calculating the Grid Size from a Reference Font” in manual “Resource Reference”

2.420 .window

This attribute queries the window that the object belongs to.

Definition

Data type
object

Access
get

C
Identifier: AT_window
Data type: DT_object

COBOL
Identifier: AT-window
Data type: DT-object

Classification
hierarchy attribute

2.421 .winsys

In the ***setup*** object, this attribute queries the window system type.

Definition

Data type	*Access*
enum	get

C	*COBOL*
Identifier: AT_winsys	Identifier: AT-winsys
Data type: DT_enum	Data type: DT-enum

Classification	*Objects*
object-specific attribute	setup

Value range

winsys_none
: The Window System cannot be identified.

winsys_windows
: The Window System is a version of MICROSOFT WINDOWS.

winsys_x11
: The Window System is an X11 compatible version of the X WINDOW SYSTEM (X Windows).

2.422 .winsys_string

In the ***setup*** object, this attribute requests the window system.

Definition

Data type
string

Access
get

C
Identifier: AT_winsys_string
Data type: DT_string

COBOL
Identifier: AT-winsys-string
Data type: DT-string

Classification
object-specific attribute

Objects
setup

See also

Attribute *.winsys_version*

2.423 .winsys_version

In the ***setup*** object, this attribute requests the window system's version.

Definition

Data type
integer

Access
get

C
Identifier: AT_winsys_version
Data type: DT_integer

COBOL
Identifier: AT-winsys-version
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

See also

Attribute *.winsys_string*

2.424 .wrap

At the ***layoutbox***, this attribute can be used to switch wrapping on and off. Wrapping means that when resizing or changing the *.visible* attribute of the ***layoutbox***, the children may be reordered so that all of them remain visible or accessible.

When used with a ***spinbox*** this attribute controls whether a circular spinning is to be executed (*.wrap = true*), or whether the spinning is to be stopped on reaching the minimum or maximum limit (*.wrap = false*). The value of this attribute is preset on *true*.

Definition

Data type	*Access*	*changed event*
boolean	get, set	yes

C	*COBOL*
Identifier: AT_wrap	Identifier: AT-wrap
Data type: DT_boolean	Data type: DT-boolean

Classification	*Objects*
object-specific attribute	layoutbox, spinbox

layoutbox

If wrapping is switched off, no more changes are made (e.g. changes to the attributes of the ***layoutbox*** *.direction*, *.ymargin*, *.xmargin*… or their children).

If wrapping is disabled from the beginning, objects are not arranged by the ***layoutbox***.

2.425 .x

In the object ***thisevent***, this attribute queries the mouse coordinate X (valid only for *select* event in window and groupbox; relative to the object to which the event refers).

Definition

Data type
integer

Access
get

C
Identifier: AT_x
Data type: DT_integer

COBOL
Identifier: AT-x
Data type: DT-integer

Classification
object-specific attribute

Objects
thisevent

See also

Attribute *.y*

Chapter “Event Object thisevent” in manual “Rule Language”

2.426 .xalignment[index]

This ***tablefield*** attribute defines the horizontal alignment of the content (left, centered, right) within the cell for the cell in row I and column J.

The value of *.xalignment[0,0]* is applied to all cells for which no horizontal alignment has been specified.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_xalignment	Identifier: AT-xalignment
Data type: DT_integer	Data type: DT-integer

Default value
1

Classification	*Objects*
object-specific attribute	tablefield

Value range

-1
: right aligned

0
: horizontally centered

1
: left aligned
Default value when not overwritten by *.xalignment[0,0]*

Note

To use this attribute *.options[opt_new_align]* has to be *true*.

2.427 .xauto

This attribute defines whether the IDM should automatically determine the object width, the distance of the object from the left edge or the distance from the right edge of the parent object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_xauto
Data type: DT_integer

COBOL
Identifier: AT-xauto
Data type: DT-integer

Classification
geometry attribute

Value range

-1
The distance from the left edge of the parent object is computed by the IDM.
The distance from the right edge and the object width (*.xright* and *.width*) need to be defined by the developer. For some objects the IDM can also determine the width automatically from the contents.

0
The object width is computed by the IDM. The actual width can be queried with the *.real_width* attribute.
The distances from the left and right edges of the parent object (*.xleft* and *.xright*) need to be defined by the developer.

1
The distance from the right edge of the parent object is computed by the IDM.
The distance from the left edge and the object width (*.xleft* and *.width*) need to be defined by the developer. For some objects the IDM can also determine the width automatically from the contents.

See also

Attributes *.width*, *.xleft*, *.xright*

2.428 .xdpi

With this attribute of the ***setup*** object the dots per inch (DPI) in horizontal direction of the screen can be queried.

In multiscreen systems (IDM for Motif only) the attribute returns the value for the default screen.

Definition

Data type
integer

Access
get

C
Identifier: AT_xdpi
Data type: DT_integer

COBOL
Identifier: AT-xdpi
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.429 .xdpi[integer]

With this attribute of the ***setup*** object the dots per inch (DPI) in horizontal direction of screen I can be queried.

The data type of the index is *integer*, with a valid range from *1 … setup.screencount*.

Definition

Data type	*Access*
integer	get

C	*COBOL*
Identifier: AT_xdpi	Identifier: AT-xdpi
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.430 .xleft

.xleft specifies the distance to the left border of the superordinate object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_xleft
Data type: DT_integer

COBOL
Identifier: AT-xleft
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.width*, *.xauto*, *.xright*

2.431 .xmargin

This attribute determines the distance of the children (***layoutbox***) or the content (***edittext***, ***poptext***, ***image***) from the left and right edges of the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_xmargin	Identifier: AT-xmargin
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	edittext, layoutbox, poptext, image

layoutbox

Value range	*0…65536*
Default value	*0*

With this attribute the right and left distance between the edges of the ***layoutbox*** and the children can be set.

image

The attribute specifies the horizontal distance from the edge (border) to the actual content of an image object in pixels. Changes to the xmargin value always affects the distance to the left margin and to the right margin.

See also chapter „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

Since IDM version A.06.03.a

edittext

The attribute specifies the distance between the text and the border in pixels. Changes of the *.xmargin* value affect the distance from the left and right border. Moreover, with *.width = 0* and *.xauto <> 0* a change of the attribute influences the width of the object.

The default values of this attribute vary for the different window systems. The default value for Motif is *5*, that for Microsoft Windows is *1*. The value range is *-127…127*.

On Microsoft Windows, additionally the attribute *.options[opt_et_margin]* of the dialog has to be set to *false* (default) for the *.xmargin* attribute to be considered. Also on Microsoft Windows and depending on the patch level, it may occur that the right margin is ignored with overlong texts in single-line edittexts.

poptext

Microsoft Windows

The ***poptext*** adheres to the attribute *.xmargin* with the styles *edittext* and *listbox*. The behavior corresponds to the behavior of the ***edittext*** (described above).

Motif

The ***poptext*** object on Motif 2.x and from IDM version A.05.02.d supports the attributes *.xmargin* and *.ymargin* to influence the horizontal and vertical distance between the text and the outside borders.

Since the ***poptext*** on Motif is a compound object, the attributes influence the distance between the text and the text field frame and therefore only has an indirect effect on the distance between the text field and the outside borders. The *.ymargin* attribute influences the height of the ***poptext*** when no height is specified (*.height = 0* and *.yauto <> 0*). Too high values can lead to the text not being visible with predetermined heights. The *.xmargin* attribute however is also effective when a width is specified and adjusts the displayed text area and distance to the border. For negative attribute values, the default values of the system are used.

Particularities

- The use of *.xmargin* is not recommended because Motif displays the cursor within the frame area when it is left or right of the text.
- The text field frame on Motif normally is only visible for the styles *edittext* and *listbox*.
- The value *-2* for *.xmargin* and *.ymargin* is a compatibility mode to the precedent version which had a slightly different height calculation and text positioning for *.height = 0*.
- The default value is *5* and therefore different from the Windows platforms.

2.432 .xml

This attribute contains a string representation of an ***XML Document*** or an ***XML Cursor***. With the XML Document, the stored DOM tree is deleted when a new value is set and a new tree is built from the new value. All existing XML Cursors become invalid. The attribute *.mapped* has the value *false* for invalid XML Cursors.

Definition

Data type
string, *object* [***text***]

Access
get (***doccursor***)
get, set (***document***)

C
Identifier: AT_xml
Data type: DT_string, DT_text

COBOL
Identifier: AT-xml
Data type: DT-string, DT-text

Inheritance
no

Classification
object-specific attribute

Objects
doccursor, document

The attribute is available for the XML Document and the XML Cursor, but it is not passed down. Please note that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

2.433 .xorigin

.xorigin specifies the shift of the object contents along the x-axis.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_xorigin
Data type: DT_integer

COBOL
Identifier: AT-xorigin
Data type: DT-integer

Classification
geometry attribute

2.434 .xraster

This attribute defines the basic grid unit on the x-axis.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_xraster
Data type: DT_integer

COBOL
Identifier: AT-xraster
Data type: DT-integer

Classification
geometry attribute

Remarks

» The grid attributes *.reffont*, *.xraster* and *.yraster* are only significant if the application is to be realized on as many different hardware environments as possible. These attributes specifies a basic unit for the object size which is independent of the previously used pixel units. The letter size of the reference font selected or the input the user makes without specifying a font, are used as the new unit. All dimensions or positions then refer to this unit.

Identifier	Data Type	Meaning
.reffont	*object* (***font***)	Identifier of the font to which the units shall refer.
.xraster	*integer*	Basic unit on x-axis
.yraster	*integer*	Basic unit on y-axis

» If a reference font is given, the DM automatically calculates the values for *.xraster* and *.yraster*, i.e. a *.reffont* specification overwrites the specifications of *.xraster* and *.yraster*.

Example

```
.xraster    8;
.yraster    16;
...
.posraster  true;
.xleft      10;
.ytop       4;
```

The thus defined object has the position 80, 64 on pixel coordinates, because the position was given in relation to the bases 8 and 16.

See also

Attributes *.reffont*, *.yraster*

2.435 .xright

This attribute specifies the distance to the right border of the superordinate object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_xright
Data type: DT_integer

COBOL
Identifier: AT-xright
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.width*, *.xauto*, *.xleft*

2.436 .xspacing

With this attribute, the **horizontal** distance between the children of the ***layoutbox*** can be set.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_xspacing Data type: DT_integer	Identifier: AT-xspacing Data type: DT-integer

Value range	*Default value*
0…65536	*0*

Classification	*Objects*
geometry attribute	layoutbox

See also

Attribute *.yspacing*

2.437 .y

In the object ***thisevent***, this attribute queries the mouse coordinate Y (valid only for the *select* event in the window and the groupbox; relative to the object to which the event refers).

Definition

Data type
integer

Access
get

C
Identifier: AT_y
Data type: DT_integer

COBOL
Identifier: AT-y
Data type: DT-integer

Classification
object-specific attribute

Objects
thisevent

See also

Attribute *.x*

Chapter “Event Object thisevent” in manual “Rule Language”

2.438 .yalignment[index]

This ***tablefield*** attribute defines the vertical alignment of the content (top, centered, bottom) within the cell for the cell in row I and column J.

The value of *.yalignment[0,0]* is applied to all cells for which no vertical alignment has been specified.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_yalignment	Identifier: AT-yalignment
Data type: DT_integer	Data type: DT-integer

Default value
1

Classification	*Objects*
object-specific attribute	tablefield

Value range

-1
: bottom aligned

0
: vertically centered

1
: top aligned
Default value when not overwritten by *.yalignment[0,0]*

Note

To use this attribute *.options[opt_new_align]* has to be *true*.

2.439 .yauto

This attribute defines whether the IDM should automatically determine the object height, the distance of the object from the top edge or the distance from the bottom edge of the parent object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_yauto
Data type: DT_integer

COBOL
Identifier: AT-yauto
Data type: DT-integer

Classification
geometry attribute

Value range

-1
The distance from the top edge of the parent object is computed by the IDM.
The distance from the bottom edge and the object height (*.ybottom* and *.height*) need to be defined by the developer. For some objects the IDM can also determine the height automatically from the contents.

0
The object height is computed by the IDM. The actual height can be queried with the *.real_height* attribute.
The distances from the top and bottom edges of the parent object (*.ytop* and *.ybottom*) need to be defined by the developer.

1
The distance from the bottom edge of the parent object is computed by the IDM.
The distance from the top edge and the object height (*.ytop* and *.height*) need to be defined by the developer. For some objects the IDM can also determine the height automatically from the contents.

See also

Attributes *.height*, *.ybottom*, *.ytop*

2.440 .ybottom

This attribute defines the distance to the bottom border of the superordinate object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_ybottom
Data type: DT_integer

COBOL
Identifier: AT-ybottom
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.height*, *.yauto*, *.ytop*

2.441 .ydpi

With this attribute of the ***setup*** object the dots per inch (DPI) in vertical direction of the screen can be queried.

In multiscreen systems (IDM for Motif only) the attribute returns the value for the default screen.

Definition

Data type
integer

Access
get

C
Identifier: AT_ydpi
Data type: DT_integer

COBOL
Identifier: AT-ydpi
Data type: DT-integer

Classification
object-specific attribute

Objects
setup

2.442 .ydpi[integer]

With this attribute of the ***setup*** object the dots per inch (DPI) in vertical direction of screen I can be queried.

The data type of the index is *integer*, with a valid range from *1 … setup.screencount*.

Definition

Data type	*Access*
integer	get

C	*COBOL*
Identifier: AT_ydpi	Identifier: AT-ydpi
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	setup

Remark Motif:

The IDM FOR MOTIF provides multi-screen support. Please note that the screen index is something else than the screen number e.g. obtained with the program **xdpyinfo**.

See alsochapter „Multiscreen Ssupport untder Motif“ in manual „Programmiertechniken“

An example dialog can be found at the display resource.

Remark Windows:

The IDM FOR WINDOWS has multi-monitor support starting with IDM version **A.06.03.a**.

See also: chapter „Multi-Mmonitor Ssupport untder Windows“ in manual „Programmiertechniken“.

2.443 .ymargin

This attribute determines the distance of the children (***layoutbox***) or the content (***edittext***, ***poptext***, ***image***) from the top and bottom edges of the object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_ymargin	Identifier: AT-ymargin
Data type: DT_integer	Data type: DT-integer

Classification	*Objects*
object-specific attribute	edittext, layoutbox, poptext, image

layoutbox

Value range	*0…65536*
Default value	*0*

With this attribute, the upper and lower distance between the edges of the ***layoutbox*** and the children can be set.

image

The attribute specifies the vertical distance from the edge (border) to the actual content of an image object in pixels. Changes to the ymargin value always affects the distance to the top margin and to the bottom margin.

See alsochapter „HighDPI UnterstützungSupport“ in manual „Programmiertechniken“

Since IDM version A.06.03.a

edittext (Motif only)

The attribute specifies the distance between the text and the border in pixels. Changes of the *.ymargin* value affect the distance from the top and bottom border. Moreover, with *.height = 0* and *.yauto <> 0* a change of the attribute influences the height of the object.

The value range is *-127…127*. The default value is *5*.

On MICROSOFT WINDOWS the ***edittext*** does not support this attribute.

poptext (Motif only)

The ***poptext*** object on Motif 2.x and from IDM version A.05.02.d supports the attributes *.xmargin* and *.ymargin* to influence the horizontal and vertical distance between the text and the outside borders.

Since the ***poptext*** on Motif is a compound object, the attributes influence the distance between the text and the text field frame and therefore only has an indirect effect on the distance between the text field and the outside borders. The *.ymargin* attribute influences the height of the ***poptext*** when no height is specified (*.height = 0* and *.yauto <> 0*). Too high values can lead to the text not being visible with predetermined heights. The *.xmargin* attribute however is also effective when a width is specified and adjusts the displayed text area and distance to the border. For negative attribute values, the default values of the system are used.

Particularities (Motif)

- The text field frame on Motif normally is only visible for the styles *edittext* and *listbox*.
- The value *-2* for *.xmargin* and *.ymargin* is a compatibility mode to the precedent version which had a slightly different height calculation and text positioning for *.height = 0*.
- The default value is *5* and therefore different from the Windows platforms.

2.444 .yorigin

.yorigin specifies the shift of object contents along the y-axis.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_yorigin
Data type: DT_integer

COBOL
Identifier: AT-yorigin
Data type: DT-integer

Classification
geometry attribute

2.445 .yraster

This attribute defines the basic grid unit on the y-axis.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_yraster
Data type: DT_integer

COBOL
Identifier: AT-yraster
Data type: DT-integer

Classification
geometry attribute

Remarks

» The grid attributes *.reffont*, *.xraster* and *.yraster* are only significant if the application is to be realized on as many different hardware environments as possible. These attributes specifies a basic unit for the object size which is independent of the previously used pixel units. The letter size of the reference font selected or the input the user makes without specifying a font, are used as the new unit. All dimensions or positions then refer to this unit.

Identifier	Data Type	Meaning
.reffont	*object* (***font***)	Identifier of the font to which the units shall refer.
.xraster	*integer*	Basic unit on x-axis
.yraster	*integer*	Basic unit on y-axis

» If a reference font is given, the DM automatically calculates the values for *.xraster* and *.yraster*, i.e. a *.reffont* specification overwrites the specifications of *.xraster* and *.yraster*.

Example

```
.xraster    8;
.yraster    16;
...
.posraster  true;
.xleft      10;
.ytop       4;
```

The thus defined object has the position 80, 64 on pixel coordinates, because the position was given in relation to the bases 8 and 16.

See also

Attributes *.reffont*, *.xraster*

2.446 .yspacing

With this attribute, the **vertical** distance between the children of the ***layoutbox*** can be set.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C	*COBOL*
Identifier: AT_yspacing Data type: DT_integer	Identifier: AT-yspacing Data type: DT-integer

Value range	*Default value*
0…65536	*0*

Classification	*Objects*
geometry attribute	layoutbox

See also

Attribute *.xspacing*

2.447 .ytop

This attribute defines the distance to the top border of the superordinate object.

Definition

Data type	*Access*	*changed event*
integer	get, set	yes

C
Identifier: AT_ytop
Data type: DT_integer

COBOL
Identifier: AT-ytop
Data type: DT-integer

Classification
geometry attribute

See also

Attributes *.height*, *.yauto*, *.ybottom*

Index

B

C

F

G

H

I

K

L

M

N

O

S

T

U

V

W

X

Y