ISA Dialog Manager

Release Notes IDM 4

A.06.03.c

The release notes report bug fixes, changes and enhancements of each version of the ISA Dialog Manager.

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 Version A.04.04.h

1.1 Windows NT

- Gnats 9754: The position of the scrollbar sliders of grouping objects with a virtual size was not updated when an attribute, that influenced this position (i.e.: .xorigin) was changed. The error did not occur when the size of the grouping object needed to be recalculated at the same time. In addition, it was also possible to set the attributes .xorigin and .yorigin to values, that fell outside of the "dynamic" value range. Now the values are automatically adjusted to the next possible value.
- Gnats 9760: The focus was incorrectly set after the destruction of a grouping object (groupbox, notepage, ...).
- Gnats 9718: Because the IDM does not support multiple indexed attributes, you have to use methods with parameters to take advantage for OLE properties with multiple indices.
  Instead of (not valid in the Rule Language!)

```
control.Prop["a"]["b"] := "x";
print control.Prop["a"]["b"];
```

  the following syntax in the IDM Rule Language must be used

```
control:Prop("a", "b", "x");
print control:Prop("a", "b");
```

  Sometimes the method name is altered by a "Set" or "Get" at the beginning:

```
control:SetProp("a", "b", "x");
print control:GetProp("a", "b");
```

1.2 Core

- Gnats 9772: .options[opt_new_align] on the tablefield is now correctly written in a binary form.

1.3 Motif

- Gnats 9665: Workaround for Solaris/Motif errors in conjunction with a different X-Display. Pop-up men can now be opened with Solaris-IDM on a NON-Solaris X-display.
- Gnats 9662: Warning messages during the creation/selection of notepages in the editor have been resolved. Crashes after making notepages invisible under Suse has also been corrected.
- Gnats 9756, 9757: Problem in displaying Chinese texts (multi-byte locales) in different objects (tablefield, notepage, menus, etc.) has been corrected. Return of file and directory names of the filerequester now correctly regards the code page.

2 Version A.04.04.g

2.1 Windows NT

- New: The poptext object now supports the automatic sorting of content under Microsoft Windows. The sorting is carried out by Microsoft Windows. The only thing, that changes is the order in which it is displayed; the indexes remain the same. The attribute *.options[opt_sort]* controls, if the content is to be sorted or not. Normally, this attribute is set to *false*. If sorting is desired, then the attribute must be set to *true*.
- New: The treeview object now supports the setting of the text color (.fgc). This can have an affect on the looks of the existing dialog.
- The method :getformat() of the edittext object (.options[opt_rtf] = true) returns *nothing* under the non-unicode-capable Microsoft system. The error exists in idmuni.dll.
- Gnats 9742: The first area within a splitbox object was not displayed – instead the background was shown.
- Gnats 9741: The method :setformat() on the edittext object (.options[opt_rtf] = true) did not function correctly. This problem occurred after the correction of Gnats 9672. It is important to note, that a format setting on an insertion marker (startsel = endsel) can only take place during run-time (.visible = true) and is only temporary. This is a characteristic of Microsoft Window objects, which unfortunately cannot be avoided. Please also refer to the note for the correction of Gnats 9672, that can be found further on in these release notes.
- Gnats 9733: An Assfail in object/slot 2284 was possible when a poptext object that had .style <> poptext possessed the focus and one of the keys `Tab`, `F4` or `Esc` was pressed. In addition, a combination of the modifier keys also had to be pressed so that the button would not be processed by the IDM. In this case, it was the `Tab` key and the modifier keys `Alt` and `Ctrl`.
- Gnats 9706: When an area within an edittext object, that had an empty format was to be marked from left to right, then the last character remained unmarked. This problem occurred only when the cursor was moved over more than one letter at a time (movement with `Shift` + `Home` or `Ctrl` + `Shift` + `Cursor Left`). This problem did not occur when a form (not empty) or no format was set.
- Gnats 9699: When a toolbar object was undocked from a window, the window cannot be activated. This problem occurred, because the focus was always set in the toolbar object.
- Gnats 9698: Setup.overridecursor was not reported to the statusbar and toolbar objects. As a result of this an Assfail in object/slot 2283 occurred in some cases.
- Gnats 9696: Under Windows 95 no additional menus could be created.

- Gnats 9680: The Microsoft Windows poptext object does not discern between capital and lower-case characters. As a result an incorrect entry was displayed in some cases, when the poptext object contained the same text with differs only in upper/lower-case.

  Notes for Windows 95
  Problem cannot be avoided under Windows 95.
- Gnats 9677: The attribute .toolhelp now supports multi-line texts.
- Gnats 9673: The treeview object has a set background color (.bgc), which is only indicated where no text existed. In order to correct this error comctl32.dll version 4.71 or higher is needed (beginning with Internet Explorer 4.0).
- Gnats 9672: When the method :setformat is used on an edittext object whose attribute .options [opt_rtf] = true, and the attribute .startsel and .endsel or the parameters Start and End have the same value, then the edittext object throws out the format setting when .endsel is altered. This also happened when one of the methods, which existed for an edittext object whose attribute .options[opt_rtf] = true, was called up with the parameters Start and End. This can now be avoided. The above-mentioned standard behavior of the Window objects, however, remains the same.

  Note
  - Method :setformat()
    A format setting to the insertion marker (.startsel = .endsel or Start = End) is only temporary. The set format remains valid only until .endsel has been changed. Here, it doesn't matter if .endsel is changed by the user or via the dialog script. For this reason a format setting with Start = End remains ineffective, when Start <> .startsel and End <> .endsel. In the latter case the method :setformat() returns the wrong value. In addition, please take note, that a format setting made to the insertion marker will not be assumed in the RTF text and as a result is *void*, when it is carried out on an object whose current status is "invisible".
  - Attribute .content and method :gettext()
    The RTF text is changed by the object itself (reformatted). It exists only logically. In other words, for example, a letter is bold and cursive; the order of the format alterations can change (first bold and then cursive or vice versa).
  - Use of the parameters Start and End
    When one of the following methods :findtext(), :getformat(), :gettext(), :replacetext() and :setformat() are called up with Start and End parameters, then please take note, that the marked area remains the same, however, the insertion marker located to the right of the marked area can become somewhat displaced.
- Gnats 9671: When a non-valid tile resource was assigned to an attribute, the attribute was handled incorrectly - as if no tile resource was assigned to it. Now the attribute is viewed as being set.

  Example
  The tile resource TInSens refers to the GIF file "NotExisting.gif", which does not exist. This resource is assigned to the attribute .picture[tile_insensitive] of an image object. When the image

object is insensitive, the image object is now filled with the background color – previously .picture [tile_default] was drawn.

- Gnats 9668: When a window does not contain an object, that can be focused on, then this window was not able to be activated when an undocked toolbar object was displayed. This problem arose, because the focus was always set in the toolbar object.
- Gnats 9666: Memory leak and increased memory needs during multiple settings of the .title attribute on the notepage object has been corrected.
- Gnats 9654: When a visible splitbox object initial did not possess visible children and one of these children was then switched to a visible status, the child did appear but no events were triggered.
- Gnats 9652: The filerequester dialog no longer functioned under Microsoft Windows 98. This is corrected in idmuni.dll.
- Gnats 9639: When a poptext object with attribute .style=edittext contains multiple entries which begin similar, then the first entry and not the entry that corresponds to the active item is marked. Since this is the standard behavior of a poptext object under Microsoft Windows, this cannot be changed. However, now an event (select or activate depending upon .options[opt_old_select]) is created when the .activeitem attribute is changed when it is pop-uped. The content of the .content attributes remains the same.

  Notes for Windows 95
  The problem cannot be repaired under Windows 95 – for this reason no event is generated.
- Gnats 9617: Under Win95, Win98 and WinMe the content of an edittext object with *.options[opt_rtf] = true* could not be set to a normal text.

2.2 Motif

- Gnats 9746: The &-symbol (mnemonic character) was incorrectly handled by the listbox, treeview, poptext and filereq objects.
- Gnats 9740: Maximized windows are now completely visible including all of their decorations.
- Gnats 9704: A bug in the Motif library can trigger the warning *"DeAssocNavigator requires a navigator trait"* under Linux (reproducible with OpenMotif 2.1.10 & OpenMotif 2.1.30) by the destruction of a list widget or text widget with scrollbars.
  This error is registered in OpenMotif.org under the number 1168 and has been corrected in OpenMotif version 2.2.3.
  In IDM a workaround for this has been implemented for Linux which avoids this error.
- Gnats 9667: No more X-warnings occur when the title is dynamically reset (via .title-attribute) on the notepage object.
- Gnats 9666: Memory leak and increased memory needs during multiple settings of the .title attribute on the notepage object has been corrected.
- Gnats 9655: Notebook keeps the initial correct size, just like when dynamically adding children.
- Gnats 9648: The window title is now also properly displayed under Solaris 9 in ja_JP.PCK locale.

2.3 Core

- Changed: Error messages, that occur when DM is called up in the wrong run-state are now only written in the trace file and no longer in the log file.
- Gnats 9735: If the format code page was not UTF8, then no umlaut was displayed in an edittext with format. Instead, two characters were displayed. This error was a result of the correction of Gnats 9695, but was corrected in the meanwhile.
- Gnats 9711: Binary writing via xidm no longer fails by the path resolving through the master import.
- Gnats 9709: When reading dialogs with reposid() information the reading is no longer aborted, but rather skipped over.
- Gnats 9697: The start sequence of modules, which changed as a result of the correction carried out on 9577, has been corrected.
- Gnats 9695: applyformat() with format string now works in a correct manner by an AppFormatCodepage!=CP_internal.
- Gnats 9683: No more busy-loop when decoding ACP strings on UNIX/Linux systems.
- Gnats 9682: Checking for available object storage space has been corrected.
- Gnats 9661: An assfail occurred when calling up the :delete method on a tablefield, when the previous focus was made invalid through deletion. This has been corrected.
- Gnats 9658: Setvalue on .content on the poptext is now possible in the :init rule.
- Gnats 9656: Accelerators, that have key definitions (separate character) are now saved as correctly coded in the binary file; code page corrections of accelerator strings and characters in Motif-WSI. Now, under Motif, a warning appears when the xkey string is not valid.
- Gnats 9649: Also binary file (ModuleID!=0x20001) written with the help of XIDM/dump can be read without “Can't identify” problems occurring.
- Gnats 8697: A static assignment to an path-ID on a shadow attribute resolves now correct.

2.4 Network

- Gnats 9712: The DDM for Java-WSI also uses DM_InputHandler in order to understand asynchronous Ext-Events from the server side.

2.5 COBOL Interface

- Gnats 9710: Strings by DMcob functions are interpreted in the correct code page during tracing.

3 Version A.04.04.f

3.1 Windows NT

- Gnats 9616: The calling of DM_GetToolkitData with the attributes AT_IUnknown and AT_OLEInterface no long functioned on the sub-control object. This sometimes led to a crash, when a sub-control object was used as an input parameter for an OLE-method.
- Gnats 9597: When a poptext object that possessed the old selection style was opened and then directly edited, then the select event from a `Return` key delivers the correct corresponding text index.
- Gnats 9595: An inconsistency occurred by the treeview object under Windows XP, when the treeview object did not possess any scrollbars. Now the treeview object is always displayed in the same way.

  Attention
  The additional border is still there. This is caused by the IDM, which makes sure that the objects in question are correctly drawn under the previous operating systems from Windows XP. The future IDM for Windows XP will contain additional code that will handle these objects in a different manner as before.
- Gnats 9592: When inquiring the attributes .screen_width and .screen_height on the setup object the actual workplace size is given. This means that when the taskbar is set to be always visible and is always displayed at the top, then the size of the taskbar is subtracted from the size of the screen. If the settings of the taskbar are changed, then the new values will be displayed upon request.
- Gnats 9568: There are a number of dialog events that can be triggered without an user action, but rather as a result of a different action. By such events, call-ups that block the rule execution, for example querybox(), should not be used because this can trigger a repeated execution of the rule. In this case the reference is to the events :deselect and deactivate.
  This condition is not recognizable by IDM and therefore cannot be avoided. If it is essential to execute such a blocking-type of call, then a stop condition should be added to avoid repeated execution.

  Example

  False

```
edittext {
   on deselect {
    if atoi(this.content) > 99 then
      !! leads to an endless loop
       querybox(InputError);
     endif
  }
}
```

Correct

```
edittext {
  on deselect {
    if atoi(this.content) > 99 then
      !! prevents repeated openings
       this.content := "0";
      querybox(InputError);
      endif
  }
}
```

Also Correct (Eventually Better)

```
edittext {
  on modified {
    if atoi(this.content) > 99 then
      !! prevents repeated openings
      querybox(InputError);
    endif
  }
}
```

» Gnats 9567: By double clicking on an image object the following error message was displayed: "THING YIHANDLER CALLED FOR: D.Wn.Img (task: 9)" in the log file.

» Gnats 9560: Sometimes an inconsistency of the attributes .rowfirst and .colfirst occurred by a tablefield object when the rows and or columns were set to 0 and then later increased, and when there were visible and invisible rows or columns. This lead to scroll operations being shifted by one row or column until the inconsistency was corrected (scrolled back to the beginning). The scroll problem only occurred when invisible rows or columns existed, because due to improvement reasons relative to the previous scroll position the new position was calculated. The inconsistency already existed, but had no affect, because it was possible to position the scroll position as long as there were no invisible lines or columns.

» Gnats 9552: As the cursor position can sometimes change during the setting of the focus on the edittext object under MS Windows, now an additional calling up of the format function with the task FMT_setcursorabs takes place.

» Gnats 9551: When an RTF edittext object contained an RTF text as its content, and a special RTF method was called in the invisible state, the text was mishandled as normal text. As a result, non-readable RTF text was displayed.

» Gnats 9550: The error in accessing the text_underline status via :getformat() has been corrected. Now the access returns a value when the text_bold condition is invalid.

» Gnats 9548: A visible multi-line edittext object a GetValue call on .startsel or .endsel returned a wrong value, when both of these attributes marked positions, that did not lie within the first line.

- Gnats 9539: When using a Microsoft wheel mouse, the application sometimes froze up (100% CPU capacity). This occurred when the mouse was set to page-scrolling and the wheel was used on a tablefield object.
- Gnats 9537: It was not possible to enter a multi-byte character, that required more than one byte. This error only affected the unicode-incapable Microsoft systems, that used real multi-byte character sets (i.e.: Czech Windows 98). In addition, only those objects whose characters were processed by Windows (i.e.: edittext without format) were affected. Microsoft Windows NT, 2000 and XP did not have this problem, because the error was located in idmuni.dll.
- Gnats 9536: The quoting of commands when calling up shell programs via execute() has been corrected. A quoting of a command parameter is not necessary, whereas for the other parameters quoting remains necessary.
- Gnats 9535: An OLE method that possessed a BSTR type of argument was not able to be called up via the OLE interface within IDM: when, for example, the OLE control was created with the Microsoft Developer Studio .NET and the ATL library.

  Note
  Not all applications and compilers stick to the conventions, that describe how arguments are to be handled. Unfortunately, this leads to problems, that are not always able to be fixed. In order to remain as independent as possible, IDM uses the TypeInfo interface for calling up OLE methods. If the TypeInfo interface is not available (no TypeLibrary) or when the TypeInfo interface, which is required for IDM methods, was not implemented, then the OLE method is called up directly. In this case, the known problems with the arguments exist. For example, problems can arise with string parameters that only serve for the input of data.
- Gnats 9474: Corrections after the fact to the "The focus was not reset to an object that was destroyed due to an attribute change, but later newly created". This error only occurred when the object was a direct child of an object, that was allowed to hold the focus (layoutbox, notebook). In addition, the error also occurred principally for children of a splitbox object.

3.2 Motif

- Gnats 9606: The quick-navigate-feature on the listbox and poptext (Motif 2.1) can be turned off via the option .options[opt_quick_navigate] := false (default is true). This is a way of working around the Motif bug (beginning with Motif 2.1) in the quick-navigate-feature of the list widgets, which triggers a beep when the keyboard is used.
  Quick navigate on the poptext (Motif 2.1) works, but only in the poptext style!
- Gnats 9573: Bypass the double FocusOut (Motif bug) in order to avoid a doubled select event at the poptext.
- Gnats 9566: A poptext with background color no longer triggers an error on IRIX. Workaround on a Motif bug.
- Gnats 9553: When a window was opened, the workaround on a Motif2.1 bug (drop-down list of the last focused poptext was not closed => Gnats 9492) leads sometimes to a core dump. This has now been corrected.

- Gnats 9544: Checkboxes now have the correct size.
- Gnats 9528: The dynamical change of .navigable on a visible edittext has been corrected.

3.3 Java Interface

- Gnats 9602: Assfail no longer occurs when starting an IDM-Java application with visible note-pages.

3.4 Core

- Gnats 9614: The row count/row header combination on a tablefield is now correctly written in a binary form.
- Gnats 9612: Incorrect codes in UTF8 characters are now correctly recognized and an error message is written in the trace/log file.
- Gnats 9610: The indeces of the attribute .path in the doccursor object were one too small.
- Gnats 9605 and 9607: An object/module stack overflow no longer occurs during parsing. Also export/import constructions are correctly identified by the interface parser, and subsequent exports are no longer mixed up.
- Gnats 9599: The datatype of the .index attribute in the thisevent object was always of type index during a paste event. Now, the datatype is once again object specific.
- Gnats 9596: Edittext with a set format and format function is now saved correctly in the binary file.
- Gnats 9593: The :clear method on a tablefield object did not reset the selection anchor when this was positioned within the deleted area. As a result of this, strange things appeared when subsequently a selection was to be enlarged. Now, if necessary, the selection anchor is reset. However, this has the side effect that an enlargement of the selection cannot be carried out, because the selection anchor needs to be reset before this can take place. If this is not desired, then in place of calling up the :clear method, the attribute .content[row, col] of the corresponding cells needs to be set to null.
- Gnats 9579: When a path is written in a binary form, an Assfail is no longer returned when an import existed that did not have a MasterID.
- Gnats 9578: The codepage conversion of Extevent data will only be carried out once when the event queue is full.
- Gnats 9577: Shared modules are now considered when imports are created.
- Gnats 9569: The tablefield object was extended by adding the attributes .xalignment[I,J] and .yalignment[I,J]. The activation is carried out through .options[opt_new_align].
  The horizontal and vertical alignment of text within a tablefield cell was, up to now, able to be set via the attributes colalignment[I] and rowalignment[I]; however, the horizontal alignment could only be done column by column and the vertical alignment row by row.
  Through the option .options[opt_new_align] the new attributes .xalignment[I,J] and .yalignment [I,J] are now considered in place of .colalignment[I] and .rowalignment[I] These have the same

value range/meaning as the old attributes; however, these are double indexed and the heredity follows according to the direction just as it is with other double indexed attributes.
Supported under Motif and MS Windows.

- Gnats 9558: Access to the .nextactive attribute on a tablefield with an invalid value triggered an access error in place of an Assfail.
- Gnats 9557: No more Assfails by multiple calls of :delete on an editable tablefield whose format and focus was in the area to be deleted.
- Gnats 9542: The -IDMtracefile option now accepts % place holders; the same as -IDMerrfile.
- Gnats 9540: The focus field was not corrected on a tablefield object (transferred/implemented), when the column or row was made invisible by setting the column or row to 0. This error led to an "Assertion failed in itable.c line 3005" being written in the log file under Windows-WSI.

  Note
  Other assertions were also possible.
- Gnats 9532: DM_SetVectorValue on .format on a tablefield now also accepts DT_instance as a type.
- Gnats 9530: The .active attribute on a tablefield is now correctly written in a binary form.
- Gnats 9525: A DM_GetVectorValue call, where first or last is NULL on .colvisible/.rowvisible now functions without problem.
- Gnats 8707: When altering a .model attribute the changes made to the inherited attribute values are passed on to the WSI.

3.5 Editor

- Gnats 9451: The assigning of paths as shadow objects is now possible.
- Gnats 8755: The notepage sliders in the object/attribute window under MS Windows are now labeled in German.

3.6 Network

- Gnats 9623: The avoidance of a premature clearance of strings in array attributes of records when calling up record functions.
- Gnats 9522: Memory leak when calling up server functions via output parameters has been corrected.
- Gnats 9496 Amendment/Changes: The use of temporary string buffers, and the life span of strings, that go along with this, must be indicated with the new flag DMF_UseStringBuffer by calling up DM_GetVectorValue. In this case a clearance happens at the next call of an DM function without a set DMF_DontFreeLastStrings option.
- Gnats 8682: Under Microsoft Windows the IDM network application page had a HANDLE leak when it was started via the option -IDMserve.

» Gnats 8361: The options -IDMtracefile, -IDMtracetime, -IDMerrfile, -IDMerrwinfile are passed to the child process in the service mode (-IDMserve) of a DDM server application (this is only true for Windows - with UNIX this always occurs through forking). Here it is wise to use the % place holders because otherwise the trace file and log file could be overwritten.

3.7 COBOL Interface

» Gnats 9600: The COBOL separators (DMF_CobolSetTerminator and DMF_CobolGetTerminator) did not function when these were larger than 127; independently if these were set in COBOL or via DM_Control. This has now been corrected.

» Gnats 9565: Processed start options have been removed from argv vector. This led to COBOL applications being aborted when, for example, they were called up with the options "-IDMtracefile id.log -IDMenv X=X".

4 Version A.04.04.e

4.1 Motif and MS Windows

» Gnats 9108: New Attribute .options for Poptext/Combobox

Attribute .options[opt_old_select]
Since this attribute controls the creation of the activate and select events, it is normally set to *true*, as it has been in the past (analog to A0404d). The default value will be changed in future versions (in a major release, not in a patch).
If the attribute .options[opt_old_select] is set to *false*, then the select/activate events will be generated analog to the other sample objects, such as the listbox and treeview.
First modification to the "old" behavior is that the activate events no longer indicate the receiving of the focus (edittext and listbox style), but rather show a change in the activation status (.activeitem attribute). Secondly, a select event is only created by a "completed" selection of the user. This is true even when the same entry is selected a second time. In this sense "completed" means that the list is closed (exception: comments 1 and 2) or when the modification does not require that the list be made visible.

Data type:	boolean
Value range:	true, false
Default value:	true
Access:	get, set
Valid for:	poptext
Supported by:	UNIX & MS Windows

Summary
For an poptext with the attribute .options[opt_old_select]=false, the select and activate events are triggered under the following conditions

Action	Motif 1.2	Motif 2.1	MS Win	Event
cursor selection (close list, on different entry)	*	*	*	- activate and select
cursor selection (open list, on different entry)	*	*	*	- activate (and select in listbox style)

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

Comments

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

Example for the use of separate events with .options[opt_old_select] = false:
In order to be aware of all changes that have taken place (even in open lists):

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

In order only to react to the selection of another

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

In order to react to every selection

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

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

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

```
  on focus {...}
}
```

4.2 Windows NT

- Gnats 9511: In the treeview object the name of the picture, to some extent, was displayed in place of the text. This problem always occurred when a window resource was used as a graphic, or when an error occurred during the loading of a GIF data file.
- Gnats 9508: When an entry within an editable poptext object with format was chosen, this was not, as is to be expected, taken over into the edit field but rather was inserted into the cursor position. The error occurred when the cursor was moved, after the opening of the list in the poptext object, before an entry was selected.
- Gnats 9504: For the position and size calculation of the child objects within a splitbox object, a raster that is defined on the splitbox object is to be used. Due to an error in the previous version the splitbox object used the raster of its father object. As a result of the correction of this error it is now possible that the child object receives a different position or size as compared to what it had in previous versions. This is true when the splitbox object possesses a different raster definition as its father.
- Gnats 9503: In the treeview object no dbselect event was created, when the picture was changed in the select event, for example, in order to display the active entry in a different way. The error occurred due to the fact that the treeview object for graphic alternation was destroyed and then newly applied. The treeview object is no longer destroyed and newly applied when an attribute change takes place.
- Gnats 9498: In the case that an object was a child of a layoutbox, a change in the layout attribute in a visible state had no effect. This means that the object did not assume the modification, i.e. the layoutbox object did not re-arrange its children. It was observed that a change from .xauto had no effect on the object.
- Gnats 9497: A new symbol for the horizontal scrollbar of a listbox object was not enforced in every situation. As a result, it did occur that the horizontal scrollbar was not displayed, even though text entries were only partially shown. This situation came about, for example, when a text entry was changed in a visible state.
- Gnats 9495: When a window was made visible with .maximized := true; then the content of the window was displayed in the normal size. Thus, one saw the window frame in its maximum size, however the content was displayed in the normal (not the maximum) size.
- Gnats 9493: The poptext object in MS Windows NT 4.0 caused a memory leak. The object management has been changed so that this will never happen again.
- Gnats 9491: A system crash occurred when the .title attribute of a notepage object was set to *null*. For example, a crash was caused when the "inherit"-bit was deleted in the IDM editor.
- Gnats 9490: A window, which was directly located next to the screen, appeared on the screen. This happened because the windows in MS Windows, which are positioned completely outside of the visible area, were shifted into the visible area. In order to avoid this from happening again,

windows, which are positioned completely outside of the visible area, will be shifted by IDM so far that at least one pixel lies in the visible area.

- Gnats 9482: No event was created when the reason for the focus change was the transformation of an object from visible to invisible.
- Gnats 9478: The cursor control with multiline edittext objects with formats did not work. In addition, it was not possible to move word-wise with formatted edittext objects (even in single line mode).
- Gnats 9474: The focus was not restored to an object that was destroyed and newly applied due to an attribute modification. This error occurred when the object was situated under a layoutbox, a splitbox or a toolbar object, whereby the object did not have to be a direct child. In case of error the direct father of the object received the focus.
- Gnats 9471: The scrollbar slider in a tablefield object could not be interactively moved beyond the position 32767. Scrolling side to side, line by line or from column to column was not affected in any way.
- Gnats 9472: The attributes .startsel and .endsel were displayed shifted towards the front in multiline edittext objects. The error was first noticed after a few lines.
- Gnats 9463: In the transparent area of icon resources the colors were shifted, when the icon resource was used in a listbox, image, groupbox or poptext object.
- Gnats 9430: The attributes .xmargin and .ymargin were only interpreted in the tablefield object when thetext fit into the field. Because this behavior is not specified anywhere, it now stands that the attributes .xmargin and .ymargin are always valid.
- Gnats 9412: When a context menu was opened through an insensitive entry of a tablefield object, then the .index attribute of the thisevent-object was not set.
- Gnats 9410: When an edittext object has the following attribute value:

```
.maxchars := 0;
.options[opt_rtf] := true;
```

  then the text limit (in MS Windows) will be set explicitly to MAXINT (2147483647) in order to allow for unlimited text entry. If this is not desired, then the attribute .maxchars must be set to an appropriate low value.
- Gnats 9145: When both scrollbars were optional in a tablefield object and they were not displayed, then the attribute modification, which made it imperative that one of the scrollbars was displayed, was not checked if it was necessary to have a different type of scrollbar in its place. This error was discovered when the size of the columns was changed interactively. This affects all attributes, which are able to change the layout of the tablefield object.
- Gnats 9137: No charinput event was created when the content of an edittext object without format, which Windows normally provides for edittext objects, was changed.
- Gnats 9101: Accelerators on menu entries did not trigger a select event.
- Gnats 8687: When the help key `F1` was pressed while a context menu of a tablefield object was open, then the context menu could not be closed. The help event was created, but only after the context menu was closed was it possible for the help event to be executed.

- Gnats 5991: The opening of the context menu has been adapted to the standard behavior under MS Windows. When a context menu is opened, the object under the mouse now receives the focus. If the object under the mouse is a grouping object, for example a window or a groupbox object, then the focus will be set to the first child object that is able to receive the focus. This is only possible when the focus object is not the child of a grouping object.
- Corrected: When a context menu of an edittext object was open, then no help event was created when the `F1` key was pressed.
- Corrected: The attributes .startsel and .endsel are no longer falsely displayed (cursor in the front instead of in the back).
- Gnats 9517: Some special characters could not be entered when a format was set. For example, the Czech letter "small e with caron " could not be entered.

4.3 Motif

- Gnats 9515: During the conversion of .activeitem the active string was falsely converted when one linked with startup.o.
- Gnats 9492: When opening a dialogbox from a poptext/combobox object (Motif2.1), then its list did not remain open.
- Gnats 9380: Line-wise vertical scrolling with the scroll arrow functions again.
- Gnats 9515: If the user linked with startup.o during the transformation from .activeitem the active string was not converted correctly.

4.4 Core

- Gnats 9514: The memory leak in the :delete-method has been corrected. This appear when trying to delete the last entry of a user-defined array, in which strings were stored.
- Gnats 9513: Dynamic resizing of user-defined attributes now allocate memory in the correct size.
- Gnats 9512: Memory leak when using applyformat() with only one format string has been corrected. Clean-task is also called up here.
- Gnats 9509: DM_SetContent now also carries out a code page conversion from strings in the user-data field when DMF_OmitString- & DMF_UseUserData-Option is activated.
- Gnats 9499: Assfail no longer occurs by calling parsepath up with null-object and Index>0.
  Notation for the call:

```
object parsepath( string objectname input
  { , object parent input
  { , object dialog input
  { , integer index input } } } )
```

  For index >0 the following is also valid:
  A parent or a dialog/module, from which the search is to be started from, must always be given. An overall dialog/module search among the children does not take place. When using the Rule

Language the dialog/module, which belongs to the rule and from which the call was originally triggered, is automatically used.

- Gnats 9496: DM_GetRecord (function for carrying over records from the IDM side to the client side, which is used from the code generated by +/-writetrampolin) should no longer indicate a memory leakage when using field attributes. In addition, strings from DM_GetVectorValue are stored in a normal temporary buffer and are then deleted at the next DM function call, whereby no DMF_DontFreeLastStrings has been set. DM_FreeVectorValue and DM_GetVectorValue now support the DMF_DontFreeLastStrings option.
- Gnats 9494: Dynamic format change can now be carried out without causing the system to crash; dpy-string with new content will be reallocated.
- Gnats 9486: The string is now taken care of correctly by ID-replacing in associated arrays with object-index and string-value.
- Gnats 9484: Default format handler now leads to a correct reallocation for the display string, when a codepage conversion must be carried out.
- Gnats 9483: Access to .index attribute on a toolbar now returns the correct value.
- Gnats 9455: Invalid ID's in the parameter of DM_CallRule-, DM_CallFunction-, and DM_CallMethod functions are transferred on as null-ID.
- Gnats 9317: A stack-overflow no longer appears, when in the rule/code window of the IDM editor a rule/object was shown whose source text was >64k.
- Corrected: The function DM_GetContent will once again be written out in the tracefile when DMF_OmitActive option is activated.
- Gnats 9516: Treeview method :insert no longer builds up unnecessary references to .pictute[0] object.

4.5 Editor

- Gnats 9481: The special attributes of a toolbar object were not adopted under MS Windows. This has now been corrected.
- Gnats 8990: Parsing is now correctly discontinued during the definition of an invalid method overloading (predefined methods).

4.6 Network

- Gnats 9510: The transferring of strings >64k between client and server is now possible with the C-functions (as parameter or through records). Please take note, that string transfers >64k with mixed client/server versions is not possible, when one of the IDM versions being used is lower than A.04.04.e.

5 Version A.04.04.d

5.1 Windows NT

» Gnats 9467: When the attribute .contentfunc is set on a tablefield object, then it was not possible to scroll through the content with the wheel of a Microsoft roll-mouse. This was due to the fact that no window message was sent relaying that the scroll action had been ended. Now a timer is started which allows for scrolling after a short lag of time.

» Gnats 9465: Support for WinRunner through the tablefield object.

» Gnats 9462: Conversion of attribute .activeitem to 0 with an empty treeview now delivers correct changed-events and does not provoke a access violation.

5.2 Unix

» Gnats 9477: Documentation is now once again installed from install.sh-script.

5.3 Motif

» Gnats 9468: When .maxchars was set on edittexts, invalid memory access occurred after longer strings were shortened.

5.4 Core

» Gnats 9464: External events are no longer lost when the event queue is completely full. The DM function for sending external events now indicates this to the user.

5.5 Editor

» Gnats 9420: Objects that were set to ".mapped = false" were not displayed in the editor under Microsoft Windows.

5.6 COBOL-Interface

» Gnats 9476: The re-transmission of strings in records from a COBOL server function to a client function is now working again.

6 Version A.04.04.c

6.1 Windows NT

- Gnats 9454: The spinbox object could not be handled with a roll-mouse (scrolling).
- Gnats 9453: The child objects of a layoutbox object were not arranged correctly and were displayed simply on top of each other.
- Gnats 9450: Windows, which were defined very small but larger than the minimum size allowed by MS Windows, were displayed too big. One window, whose height was defined at 1 pixel, was affected by this problem.

  Note
  The minimum window size set by Microsoft Windows remains as the valid size limit.
- Gnats 9449: The MDI child windows displayed no title line when the attribute .borderwidth was set to 0.
- Gnats 9442: MDI child windows, which did not possess the standard attribute value, were not correctly displayed. Recommendation: an MDI child window should have the following attributes set to true: .sizeable, .iconifyable, .movable and .closeable. In addition, the window should also possess a title.
- Gnats 9439: When a window, which contained an attached notepage to the top and bottom of it, was either interactively enlarged or decreased in size, then the attribute .real_height of the notepage object (which was active at the time) had a different value as notepage objects that were not active.This error only occurred when the attribute .borderwidth of the notepage was not equal to 0.
- Gnats 9437: When the window belonging to the tablefield object was closed when filling the tablefield object, the program crashed.
- Gnats 9436: The scrollbars of a tablefield object were disabled when clicked upon.
- Gnats 9431: Unicode functions were used which were not available in the Windows 95 System DLLs. These functions included

Function	DLL	Not in Version	Included in Version
SHBrowseForFolderW	shell32.dll	4.00	4.72
SHGetPathFromIDListW	shell32.dll	4.00	4.72
_TrackMouseEvent	comctl32.dll	4.00	4.72

  This error only occurred in an original Windows 95 system. This error no longer occurred after the appropriate version was installed. Both DLLs are installed, for example, from the Internet Explorer 4.01 when the shell integration is chosen.

- Gnats 9429: Problems with the "on open" rule came about, when network functions in this rule were called up and the user began to wildly click around. In such a situation the opening up of the menu is now prevented.
- Gnats 9426: The WIN32 IDM Version was delivered in place of Java IDM Version for Java-WSI.
- Gnats 9423: Model attributes were not taken over when an instance was dynamically applied. This happened when the application was carried out in a visible state and when an attribute on the model was changed, which as a result required it to be newly applied. This was observed on a window whose coordinates were not assumed because the attribute .titlebar was changed.
- Gnats 9422: When the attribute .level on a treeview object was changed, then the attribute .activeitem was set to the last top-level entry. As a result, the entry, which was closed and hung on a child, opened.
- Gnats 9419: The desktop flickered when a dialog or a module object was closed under Microsoft Windows NT 4.0. This happened when the dialog or the module object possessed a visible poptext object, which contained graphics.
- Gnats 9418: When a listbox or a poptext object possessed a text line, which had between 512 and 1024 characters in it, then this sometimes resulted in a crash of the program. This problem has been corrected.
- Gnats 9417: When the attribute .statushelp is set to an empty string, then this was displayed in the status line and not, as specified, in status help.
- Gnats 9416: When an attribute change was carried out on an object, which had the focus, then the focus was set to the first object in the window. This resulted in the necessity of having to destroy the object and then newly apply it. In this case the treeview lost the focus when a newline was inserted via :insert method (for example by double clicking).
- Gnats 9413: The ISA Dialog Manager also displays insensitive scrollbars, which cannot be used when the object in question is insensitive. This is not the Microsoft Windows standard, however, it was implemented because it gives the user useful feedback. Another form of positive feedback is the scroll arrow of objects, which are directly implemented from the ISA Dialog Manager. These are displayed as insensitive when it is not possible to further scroll in one or the other direction. This is valid for the following objects: groupbox, layoutbox, notepage, scrollbar and the tablefield.
- Gnats 9403: When an application was dynamically linked, the tracing was switched on, and the application was ended before DM_Inititialize was called up, then this led to a crash of the program (abnormal termination).
- Gnats 9401: It was not possible to close an open poptext by pressing the `Escape` key. This was the case when a certain accelerator was defined.
- Gnats 9400 and Gnats 9399: When the object edittext had the focus, the cursor position was changed when one of the following methods were called: :getformat(), :gettext(), :replacetext() or :setformat().
- Gnats 9398: Unresolved externals existed when a COBOL application was linked with the Dialog Manager DLLs.

- Gnats 9396: When a popup menu was opened through an RTF edittext, the mouse pointer did not change into an arrow. This is an error of the Microsoft Windows RTF object. This error is avoided in the Dialog Manager.
- Gnats 9387: When during the runtime of a menubox the menubar was made invisible, then the menubar was not completely updated.
- Gnats 9371: The following objects now display an optional scrollbar: groupbox, layoutbox, notepage and window. This is true when the virtual size is smaller than the object size, but the scrollbar covers up a part of the virtual area in the other direction. Up to now an optional scrollbar was only shown when the object size was smaller than the virtual size.
- Gnats 9369: The text of a statictext object is cut off at the 127th character. This is true when the statictext object is situated within a statusbar object. This limit is set by Microsoft Windows.

  Note regarding the "statusbar" under Microsoft Windows
  A statusbar can only display a maximum of 256 characters. For this reason only up to 256 child objects (statictext) can be visible in the statusbar at the same time. Furthermore, a text field within a statusbar can only display up to 127 characters.

  Note to object "statictext" under Microsoft Windows
  A text field in a statusbar can only display a maximum of 127 characters. Due to this, the text (*.text* attribute) should not contain more than 127 characters, when the text field is a child of a ***statusbar*** object.
- Gnats 9354: The Windows poptext marks the text when the size of the poptext is changed. As a result, after the poptext has been applied, it is displayed differently after it receives the focus. As long as the poptext does not have the focus, then this conversion will be suppressed. Consequently, one is able to set .startsel and .endsel in advance. Please keep in mind that the Windows poptext automatically converts the selected area as soon as it receives the focus.
- Gnats 9289: The activation of the main window did not function correctly when it owns a visible toolbar. The main window was displayed as active, however none of the child objects had the focus.
- Gnats 8862: When within a splitbar object the size of an area was changed, then the focus was set to an incorrect object. This happened when the original focus object did not lie within the father of the splitbox object. The behavior is now adjusted to the behavior of the tablefield object. The focus is no longer transferred to the splitbox object.
- Gnats 7266: Character problem when scrolling within a layoutbox.
- Corrected: Debugger was unable to find its module even though IDM_HOMEDIR was set.
- Corrected: When YFileGetBuffer was called up and 500 bytes were already in the buffer, then it was possible that the system crashed. This happened, for example, in the debugger or the editor when a rule was displayed.

6.2 UNIX

- New: Platform AIX5.1+Motif2.1 is now supported.
- Gnats 9391: Warning in install.sh-script with respect to missing data, libIDMapp.a is now avoided.

6.2.1 Installation

The install-sh-script now contains additional options to avoid the editing of data.

New options:

-h	indicates a short help
-nodoc	documentation is not installed

only with root-rights:

-uid <id>	User-ID for the installed data (default: root).
-gid <id>	Group-ID for the installed data (default: bin).

The conversion of User-ID and Group-ID takes place in a standard fashion only under the user "root". If another user installs the data then his User-ID and Group-ID will be used for the installed data.

6.3 Motif

- Gnats 9444: The 64k limit for text buffer in Motif-WSI has been repealed.
- Gnats 9392: Inquiry/setting of .topitem on the treeview now correctly takes into consideration the nodes that are not open.

6.4 Core

- Gnats 9448: An error is no longer triggered, when an invalid object ID is given as an argument for an external event.
- Gnats 9447: When deleting entries in an associated array (via :delete method), this went beyond the allocated memory area that then lead to a program crash as well as other problems.
- Gnats 9446: When using strings as data or parameters in external events, which were sent from the DDM server side, this resulted in strange program crashes (destroyed memory management). This has now been corrected.
- Gnats 9443: Problems with string returns of closequery, ID replacing, codepage conversion have been corrected. Support of the options DMF_GetMasterString, DMF_GetLocalString and DMF_DontFreeLastStrings have been implemented.
- Gnats 9438: The out of date attribute .focusable is now ignored; no more side effects for .sensitive & .editable.

» Gnats 9425: Correct iconv-codepage name for the conversion via CP_ucp (iconv) is now used on HP-UX. The UserCodePage can be changed several times.

» Gnats 9408: Resource changes from the Rule Language are once again correctly transmitted to WSI.

» Gnats 9404 - During the coding of unicode characters according to UTF8 bits 13/14 disappeared. As a result, \u8fe8 was displayed as \u83e8.

» Gnats 9402: The maximal length of display strings was not set to .maxchars when the format string was empty. As a result, an edittext object, whose width was not set, was too narrowly displayed.

» Gnats 9397: The setval-event of an object, which was then immediately destroyed, is now deleted correctly.

» Gnats 8596: The attributes .startsel and .endsel (in numerical scanf-format) were incorrectly calculated when the part after the comma ended with null. The nulls were suppressed in the calculation although these were contained in the attribute .content. This error was discovered in format "%008'.6,2d".

6.5 Editor

» Gnats 9427: The auto-completion-list no longer lists the non-accessible, non-exportable hierarchical children.

6.6 COBOL-Interface

» Gnats 9407: The byte sequence in the output parameter was switched around. This was due to the fact that namely the sequence was swapped before the COBOL function was called up, and not afterwards. This error only occurred on architectures whereby the byte sequence between COBOL and C are different (i.e. Intel). This error did not occur with Windows. The error was observed for integer parameters under Linux with Intel processors.

» Gnats 9405: The COBOL SetTerminator was not handled correctly. Now, the terminator character halts the text. In addition, the terminator characters are now converted in the internal code page.

Note
The conversion takes place directly through the function call-up. This means that the conversion in the network version takes place on the server, and that the Dialog Manager texts and not the COBOL texts are displayed in the tracefile.

6.7 Net

» Changed: Strings are now read in full length. When strings for calling up COBOL functions were transferred, then only the COBOL size was read. Unfortunately, this can lead to a crash of the system.

» Corrected: As a result of a network error, it was possible that the program under Windows crashed on the client side.

» Gnats 9460: Multiple conversions of the UserCodePage should function now without any problems.

7 Version A.04.04.b

7.1 Windows NT

» New: The filerequester object with .style = fr_directory now uses the new user interface. An additional edittext is now shown when the attribute .mustexist possesses the wrong value.

Requirement
The changes are only available when version 5.0 of the system data shell32.dll is available on the system. This is the case with Window versions beginning with Windows 2000 and Windows ME. A limited version (only an additional edittext) is already possible with the version 4.71 of the system data shell32.dll. This is the case with Windows 98 and the versions that follow. The applicable version can be installed on older systems with the Internet Explorer 4.0 or 4.1 (additional option).

» Gnats 9376: The object menuitem was displayed as a menu separator when no text was given (Microsoft Windows feature). This lead to the problem that a menuitem, which received a text at a later time, was shown as a menuitem but it remained insensitive. This error seemed to occur when a menuitem was dynamically set.

7.2 Motif

» New: Transparency of graphics in the treeview has now been taken into consideration.

» New: The image object now supports the attribute .borderwidth. This allows graphics without borders. The default value for the width is 2.

» Gnats 9381: When implementing .activeitem on a poptext, the umlaut was not correctly displayed.

» Gnats 9379: Memory leak when marking tablefield texts has been corrected.

» Gnats 9362: The rider of a notebook receives the color of the corresponding notepage.

7.3 Core

» Gnats 9377: getvalue on .activeobject now delivers correct values.

7.4 Editor

» New: Attribute .borderwidth is now available for the image object in the layout page.

7.5 Debugger

» Gnats 9383 and 9384: Breakpoints can now be set in the given lines.

8 Version A.04.04.a

8.1 Windows NT

- Gnats 9375: When a userdefined attribute was set to a model of a control object, whose attribute .mode was set to mode_server, then the program crashed.
- Gnats 8735: On the treeview object it was not always possible to start a drag and drop action. This error always came about, when .activeitem possessed an invalid value. For example, this happened directly after applying .activeitem = 0, which is true when the treeview object is empty.
- Gnats 9359: It is possible that a connected OLE control crashes. This happened after Gnats 9349 was corrected, whereby the OLE control was activated after the control object received the focus. This could be the reason why the OLE control itself transferred the focus, which actually is not allowed. It could be that the OLE control was so mixed up that it caused a crash. This problem has been taken care of through a delay in the activation of the OLE control.
- Gnats 9357: It could be that an assfail was caused in module table2 line 691, when the user directly clicked on the tablefield object with the right mouse button. This error could not be reproduced; code review revealed a problem, which was changed. Similar problems occurred when the user navigated on a tablefield object by using the keyboard, and when the tablefield object was the target of a drag and drop action.
- New: The tile resource can now load images in ICO format. It is no longer necessary to connect an Icon through a resource data. If the ICO data contains Icons of various sizes, then the first size will be used. If the attribute ".scale" is set to the tile resource, then the data will not be newly read (due to reasons of speed). Instead, the first loaded Icon will either be enlarged or reduced in size. An exception to the rule includes treeview and notepage objects, whose graphics are retrieved at the time of application. These objects force a new loading of the ICO data.
- Gnats 8841: The object statictext now supports multilined texts under MS Windows. Please take note that an automatic calculation of the size for multilined texts is not available.
- New: Support for the attribute .mapped. When this attribute is set to false, the windows object will be applied, however it will not be made visible.
- New: Background graphics for the grouping objects layoutbox, notebook, notepage, groupbox, splitbox, spinbox, toolbar and window are now possible. The object statusbar is not supported.
- New: The open-event on the menubox object now delivers the object and the coordinates of the mouse click, which lead to the opening of the menubox. Where applicable, the index of the entry that was clicked on will also be delivered. When the open event is created via the keyboard then the object, and if necessary the index, will be delivered.
  The object thisevent now supports the following attributes when opening an event on the object menubox

Attribute	Type	Description
.value	object	The object that was clicked on, or rather which held the focus as the open event was triggered.
.x	integer	The x-coordinate of the mouse click regarding the object thi-sevent.value. When the open event was triggered via the keyboard, then the attribute was not set (type *void*).
.y	integer	The y-coordinate of the mouse click regarding the object thi-sevent.value. When the open event was triggered via the keyboard, then the attribute was not set (type *void*).
.index	integer or index	The entry index which was clicked on or which held the focus when the open event was triggered. Value is only set when the object thi-sevent.value supports an integer (is the case with listbox and treeview) or an index (only with tablefield). Otherwise, the attribute is not set (type *void*).

» New: The image object can now react to mouse-over events (mouse pointer is either over the object or not). For this purpose, the object received the attribute .mouseover and the attribute .picture was expanded. Now, graphics can be set in the following cases

Attribute	Type	Description
.picture[tile_default]	tile	Default Graphic This is always shown when the object is in a normal state (not activated).

Attribute	Type	Description
.picture[tile_active]	tile	Active Graphic Is shown when: 1. The image object is active and it possesses the checkbox style. 2. The image object possesses the checkbox style, is not active and the mouse button is pressed down over the image object. 3. The image object possesses the push-button style and the mouse button is pressed down over the image object. Exception: when tile_active is not set, then tile_default will be shown. 4. This will not be shown, when .mouseover is set and the position of the mouse is over the image object.
.picture[tile_insensitive]	tile	Insensitive Graphic Is shown when the image object is insensitive (.real_sensitive = false). Exception: when tile_insensitive is not set, then tile_default and/or. tile_active (checkbox style and active) will be shown.
.picture[tile_mouse_over]	tile	Mouse Over Graphic This is shown when the following cases apply: the .mouseover is set, the mouse is located above the image object, the mouse button is not pressed down, and when it is not active given the image object does not possess the checkbox style. In this state the borders are accentuated.

Attribute	Type	Description
.picture[tile_active_mouse_over]	tile	Active Mouse Over Graphic Is shown when .mouseover is set, when the mouse is located above the image object and: » the mouse button is pressed down on top of the image object. » the image object possesses the check-box style and is active.
.mouseover	boolean	The attribute default value is normally set to *false*, so that the image object behaves, as it should. When the value is set to *true* then the tile resources in .picture[tile_mouse_over] and .picture[tile_active_mouse_over] are taken into consideration. The behavior corresponds to that of the image object in MS Internet Explorer.

Note

1. There is no strategy, except in the previously mentioned cases, to display a different tile when a tile is not set. For this reason it is important that all necessary tiles are defined.
   The following is necessary

tile_default	always
tile_active	with checkbox style
tile_mouse_over	when .mouseover is set
tile_active_mouse_over	when. Mouse over is set

2. A mouse button is considered depressed, when the left mouse button is pressed down while the mouse is located over the image object.

» New: In Windows-WSI the progressbar object is now supported. Object-specific attributes include

Attribute	Type	Description
.textfgc	color	color of the labeling text
.minvalue	integer	minimum value of the progress area
.maxvalue	integer	maximum value of the progress area

Attribute	Type	Description
.curvalue	integer	actual position of the progress
.direction	integer (1|2)	alignment of the progressbar, vertical (=1) or horizontal (=2)
.style[style_continuous]	boolean	indicates if the bar is to be displayed in blocks or in a continuous manner
.style[style_labeled]	boolean	indicates if the bar should be labeled. This is only valid when .direction = 2 (horizontal)

- Gnats 9187: It was not possible to leave an OLE control via keyboard navigation buttons, which was activated from a control object with the attribute .mode = mode_client, and which at the same time had the focus.
- Gnats 9349: The focus was not correctly set to the OLE control, which is connected through a control object via the attribute .mode = mode_client. As a result, it was not possible to navigate with the keyboard buttons within the OLE control.
- Gnats 9025: The accelerators on entries within context menus did not function after the context menu was opened.
- Gnats 9347: The statusbar help was also displayed for the transparent areas of an object. As a result, the statusbar help was constantly switching back and forth between object help and father help. This was observed on a notebook in an area next to a rider.
- Corrected: When the mouse was positioned over the last field of a listbox object during a drag and drop operation, the user was not notified when the attribute .style did not possess the value style_single.
- Gnats 9344: During a paste operation in an edittext object, the selection (.startsel and .endsel) was always set to the mouse position. This also happened when the action was carried out over the keyboard.
  The attributes .startsel and .endsel of the object edittext are no longer set to the mouse position when carrying out paste events. The mouse position is saved in the attribute .index of the paste event.
  The attribute .index of the paste event contains the target area for the object index. This is:
  - Keyboard Operation
  - first(thisevent.index): is equal to this.startsel
  - second(thisevent.index): is equal to this.endsel
  - Mouse Operation
  - first(thisevent.index) and second(thisevent.index) is equal to the mouse position in the text, whereby 1 marks the position behind the first letter.

Note

Second(thisevent.index) is no longer the line above which the mouse is positioned. The first/second values are the same with the mouse operation, because during the drag and drop operation it is not possible to mark an area.

- Corrected: During the management of the interface object, it came about in some cases that reading and writing beyond a memory block limit took place. This error only occurred when too many (>256) interface objects were temporarily created. As these temporary interface objects were released, the memory block, which contains cross references for all interface objects, was greatly reduced in size.
- Corrected: When a child window was dynamically created as a child of an empty window, then an assfail in rule\event line 1345 could occur when the child window was deleted. The reason for this was that the father window was not an MDI father, which had to be destroyed and newly created when creating the child window. Unfortunately, this was not cleared up by the children windows, which were already partially created.
- Gnats 9307: When a window possessing a size raster was not maximized, then the window did not take up the entire display area. If the window was maximized from its iconified state, then the window took up the entire display area. This problem came about because in the calculation of the maximum window size it was not taken into account that Windows, as a rule, enlarges a window to a size in which the window frame is positioned beyond the normal display area.
- Gnats 9324: When the attribute .picture of the active (.active = true) notepage object was set to 0, then the first child of the preceding notebook object was activated.
- Gnats 9313: In maximizing a window that was tethered at the top and bottom, the child object attached to the bottom was initially positioned incorrectly (flickering). This error is not understandable. We believe this has been taken care of with the correction of Gnats 9314.
- Gnats 9338: When an object (type interface) of an OLE method was called up, which returned an object from type interface as a return value, this sometimes led to an ASSFAIL in object/slot line 2282. This ASSFAIL came about when the interface object, whose method was called up, was deleted from the garbage container. This was due to the fact that the interface object was no longer referenced by itself or by its children. The new child object was not entered as a child yet.
- Gnats 9314: In windows that had multilined menubars, it came about that child objects were sometimes positioned falsely. This happened when the size of the window was changed within a rule. This error only occurred on windows with ".yauto := 0;" and on child objects with ".yauto := 0;" or ".yauto := -1;". The error also appeared when the size of the window was newly calculated as a result of other actions that took place. In the previous Gnats entry this happened as a result of changing the width of a docked toolbar.
- Gnats 9335: Error messages are now given out in the log file, when COM or MSXML was unable to be installed. In addition, every COM error from MSXML is now written in the trace file.
- Gnats 9334: When the adding of an XML node failed, a program crash was possible. This happened, for example, when an XML attribute was to be added via the method :add:

```
doccursor:add(nodetype_attribute, "A", "A B", false);
```

- Corrected: There was no empty space between the menu text and the accelerator.
- Corrected: When querying the filerequester object, no path was delivered.
- Gnats 9311: In pushbuttons, in which the size was set to 0, the nmemonic letter was no longer marked.
- Gnats 9305: It sometimes came about that a messagebox was not applied/created. This happened because a window was designated as the father for the messagebox, which had just been destroyed. Now, the suggested father is checked and if necessary the messagebox is opened without the father. As a result of this, the messagebox can only be activated directly, and not by clicking on the application window.
- Corrected: When a popup menu was opened with the method :openpopup, then internal WSI-Id's were not released when the window that belonged to it was closed directly without having closed the popup menu.
- Gnats 9306: When a popup menu, which was defined to a groupbox or a notepage, was opened with the method :openpopup, then no select event was created. In this case the status text also remained the same, i.e. no changes were carried out.
- Gnats 9309: The selected entry of a poptext was changed when a statusbar entry was changed. Here, the user had to open the poptext with the mouse and then leave the mouse over an entry. Now, when the user changed selected entry via the cursor button, the selected entry sprang back when a statusbar entry was changed in the select rule. This does not happen anymore. Attention: this kind of “error” can happen again at any time. The springing back was caused by a mouse-move-event, which was caused through the change in the statusbar. In principle, this can always take place when visible objects in the poptext select rule are manipulated. An additional mouse move is necessary, for example, when the mouse pointer needs to be changed.
- New: The search for idmuni.dll has been improved. Now these are found more often. It is also possible to use an idmuni.dll of another version, as long as it provides all necessary functions.

8.1.1 OLE Events with Return Parameters

Up to now the OLE expansion of the IDM has only allowed for events (also known as Messages) without return parameters. These restrictions no long exist in the A.04.04.a version.

8.1.1.1 OLE Events without Return Parameters

Up to now the definition and use of OLE events took place through the message object on the server side. The event was received as an “external” event on the client side. This is what it looked like:

Client

```
dialog OLEClient
window Wi {
  control Co {
    .xauto 0;
    .yauto 0;
```

```
    .mode mode_client;
    .uuid "499593d1-a159-11d1-a7e3-00a02444c34e";
    .connect true;
    .active true;
    on extevent "Msg" (integer I, string S, boolean B) {
      Wi.title :="Received Msg: "+I+", "+S+", "+B;
    }
  }
  on close {
    exit();
  }
}
```

Server

```
dialog OLEServer
message Msg(integer I, string S, boolean B);
model control MCo {
  .mode mode_server;
  .uuid "499593d1-a159-11d1-a7e3-00a02444c34e";
  .message[1] Msg;
  pushbutton PbSendEvent {
    .text "Send Event";
    on select {
      sendevent(this.control, Msg, 123, "hello", true);
    }
  }
}
```

8.1.1.2 Return Parameter on the Server Side

This process still functions. In order to support events with return parameters two new keywords exist in the Rule Language: **extern** and **public**.

By **extern** one declares rules on the server object that should be handled as events, and which should be sent to the client. A rule body is not allowed for such a declared rule. The only data types that are allowed include: *integer*, *string*, *boolean* and *object*. The parameters that are allowed are: **input**, **output** and **input output**. The return of such a declared event can take place directly as a return value or through the use of an output type. By a parameter of an *object* data type, only the null object or the IDM object of the control or subcontrol class can be transferred because these are the only ones that have an IDispatch interface that is relevant for OLE. Objects of other IDM object classes are transferred to the client as null objects.

The sending of such an event can no longer take place through the sendevent built-in. This is true because built-in calls do not support return parameters, but rather are executed through a direct call of the rule that is declared as “external”. One cannot see if the client has handled the event or not. This can only be determined in examining the return value.

Example Server

```
dialog OLEServer
model control MCo {
  .mode mode_server;
  .uuid "499593d1-a159-11d1-a7e3-00a02444c34e";
  extern rule boolean Msg(integer I, string O output);
  pushbutton PbSendEvent {
    .text "Send Event";
    on select {
      variable string Str;
      if this.control:Msg(123, Str) then
        this.text := Str;
      endif
    }
  }
}
```

8.1.1.3 Return Parameter on the Client Side

On the client side of an IDM-OLE application, the **public** keyword serves as a way of marking rules, which are allowed to be called up as event methods from the “outside” (i.e. from the OLE server side). This way the user is protected from unwanted calls of “private” methods. The same data types and parameter types, as is the case on the server side, are allowed with public rules. In contrast to the previous ***Messages***, these events are called synchronous. When an event is not available as a “public” rule, then this is asynchronously processed, just as it has been in the past, as an external event. However, this is done without support of the return value. Public methods are handled on the client side as completely normal rules and only have an additional meaning as an event under a control object in *mode_client*.

Example Client

```
dialog OLEClient
window Wi {
  control Co {
    .xauto 0;
    .yauto 0;
    .mode mode_client;
    .uuid "499593d1-a159-11d1-a7e3-00a02444c34e";
    .connect true;
    .active true;
    public rule boolean Msg (integer I, string O output) {
      Wi.title :="Received Msg: "+I;
      O := "hello";
      return true;
    }
  }
```

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

8.1.1.4 Conversion between OLE and IDM Types

Here's a small tip for dealing with data conversion when transferring data to the IDM OLE client side. The data is received by the IDM as a VARIANT (MS Windows data type) and then it is converted into an IDM data type according to the following table

Variant Data Type	IDM Data Type
VT_BOOL	boolean
VT_BSTR	string
	integer
	object (dynamically set subcontrol)

Parameters, which expect a return value from the client side, are additionally marked through VT_ BYREF in the VARIANT structure.

8.1.1.5 Tracing the IDM-OLE Activities

When the Client tracing function is turned on - as is also true on the server side - the tracing lines give hints to the actions that take place in the IDM's OLE interface through the use of [OC] and [OR] tags. The following short overview should explain the meaning of the identifier following the tag, which also leads to an improvement in debugging. Tracing is carried out regardless if the attribute/method or event actually exists. This makes tracing the activities on the OLE interface easier to carry out.

OC/OR-Tracing-Output		Description
Client	Server	
	DM_OLE_CallMethod	Calls up a method on the server
	DM_OLE_GetAttr	Client requests the value of an attribute from the server
	DM_OLE_SetAttr	Client sets the value of an attribute on the Server
DM_OLE_AttributeChanged		Client receives the message, that an attribute has been changed on the server side

OC/OR-Tracing-Output		Description
Client	Server	
	DM_OLE_CallEvent	An event is sent from the server side
DM_OLE_HandleEventInvoke		Client should handle an event

8.2 Motif

- Gnats 9312: The access to topitem should now be OK. Redrawing of the treeview should, when possible, show the original (pixel correct) "topitem".
- Gnats 9368: Umlauts, used in edittext formats, are now correctly shown on the screen.
- Gnats 9373: A sudden crash caused by an arithmetic exception in Motif layout routine when making a notebook visible through the conversion of the activation calculation has been corrected.
- Gnats 9374: The performance, when switching over to "invisible" or when creating a new treeview, has been improved.

8.3 Java Window Interface

- **Java Version**
  The ISA Dialog Manager Internet interface now needs Java 2, i.e. a Java runtime environment (JRE) or a Java development environment (JDK) in the versions 1.2.x or 1.3.x. This means that the Java Plugin is now needed for an applet. As a result, HTML-pages must be adjusted accordingly (see examples\idmapplet.html).
- **New**
  The ISA Dialog Manager for Internet now supports the attribute .dialogbox on the window object. Please take note, that a dialogbox always needs a main window so that it is possible under Microsoft Windows to activate it from the taskbar (selecting the main window). The main window is automatically calculated by the ISA Dialog Manager.What actually happens is the application window, which was lastly active, takes over the status as main window. This window should not be closed so long the dialogbox is open. For this reason the following rules should be avoided:

```
on select
{
  MyDialogbox.visible := true;
  this.window.visible := false;
}
```

  This rule closes the main window of the dialogbox, and the window begins to flicker. A change in the sequence avoids the flickering, but at the same time the problem arises that the Dialog Manager is not able to designate a main window. This is why it is better to leave the window open until the dialogbox has been closed.

When the ISA Dialog Manager was not able to designate a main window, or if the main window was closed, then it was possible that the dialogbox was partially obscured by another window. In this case all application windows are blocked, i.e. however, no further window in the taskbar can be seen.
Only through the iconification of all other windows, is the appearance of the dialogbox possible.

» Gnats 9358: The following error could occur, when an object was set to invisible:

```
Error: YNodebox.inqVal: <Label> has no widget
```

whereby <Label> is a Dialog Manager name.

8.4 Core

» New: Background graphics for grouping objects have been added. The following attributes are available for this:

Attribute .tile

Name:	.tile (C: AT_tile; COBOL: AT-tile)
Type:	object (C: DT_tile; COBOL: DT-tile)
Index:	none
Access:	set/get
Inheritance:	yes
Default:	null
Objects:	groupbox, layoutbox, notebook, notepage, spinbox, splitbox, statusbar, toolbar, window

The attribute .tile indicates a tile resource, which is displayed as a background. The way in which the graphic is displayed is determined through the attribute .tilestyle.

Attribute .tilestyle

Name:	.tilestyle (C: AT_tilestyle; COBOL: AT-tilestyle)
Type:	enum (C: DT_enum; COBOL: DT-enum)
Index:	none
Access:	set/get
Inheritance:	yes
Default:	tilestyle_tiled
Objects:	groupbox, layoutbox, notebook, notepage, spinbox, splitbox, statusbar, toolbar, window

The attribute .tilestyle indicates how the background graphic is to be displayed. The following values are possible:

- tilestyle_tiled
  The background graphic is repeatedly displayed next to or underneath each other until the background is filled.
- tilestyle_centered
  The background graphic is centered directly in the middle.
- tilestyle_stretched
  The background graphic is changed in size so that it fills out the entire area available.
- tilestyle_parent_tile
  The definition of the father object is taken over. This gives the impression that the object is see-through. This value is not valid for the object statusbar, toolbar and window.

- Changed: The object thisevent now supports the access of all attributes, which have been changed in comparison to the default value. Up to now, a access was either allowed or denied depending on the event and the object. As a result, it was not possible for a WSI to deliver additional information. Attention: not every attribute of thisevent object is available on every platform. The attributes that are available can be looked up in the documentation.
- Gnats 9011: Initial selection of filerequester can be preset.
  When the new boolean attribute .startsel (default is false) is set to true on the filerequester object, then the file/directory-selection is preset with the value in .value-attribute.

  Restrictions

  - Only one selection (.value must be scalar) is possible.
  - Under Windows NT presets are only possible for loading or saving files but not for directories.
  - The preset value in .value-attribute must contain the entire path and should correspond to the given directory (.directory-attribute).
  - The initial selection is displayed in the entry field. A selection in the selection list does not take place. This problem is due to the system functionality of the nativ Motif widget/ MS Windows control.
- New: The methods :instance_of() and :parent_of(), can be called up from all object classes.

```
boolean :instance_of(object Model);
```

  New method :instance_of() allows the user to determine if an object was derived from a particular model. The call returns the value true as soon as the given model in the .model chain of the instance appears. Otherwise, false is returned.

```
boolean :parent_of(object Child);
```

  With this new method :parent_of() it is possible to determine if one is a father, grandfather or a great grandfather,…. of a child; the object via single or multiple recursive calls of .parent from the child is able to be reached.
- Gnats 9327: An error message occurs for a non-resolvable export/import problem.
- Gnats 9329: Application of labels larger than 64k is now possible.

- Gnats 9343: Methods :findtext, :gettext and :replacetext only functioned for edittext objects under MS Windows, in which the attribute .options[opt_rtf] was set to true. Now the methods function for all edittext objects. However, please be aware that the parameter type is only allowed to possess the value content_rtf on systems that support RTF.
- Gnats 9350: The time needed for loading modules (between unloading & start) has been reduced.
- Gnats 9339: The .order attribute is now written out with -writedialog or from the editor through the objects (menubox, menusep, menuitem).
- New: Attribute .real_modified has been added to the objects edittext and poptext.

  Attribute .real_modified
  This attribute indicates if the content has been changed after the object received the focus. The setting of the content resets the value of this attribute.

Data type:	boolean
Access:	get, when the object is visible (.real_visible = true)
Allowed by:	edittext and poptext
Supported by:	Microsoft Windows, on all other platforms true is always returned.

  Example
  The attribute can be used to find out if a "charinput" has really changed the content:

```
edittext {
  on charinput {
    if this.real_modified then
      print "" this + " changed";
    endif
  }
}
```

- Gnats 9353: The unnecessary setting of default objects is avoided when the model originated from a different module.
- Gnats 9372: Strings with 65533 characters are now correctly transferred through the C-interface.

8.5 Editor

- Gnats 9298: Memory access infringement in a format used by the editor has been corrected.
- Gnats 8611: When the source text window is open, the actual object is immediately shown, when allowed.
- Gnats 8611: A double-click on a "page" object in the object browser opens the corresponding editing window. For example, the object/attribute window is opened, when a pushbutton is clicked on twice. For objects belonging to children, the behavior remains the same, i.e. the children are either shown or not. No window is opened.

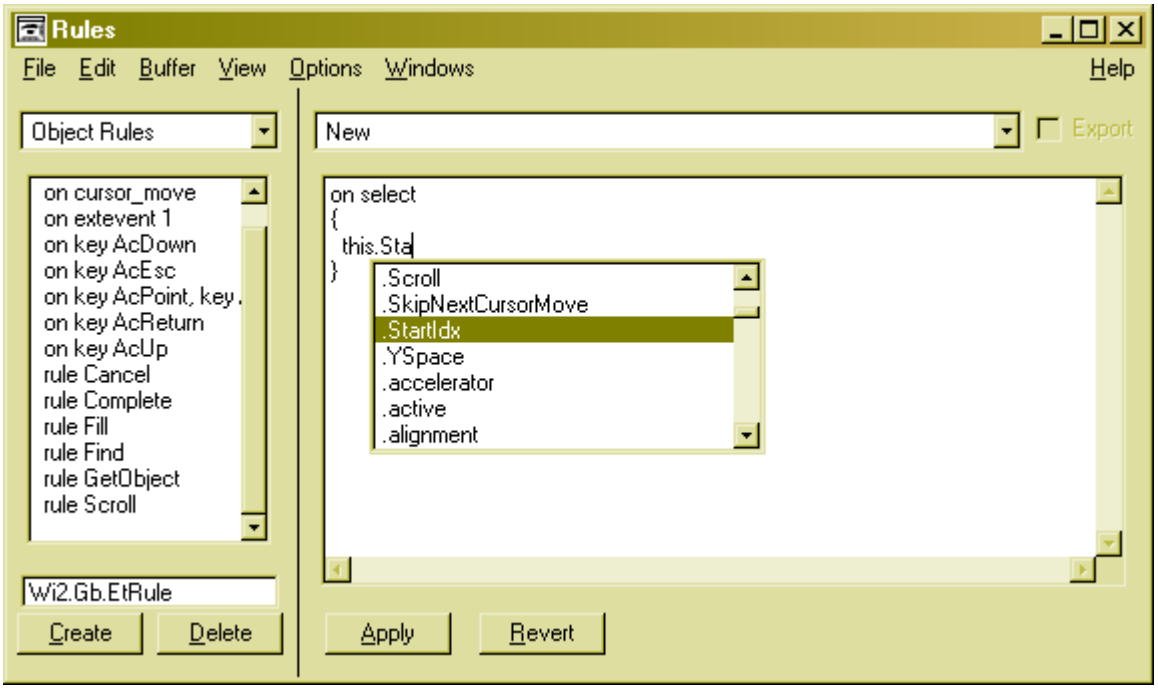
- Gnats 9323: The entry of control characters, which did not change the content of an edittext, brought about that the "Apply" and "Revert" buttons had to be used.
- Gnats 8496: By "Save file as..." the user was not asked if an existing data was already selected.
- Gnats 9315: The setting of COBOL as the language for application functions works again and no longer created any Java functions.

8.5.1 Auto-Complete Feature in the Rule Window

The rule window in the editor is now extended with an auto-complete feature in order to simplify the development. This is always turned on, but can be turned off in the configuration window.

This service helps the programmer when entering attributes and methods, and it functions as follows:

- When the button `.` or `:` is pushed, then a selection list will pop up near the cursor, when an identifiable object is positioned to the left of it.
- What ever is positioned to the left of the cursor is handled as a path on an object and the system will try to identify this object.
- The selection displays possible attributes or child objects (by the last `.` used), or the methods available on an object (with a `:`). Predefined and user-defined attributes and methods are displayed.
- The programmer can continue as usual with his typing. The suitable choice will then be marked in the selection. It is also possible to navigate within the selection list with the Cursor `Up` and `Down` keys.
- The marked selection is then taken over to the source text either via the `Return` key or by selecting via mouse.
- Pressing the `Escape` key will halt a selection. Another way of interrupting the selection is by moving the cursor to another area either with the mouse or by using the Cursor `Left` or `Right` keys.



Limitations

As object, a path expression, which does not contain any dynamic values, is recognized in front of the . or the : (i.e. variables, attribute values, expressions). Also, keywords such as this, thisevent and setup are recognized. The completion can only take place within rules, whose affiliation to an object is known and where no parsing is needed. For this reason it is always better to work in the “Object Rules” mode (rules for objects).

Example

```
dialog Dialog

window Wi
{
  object ObjAttr := Pb;
  pushbutton Pb {}
  pushbutton {}
  pushbutton {
    record Rec {}
  }
}
record Rec2 {
  object Attr;
}
on dialog start {
  variable object LocalVar := Dialog;
  variable integer Count := 2;
}
```

In the start rule the following paths should be completed

OK	Not OK
this.	LocalVar.
Dialog.Wi.Pb:	Wi.ObjAttr:
Dialog.Wi.PUSHBUTTON[2].Rec.	this.parent.
	Dialog.lastrecord.Attr.
	Wi.PUSHBUTTON[Count-1]:

8.6 COBOL Interface

» Gnats 9333: Truncation from return strings in COBOL functions corresponding to the put-separator has been corrected.

» Gnats 9336: Supports DMF_SetUserCodePage in the DMcob_Control function.
The DMcob_Control function now also supports the DMF_SetUserCodePage action. Here, the field DM-usercodepage in DM-StdArgs must be additionally documented with the name of the code page to be used. It is recommended to do this on CP_ucp before the conversion of the application code page via DMF-SetCodePage. This will guarantee the coding of the DM-usercodepage strings.

Example

```
move DMF-SetUserCodePage to action.
move "ISO-8859-15" to DM-usercodepage.
call "DMcob_Control" using DM-StdArgs DIALOG-ID action.

move DMF-SetCodePage to action.
move CP-ucp to DM-Options.
call "DMcob_Control" using DM-StdArgs DIALOG-ID action.
```

8.7 Demos

» The drag & drop examples have been extended with an additional treeview.

8.8 Network

» Gnats 9316: When calling up DM_Control() from the server side, a DMF_IsInternalCP as optional parameter is not automatically ordered.
In addition, DMF_Cobol* actions are no longer transferred to the client side, but rather are handled on the server side. Inasmuch, now various settings on various DDM servers are supported.

9 Version A.04.03.c3

9.1 Motif

» Gnats 9846: Avoid doing a short flickering of an typed character at an edittext width covered format (S, password) and slow X-connection.

10 Version A.04.03.c

10.1 Windows NT

- Gnats 9261: During multiple selection the filerequester object only returns the path of the file name instead of the file name itself. Now, the path is returned together with the file name.
- Gnats 9269: The help key `F1` triggered a select event for any object. This was due to the fact that the help key `F1` was interpreted as a mnemonic for an object, which did not have a text and which principally supports mnemonics.
- Gnats 9285: The color of GIF files, which contained a transparent color and whose index was larger than 0X75, or whose RGB values resulted by an bitwise AND operaton, were not displayed in a transparent manner.

  Important
  Please note that transparency is only supported for tile resources and not when a GIF file is assigned to a .picture attribute in the form of a text string.
- Gnats 9286: Fewer pictures were represented in comparison to earlier versions. Now, only one Windows bitmap is created for a tile object when possible. This will allow for more pictures to be shown at the same time.
- Gnats 9287: The sequence of events in the Dialog Manager has changed with respect to the object. Now the focus event comes before the select event. As a result of this, the deselect event now comes before the select event when an edittext has the focus and when an object image is clicked on. Take note: the events 'deselect' and 'modified' are triggered as a result of the loss of the focus. This is why there is no 'deselect' or 'modified' when an object image having the attribute .focus_on_click = false clicked on.
- Gnats 9291: When the Dialog Manager was started on a non-Unicode-capable system (Win9x), no pictures were displayed in the treeview object. This error has been corrected in the idmuni-DLL.
- Gnats 9292: The attribute .activeitem was not actualized in the treeview object when the Dialog Manager was started on a non-Unicode-capable system (Win9x).
- Gnats 9294: No MDI children windows were shown when the Dialog Manager was started on a non-Unicode-capable system (Win9x). This error has been corrected in idmuni-DLL.
- Gnats 9297: The setting of setup.overridecursor did not change the cursor by longer calculation work when updatescreen() was not called up.

  Important
  When the attribute .overridecursor is set to a cursor on the object setup, then the mouse pointer will be explicitly altered. The reason for this is that the mouse pointer should change automatically when updatescreen() is not called up once in a while during a longer calculation. This setting of the mouse pointer can lead to the mouse changing itself short-term, for example when the mouse is situated over a window that does not belong to the application.

» Concerning pictures in menus, now an eventual defined transparent color is replaced by a menu background color. In earlier versions only the transparent color was displayed.

10.2 Motif

» Gnats 9259: As a result of setting the focus to a treeview that was previously enlarged, the active element receives the focus frame again.

» Gnats 9278: When .activeitem is set not equal to 1 in the poptext and then the poptext is opened, then the selection mark will be on the first entry in the poptext and the focus frame will be on the entry of the .activeitem. Actually, both should be on .activeitem. This error has now been corrected.

» Gnats 9283: Multiple-line fields were not correctly displayed in tablefields. This error has been corrected.

» Gnats 9295: The poptext list appears behind the active window when it is opened. This error has been corrected.

10.3 Core

» Gnats 9190: New functionality of the :find-method.
Via the new comparison form *match_substr* it is now possible to search for partial strings.

Example

```
listbox Lb
{
  .content[1] "Hannover";
  .content[2] "Stuttgart";
}
```

The calling up of `Lb:find("tt", match_substr)` returns a *2* in this case.

» Gnats 9263: After the reading in binary files, which were created with an earlier version of A.04.02.h and which contained rules, errors occurred.

```
"Error in eval: internal value stack overflow"
```

The error has been corrected in the binary loading.

» Gnats 9264: Status return value of the function DM_SetVectorValue() has been corrected. It now returns a true value for .picture & .picture_hilite when everything is OK.

» Gnats 9266: The attribute .startsel is now taken into consideration when writing a dialog in an ASCII file.

» Gnats 9271: The double release (detaching) of a recently destroyed text objects through a second destruction in a recursive event loop, which is triggered by opening a popup menu, is now avoided.

» Gnats 9273: Binary writing/calling of the :index-method from associative arrays of the menubox object now functions again.

» Gnats 9275: Substring()-function now returns "" when the given string is an empty string.

» Gnats 9277: No more access to the content strings by DM_Set Content when DMF_OmitStrings is set.

» Gnats 9280: Additional functionality for the use of strings in "any" code pages:
There is a new code page CP-ucp (user codepage). When using this code page a conversion of characters/strings is carried out via the iconv-routines which exists under UNIX/LINUX. The user can freely choose a code page. Because the iconv-functionality is not available under Windows or VMS, this is not supported on these platforms (default conversion leading to ?-character).
The conversion to/from a code page can be set by the user. It is necessary that iconv supports the conversion from/to UTF-8. The default for the user code page is set to "8859-1". The application programmer can set the codepage, in which his own strings appear, via DMF_SetUserCodePage.

New Interface Function

```
DM_Boolean DML_default DM_EXPORT DM_ControlEx __4(
  (DM_ID, objectID),
  (DM_UInt, action),
  (DM_Pointer, data),
  (DM_Options, options))
```

This function has to be seen analog to DM_Control(), but expands this by an additional data parameter. It supports all of the other known actions within DM_Control plus the following as well:

DMF_SetUserCodePage

For DMF_SetUserCodePage a pointer on a string must be delivered in the data parameter, which contains code page identifier.

Example

```
/* set application code page to CP_ucp and use CP1250. */
:
DM_Control((DM_ID)0, DMF_SetApplCodePage, CP_ucp);
DM_ControlEx((DM_ID)0, "CP1250", DMF_SetUserCodePage, 0);
:
```

» Gnats 9293: Loading of mixed modules ASCII/Binary confused the internal import stack of the interface parsers. This has been corrected.

» Gnats 9299: The built-in function sprintf() interpreted the format string in the wrong code page. As a result, umlauts in format strings did not appear as such in the result string.

10.4 Editor

» Gnats 9265: Wrong character in the rule/code window of the editor was avoided through correct scheduling in the right code page (CP_utfwin for Windows NT).

» Gnats 9268: When entering a config-file the argv-vector is now enlarged correctly.

10.5 Network

» Gnats 9280: The repeated conversion of strings, which are returned as return values or output parameters when calling up server functions, has been corrected. The conversion takes place on the server side in the internal code page, before strings manage to get to the client side via the network.

11 Version A.04.03.b

11.1 Windows NT

- The select event in the splitbox is now generated after letting go of the mouse button. For this to take place, the pointer must be inside the splitbox at the time of releasing the mouse button.
- Gnats 9256: When using format resources on tablefields or poptexts, and when AppFormatCodepage != CP_internal was used, characters were not correctly converted. This was the case, for example, with umlauts. This problem has now been taken care of.
- Gnats 9248: By objects with graphics (image, menuitem, notepage,...), the graphics and some texts were not displayed correctly. This has now been corrected.
- Gnats 9215: An "Access Violation" in an ISA Dialog Manager application can occur when C-Runtime is not equivalent to Service Pack 4 or higher. It was noticed that "Access Violation" in the Microsoft debugger occurred, but not when the application was started outside of the debugger. This problem is discussed in the Microsoft Knowledge Base Article Q225099. The "Access Violation" takes place when lots of small storage areas exist (> 16 MB), and when they are newly allocated. This problem can only occur in the ISA Dialog Manager when dealing with large dialogs. The problem occurred in other applications when it was linked with an older C-Runtime library.
- Gnats 9166: On a poptext where *.style = edittext*, no *modified* event was sent after something was entered into content and directly after the `Return` key was pressed. As a result of this the content was also lost. This has now been corrected.
- Gnats 9208: On a poptext where style = poptext and a popup-menu with accelerators, the accelerators were not triggered. Now the accelerators function in this configuration as well.
- Gnats 9243 and Gnats 9242: The setting of .activeitem on a treeview from the Rule Language does not function. For the same reason the option opt_rightklick_selects also does not function.

11.2 Motif

- Gnats 9237: Font assignments to tablefield objects did not function. This error has now been corrected.

11.3 Core

- Gnats 9249 and Gnats 9250: During binary writing, exporting or similar actions, a storage access error occurred when .format and or .formatfunc-attribute was used on an object. This problem has now been corrected.
- Loading of binary modules and dialogs, which were written with older versions of IDM (older than A.04.02.h), resulted in Assfail. This problem has been corrected.

- Error message *"[W: formatter couldn't convert display string from internal codepage into codepage 22]"* appeared when a format function was used without a format string
- Gnats 9241: In the makefile from the srclib-index, which can be used to build your own startup.obj on the Windows platform, the ustartup.obj is now built correctly.
- Gnats 9229: The attribute *.options[]* of the edittext was not written out in the binary writing. The attribute received the default value, i.e *.options[opt_rtf] = false*, in the binary dialog. This error has now been corrected.
- Gnats 9200: When destroying models, which have hierarchical children and inherited instances, the instances are now correctly taken into consideration in the reference counter. This is important in order to achieve a complete replacement of IDs and to avoid similar errors.
- Gnats 9244: The storage access error that occurred when calling up the built-in functions apply-format() has been corrected.
- Gnats 9168: The option DMF_DirectCall when calling up DM_LoadDialog() is not supported (error in the documentation). The calling up of :init-method cannot be prevented by this. This call-up happens later in DM_StartDialog().
- Gnats 9011: Initial choice for the filerequester can be preset.
  If the new boolean attribute .startsel (default is false) is set to true on the filerequester object, then the data/directory-choice will receive the same value as the .value attribute.

  Restrictions
  - Only one choice is possible (.value must be scalar).
  - Under Windows NT pre-settings are only possible by loading/saving data, but not for directories.
  - The pre-set value in .value attribute must have the complete path and should also correspond to the given directory (.directory attribute).
  - The initial choice appears in the entry field. A selection in the selection list does not occur. This has to do with the system functionality of the selection box under Windows/Motif.
- Gnats 9164: The DM_GetVectorValue call to the attribute .colvisible or .colsizeable no longer checks the first/last indices with .rowcount, but rather correctly with .colcount.

11.4 Editor

- The moving of children within a splitbox with the right mouse button now functions correctly. Earlier, an incorrect offset was calculated when this action was carried out. By double clicking the splitbox, the object/attribute window is now opened, as one is used to with other objects.

11.5 COBOL Interface

- Gnats 9238: When starting an IDM application, which is linked to a COBOL option (**ufcob.obj**), it was internally assumed that the actual platform was not Unicode-capable. Under

WindowsNT/2000 and higher versions, the drawing near of idmuni.dll is instigated. With the new **ufcob.obj** Unicode-capable platforms are now recognized.

» Gnats 9211: Storage leak in **DMcob_SetVectorValue** has been corrected. Earlier set strings are now released and can be written over.

12 Version A.04.03.a

12.1 Motif

- Gnats 8774: When an edittext is visible and it is given a new content, then start/endsel will be set to 0. Now startsel/endsel is no longer set to 0, but rather remains this way.
- Gnats 9118: If an item in a poptext was previously selected, then a list of another poptext could only be opened with a double click of the mouse. Now, the opening of a poptext is no longer a problem.
- Gnats 9129: When the .format attribute of a poptext is set then the pointer to the right disappears. This is independent of the poptext style. This error has now been corrected.
- Gnats 9185: Poptext does not allow itself to be set to sensitive if the poptext was originally created as insensitive. This error how been corrected.
- Gnats 9183 (A0402h,A0310q): No Assfail due to lock-overflow with sensitive toggle of visible objects with popup menus as children.

12.2 Microsoft Windows-NT

- The additions for the compilation with CYGWIN according to IDMuser.h have been taken on (no guarantee).
- The options for the packing of structures have been changed. Now the system default is used. This change requires that all C/C++ sources, which use Dialog Manager Include (i.e. IDMuser.h), be **newly compiled** In the makefile file the option "-Zp1" can be removed if it is not needed for other reasons.
- The tablefield now supports the xmargin and ymargin attributes. These attributes can only be used for columns and rows (.xmargin[tablefield_column]).
- Gnats 8747: Under Win32 it is now possible with C to call up the number of free WSI-Ids. In order to do this the attribute maxsize needs to be called up on the setup object via DM_GetToolkitdata.

  Example

  ```
  DM_GetToolkitData(idSetupObject, AT_maxsize);
  ```

  Note

  The number of available WSI-Ids does not have anything to do with the number of objects that can still be made visible!
- The notebook tabs now show the tool helps of each notepage.
- Gnats 9176 (A0401j,A0402h) – supplement: A crash of the system is possible when the linked OLE control has also closed down without reason (crashed!)
- Gnats 9176 (A0401j,A0402h): There were OLE controls that claimed the keyboard entry as soon as they were made active. This error appeared for the first time after the correction of Gnats 9064.

Now, an active OLE control will be deactivated as soon as another (Dialog Manager) object is clicked or when another object receives the focus. As a result of this, the Dialog Manager takes over the keyboard entry again.

» Gnats 8979 (A0402i): With modal windows (dialogbox type) all other windows will become insensitive. Up to now the out-docked toolbars that belonged to the insensitive windows could still be worked with. Now these will be set to insensitive.

Note
When a window is modal this does not mean that it is positioned in front of non-modal windows in the Z-order. Windows has 3 guidelines set for the Z-Order: windows with *topmost*-property are positioned in front of all windows without this property (application-wide). Windows having an owner relationship: the owner is always positioned behind the “owned” window. Windows of an active application are positioned in front of windows with inactive applications. One exception to this rule is the top-most window. This allows the user to set an insensitive window in front of a modal, sensitive window. In reality this does not happen very often because an insensitive window cannot be clicked to the front, but can only be brought forward through other programs.

» Gnats 6748: Versioning of the dynamic libraries

```
idm.dll
idmrt.dll
idmndx.dll
idmnrt.dll
idmnet.dll
idmuni.dll
idmjava.dll
idmjavart.dll
idmjavandx.dll
idmjavanrt.dll
```

Tip
The version information can be shown under windows with the explorer by clicking the right mouse button on the file in the sub-menu *“Properties”*.

» Gnats 9087 (A0402e3): When images are displayed for a longer period of time errors can arise. This is because the GDI-handle was not released and at some point in time the system ran out of handles. The handle-leaks have now been removed.

12.3 Java-Window-Interface

» Gnats 9170 (A0402h): It was not possible to set colors in the Java interface via DM_ Object.SetValue. Each time an “non-convertible type” error occurred.

» Gnats 9169 (A0402h): The function with anyvalue parameter was not called up.

» When an empty status report was sent, the system crashed. This happened for example when a filereq dialog was supposed to be opened.

» Accelerators are now supported.

Take Note

There is no support for accelerators with letters. Accelerators with function buttons and buttons with special functions are supported in any case. Buttons with special functions are displayed on the virtual buttons that are defined under the Java class java.awt.event.KeyEvent. For example, *accelerator A "ADD"* corresponds to the virtual button *java.awt.event.KeyEvent.VK_ADD*.
The following should be taken into consideration with respect to how the buttons in Java (Swing) are worked on:

1. It cannot be avoided that a “key” event is created for an accelerator when this accelerator is assigned to an object. For example: when in the following dialog

```
dialog D
accelerator A F2;
window W
{
  pushbutton P
  {
    .accelerator F2;
  }
}
```

   The function key `F2` is pressed, the following event is triggered:
   P key A
   P select
2. System accelerators that are directly assigned to Java (Swing) objects have priority when the object in question has the focus. For example, when the <BackSpace> button is being processed from an edittext; when this has the focus. In this case an accelerator on <BackSpace> will not be triggered (this is not valid for the “key” event that is triggered).

» Gnats 9007: Support of +<option> with the Java-WSI-options.

» The timer object is now supported. For its use refer to timer object. In this stretch the option “Encryption in Java-wsi“ was also changed. Now the network protocol between Java-IDM-Server and Java-Applet/Application (idm.jar) no longer compatible to older versions. The application function **DM_ExistsMessageProc** received an additional parameter:

```
int DM_ExistsMessageProc (void *connptr, long timeout, char *message)
```

The parameter *timeout* is new. This parameter shows during tests (in milliseconds) how long file is available and how long they should be maintained. The value *0* for *timeout* means test without maintenance (old behavior). A value not equal to *0* means, when data are available, immediately return *0*, otherwise wait at least the maximal timeout shown in milliseconds. If still no data is available, then return *1*.

» The timeout attributes on the dialog object are now supported.

» Corrected (0402d): Earlier only one DM_VectorValue could be carried over in a block. Now it is possible to do more than just one.

» A comment to Unicode: No Unicode characters are shown in the title lines of the object windows and messageboxes. AWT classes in Java cause this error (see Java Bug Parade Bug ID: 4414631).

In the title line of the object windows and the messageboxes only letters, which belong to the system language, are shown. For example, on a German system umlauts are shown, but no Greek characters. This problem arises because the highest graphical objects are based on window systems. These objects are handled differently from all other objects. This is why the errors do not appear in the children windows or in other objects.

12.4 Kernel

- Two new objects, transformer and mapping, have been added. These objects allow for the transferring of XML file to IDM and vice versa.
- XML Interface
- Gnats 8977 (A0402d,A0401h) – Read performance from binary file has been improved.
- Gnats 9117: Name of Tracefile/Logfile/Errwinfile through the attribute .tracefile, .logfile, .errfile on the setup object can be called up.
- Gnats 8470: Escape sequence that is placed in brackets when an error takes place in the rule code, is now configurable through the option IDMhilite="<prefix>[...<suffix>]".
- Gnats 8977 (A0402d,A0401h): Read performance from binary file had been improved.
- **Note**
  The compatibility to the options Dia, Draw, DDE, Wire, Commarea in the A0310 version is temporarily not given due to the transformation of the UNICODE. New libraries for this option will be delivered with the next patch for the version A0310.
- Conversion of the handlings of format functions:
  The structures DM_FmtContent, DM_FmtDisplay contain strings in the AppFormatCodePage, which can be set with DM_Control. Strings in DM_FmtRequest-Strukture are in the same code page when the call-up is initiated in IDM. This is additionally noted in the code page field in the respective sub-structures.

```
struct {  /* FMTK_parseformat */
  DM_UInt1 codepage;    /* Codepage of format string */
  DM_String formatstr;  /* String describing the format */
} parseformat;

struct {  /* FMTK_setcontent */
  DM_UInt1 codepage;  /* Codepage of content string */
  DM_String string;   /* String to use as new content */
} setcontent;

struct {  /* FMTK_modify */
  DM_UInt1 codepage;   /* Codepage of modify string */
  DM_String string;    /* String to be inserted at pos */
  DM_UInt4 strlength;  /* length of insertion string */
  DM_UInt4 dpycurpos;  /* Cursor in display string */
  DM_UInt4 dpyselpos;  /* Start of selection in display string */
```

```
} modify;
```

Otherwise, a different code page can be transferred to the IDM here. An additional task FMTK_getdpymaxchars has been added. Here, the format function is asked about the maximal number of characters in the display string. The return value must be returned in the DM_FmtRequest-Structure within the getdpymaxchars area.

```
struct {  /* FMTK_getdpymaxchars */
  DM_UInt4 maxchars;  /* Maxchars */
} getdpymaxchars;
```

- Gnats 9172: Release of attribute annotations has been corrected.
- Gnats 9140 (A0402h): In the runtime draft of the IDM the available interface file is now correctly passed.
- Diverse performance and error corrections from the A0402e3 version have also flown in; this includes Gnats such as Gnats 9087, 9172, 1940 as well as:
  - Interface data are now once again written and read in their original format (without export numbers). Format "Version2" can still be read. A new generation of interface data is not necessary.
  - Transformation from YInitCA analog to A0310, so that an application, which uses the Commarea option, can also be built and carried out with the DLL variables of the IDM library.
  - The attribute .repos_id is no longer supported!
  - Storage leaks with respect to data/path names by module/import has been corrected.
  - Internal Object-ID-Management has been slimmed and its memory needs reduced.
  - Object replacing (i.e. after destruction of an object) and clean up of object IDs by user-defined ass. arrays (default value, index, de-referencing) has been corrected.
  - In the case that a module is unloaded, or that objects are exported which are still actively used by other objects (example: a format resource is linked to a visible edittext and therefore contains status information about the current entry progress (conPriv)), then the unloading of the module will be refused. Through the implicit re-loading, the module would once again be immediately reloaded.
- Unicode: In addition to startup.obj, the data ustartup.obj is also delivered, which the application code page sets to utf8. Because of this the simple linking or building of unicode-capable applications is possible => DM_BootStrap etc. from startup.c contains unicode strings for correct processing.
  Application programmers can receive the arguments in AppMain() in a different code page by transforming the application code page and afterward DM_GetArgv(). In startup.c the code page for the application can be defined through Define XXX_APPLCODEPAGE.
- Unicode: When tracing function call-ups (FC/FR-Marks) strings are no longer given out in the application code page, but rather according to the distribution code page.
- Unicode: As a rule the string parameters of all DM_Functions are transferred in the application code page. However, there are a few exceptions:

```
DM_BindCallBacks  (DM_FuncMap far *funcMap,
                   DM_UInt funcCount,
```

```
                    DM_ID dialogID,
                    DM_Options options);
```

Here the funcMap in CP_ ascii code page is expected.

```
DM_BindFunctions  (DM_FuncMap far *funcMap,
                    DM_UInt funcCount,
                    DM_ID objID,
                    DM_ID rehashID,
                    DM_Options options);
```

Here the funcMap in CP_ ascii code page is expected.

- Unicode: The function DM_ErrMsgText() receives as a return value a char-Pointer (DM_String). Now, if CP_utf16, CP_utf16l or CP_utf16b is used an application code page, the return string must be converted. As a result of this, the string in this application must be un-copied in case someone wants to use it at a later date. Otherwise the storage space will be released the next time that DM_ -call-up is used.
- A note to the customer: those users who used their own Makefile, whereby when building an IDM application oleaut32.lib and advapi32.lib were not linked to it, this must now take place. Errors such as unresolved externals:

```
dm.lib(variant.obj) : error LNK2001:
unresolved external symbol __imp__VariantInit@4
dm.lib(xml.obj) : error LNK2001:
unresolved external symbol __imp__VariantClear@4
```

  etc. are otherwise to be expected.
- Unicode: When DM_QueueExtEvent and DM_SendEvent are used, the following should be considered:
  When using these functions it is not certain that the call-ups are correctly recorded in the trace file. If strings are used in the parameters, then it can happen that they are falsely represented in the trace file. The string conversion is done at a later time, otherwise one would not be “thread-sure”.

12.5 Network

- Unicode: The transferring of strings takes place in the internal code page. Earlier, this was done in the application code page. Because of this, false symbols can occur when the application page and the display page use different versions of the Dialog Manager library.
- Unicode: The action DMF_SetCodePage of the DM_Control function now sets the code page on the page it was called up on. Now it is possible that different application pages work in various code pages. It is unfortunately no longer possible to set the code page for the application page on the display page.

12.6 Editor

- Support for layoutbox & splitbox in the editor. This means they have their own pages in the Obj/Attr. window.

- Relative (+/-) & absolute positioning over the extended main window is now possible.
- When creating objects underneath a layoutbox they receive posraster/sizeraster:=false. Children of layoutboxes can not be interactively moved.
- Unicode: The editor allows for setting the code page with the dialog/module in the storage page of the configuration window.
- Unicode: In the rule and code windows, the rule code is displayed in the code page that is governed by the window system. This means: ISO8859-1 under Motif (Unicode not active) and with Unicode-Active as Unicode-Symbol! Under Windows (NT/W95/W98) accordingly. This means that the escaping from symbols is dependent upon the system environment. It has also been tried to show the symbols instead of escaping from them. Symbols in substitute displays actually point to a font that is not available!
  The code page that is shown through the rule/code window can be set or varied through the editing page of the configuration window. The changes are first realized by a repeated display of a rule code that is not yet found in a buffer (this means one can/should first remove all the old buffers with Buffer/Kill and then select the rule again).

12.7 Unicode Support of the ISA Dialog Manager

12.7.1 Unicode

Unicode is a standard for the universal coding of symbols within texts. The goal is the timely replacement of ASCII and other coding tables (i.e.: ISO8859, CP850, CP1252 etc.) through a uniformed standard, which covers most all languages in the world. Das Unicode consortium (**https://home.unicode.org**) is dealing with the development, expansion and distribution. Unicode internationally standardized under ISO/IEC 10646 (also known as UCS – Universal Character Set).

As a result of this, the creation of applications and documents that function everywhere in the world can be done easily. Systems that support Unicode can be found unknowingly on most desks. These include operating systems such as Windows NT, Solaris 7, HP-UX 11.0 etc., as well as programming languages or formats like Java or XML.

12.7.2 Support

IDM allows for the creation of Unicode-capable applications beginning with the version A.04.03.a. In detail this means:

- IDM processes internally all strings as Unicode strings.
- IDM supports UTF-8 and UTF-16 (big endian and little endian) as coding of Unicode strings. UTF-16 is also known as UCS-2. Surrogate partners or combined symbols are not handled as one symbol.
- Data that is read in with IDM, for example: dialogs/modules, interface data or ini-data, can exist in various supported codes. This way it is easy to write a dialog in Unicode (i.e. with an UTF8-

capable editor dtpad in Solaris 7 or with the UTF16-capable notepad.exe in WindowsNT). The coding can then be set in the data so that a mixture of various codes is possible.

- When a support for the Unicode of an operating system already exists (Windows NT as well s Solaris 7, HP-UX 11.0), then the display/editing will also be in Unicode. Whether the Unicode support is “active” depends upon the chosen “locale” under UNIX.
- Log & trace data, as well as dialog-, module-, interface- and ini-data, can also be written out in a supported Unicode coding.
- For application functions and format functions the use of supported Unicode coding is also allowed.
- The editor, the debugger and the IDD support the Unicode with respect to display and entry.
- There are various entry methods for Motif applications for the setting of the .preedit-attribute on the window.
- The Motif variants of IDM now also support font sets, which are a compilation of various fonts, in order to allow for the correct display of Unicode symbols.

12.7.3 Compatibility

Fundamentally, there should be no necessity for changes to IDM applications, which use ISO8859-1 or CP1252 as a code page, so that they function as they did earlier. Although, one should take to heart a few special points, or at least understand the connection of how it was earlier and how it is now.

12.7.3.1 A0310-Options

The compatibility to the options Dia, Draw, DDE, Wire, Commarea in version A0310 is for the meantime due to the conversion of the UNICODE not available. They are still linkable and function to a certain extent, but the adaptation can only be carried out in the next delivery of the options in the next A0310 patch.

12.7.3.2 Data Input and Output

Normally, the output of IDM in ASCII (errors are additionally marked with control symbols to bring out the error), this means that non-ASCII symbols are “escaped”, or otherwise stated they are brought into an ASCII-conform display (see section 1.7.4).

Data, especially dialogs and modules, are also read into ASCII from the IDM. Nevertheless, not completely in ASCII; if that was the case this would mean that all symbols whose coding was greater than 0x7f, would be rejected. In actuality, the symbols are accepted as they are. This has two “positive” consequences:

- Up to now IDM has used the code page ISO8859. This means that the data symbols were quasi in ISO8859.

» This is not 100% true – the code page, in which the symbols are displayed under Windows (NT & 95/98), is normally for Europe CP1252 (WinLatin1). This is practically identical with ISO 8859-1 except for the C1 control symbol (code area 0xs80-0x9f). This has the positive effect that one can enter the symbol for the EURO € as code \200 (the same as (0x80) into a dialog.

Note
This symbol is not conform to ISO8859, because the symbol there is noted as 0xA4 in ISO8859-15 as Latin9. It was not correctly displayed under UNIX (by ISO8859-15 “locale”).

With the Unicode support of IDM dialogs, modules and ini data can be read into the ISO8859-1 code page under UNIX/VMS ; under Windows NT in ACP (Ansi code page of the application.). This should insure the running of earlier dialogs without having to make any changes to them. Unfortunately, there is no 100% guarantee for the standard exchange of ASCII data between UNIX and Windows (the same as earlier).

The effect on IDM applications are first seen when a data shows the symbol in the code area 0x80-0x9f, when it is written out under UNIX/Windows NT, or when a “code” is created or manipulated (i.e. over sprintf() or an application function / format function). If the symbol € has the code 0x20AC in the Unicode standard, then when it is written via the editor it would have the character sequence *\u20ac* as a result.

12.7.3.3 International Applications

ISO8859 or Country/Language-Related Symbol Tables

ISO 8859-* and Windows with CP1251, CP1252, etc., define many symbol tables which are to be used according to the various countries and languages in question. Similarities with these symbol tables are a cooperative constant code area (i.e. ASCII-symbols), and also an extended area with the specific symbols (i.e. exclusive for Greek, Latin and Cyrillic).

This means that an integration of symbol code and local font settings and language and country settings allows for the correct display of the symbols through the system.

UNIX

When one builds and internationalized application (i.e. by using ISO8859-5 for the support of Cyrillic) on UNIX, and one plans on using this in the future, then this is possible without having to convert anything. Nevertheless, the symbols are not necessarily equipped with the correct Unicode code because it is only possible to convert from ISO8859-1. The display functions without problem.

If one wants to make his application Unicode-capable, it is recommended that he change his data to UTF-8. If one has coded his data according to ISO8859-5, then the following way of changing data is recommended:

1. Via `idm –IDMcp_io=iso8859 my.dlg –writebin my.8859` change the form so that no “escaping” from ISO8859 symbols takes place.
2. Change the file on UNIX into UTF-8 from the desired country specific ISO code page, i.e. ISO 8859-15, via `iconv -f 8859-15 -t UTF-8 <my.8859 >my.utf8`.

3. Put *// UTF8* at the beginning of the file so that IDM can automatically identify the file as Unicode file.

Windows NT

The conversion of dialog data is also possible, as is under UNIX. The writing of a UTF-16 data can be done with **notepad**. For programs that used Ansi code page (ACP), this is not necessary.

Windows 95/98

Microsoft does not support Unicode in their Windows 95 and 98 operating systems. Only conversion routines are provided by the system. Therefore, IDM must convert strings for display and entry purposes accordingly. There is no special executable for the use under Windows 95. Nevertheless, it is **absolutely necessary** that the **idmuni.dll** library be installed parallel or in the search path of the DM application.

12.7.3.4 Binary Data

Principally, binary files that have been written before IDM version A.04.03a, receive the strings in ISO8859. These binary files can be read without problem. The symbols are interpreted as ISO8859-1 (Latin-1) and are converted internally in Unicode. According to the course of action in the previous section, there is no guarantee of conformity between different “sub-code” pages. Otherwise, the following is true beginning with the A.04.03 version: all strings in IDM binary files are coded as Unicode strings.

12.7.3.5 Format Functions

When format functions are implemented in self-made applications, then these must be re-written for the entire Unicode support. Due to compatibility reasons, strings are always transferred and expected in the code page of the window system (this means ISO8859 for UNIX, ACP for NT). A non-adapted format function has the consequence that non-supported symbols within the window system code page are passed as “?” In order to avoid this, the code page for format functions must be set via DM_Control(DMF_SetFormatCodePage,...). It is also important that the format functions are changed accordingly to UTF-8 or UTF-16.

In addition, the handling of format functions has been changed to a certain extent:

The structures DM_FmtContent, DM_FmtDisplay receive strings in the AppFormatCodePage, which can be set with DM_Control. Strings in DM_FmtRequest-Structure are in the same code page when the call is initialized in IDM. This is also marked in the code page field in the separate sub-structures.

```
struct {  /* FMTK_parseformat */
  DM_UInt1 codepage;    /* Codepage of format string */
  DM_String formatstr;  /* String describing the format */
} parseformat;

struct {  /* FMTK_setcontent */
  DM_UInt1 codepage;  /* Codepage of content string */
  DM_String string;   /* String to use as new content */
```

```
} setcontent;

struct {  /* FMTK_modify */
  DM_UInt1 codepage;   /* Codepage of modify string */
  DM_String string;    /* String to be inserted at pos */
  DM_UInt4 strlength;  /* length of insertion string */
  DM_UInt4 dpycurpos;  /* Cursor in display string */
  DM_UInt4 dpyselpos;  /* Start of selection in display string */
} modify;
```

Otherwise, it is possible to give other code pages over to IDM here. An additional task FMTK_getdpymaxchars has been added. Here, the maximal number of symbols within the display string in the format function is inquired. The return value must be returned into the DM_FmtRequest structure in the getdpymaxchars area.

```
struct {  /* FMTK_getdpymaxchars */
  DM_UInt4 maxchars;  /* Maxchars */
} getdpymaxchars;
```

12.7.3.6 DM-Functions

12.7.3.6.1 String Parameters

As a rule all DM_-functions transfer string parameters to the application code page. However, there are a few exceptions to this rule:

```
DM_BindCallBacks  (DM_FuncMap far *funcMap,
                   DM_UInt funcCount,
                   DM_ID dialogID,
                   DM_Options options);
```

Here, the funcMap in the CP_ascii code page is expected.

```
DM_BindFunctions  (DM_FuncMap far *funcMap,
                   DM_UInt funcCount,
                   DM_ID objID,
                   DM_ID rehashID,
                   DM_Options options);
```

Here, the funcMap in the CP_ascii code page is expected.

12.7.3.6.2 DM_ErrMsgText

The function DM_ErrMsgText() receives a char-pointer (DM_String) as a return value. If CP_utf16, CP_utf16l or CP_utf16b is used as an application page, the return string must be converted. As a result of this, this string in the application must be un-copied if it is to be used again at a later date. Otherwise, the storage space at the next DM call-up will be released.

12.7.3.6.3 DM_SendEvent

When using the functions DM_QueueExtEvent and DM_SendEvent it cannot be ensured that the call has been documented correctly in the tracefile. If parameters are used in the strings, it can happen that they are displayed incorrectly in the tracefile. The string conversion is done later. Otherwise you can't be sure that one is thread-sure.

12.7.3.7 Network (DDM)

The transferring of strings takes place in the internal code page. Earlier, the strings were transferred in the application code page. Due to this, it is possible that symbols are misrepresented, if the application page and the display page have used different versions of the Dialog Manager Library.

The action DMF_SetCodePage of the function DM_Control sets the code page on the page it was called up on. This way it is possible that different application pages work in different code pages. Unfortunately, it is no longer possible to set the code page for the application page on the display page.

12.7.3.8 Applications that use the System Functions

When using system functions (Win32, X, Motif, etc.), it is important to note that at the moment strings are expected/returned in a different coding. Especially with respect to programming in Windows, it is important to pay attention to the correct Ansi/Unicode-Use in order to guarantee the cooperation with IDM, which expects Unicode messages.

12.7.4 Unicode Symbols

The Unicode (Standard 3.0) defines approximately 50.000 symbols. In order to be able to use all these symbols in an IDM dialog, the following two possibilities are available:

- Writing dialog files via an Unicode-capable editor. When loading into IDM one must state the corresponding code page.
- Or/and – substitute display of symbols within "" strings through a number code in the form of *OOO* (octal) or \u*XXXX* (hexadecimal). Whereby *O* stands for an octal-figure in a range from 0-7 and *X* stands for a hexadecimal figure in the range from 0-9,A-F. The Euro symbol is written in the substitute display as *u20AC*.

Functions that manipulate strings or compare, arrange or transform separate symbols are implemented in IDM according to the following rules:

- Numerical symbols are figures ranging from 0 to 9
- Hexadecimal symbols include the numbers 0 to 9 and the letters a to f or A to F
- Alphabetical symbols include only the letters a to z or A to Z

» The conversion of capital and lower case writing is only carried out in areas with alphabetical symbols

» Punctuation marks include the following symbols: ! " # $ % & ' ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~

12.7.5 Display/Representation

A Unicode application is not the same as: “all symbols in the Unicode standard are displayed on the screen”. The way symbols are displayed on the screen depends on the support of the system, on the chosen ***font***, and on the installed fonts and codes.

See Also

Chapter “Comments UNIX”

12.7.6 Code Pages and Conversion

The following new code pages have been added to IDM:

Code page	Term in IDMuser.h	Options-term
UTF-8 Unicode, 8-bit coding with variable lengths, within 0..127 compatible with ASCII.	CP_utf8	utf8
UTF-16 UTF-16 BE UTF-16 LE Unicode, codes a symbol with a 16 bit code. Two variations: » BE (Big Endian – High,Low-Byte-Sequence) » LE (Little Endian, Low-High-Byte-Sequence). Under Windows NT UTF-16 corresponds to the LE variation, under all other systems (i.e. UNIX) displays are based on the BE variable.	CP_utf16 CP_utf16b CP_utf16l	utf16 utf16b utf16l
CP 1252 WinLatin1 code page from Microsoft Windows for the European language area.	CP_cp1252	cp1252
Active Ansi code page of the program. Conversion is only possible under MS Windows.	CP_acp	acp

During the conversion of strings into another code page, i.e. when calling up an application function, it can happen that the target code page cannot recognize one of the symbols. When this happens, a “?”

is given instead of the symbol which was not recognizable. During the output of strings, which contain Unicode symbols, in a file, i.e. a trace file or a dialog, a substitute display is chosen depending upon the desired target code page:

Target Code page	Substitute Display through Octal Figure \OOO in the Range Between 127...255	Substitute Display through Hexa-decimal Figure \uXXXX in the Range Between 256...65535
ASCII	x	x
ISO8859		x
UTF-8, UTF-16		
Other	x	x

12.7.7 New IDM Options

The IDM library has received the following new options. These apply to the way files and data are handled and the code page they belong to:

Option	Description
-IDMcp_input <codepage>	This option defines the code page with which dialogs, modules, interfaces and init files are interpreted from a Dialog Manager application. This only applies to the files that do not have an identification mark (see also chapter “Files”).
-IDMcp_output <codepage>	With this option once can determine the code page with which IDM gives out symbols in a file. In this case file means all dialogs, modules, interfaces, and init files as well as log or trace files and stdout and stdin (UNIX).
-IDMcp_io <codepage>	This is an abbreviation for the simultaneous setting of input and output code pages (-IDMcp_input and -IDMcp_output).
-IDMcp_appl <codepage>	The code page, in which the application functions expect or return strings, is defined via this option. Note! The easiest way is through the calling up of DM_Control within the application.
-IDMcp_format <codepage>	This option influences the code page, in which format functions that have been written in C, receive or return their strings. As is with –IDMcp_appl<codepage>, it is easier to accomplish this via DM_Control ().

The following identifiers are allowed for the specification of the code page in <codepage>:

ascii, iso8859, iso6937, utf8, utf16, utf16b, utf16l, no_conv, winansi, cp437, cp850, dec169, hp15, cp1252, acp.

New Action for DM_Control(): DMF_SetFormatCodePage

This defines the code page in the string to format functions (in C) that are transferred and also expected in return.

12.7.8 Files

If a module or dialog file exists in the Unicode, then IDM automatically recognizes this by the prefix within the first 8 bites of the file. An existing prefix has priority over the code page that was given via options.

Prefix	File Type
BOM (Byte Order Mark = 0xfeff) and reversed byte sequence of BOM is defined from Unicode standard.	
Byte sequence 0xfe 0xff	UTF-16 BE
Byte sequence 0xff 0xfe	UTF-16 LE
// UTF8 or //utf8	UTF-8
// 8859	ISO-8859-1
// 1252	CP1252

When the output code page is set with DMcp_io or DMcp_output on UTF-7, ISO-8859, CP1252 or UTF16, then the dialog/module files and the interface files automatically receive this prefix at the beginning of the file in order to insure for automatic recognition.

12.7.9 General Comments

In almost all window systems the window frames and the titles within these frames are presented from a sub-window system. The font used for the title (i.e. by window and messageboxes) and the display/code pages that are linked to it can only be set by the system/user and are not configurable through IDM. In this respect, effects that are characters, which are correctly represented as push-button text, are not visible in the title line. These are unfortunately dependent upon the system settings and are not releasable from IDM. Therefore it could be that e.g. text of a Pushbutton is not correct displayed in an title of a window.

12.7.10 Comments Editor

The IDM Editor, as well as the IDM Debugger, supports the input and display of Unicode symbols.

The editor allows for the setting of the code page, with which dialogs and modules are written out in, in the saving page of the configuration window.

In the rule, respectively code window, the rule code is displayed in the code page in which the window system masters. This means ISO8859-1 under Motif (Unicode not active) and when Unicode is active as a Unicode symbol! As is under accordingly under Windows (NT/W95/W98). This means that the escaping from symbols dependent upon the system environment. Naturally, it is attempted to show the symbol instead of trying to escape from it. Symbols within substitute displays clearly point to a font that is not available!

The code page, which is displayed in the rule/code window, can be set or changed through the editing page of the configuration window. These changes can first be seen by a renewed display of the rule/-code that is not yet found in a buffer (this means that one can/should first remove all of his old buffers via Buffer/Kill and then select the rule once again).

12.7.11 Comments UNIX

IDM supports Unicode on all platforms. However, only the following UNIX platforms are capable of “transferring” these. This is true because their Motif library supports UTF-8 strings:

- Solaris beginning with Version 7
- HP-UX beginning with Version 11.0 and the Extension Pack May 1999

In order for an IDM application under UNIX to actively use Unicode, the following conditions must be fulfilled:

- Motif library must be dynamically linked to it.
- Local must be set to Unicode local. This is recognized by Motif/X-library as well as from IDM. This happens very easily through:
  - Under SOLARIS i.e. with: `setenv LC_ALL en_US.UTF-8` in one csh.[1]
  - Under HP-UX i.e. with: `setenv LC_ALL de_DE.utf8` in one csh.

  The IDM library recognizes a set local as UTF-8 local, when LC_ALL (or when not set in LC_ CTYPE or when not set in LANG) of the partial string “UTF-8” or “utf8” appears in the environment variables. Only after this has happened will the coded exchange between the IDM strings and the Motif library in UTF-8 take place. Otherwise, strings are transformed according to ISO8859-1 and correspondingly invalid symbols are represented with a “?”.
- The dialog to be carried out should either use the default fonts of the system so that Motif itself looks for the necessary fonts, or an appropriate Unicode should be chosen. Normally, one font does not do the job; the choice happens through a set of fonts. Such font sets include, for example, the system fonts for CDE.

```
font Fn "-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-1,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-2,
```

[1] See also the documents “Unicode Support in the Solaris 7 Operation Environment” and “International Language Environments Guide” from SUN

```
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-4,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-5,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-6,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-7,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-8,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-9,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-iso8859-15,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-big5-1,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-jisx0208.1983-0,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-jisx0201.1976-0,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-ksc5601.1992-3,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-gb2312.1980-0,
-dt-interface system-medium-r-normal-s*utf*-*-*-*-*-*-*-tis620.2533-0";
```

This can also be abbreviated as (important is the comma):

```
font Fn "-dt-interface system-medium-r-normal-s*utf*,";
```

This font set is available in various sizes: xxs, xs, s, m, l, xl, xxl (Extra Extra Small, Extra Small, Small, Medium, Large, Extra Large, Extra Extra Large). These symbols are to be set in front of *utf instead of s.

Things to take note of:

- The X-display should also possess the same chosen fonts!
- In principle, the number of displayed symbols varies from system to system. In other words this is dependent upon the installation status and the configuration.
- For the purpose of input and editing in different locals it is important to check the manuals/infos for internationalizing the system.
- Accelerators and mnemonics are dependent upon the connected keyboard. For example, an accelerator can be defined to `Ctrl` + `Ω`, but the problem is that a German keyboard is not able to produce a `Ω` key.

12.7.12 Unicode in the Application

It is left up to the application programmer to decide if he wants to leave his application the way it is or change his interface functions to IDM in Unicode.

Via the function DM_Control(DMF_SetCodePage,...) or DM_Control(DMF_SetFormatCodePage,...) the code page, in which strings are exchanged from/to application functions and format functions, can be set.

The application programmer always receives the string within a DM_String-File types, regardless of which code page / coding he has chosen.

In order to use Unicode strings in the application, it is recommended to use one of the following code pages:

- utf8
- utf16b or utf16l

The following file types for the storage of symbols are usually available on C-page:

- char
- wchar_t
- TCHAR (only MS Windows)

Tips

1. The use of wchar_t should only be considered in extreme cases because wchar_t does not have a defined bit size. The behavior of the w-function is system dependent and should be considered in local context. A cast of DM_String that is coded in code page UTF-16, upon which wchar_t is not 100% sure. This is true because UTF-16 fix has 16 bit and a lot of wchar_t-implementations work with 32 bit. The most compatible way is most likely through the use of UTF-8.
2. Under MS-Windows it is recommended to use UTF-16LE as the “native” coding of symbols, which is supported by the data type TCHAR from the compiler.
3. IDM does not offer string manipulation functions on the C-page. However, there is support from the system manufacturer, from third-party companies, or “shareware” software that can be down-loaded at no cost (i.e. ICU from IBM).

Other ways of switching the application to Unicode-capable:

In addition to startup.obj, the file ustartup.obj is also sent with it. This sets the application code page to utf8. This then allows for a simple building/linking of a Unicode-capable application => DM_BootStrap etc. from startup.c receives Unicode strings for correct processing.

Application programmers themselves are able to receive the arguments in a different code page in AppMain() by converting the application code page and then DM_GetArgv(). In startup.c it is possible to pre-set the code page for the application through Define XXX_APPLCODEPAGE.

12.7.13 Tracing

When tracing function call-ups (FC/FR marks, strings are no longer issued in the application page, but rather in the output code page.

12.7.14 New Attribute

Attribute .preedit on the Window. Support only for Motif with active Unicode support. Functionality is dependent upon the operating system or Motif version.

This serves as a way to regulate the display/choice of the input modes in edittexts, which lie within a window.

Attribute Value	Description
preedit_root (default)	Input mode display, only when one finds oneself within the edit-text.

Attribute Value	Description
preedit_overthespot	Input mode display is always available on the lower edge of the window (Note: the window will increase in size).
preedit_offthespot	No support through IDM
preedit_onthespot	No support through IDM

12.8 XML Support

IDM XML support allows for the processing of XML Documents in the Rule Language. The main components include the new object class documents and doccursor, which are described in the following section.

12.8.1 XML Document (document)

Keyword:

document

Definition

```
{export} {model} document {<Identifier>}
{
  [<Attributedefinition>]
  [<Methoddefinition>]
}
```

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

XML and DOM are standards of the World Wide Web Consortium (W3C). The specifications can be seen in detail on the Internet page of W3C (**https://www.w3.org**).

Event	–
Children	doccursor
	document
	record
	transformer
Parent	application
	canvas

checkbox

dialog

doccursor

document

edittext

groupbox

image

import

layoutbox

listbox

mapping

menubox

menuitem

menusep

messagebox

modul

notebook

notepage

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

statusbar

tablefield

timer

toolbar

transformer

treeview

window

Menu –

Attributes

Attribute	RSD	PSD	Properties	Short Description
doccursor[integer]	identifier	object	S,G/-/-	XML Cursor of the XML Documents
document[integer]	identifier	object	S,G/-/-	XML Documents of XML Documents
firstrecord	identifier	object	S,G/-/-	First record of XML Documents
function[integer]	identifier	object	S,G/-/-	Functions of XML Documents
idispatch D	pointer	pointer	S,G/-/-	Microsoft Windows Interface Pointer
ixmldomdocument2 D	pointer	pointer	S,G/-/-	Microsoft Windows Interface Pointer
label	string	string	S,G/D/C	Name of XML Documents
lastrecord	identifier	object	S,G/-/-	Last record of XML Documents
model	identifier	object	S,G/D/C	Object model /master copy
namedrule[integer]	identifier	object	S,G/-/-	Named rules of XML Documents
parent	identifier	object	S,G/-/-	Parent of XML Documents
record[integer]	identifier	object	S,G/-/-	Records of XML Documents
recordcount	integer	integer	-,G/-/-	Number of records in XML Documents
rule[integer]	identifier	object	S,G/-/-	Rules of XML Documents

Attribute	RSD	PSD	Properties	Short Description
scope	integer	scope	-,G/-/-	Type of XML Documents (Default, Model or Instance)
transformer[integer]	identifier	object	S,G/-/-	Transformer of XML Documents
userdata	anyvalue	anyvalue	S,G/D/C	Application data of objects
xml	string	string	S,G/D/C	String representation in XML Documents

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

Methods

Methods	Return Value	Arguments	Short Description
load	boolean	string URL	Loads XML Documents from records or URL
save	boolean	string URL	Saves XML Documents from records or URL
transform	boolean	object Schema {anyvalue Tar-get} {enum Type}	Transforms the XML Document with given scheme.
validate	integer	{string ErrorMsg}	Validates the XML Document

12.8.1.1 Note for Microsoft Windows

Here, the Microsoft XML Core Services (MSXML) Version 3.0 or higher is needed.

12.8.1.2 Note for UNIX and VMS

The XML Document object (***document***) is not available.

12.8.1.3 Object-Specific Attributes

doccursor

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

idispatch

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

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

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

ixmldomdocument2

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

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

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

xml

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

12.8.1.4 Object-Specific Methods

load

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

save

Saves XML Documents in records or URL.

transform

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

As an alternative, the target of the transformation can be a text or record. If the result of the transformation is not a legal XML format, then this must be directly converted into a text or a record. Otherwise, the result cannot be assigned to an XML Document. This is true, for example, for HTML conversions.

validate

This checks if the XML Document is equivalent to the document type given in the XML Document. If it does not contain a document type, an error will be registered.

12.8.2 XML Cursor (doccursor)

Keyword:

doccursor

Definition

```
{export} {model} doccursor {<Identifier>}
{
  [<Attributedefinition>]
  [<Methoddefinition>]
}
```

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

Via other methods it is possible to set the reference to a different node of the DOM tree. Understandably formulated means that the XML Cursor can be moved via methods in the DOM tree. In any case, this will remain a child of the XML Document.

The XML Cursor attributes contain, in addition to the "normal" Dialog Manager attributes, other attributes which make it possible to access properties such as "name", "value" or other attributes of the DOM nodes. Because these are runtime attributes, they cannot be passed down. In addition, many of these attributes can only be read because the corresponding properties of the DOM nodes cannot be changed. The XML Cursor is invalid for the time being, but as soon as it is accessed for the first time it will be automatically positioned to the root of the DOM tree. Note, this also happens when the XML Cursor is made invalid through a certain action. The attribute *.mapped* lets the user know if the XML Cursor is valid or not.

XML and DOM are standards of the World Wide Web Consortium (W3C). The specifications can be seen in detail on the W3C Website (**https://www.w3.org**).

Event	–
Children	document
	record
	transformer
Parent	document
Menu	–

Attributes

Attribute	RSD	PSD	Properties	Short Description
attribute[integer] D	string	string	-,G/-/-	Names of the DOM node attributes
attribute[string] D	string	string	S,G/-/-	Attributes of the DOM nodes
data D	string	string	S,G/-/-	Data of DOM nodes
document[integer]	identifier	object	S,G/-/-	XML Documents of the XML Cursors
idispatch D	pointer	pointer	-,G/-/-	Microsoft Windows Interface Pointer
ixmldomnode D	pointer	pointer	-,G/-/-	Microsoft Windows Interface Pointer
ixmldomnodelist D	pointer	pointer	-,G/-/-	Microsoft Windows Interface Pointer
firstrecord	identifier	object	S,G/-/-	First record of XML Cursor
function[integer]	identifier	object	S,G/-/-	XML Cursor functions
label	string	string	S,G/D/C	Name of the XML Cursor
lastrecord	identifier	object	S,G/-/-	Last record of XML Cursor
mapped D	boolean	boolean	-,G/-/-	Shows XML Cursor on a DOM node?
model	identifier	object	S,G/D/C	Model of the XML Cursor
name D	string	string	-,G/-/-	Name of the DOM nodes

Attribute	RSD	PSD	Properties	Short Description
namedrule[integer]	identifier	object	S,G/-/-	Names rules of the XML Cursor
nodetype D	enum	enum	-,G/-/-	Type of DOM node
parent	identifier	object	S,G/-/-	Parent of the XML Cursor
path D	string	string	-,G/-/-	Position of the XML Cursor in the DOM tree
publicid D	string	string	-,G/-/-	Official identifier of the DOM nodes
record[integer]	identifier	object	S,G/-/-	Records of XML Cursor
recordcount	integer	integer	-,G/-/-	Number of records of the XML Cursor
rule[integer]	identifier	object	S,G/-/-	Rules of the XML Cursor
scope	integer	scope	-,G/-/-	Type of XML Cursor (default, model or instance)
specified D	boolean	boolean	-,G/-/-	Is the attribute of the DOM node explicitly given?
systemid D	string	string	-,G/-/-	System identification of the DOM nodes
target D	string	string	-,G/-/-	Instruction name of the DOM nodes
text D	string	string	S,G/-/-	Vale of all sub-node of the DOM nodes
transformer[integer]	identifier	object	S,G/-/-	Transformer of XML Cursor
userdata	anyvalue	anyvalue	S,G/D/C	Application data of the XML Cursor
value D	string	string	S,G/-/-	Value of DOM nodes
xml D	string	string	-,G/-/-	String representation of DOM nodes and all sub-nodes

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

Methods

Methods	Return Value	Arguments	Short Description
add	boolean	enum node type {string Name} {string Value} {boolean Select}	Adds a child node to the DOM nodes.
add	boolean	pointer IXMLDomNode {boolean Select}	Adds the COM object as a child node to the DOM nodes under Microsoft Windows
delete	boolean		Deletes the DOM node and all its child nodes
match	boolean	string pattern	Tests if DOM nodes correspond to the model
reparent	boolean	{object Newparent} {integer Index}	Transfers the DOM nodes with all of its child nodes
select	boolean	enum direction	Moves the XML Cursor in the given direction
select	boolean	string pattern	Moves the XML Cursor to the first DOM node which corresponds to the model
transform	boolean	object scheme {anyvalue Target} {enum Type}	Transforms the DOM nodes with given scheme

12.8.2.1 Note for Microsoft Windows

The Microsoft XML Core Services (MSXML) Version 3.0 or higher is needed.

12.8.2.2 Note for UNIX and VMS

The XML Cursor object (***doccursor***) is not available.

12.8.2.3 Object-Specific Attributes

attribute[integer]
attribute[string]

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

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

If the index is a string, then the index will display the name of the attribute and the value of the attribute will be returned. The assignment to an attribute that is indexed with a string will apply a corresponding attribute to the DOM nodes. The assignment of an empty string deletes the corresponding attribute of the DOM nodes.

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

data

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

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

idispatch

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

In the Rule Language the attribute can only be assigned to the same attribute of a different 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.

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

ixmldomnode

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

In the Rule Language the attribute can only be assigned to the same attribute of a different 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.

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

ixmldomnodelist

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

accessed through the interface pointer.

In the Rule Language the attribute can only be assigned to the same attribute of a different 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.

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

mapped

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

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

name

Refers to the name or tag of the DOM node.

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

nodetype

This serves for calling up the type of DOM node.

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

path

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

If the value of the path attributes is noted somewhere else (i.e. in the userdata attribute), then it is important to know that these noted values will not be adjusted when the structure of the DOM tree changes. When the select method with the noted value is called up after this, then the XML Cursor will end up pointing to an incorrect DOM node.

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

publicid

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

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

specified

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

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

systemid

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

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

target

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

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

text

Is the value of all sub-nodes within the DOM nodes. A string is delivered which represents the text of all sub-nodes. This attribute is mainly helpful when one needs the text of an XML element. This way one can avoid first having to go to the child nodes that contain the actual text.

Please note, the setting of this attribute will automatically delete all child nodes and a new text node will be inserted.

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

value

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

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

xml

String representation of DOM nodes and all of its sub-nodes.

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

12.8.2.4 Object-Specific Methods

add

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

delete

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

When the value of the path attributes is noted elsewhere (i.e. in userdata attribute), then it is important to know that these noted values will not be adjusted when the structure of the DOM tree changes. When the select method with the noted value is called up after this, then the XML Cursor will end up pointing to an incorrect DOM node.

match

Test if the DOM node is the same as the given model (see also "Model for the Methods match and select").

reparent

Re-hangs the DOM nodes with all of its child nodes.

When the value of the path attributes is noted elsewhere (i.e. in userdata attribute), then it is important to know that these noted values will not be adjusted when the structure of the DOM tree changes. When the select method with the noted value is called up after this, then the XML Cursor will end up pointing to an incorrect DOM node.

select

Moves the XML Cursor in the given direction or moves the XML Cursor to the first DOM node which corresponds to the given model (see "Model for the Methods match and select").

transform

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

As an alternative, the target can also be a text or a file. If the transformation does not have a legal XML format, then this must be directly transformed into a text or a file. This is true because otherwise this cannot be assigned to an XML Document. This is the case when documents are transformed to HTML.

12.8.2.4.1 Model for the Methods match and select

A model is very similar to a Dialog Manager identifier. The model builds a path of element names. The path begins at the root of the tree. Each hierarchical level is compared to the corresponding part of the path. Here it is possible to define certain characteristics of the father such as the possession of a certain attribute or the position within the children.

In principal, all symbols except **.**, **[**, **"**, **]**, **<**, **>**, **=**, ****, tab, space and page breaks are allowed to be used in the model. If one of the above mentioned symbols, not including page break, is used, then **** must be set in front of it. Between two parentheses (**"**), within a string, the following symbols are also allowed: **.**, **[**, **]**, **<**, **>**, **=**, tab and page break. Please note, in the dialog script the symbol **** within a string is

already an escape symbol. Due to this it is important to use \\ in a dialog script whenever a \ is needed. The symbol \" must also be given in a dialog script whenever the symbol " is needed.

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

<Name>

Is compared to the name attribute of the XML Cursor. The XML Cursor must possess the node type *nodetype_element*. As an alternative, the symbol * can also be given. In this case the name attribute will not be taken into consideration.

The name of an XML element begins with a letter or an underline. It can also contain letters, numbers, hyphens, underlines and periods. The exact definition of an XML element name can be found in the XML specifications (**https://www.w3.org**).

<Attr>

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

The name of an XML attribute begins with a letter or an underline. It can also contain letters, numbers, hyphens, underlines and periods. The exact definition of an XML attribute name can be found in the XML specifications (**https://www.w3.org**).

<Op>

Is a comparing-operator which compares the attribute given in <Attr> with <Value>. It can be tested for equal **=** and unequal **<>**.

<Value>

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

<Idx>

The DOM node must be positioned within the child nodes of the father nodes. The first child has position 1.

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

Specialties

.

When the model begins with a dot, then this is relative to the actual DOM node. This means that the path begins by the actual DOM node.

..

When two dots are together, then numerous hierarchy levels will be passed by.

[<Idx>]

An index without any further information selects the DOM node at this position. In this case, the type of DOM node remains clear. Each DOM node can be clearly referenced through the path attribute {**[**<Idx>**]**[**.[**<Idx>**]**]}.

Additions Under Microsoft Windows

XPath can also be used as model syntax. Here, the model must begin with either / or ./. The use of XPath models is/will not be supported for each platform which makes it non-portable.

12.8.3 Attributes

12.8.3.1 attribute[I]

Identifier:

.attribute[I]

Classification: Object-Specific Attribute

Definition

Argument type	string, object
C definition:	AT_attributes[I]
C data type:	DT_string
COBOL definition:	AT-attributes[I]
COBOL data type:	DT-string
Access:	set (only with string index, cannot be passed down), get

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. 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 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.

Example

The attribute order number of the XML element, upon which the XML Cursor points, will be set with the following instruction to the value 0815:

```
this.attribute["Order Number"] := "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
```

12.8.3.2 data

Identifier:

.data

Classification: Object-Specific Attribute

Definition

Argument type	string, object
C definition:	AT_data
C data type:	DT_string
COBOL definition:	AT-data
COBOL data type:	DT-string
Access:	set (cannot be passed down), get

This serves for setting and calling up data of a DOM node. The attribute is only available when the node type is either *nodetype_cdata_section* or *nodetype_processing_instruction*.

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.

12.8.3.3 doccursor[I]

Identifier:

.doccursor[I]

Classification: Object-Specific Attribute

Definition

Argument type	object
C definition:	AT_doccursor[I]

C data type:	DT_doccursor
COBOL definition:	AT-doccursor[I]
COBOL data type:	DT-doccursor
Access:	set, get

The XML Cursor of the XML Documents can be accessed through the doccursor attribute. The attribute is indexed with the object index. The initial child cursor possesses the index 1. The attribute is available for the XML Document.

12.8.3.4 document[I]

Identifier:

.document[I]

Classification: Standard Attribute

Definition

Argument type	object
C-Definition:	AT_document[I]
C-Data type:	DT_document
COBOL definition:	AT-document[I]
COBOL data type:	DT-document
Access:	set, get

The XML Documents can be accessed through the document attribute. The attribute is indexed with the object index. The first child document possesses the index 1. The attribute is available for all objects, whose record attribute is accessible.

12.8.3.5 idispatch

Identifier:

.idispatch

Classification: Object and Window-System-Specific Attribute

Definition

Argument type	pointer

C definition:	AT_idispatch
C data type:	DT_pointer
COBOL definition:	AT-idispatch
COBOL data type:	DT-pointer
Access:	set (not with XML Cursor and control, cannot be passed down), get

Can only be read by XML Cursor and control.

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. 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 sub-control. 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 that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

12.8.3.6 ixmldomdocument2

Identifier:

.ixmldomdocument2

Classification: Object and Window-System-Specific Attribute

Definition

Argument type	pointer
C definition:	AT_ixmldomdocumnet2
C data type:	DT_pointer
COBOL definition:	AT-ixmldomdocumnet2
COBOL data type:	DT-pointer
Access:	set (cannot be passed down), get

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* for invalid XML Cursors.

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.

12.8.3.7 ixmldomnode

Identifier:

.ixmldomnode

Classification: Object and Window-System-Specific Attribute

Definition

Argument type	pointer
C definition:	AT_ixmldomnode
C data type:	DT_pointer
COBOL definition:	AT-ixmldomnode
COBOL data type:	DT-pointer
Access:	get

Can only be read.

The IXMLDOMNode COM interface pointer of the XML Cursor can be accessed through this attribute under Microsoft Windows. 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 that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

12.8.3.8 ixmldomnodelist

Identifier:

.ixmldomnodelist

Classification: Object and Window-System-Specific Attribute

Definition

Argument type	pointer
C definition:	AT_ixmldomnodelist
C data type:	DT_pointer
COBOL definition:	AT-ixmldomnodelist
COBOL data type:	DT-pointer
Access:	get

Can only be read.

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. 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 that an XML Cursor, whose attribute *.mapped* possesses the value *false*, will automatically be positioned to the root of the DOM Document.

12.8.3.9 mapped

Identifier:

.mapped

Classification: Object-Specific Attribute

Definition

Argument type	boolean
C definition:	AT_mapped
C data type:	DT_boolean
COBOL definition:	AT-mapped
COBOL data type:	DT-boolean
Access:	get

Can only be read.

Is true when the XML Cursor points to a node in the DOM tree. As a rule, the access to a different attribute of XML Cursors is positioned at the root of the DOM tree.

The attribute is available for the XML Cursor.

12.8.3.10 name

Identifier:

.name

Classification: Object-Specific Attribute

Definition

Argument type	string
C definition:	AT_name
C data type:	DT_string
COBOL definition:	AT-name
COBOL data type:	DT-string
Access:	get

Can only be read.

Name or tag of the DOM node.

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.

12.8.3.11 nodetype

Identifier:

.nodetype

Classification: Object-Specific Attribute

Definition

Argument type	enum
C definition:	AT_nodetype
C data type:	DT_enum
COBOL definition:	AT-nodetype
COBOL data type:	DT-enum
Access:	get

Can only be read.

This is for calling up the type of DOM node. Possible values can be:

- nodetype_element
- nodetype_attribute
- nodetype_text
- nodetype_cdata_section
- nodetype_entity_reference
- nodetype_entity
- nodetype_processing_instruction
- nodetype_comment
- nodetype_document
- nodetype_document_type
- nodetype_document_fragment
- nodetype_notation

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.

12.8.3.12 path

Identifier:

.path

Classification: Object-Specific Attribute

Definition

Argument type	string
C-Definition:	AT_path
C-Data type:	DT_string
COBOL definition:	AT-path
COBOL data type:	DT-string
Access:	get

Can only be read.

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

If the value of the path attributes is noted somewhere else (i.e. in the userdata attribute), then it is important to know that these noted values will not be adjusted when the structure of the DOM tree changes. When the select method with the noted value is called up after this, then the XML Cursor will end up pointing to an incorrect DOM node.

The attribute is available for the XML Cursor.

12.8.3.13 publicid

Identifier:

.publicid

Classification: Object-Specific Attribute

Definition

Argument type	string
C definition:	AT_publicid
C data type:	DT_string
COBOL definition:	AT-publicid
COBOL data type:	DT-string
Access:	get

Can only be read.

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

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.

12.8.3.14 specified

Identifier:

.specified

Classification: Object-Specific Attribute

Definition

Argument type:	boolean
C definition:	AT_specified
C data type:	DT_boolean
COBOL definition:	AT-specified
COBOL data type:	DT-boolean
Access:	get

Can only be read.

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

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.

12.8.3.15 systemid

Identifier:

.systemid

Classification: Object-Specific Attribute

Definition

Argument type	string

C definition:	AT_systemid
C data type:	DT_string
COBOL definition:	AT-systemid
COBOL data type:	DT-string
Access:	get

Can only be read.

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

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.

12.8.3.16 target

Identifier:

.target

Classification: Object-Specific Attribute

Definition

Argument type	string
C definition:	AT_target
C data type:	DT_string
COBOL definition:	AT-target
COBOL data type:	DT-string
Access:	get

Can only be read.

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

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.

12.8.3.17 text

Identifier:

.text

Classification: Object-Specific Attribute

Definition

Argument type	string, object
C definition:	AT_text
C data type:	DT_string
COBOL definition:	AT-text
COBOL data type:	DT-string
Access:	set (cannot be passed down), get

The value of all sub-nodes within the DOM nodes. A string is delivered which represents the text of all sub-nodes. This attribute is mainly helpful when one needs the text of an XML element. This way one can avoid first having to go to the child nodes, which contain the actual text. Please note, the setting of this attribute will automatically delete all child nodes and a new text node will be inserted.

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.

12.8.3.18 value

Identifier:

.value

Classification: Object-Specific Attribute

Definition

Argument type	string, object
C definition:	AT_value
C data type:	DT_string
COBOL definition:	AT-value
COBOL data type:	DT-string
Access:	set (cannot be passed down), get

Value of the DOM Nodes. The attribute is only available when the attribute nodetype possess one of the following values: *nodetype_attribute*, *nodetype_text*, *nodetype_cdata_section*, *nodetype_processing_instruction* or *nodetype_comment*.

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.

12.8.3.19 xml

Identifier:

.xml

Classification: Object-Specific Attribute

Definition

Argument type	string, object
C definition:	AT_xml
C data type:	DT_string
COBOL definition:	AT-xml
COBOL data type:	DT-string
Access:	set (for XML Document only), get

"changed" (only for XML Document), this means the attribute can be used for triggering rules.

The attribute can only be read for an XML Cursor.

The string representations of the XML Documents or XML Cursors can be called up via this attribute. When a new value is set in the XML Document, the saved DOM tree will be deleted, and a new tree is erected with the set values. All existing XML Cursors will become invalid. The attribute *.mapped* has the value *false* for invalid XML Cursors.

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.

12.8.4 Methods

12.8.4.1 add

:add

Adds a child node as the last node to an actual DOM node. The XML Cursor is then set to the new element.

Syntax

```
boolean :add
(
  enum Nodetype
  { , string Name }
  { , string Value }
  { , boolean Select }
)

boolean :add
(
  pointer IXMLDOMNode
  { , boolean Select }
)
```

Parameters

enum Nodetype

States the type of DOM node, which will be created. The following values are possible:

- *nodetype_attribute*
- *nodetype_cdata_section*
- *nodetype_comment*
- *nodetype_document*
- *nodetype_document_fragment*
- *nodetype_document_type*
- *nodetype_element*
- *nodetype_entity*
- *nodetype_entity_reference*
- *nodetype_notation*
- *nodetype_processing_instruction*
- *nodetype_text*

string Name

Is an optional parameter, which states the name or the tag of the new DOM node. When the nodetype parameter has the following value, then the name will be ignored and a pre-determined name will be assigned to it:

Nodetype	Predefined Name
nodetype_cdata_section	*#cdata-section*
nodetype_comment	*#comment*
nodetype_document	*#document*
nodetype_document_fragment	*#document-fragment*
nodetype_text	*#text*

string Value

Is an optional parameter, which states the value of the new DOM node. This is only allowed when the nodetype parameter possesses one of the following values:

- *nodetype_attribute*
- *nodetype_cdata_section*
- *nodetype_comment*
- *nodetype_processing_instruction*
- *nodetype_text*

pointer IXMLDOMNode

Indicates the Microsoft Windows COM object, which is to be added as a child node. A Microsoft Windows IXMLDOMNodeList COM object can also be indicated, whereby then all XML nodes, which are contained in the list, must be added.

The pointer parameter is only available under Microsoft Windows.

boolean Select

Is an optional parameter, which states if the XML Cursor is to point to the newly created DOM node or not. When the parameter is not given, then true will be selected. In other words, the XML Cursor will be set to the new DOM node.

Return Value

Indicates if the creation of the new DOM node was successful or not. In the case of an error, the XML Cursor will remain on the actual DOM node.

Permitted method for the XML Cursor.

12.8.4.2 delete

:delete

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

If the value of the path attributes is noted somewhere else (i.e. in the userdata attribute), then it is important to know that these noted values will not be adjusted when the structure of the DOM tree changes. When the select method with the noted value is called up after this, then the XML Cursor will end up pointing to an incorrect DOM node.

Syntax

```
boolean :delete
(
)
```

Parameters

None.

Return Value

Indicates if the deletion was carried out successfully or not.

Permitted method for the XML Cursor.

12.8.4.3 load

:load

This loads the XML Document from the given file or URL. The saved DOM tree will be deleted and a new tree will be erected. All existing XML Cursors will become invalid. The attribute *.mapped* has the value *false* for invalid XML Cursors.

Syntax

```
boolean :load
(
  string URL
)
```

Parameter

string URL

This is the data path of the document, which is to be loaded. It is also possible to state an URL in place of a path.

Return Value

Indicates if the file was able to be loaded or not. In the case of an error, the DOM tree will either remain as it is or it will be deleted. A partial transformation does not exist.

Permitted method for the XML Cursor.

12.8.4.4 match

:match

Tests if the DOM node is the same as the given model (see also “Model for the Methods match and select).

Syntax

```
boolean :match
(
  string Pattern
)
```

Parameter

string Pattern

The DOM node is compared to this model. The erection of the model is described in “Model for the Methods match and select”.

Return Value

States if the DOM node corresponds to the node.

Permitted method for 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.

12.8.4.4.1 Model for the Methods match and select

A model is very similar to a Dialog Manager Character identifier. The model builds a path from element names. The path begins at the root of the DOM tree. Each hierarchy level is compared to the corresponding part of the path. In addition, certain properties of the parent, such as the possession of a certain attribute or the position within the children, can be given.

Principally, all symbols except for ., [, ",], <, >, =, \, tab, space and page break are allowed. If one of the symbols mentioned above is used, with exception to the page break, then a \ must be placed in front of it. Between two parenthesis ("), in other words within a string, these symbols ., [,], <, >, =, tab and space are also allowed. Please note that in the dialog script the symbol \ within a string is already an escape symbol. This means that \\ must always be given in a dialog script whenever a \ is needed. Also, the symbol \" must be given in a dialog script whenever a " is needed.

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

<Name>

Is compared with the name attribute of the XML Cursor. The XML Cursor must possess the node type *nodetype_element*. As an alternative * can also be given. In this case the name attribute will be ignored. The name of an XML element begins either with a letter or an underscore. The name can also contain letters, numbers, hyphens, underscores and periods. The exact definition of an XML element name can be read in the XML specifications (**https://www.w3.org**).

<Attr>

The DOM node, upon which the XML Cursor points to, must have the indicated attribute.

The name of the XML attribute begins with either a letter or an underscore. The name can also contain letters, numbers, hyphens, underscores and periods. The exact definition of an XML attribute name can be read in the XML specifications (**https://www.w3.org**).

<Op>

Is the compare-operator for comparing the given attribute in <Attr> with <Value>. It can be tested for equal **=** and not equal **<>**.

<Value>

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

<Idx>

The DOM node must be placed to this position within the child nodes of the parent nodes. The first child possesses the position 1.

<Idx> is made up of figures (**0** – **9**), which are interpreted as numbers.

Special Situations

.

When the model begins with a period, then this is relative to the actual DOM node. This means the path begins at the actual DOM node.

..

When two periods follow one another, then many hierarchy levels will be skipped over.

[<Idx>]

An index without further specifications selects the DOM node at this position. Here, the node type of the node remains disregarded. Each DOM node can, as a result, be clearly referenced through the following form: {**[**<Idx>**][.[**<Idx>**]]**} (path attribute).

Additions under Microsoft Windows

XPath can be used as a model syntax. Hereby, the model must either begin with **/** or **.//**. The use of XPath models is not supported on every platform, which means that it is not portable.

12.8.4.5 reparent

:reparent

Transfers the DOM node with all of its children.

If the values of the path attributes are noted somewhere else, (i.e. in the userdata attribute), then it is important to note that these noted values will not be adjusted, when the structure of the DOM tree has been changed. When the method select is called up at a later date, then it is possible that the XML Cursor will point to an incorrect DOM node.

Syntax

```
boolean :reparent
(
  object Parent
  { , integer Index }
)

boolean :reparent
(
  integer Index
)
```

Parameter

object Parent

Indicates the new parent node. This must be a doccursor type. When no index is given, then the DOM node will be added at the end.

integer Index

Indicates the position where the DOM node is to be added. If –1 is given as the index, then the DOM node will be added as the last node.

Return Value

Indicates if the action was successful.

Permitted method for the XML Cursor.

12.8.4.6 save

:save

Saves the XML Document in a file or an URL.

Syntax

```
boolean :save
(
  string URL
)
```

Parameter

string URL

This is the file path for saving information. In the place of a path, an URL can also be used.

Return Value

Indicates if the saving process was carried out correctly or not.

Permitted method for an XML Document.

12.8.4.7 select

:select

Moves the XML Cursor in the given direction, or it moves the XML Cursor to the first node in the DOM tree, which corresponds to the given model. (see also "Model for the Methods match and select").

Syntax

```
boolean :select
(
  anyvalue DirectionOrPattern
)
```

Parameter

anyvalue DirectionOrPattern

The parameter is only allowed to be from one of the following types:

enum DirectionOrPattern

Indicates the direction the XML Cursor will be moved within the DOM tree. The following values are possible:

select_document
The XML Cursor is set to the document. The XML Cursor does not necessarily reference a DOM node yet.

select_first
The XML Cursor is set to the first DOM node, which matches the pattern that has been searched for last.

select_first_child
The XML Cursor is set to the first direct child node.
select_first_sibling
The XML Cursor is set to the first DOM node of the same parent.
select_last
The XML Cursor is set to the last DOM node, which matches the pattern that has been searched for last. This operation searches for the respective DOM node.
select_last_child
The XML Cursor is set to the last direct child node. This operation searches for the respective DOM node.
select_last_sibling
The XML Cursor is set to the last DOM node of the same parent. This operation searches for the respective DOM node.
select_next
The XML Cursor is set to the next DOM node, which matches the pattern that has been searched for last.
select_prev
The XML Cursor is set to the previous DOM node, which matches the pattern that has been searched for last.
select_next_sibling
The XML Cursor is set to the next DOM node of the same parent.
select_prev_sibling
The XML Cursor is set to the previous DOM node of the same parent.
select_root
The XML Cursor is set to the root node of the DOM tree. This is the initial value of an XML Cursor. The XML Cursor does not necessarily reference a DOM node yet.
select_up
The XML Cursor is set to the parent node.

string DirectionOrPattern

Indicates a pattern. The XML Cursor will be set to the first node in the DOM tree that corresponds to the model. The model will be noted so that the methods of the XML Cursor :select and select_first, select_last, select_next and select_prev, which also correspond to the model, can be set to other DOM nodes. The erection of the model is described in "Model for the Methods match and select".

Return Value

Indicates if the XML Cursor could be converted. In the case of an error, then the XML Cursor will remain pointed at the initial DOM node.

Permitted method for 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.

12.8.4.7.1 Model for the Methods match and select

A model is very similar to a Dialog Manager Character identifier. The model builds a path from element names. The path begins at the root of the DOM tree. Each hierarchy level is compared to the corresponding part of the path. In addition, certain properties of the parent, such as the possession of a certain attribute or the position within the children, can be given.

Principally, all symbols except for **.**, **[**, **"**, **]**, **<**, **>**, **=**, ****, tab, space and page break are allowed. If one of the symbols mentioned above is used, with exception to the page break, then a **** must be placed in front of it. Between two parenthesis (**"**), in other words within a string, then these symbols **.**, **[**, **]**, **<**, **>**, **=**, tab and space are also allowed. Please note that in the dialog script the symbol **** within a string is already an escape symbol. This means that **** must always be given in a dialog script whenever a **** is needed. Also, the symbol **\"** must be given in a dialog script whenever a **"** is needed.

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

<Name>

Is compared with the name attribute of the XML Cursor. The XML Cursor must possess the node type *nodetype_element*. As an alternative * can also be given. In this case the name attribute will be ignored. The name of an XML element begins either with a letter or an underscore. The name can also contain letters, numbers, hyphens, underscores and periods. The exact definition of an XML element name can be read in the XML specifications (**https://www.w3.org**).

<Attr>

The DOM node that the XML Cursor points to must have the indicated attribute.

The name of the XML attribute begins with either a letter or an underscore. The name can also contain letters, numbers, hyphens, underscores and periods. The exact definition of an XML attribute name can be read in the XML specifications (**https://www.w3.org**).

<Op>

Is the compare-operator for comparing the given attribute in <Attr> with <Value>. It can be tested for equal **=** and not equal **<>**.

<Value>

The value upon which <Attr> is compared to.

<Idx>

The DOM node must be placed to this position within the child nodes of the parent nodes. The first child possesses the position 1.

<Idx> is made up of figures (**0** – **9**), which are interpreted as numbers.

Special Situations

.

When the model begins with a period, then this is relative to the actual DOM node. This means the path begins at the actual DOM node.

..

When two periods follow one another, then many hierarchy levels will be skipped over.

[<Idx>]

An index without further specifications selects the DOM node at this position. Here, the node type of the node remains disregarded. Each DOM node can, as a result, be clearly referenced through the following form: {**[**<Idx>**]**[.**[**<Idx>**]**]} (path attribute).

Additions under Microsoft Windows

XPath can be used as a model syntax. Hereby, the model must either begin with */* or *./*. The use of XPath models is not supported on every platform, which means that it is not portable.

12.8.4.8 transform

:transform

Transforms the object with the indicated scheme. When the target is an XML Document, the saved tree will be deleted and a new tree will be erected. All existing XML Cursors will become invalid. The attribute *.mapped* has the value *false* for invalid XML Cursors.

As an alternative, the target of the transformation can also be a text or a file. When the result of the transformation does not have a legal format, then it must be immediately transformed into a text or a file. This is true because the result can therefore not be referenced to an XML Document. This is the case by transformations to HTML.

Syntax

```
boolean :transform
(
  object Scheme
  { , anyvalue Target }
  { , enum Type }
)
```

Parameters

object Scheme

This is the XSL transformation document with which items are transformed. The parameter must be an XML Document.

anyvalue Target

This is an optional parameter, and is the target of the transformation. If the parameter is not given, then this object will be taken as the target. The parameter can be one of the following types:

object Target

This parameter is the input parameter and it must represent an XML Document. The saved DOM tree will be deleted and a new DOM tree will be erected according to the transformation.

string Target

Dependent on type parameter, this parameter is the input parameter and it indicates a data name. Or it is an output parameter, which contains the string through which the transformation has taken place.

enum Type

This is an optional parameter, which indicates the type of transformation. Possible values include:

» *type_object*
Indicates that the target parameter is an XML Document. This value is taken as a rule when the target parameter is from type object.

» *type_text*
Indicates that the target parameter is an output parameter from type string. This value is taken as a rule when the target parameter is from type string.

» *type_file*
Indicates that the target parameter describes a file.

Return Value

Indicates if the transformation was successful or not. In the case of an error, the DOM tree will either remain the target of the XML Documents or it will be deleted. A partial transformation does not exist.

Permitted method for XML Cursor and XML Document.

12.8.4.9 validate

:validate

Checks the XML Document if it is the same as the document type given in the DOM tree. When it doesn't contain a document type, then an error message will appear.

Syntax

```
integer :validate
(
  { string ErrorMsg }
)
```

Parameter

string ErrorMsg

Is an optional output parameter. When this is given, then the parameter will be initialized with "". In the case of an error, a system error message will be assigned to the parameter. It is, however, better to call up the return value.

Return Value

Is either the DOM error code or 0. The value –1 indicates that an internal error has occurred. In this case, ErrorMsg is not installed.

Permitted method for the XML Document.

12.9 XML Transformation

The functionality of “XML Transformation” builds upon the XML interface and is made up of the objects “transformer” and “mapping”. It serves for the transformation of XML files to IDM objects/attributes and vice versa.

12.9.1 Transformer Object

Keyword:

transformer

Definition

```
{export} {model} transformer {<Identifier>}
{
  [<Attributedefinition>]
  [<Methoddefinition>]
}
```

The transformer object allows for the scanning through of an XML Document or an IDM hierarchy, whereby during the scanning semantic actions can be carried out on separate nodes. Through this it is easy to implement a transformation of files. For example, XML files can be transferred into IDM objects or vice versa; with data that is contained in IDM objects it is possible to generate an XML Document. Because semantic actions are described through user-defined codes, the transformation can have many different forms.

The following means are available to the IDM programmer regarding the definition of a transformation.

1. A transformation is started through the calling up of the :apply() method of a transformer object. As parameter, the source and the target of the transformation are handed over (i.e. a doccursor object as source and an IDM object as target, which should either take on the data or delegate it

elsewhere).

2. It is normally desired that during each transformation a certain amount of nodes, regardless if these are IDM objects or nodes of an XML tree, are run through in a certain sequence. The :apply () method realizes such a sequence when beginning with the start node (source) in a loop the :select_next() method of the transformer is called up, which determines the successor of the actual node. The sequence is ended as soon as the :select_next() method returns a 0. the default implementation of select_next() method defines a pre-order sequence. The IDM programmer is able to intervene at two different points in this process. First of all, the :apply() method can be newly defined in order to set the start node or to change the abort conditions. Secondly, the :select_next () method can be newly defined and with it the entire sequence in which the nodes are visited.
3. After is has been guaranteed that all nodes have been visited at one time or another, then a possibility must exist for setting the semantic action(s) that are to be carried out on certain nodes. For this purpose mapping objects are defined to the transformer object as children. Each of these objects defines a model in its .name attribute, which is then referred to, in order to find out if the node that is currently being visited should release and action or not. Such an action is set through the :action() method of the mapping object, which will be newly defined from the IDM programmer. This method receives, as the first parameter, the node that was just visited, and as the second parameter it will receive the target object, which comes from the :apply() method of the transformer.

In order to complete the picture: The transformer object also has a :action() method. This method will be called up in the above mentioned loop of the :apply() method for each visited node. It is then checked if a model of one of the mapping children passes to the node. The sequence in which this happens is the same as the definition sequence of the mappings by the parent transformer. The inherited mapping objects are examined at the very end. If during the examination a fitting mapping object is found, then the :action() method of the transformer calls up the :action() method of the mapping object. This method can transfer the data from the actual node to the target object. As a return value, this method can return *true*. In this case it is assumed that the node has been completely worked through and no further mapping objects should be examined. Otherwise, the left over mappings will be examined and their :action() methods will be called up until either all mappings have been gone through or the :action() method from one of these mappings returns the value *false*.

Event	–
Child	mapping
	record
Parent	application
	canvas
	checkbox
	dialog

doccursor

document

edittext

groupbox

image

import

layoutbox

listbox

mapping

menubox

menuitem

menusep

messagebox

modul

notebook

notepage

poptext

pushbutton

radiobutton

record

rectangle

scrollbar

spinbox

splitbox

statictext

statusbar

tablefield

timer

toolbar

transformer

treeview

window

Menu –

Attributes

Attribute	RSD	PSD	Properties	Short Description
firstrecord	identifier	object	S,G/-/-	First record of the transformer
label	string	string	S,G/D/C	Name of the transformer
lastrecord	identifier	object	S,G/-/-	Last record of the transformer
mapping[integer]	identifier	Identifier	S,G/-/-	Mappings of the transformer
model	identifier	object	S,G/D/C	Model of the transformer
namedrule[integer]	identifier	object	S,G/-/-	Rules with names of the transformer
parent	identifier	object	S,G/-/-	Parent of the transformer
record[integer]	identifier	object	S,G/-/-	Records of the transformer
recordcount	integer	integer	-,G/-/-	Number of records of the transformer
root	anyvalue	Anyvalue	S,G/-/-	Here, the root of the search is noted during a transformation
rule[integer]	identifier	object	S,G/-/-	Rules of the transformer
scope	integer	scope	-,G/-/-	Defines if the object default is a model or an instance
userdata	anyvalue	Anyvalue	S,G/D/C	Application data of an object

Methods

Method	Return Value	Arguments	Short Description
action	Boolean	anyvalue Src anyvalue Dest	This method searches through all mapping children of the transformer for a counterpart between their models and nodes in Src. If such a mapping object M1 is found, then its :action() method will be called up as M1:action(Src, Dest).
apply	Boolean	anyvalue Src anyvalue Dest	This method activates the transformation. This method can also be redefined.
select_next	Object	object Src	This method determines, together with the nodes, the next node in Src that is to be run through in the transformation. This method can be redefined.

12.9.1.1 Object-Specific Attribute

mapping

The mapping children can be accessed through the .mapping attribute. The attribute is indexed with the object index (similar to child). The sequence of the mappings within this vector determines the sequence in which nodes are compared to separate mappings during a transformation. The inherited mappings are not taken on in this vector. During a transformation, the comparison to mappings is done at the very end, i.e. the direct instances have priority. This is different in comparison to other inherited child objects within IDM, which are inserted at the beginning in the child vector of the parent instance.

root

After calling up the :apply() method the starting point of the transformation will be noted in this attribute. This makes it possible to decide during a transformation, if the starting point has once again been reached, and if the transformation can be stopped. The following case-decisions should be considered:

- When the Src-parameter is the :apply() method, a document or a doccursor, then a string will be saved in .root, which describes the correct position in the XML tree. The syntax of the string is exactly the convention, which is used in the .path attribute within the doccursor object (see doccursor object description). Comparisons with the .path attributes from doccursor are very easy because of this.
- If the Src-paarameter of the apply() method is an IDM object, then this object will be noted in .root. Consequently, in this case, the type of this attribute is object.

On the basis of the values in the .root attribute, the :action() and :select_next() methods of the transformer decide what they have to do (see description of these methods).

At the end of the :apply() method .root will be set to void again.

Default value is void.

12.9.1.2 Object-Specific Methods

apply

The transformation is initiated with this method. The default implementation of the algorithm can be seen below:

» If the Src-parameter is a document or a doccursor-object, then it is assumed that data from the XML tree should be transferred to IDM. In the case of a document object, a temporary doc-cursor object will be created which is used for navigation purposes within the XML tree. In the following pseudo-code (for purposes of simplicity), it is assumed that in Src a doccursor is always transferred.

```
:apply(anyvalue Src, anyvalue Dest) {
  variable object NextObj;
  this.root ::= Src.path;
  NextObj := Src;
  while NextObj <> null do
    this:action(Src, Dest);
    NextObj := this:select_next(NextObj);
  endwhile
  this.root ::= null;
  return true;
}
```

» If the Src-parameter is an IDM object (except for document and doccursor objects), then it is assumed that data from IDM should be transferred to XML or wherever else. The underlying code is identical to the code above, except that here Src and not Src.path is noted in .root. For example,

```
this.root ::= Src;
```

The apply() method can be redefined (similar to :init()).

action

This method is used by the :apply() method of the transformer (see above). Through this it is determined if the actual node fits to one of the mapping children. For this reason, the :action() method of the mapping objects must be called up.

The action() method can be can be redefined (similar to :init()).

select_next

This method is used by :apply() method of the transformer for going through all nodes of an XML tree or the IDM object hierarchy (see above). The default implementation of this method looks like

that the repeated calling up of the method realizes a pre-order run through.

The select_next() method can be can be redefined (similar to :init()).

Example of a Transformer

This example can be found in the *examples* directory.

We use the following files with the name “CD-Catalog.xml” as an XML Document:

```
<?xml version="1.0" ?>
<CATALOG>
  <CD>
      <TITLE>Empire Burlesque</TITLE>
      <ARTIST>Bob Dylan</ARTIST>
      <COUNTRY>USA</COUNTRY>
      <COMPANY>Columbia</COMPANY>
      <PRICE>10.90</PRICE>
      <YEAR>1985</YEAR>
  </CD>
  <CD>
      <TITLE>Hide your heart</TITLE>
      <ARTIST>Bonnie Tyler</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>CBS Records</COMPANY>
      <PRICE>9.90</PRICE>
      <YEAR>1988</YEAR>
  </CD>
  <CD>
      <TITLE>Maggie May</TITLE>
      <ARTIST>Rod Stewart</ARTIST>
      <COUNTRY>UK</COUNTRY>
      <COMPANY>Pickwick</COMPANY>
      <PRICE>8.50</PRICE>
      <YEAR>1990</YEAR>
  </CD>
</CATALOG>
```

The data from these files can, for example, be read out as follows:

```
dialog D {}

window Wi
{
  .title "XML-CD-CATALOG";

  on close { exit(); }

  child treeview Tv
  {
```

```
    .xauto 0;
    .yauto 0;
    .style[style_lines] true;
    .style[style_buttons] true;
    .style[style_root] true;
    integer CdIdx := 0;

    rule NewCatalog() {
        if this.itemcount = 0 then
           this.itemcount ::= this.itemcount + 1;
        endif
    this.content[this.itemcount] := "CD-Catalog";
    this.open[this.itemcount] := true;
    }
    rule AddCD() {
    this:insert(this.itemcount+1, 4);
    this.CdIdx := this.CdIdx + 1;
    this.content[this.itemcount-3] := ""+this.CdIdx+". CD";
    this.level[this.itemcount-3] := 2;
    }
    rule AddTitle(string S input) {
    this.content[this.itemcount-2] := "Title: " + S;
    this.level[this.itemcount-2] := 3;
    }
    rule AddArtist(string S input) {
    this.content[this.itemcount-1] := "Artist: " + S;
    this.level[this.itemcount-1] := 3;
    }
    rule AddPrice(string S input) {
    this.content[this.itemcount] := "Price: " + S;
    this.level[this.itemcount] := 3;
    }
  }
}

transformer Tr {
    !! transformer is for the taking over of data from an XML Document
    !! for this reason you can always expect a doccursor in Src

    child mapping MCatalog {
        .name "..CATALOG";

        :action() {
           Dest:NewCatalog();
           return true;
        }
```

```
    }
    child mapping MCD {
        .name "..CD";

        :action() {
            Dest:AddCD();
            return true;
        }
    }
    child mapping MTitle {
        .name "..CD.TITLE";

        :action() {
            Dest:AddTitle(Src.text);
            return true;
        }
    }
    child mapping MArtist {
        .name "..CD.ARTIST";

        :action() {
            Dest:AddArtist(Src.text);
            return true;
        }
    }
    child mapping MPrice {
        .name "..CD.PRICE";

        :action() {
            Dest:AddPrice(Src.text);
            return true;
        }
    }
}

document Doc {
}

on dialog start
{
  Doc:load("CD-Katalog.xml");

  Tv.visible := false;
  Tr:apply(Doc, Tv);
  Tv.visible := true;
}
```

12.9.2 Mapping Object

Keyword:

mapping

Definition

```
{export} {model} mapping {<Identifier>}
{
  [<Attributedefinition>]
  [<Methoddefinition>]
}
```

The sole purpose of the mapping object is to define semantic action, which is called up during a transformation for a specific node (in an XML tree or in the IDM object hierarchy), when a consensus between this mapping object and the node is found. The mapping objects are defined as children from a transformer object and described together with this one transformation of data.

Event	–
Child	record
Parent	transformer
Menu	–

Attributes

Attribute	RSD	PSD	Properties	Short Description
firstrecord	identifier	object	S,G/-/-	First record of the object
label	string	string	S,G/D/C	Name of the object
lastrecord	identifier	object	S,G/-/-	Last record of the object
name	string	string	S,G,/D/C	Here, a model is defined that is used for finding matches in a XML tree or in an IDM object hierarchy
model	identifier	object	S,G/D/C	Model of the object
namedrule[integer]	identifier	object	S,G/-/-	Named rules of the object
parent	identifier	object	S,G/-/-	Parent of the object
record[integer]	identifier	object	S,G/-/-	Record of the object

Attribute	RSD	PSD	Properties	Short Description
recordcount	integer	integer	-,G/-/-	Number of records in the object
rule[integer]	identifier	object	S,G/-/-	Object rules
scope	integer	scope	-,G/-/-	Defines if the object is a default, model or an instance
userdata	anyvalue	anyvalue	S,G/D/C	Application data of the object

Methods

Method	Return Value	Arguments	Short Description
action	boolean	anyvalue Src anyvalue Dest	These methods are called up from the parent transformer during a transformation when a match is found between a node in an XML tree or an IDM object hierarchy and the mapping object that belongs to it. The IDM programmer can then determine what should be done with the data from the node.

12.9.2.1 Object-Specific Attributes

name

Here, a model is given (a similar XPath expression), which defines the node within an XML tree or in the IDM object hierarchy upon which the mapping object fits. This determines if :action() method should be called up for the node during the transformation or not.

12.9.2.2 Model for .name

A model is very similar to a Dialog Manager character identifier. The model builds a path from element names. The path begins at the root of the document or by the IDM object hierarchy (i.e. dialog or module). Each hierarchy level is compared to the corresponding part of the path. In addition, certain properties of the parent, such as the owner attribute or the position within the children, can be given:

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

<Name>

XML: Is compared to the name attribute of the cursor. The cursor must possess the node type *nodetype_element*. As an alternative, * can also be used. In this case the name attribute will be ignored.

IDM: Is compared to the label of an IDM object. As an alternative, * can be use. In this case each label fits.

<Attr>

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

IDM: The IDM object must have the indicated attribute.

<Op>

XML and **IDM:** compare-operator, for comparing the attribute given in <Attr> with <Value>. The comparison is made for equals **=** and does not equal **<>**.

<Value>

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

<Idx>

XML: The XML node must be at this position within the children of its parent.

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

Special Situations

.

XML: Begins the model with a period, then it is relative to the actual node. The path then begins at the actual node.

..

XML and **IDM:** Two periods together pass an arbitrary number of hierarchy levels.

[<Idx>]

XML: An index, without other specific indications, selects the XML node at this position. The node-type of the node remains unconsidered. Therefore, each node can be clearly referenced by {**[**<Idx>**]**[**.[**<Idx>**]**]} (path attribute).

All comparisons check for capital and small letters.

Example

Model “..CD[.Title = Yellow]” references within an XML tree all nodes (regardless where they are in the hierarchy) with the tag “CD”, that have an attribute “.Title” and which also have the value *Yellow*.

When the same model is applied to IDM objects, then all objects possessing the label “CD”, and which have the user-defined attribute “.Title” plus the value *Yellow*, will be chosen.

12.9.2.3 Object-Specific Methods

action

This method is called up from the :action() method of the parent transformer, when a counterpart between a node in an XML tree or an IDM object hierarchy and the model in .name attribute of the mapping objects is found. The default implementation of this method does nothing; however, it does return the value *true*.

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

12.9.3 Attributes

12.9.3.1 mapping[I]

Identifier:

.mapping[I]

Classification: Object-specific attribute

Definition

Argument type:	object
C definition:	AT_mapping[I]
C data type:	DT_mapping
COBOL definition:	AT-mapping[I]
COBOL data type:	DT-mapping
Access:	set, get

The mapping objects of a transformer can be accessed through this mapping attribute. The attribute is indexed with the object index (similar to child).

The attribute is available on the transformer class.

12.9.3.2 transformer[I]

Identifier:

.transformer[I]

Classification: Standard attribute

Definition

Argument type:	object
C definition:	AT_transformer[I]
C data type:	DT_transformer
COBOL definition:	AT-transformer[I]
COBOL data type:	DT-transformer
Access:	set, get

The transformer can be accessed through this attribute. The attribute is indexed with the object index (similar to child). This attribute is to be uses in the same way as the record attribute.

The attribute is available on the classes, which can also have records as children.

12.9.3.3 name

Identifier:

.name

Classification: Object-specific attribute

Definition

Argument type:	string
C definition:	AT_name
C data type:	DT_string
COBOL definition:	AT-name
COBOL data type:	DT-string
Access:	set, get

Here, a model can be defined to a mapping object (see definition of mapping objects), which decides during a transformation on which node the mapping object is to be actively carried out with :action ()method in the XML tree or IDM hierarchy that is to be transformed.

The attribute is available on the mapping class.

12.9.3.4 root

Identifier:

.root

Classification: Object-specific attribute

Definition

Argument type:	anyvalue
C definition:	AT_root
C data type:	DT_string or DT_object
COBOL definition:	AT-root
COBOL data type:	DT-string or DT-object
Access:	set, get

This calls up the position where the transformation in a XML tree or in an IDM object hierarchy began. For this reason it only makes sense to call up the value of the attribute during a transformation (i.e. during the calling up of :apply() method). Otherwise, the value should be 0.

In the default implementation the :apply() method of the transformer will set this attribute as follows.

If a document object is transferred in Src, then .root will be assigned an empty string "", which represents the root of the document.

If a doccursor object is transferred, then root will be assigned a string that determines the position in the XML tree to which doccursor refers, i.e. .root := Src.path.

If a different IDM object is transferred, then .root will be assigned this object itself.

When leaving the :apply() method,.root will be set to 0.

The attribute is available on the transformer class.

12.9.4 Methods

12.9.4.1 action (transformer)

:action

This method is called up from the :apply() method of the transformer for each visited node of an XML tree or an IDM object hierarchy. The method must go through all mapping children of the transformer and test to see if model in their .name attributes fits to the actual nodes (Src parameter). The sequence in which the mappings are tested is the same as the definition sequence in the .mapping[] vector, whereby the inherited mappings are tested at the very end.

Note

The inherited mappings are not contained in the .mapping-vector of the parent. Therefore, it is only possible to access them through the heredity-hierarchy. This is different as with normal IDM inheritance.

When a match is found, then the :action() method that belongs to the mapping object will be called up, whereby Src and Dest-parameter will be skipped over. The return value of this :action() method determines if the mapping objects that are left over should be examined, and if necessary, if their :action() methods for these nodes should also be called up (return value is *false*), or if the nodes should be considered as already worked on and the :action() method of the transformer be ended (return value is *true*).

Syntax

```
boolean :action
(
anyvalue Src input,
anyvalue Dest input output
)
```

Parameters

anyvalue Src

Here, the node to be examined is transferred.

- When the .root attribute of the transformer contains a string, then a doccursor object is expected, which refers to the XML node. This node will then be compared to the mappings.
- When the .root attribute of the transformer contains an IDM object, then and IDM object is expected here, which represents the actual node. This object will then be compared to the mappings.

anyvalue Dest

Here the value, which when called up with :apply is transferred as Dest or is changed in one of the earlier calls of the :action() methods, is always channeled through.

Return Value

true when the process was completed without error.

false if otherwise.

Permitted method for ***transformer*** class.

12.9.4.2 action (mapping)

:action

This method defines how data should be transferred to separate nodes during a transformation, or in other words, how they should be transferred from Src parameter to Dest parameter.

Each mapping object possesses a :action() method, which returns the value *true* as the default implementation. This method can be redefined by the IDM programmer (similar to :init()). Therefore, it is possible to determine in which way the data is to be transferred from the Src parameter to the Dest

parameter. The method :action() is called up during a transformation by the parent of the mapping object (transformer). To be more exact, by the :action() method of the transformer, when a counterpart between the node and an XML tree or an IDM object hierarchy and the mapping object that belongs to it is found (see also Documentation of Transformer Object).

Syntax

```
boolean :action
(
  anyvalue Src input,
  anyvalue Dest input output
)
```

Parameters

anyvalue Src

Here, the actual node is transferred to the place where the transformation is currently standing. Two cases are to be considered:

- An XML tree is to be transformed.
  In Src a doccursor object is transferred which refers to the actual node in the XML tree.
- An IDM object hierarchy is to be transformed.
  In Src an IDM object is transferred which represents the actual node.

anyvalue Dest

Here the value, which when called up with :apply is transferred as Dest or is changed in one of the earlier calls of the :action()methods, is always channeled through.

Return Value

When the method returns the value *true*, then it is assumed that the actual node has been completely processed and no further mapping object will be tested for a counterpart to the node. This means no further :action() methods will be called up when other mapping objects that fit still exist.

When the value *false* is returned, then the actual node will be compared to more mapping objects, and if necessary their :action() methods will be called up.

Permitted method for ***mapping*** class.

12.9.4.3 apply

:apply

The transformation of data is initiated on a transformer object with this method. For further details please refer to the documentation of the transformer object.

This method can be redefined (similar to :init()).

Syntax

```
boolean :apply
(
  anyvalue Src input,
  anyvalue Dest input output
)
```

Parameters

anyvalue Src

Here, the root node is transferred from the place where the transformation is supposed to begin. Hereby, two cases in the default implementation should be taken into consideration:

- If Src is a document object, then the root of the XML tree, which represents the document, will be chosen as the start node. Document must be loaded. If Src is a doccursor object, then the root of the node in the XML tree will be used, upon which the doccursor object refers to. The transformation direction is from XML to IDM.
- If Src is an IDM object (except for document or doccursor), then the IDM object itself will be chosen as the root. The transformation direction is arbitrary to IDM.

anyvalue Dest

Here, the IDM programmer can transfer any value, which will only be channeled through to the :action methods of the mapping-objects. This means that within the method the call for :action(Next_Node, Des) is executed iteratively for each visited node.

Return Value

The value *true* will be returned if the transformation was a success. Otherwise, the value will be *false*.

Permitted method for ***transformer*** object.

12.9.4.4 select_next

:select_next

The transformer object calls up this method in order to determine the node that should be visited next during a transformation. The sequence in which the nodes of an XML tree or an IDM object hierarchy are to be visited can be redefined with this method. The default implementation of this method defines a deep cycle.

The method can be redefined (similar to :init()).

Syntax

```
object :select_next
(
  object Src
)
```

Parameter

object Src

In this parameter the actual, most recently processed node, is transferred. :select_next() orientates itself to the type of the .root attribute in the default implementation.

» If .root is a string, then in Src a doccursor object will be expected which refers to the actual node in the XML tree. This cursor will be set to the next node in the XML tree (in the sense of the Pre-Order-Order) and returned as a return value.

» If .root is an IDM object, then in Src an IDM object is expected which represents the actual node. The successor is based on this.

Return Value

Here, the next node is returned.

Or, in the case that the transformation needs to be stopped, a *null*, is returned.

The default implementation delivers the following:

» If .root is a string, then the doccursor will be returned which was transferred in Src and will now be placed further.

» If .root is an IDM object, then an IDM object will be returned which is to be examined next.

Permitted method for ***transformer*** object.

12.10 Callback Function DM_ExistsMessageProc

The parameters of the Callback Function DM_ExistsMessageProc, which must provide an application when you want to encrypt the data transfer between dialog page and display page in Java WSI, have changed. The changes are in bold print. The rest has been taken from the documentation.

Checks if files are available**, or waits until files are available**.

```
int DM_ExistsMessageProc (void *connptr, long timeout, char *message)
```

Parameters

connptr

Client identifier.

timeout

Time interval in milliseconds. When the value is larger than 0, then the function must return no later than the given time. If the value for the time interval is 0, then the function must return immediately (polling).

message (output)

Error message, must equal " " (blank), when **no** error has occurred**, or when the timeout interval has expired**.

Please note that the transferred buffer can only accept 1024 characters (including the 0 byte). The buffer is already pre-initialized with " ".

Return Value

0 Data is available.

1 No data is available. **This value is also returned when the timeout interval has expired.**

13 Version A.04.02.j

13.1 Windows NT

- Gnats 9436: The scrollbars of a tablefield object were disabled when clicked on.
- Gnats 9429: Problems arose with the rule "on open" in menus when network functions were called up in this rule and the user began to click around wildly. Now, in this situation the opening of the menu is prevented.
- Gnats 9426: The WIN32 IDM version was delivered in place of the Java IDM Version for Java-WSI.
- Gnats 9422: When the attribute .level on the treeview object was changed, then the attribute .activeitem was set to the last top-level entry. This lead to the opening of an entry, which was actually closed and attached to a child.
- Gnats 9413: The ISA Dialog Manager also displays insensitive scrollbars, which cannot be used because the object in question is insensitive. This is not the standard of Microsoft Windows, however this was implemented because we feel that this is a useful source of feedback for the user. An additional source of feedback is received by objects, which are directly implemented from the ISA Dialog Manager, whose scrolling arrows are displayed as insensitive when it is not possible to scroll further in any direction. This is true for the following objects: groupbox, layoutbox, notepage, scrollbar and tablefield.
- Gnats 9401: It was not possible to close an open poptext by using the `Escape` key. This was true when a certain accelerator was defined. In this special case, the window system (the poptext) should be favored over the IDM.
- Gnats 9396: When a popup menu was opened through an RTF edittext, the mouse pointer did not change into an arrow. This is an error of the Microsoft Windows RTF object. In the Dialog Manager this error is bypassed.
- Gnats 9375: When a userdefined attribute was set by a model of a control object, which set .mode to mode_server, the program crashed.
- Gnats 9371: An optional scrollbar is now displayed by the objects groupbox, layoutbox, notepage and window, when the virtual size is smaller than the size of the object, and when the scrollbar covers up a part of the virtual area in the other direction. Up to now, an optional scrollbar was only shown if the size of the object was smaller than the virtual size.
- Gnats 9369: The text of a static object is cut off after the 127th character. This is true when the static object is found inside the statusbar object. This is a limit set by Microsoft Windows.

  Note regarding the object "statusbar" under Microsoft Windows
  A statusbar can only display 256 text fields. For this reason, no more than 256 child objects (statictext) are allowed to be visible at the same time. Furthermore, a text field in a statusbar can only show a maximum of 127 characters.

Note regarding the "statictext" under Microsoft Windows
The text field in a statusbar can only show a maximum of 127 characters. This is why the text (.text attribute) can only contain up to 127 characters. This is true when the text field is the child of a "statusbar" object.

- Gnats 8862: When the size of an area within the splitbar has been changed, then the focus was set to the wrong object. This always happened when the original focus object did not lie within the father of the splitbox object. The behavior has now been adapted to the behavior of the tablefield object. Now, the focus is no longer moved to the splitbox object.
- Gnats 8735: In a treeview object it was not always possible to start a drag & drop action. The error always occurred, when .activeitem possessed an illegal value. For example, directly after setting .activeitem = 0, which is the only possible setting, when the treeview object is empty.
- Gnats 7266: Character/indication problems in the layoutbox object have been corrected.

13.2 Core

- Gnats 9447: When deleting entries in an associative array (via :delete-method) memory was accessed outside the allocated range which then leds to a program crash.
- Gnats 9448: Now no error is generated, when an invalid object ID is given as an argument for an external event.
- Gnats 8596: The attributes .startsel and .endsel were falsely calculated by numerical (scanf) formats. When the digits following the comma ended with two nulls, the nulls were not taken into account in the calculation even though these were contained in the attribute .content. The error was observed in the "%008'.6,2d" format.

14 Version A.04.02.i

14.1 Windows NT

- New: Up to now the OLE extension of the IDM only allowed events (also known as messages) without return parameters. Beginning with version A.04.04.a this restriction no longer exists. This has also been implemented in version A.04.02.i. Further details can be found in the release notes for A.04.04.a
- Gnats 9359: It was possible that a connected OLE control crashed. This came about through the correction of Gnats 9349, whereby the OLE control was activated, when the control object received the focus. This sometimes led to the OLE control transferring the focus further, which is actually not allowed. As a result, the OLE control got so mixed up that it resulted in a crash. Now, the activation of the OLE control is delayed, and because of this the error no longer occurs.
- Gnats 9349: The focus was not set correctly to an OLE control, which was connected through a control object with the attribute .mode = mode_client. As a result, navigation within the OLE controls via the keyboard was no longer possible.
- Gnats 9344: The attributes .startsel and .endsel of the edittext object are **no longer** set to the current mouse position with the paste event. The position of the mouse is saved in the attribute .index of the paste event.
  The attribute .index of the paste event contains the target area for the edittext object. This is:
  Keyboard operation
  - first(thisevent.index): is the same as this.startsel
  - second(thisevent.index): is the same as this.endsel

  Mouse operation
  - first(thisevent.index) and second(thisevent.index) correspond to the position of the mouse in the text, whereby 1 marks the position directly behind the first letter. Attention: second(thisevent.index) no longer indicates the line the mouse is sitting above.
- Gnats 9314: Child objects can be falsely positioned in windows with multiline menubars, when the size of the window was changed from a rule. This error happened in windows where ".yauto := 0;" and child objects where ".yauto := 0;" or ".yauto := -1;". The error also occurred, when the size of the window was newly calculated as a result of other actions that took place. In the previous Gnat entry, this error occurred when the width of a docked toolbar was changed.
- Gnats 9313: When maximizing a window that was bind at its top and bottom, child objects that were connected at the bottom were falsely positioned (flickering). This error is not reproducible. We believe it has been corrected through Gnats 9314.
- Gnats 9309: The selected entry of a poptext was changed, when a statusbar entry was changed. In order for this to happen, the user had to open the poptext with the mouse and then leave the mouse positioned above an entry. Then, when the user changed the selected entry with the cursor button, the selected entry sprang back, when a statusbar entry was changed in the select rule.

This no longer happens.

Attention
Such an "error" can return at any time. The reason for the springing back was a mouse-move event, which was caused by a change in the statusbar. Actually, this can always occur, when visible objects in the select rule of the poptext are manipulated. An additional mouse-move is necessary, for example, when the mouse cursor should be changed.

- Gnats 9307: The maximum allowed size for a window is now, which is normal for Microsoft Windows, the size of the display area plus the width of the frame. Through this, the behavior of the windows remains constant independent of the way in which the window is enlarged. A possible side effect of this is that a window, which is defined larger than the display area, could be displayed even larger than it was with one of the previous versions of the Dialog Manager.
- Gnats 9306: When a popup menu, which is defined on a groupbox or a notepage, was opened with the method :openpopup, then no select event was created. In this case the status text was not changed.
- Gnats 9187: An active OLE Control, which originated from a control object with the attribute .mode = mode_client, can now be exited through keyboard navigation.
- Corrected: When the mouse was positioned over the last field of a listbox object during a drag & drop operation, no message was given when the attribute .style did not possess the value style_ single.
- Corrected: When a popup menu was opened with the method :openpopup, then the internal WSI-ID's were not released, when the accompanying window was directly closed without closing the popup menu.

14.2 Core

- Gnats 9448: When an invalid object ID was given as an argument in an external event, this resulted in an assfail in object\slot 2519. This could happen, for example, when an object was deleted shortly before DM_QueueExtEvent was called.
- Gnats 9441: The data parameter is no longer changed by DM_SetValue calls on objects with :setrule.
- Gnats 9433: DM_GetVectorValue on a user-defined array now delivers the content of the field without a default value and subsequently not with any extra-uninitialized elements.
- Gnats 9353: The unnecessary setting of default objects is avoided when the model originates from a different module.
- Gnats 9350: The time needed for reloading modules (between unloading and start) has been reduced.
- Gnats 9343: The methods :findtext, :gettext and :replacetext only functioned for edittext objects under MS-Windows, whereby the attribute .options[opt_rtf] was set to true. The methods now function for all edittext objects. However, please take note that the parameter type can only possess the value content_rtf on systems that support RTF.

- Gnats 9293: The loading of mixed modules ascii/binary no longer confuses the internal import-stack of the interface parsers.
- Gnats 9273: Binary writing/Calling of the :index-method from associative arrays with menubox objects functions once again.
- Gnats 9271: The repeated release (detaching) of a text that was just destroyed through a repeated destruction in a recursive event loop, which was triggered by opening a popup menu, is now avoided.
- Gnats 9218: The employment of - use - in hierarchically inherited children is now correctly written (binary).
- Gnats 9213: Access to a superior layoutbox/toolbar via getvalue on .toolbar/.layoutbox is now possible.
- Gnats 9168. The option DMF_DirectCall when calling up DM_LoadDialog() is not supported (error in documentation). It is not possible to prevent the calling of :init-method with this. This call comes up later in DM_StartDialog().
- Gnats 9164: The DM_GetVectorValue call of the attribute .colvisible or .colsizeable no longer checks the first/last indices with .rowcount but rather with .colcount, which is the correct way to do this.
- Gnats 9011: The initial selection for filerequester can be provided.
  If the new boolean attribute .startsel (default is false) on the filerequester object is true, then the file/directory selection is initialized with the default value in .value attribute.

  Restrictions
  - Only one single selection is possible (.value must be scalar).
  - Under Windows NT it is only possible to initialize with a default value when loading and saving, but not for directories.
  - The initialized value in the attribute .value must contain the complete path and should correspond to the given directory (.directory attribute).
  - The initial selection appears in the entry field. However, a selection in the selection list does not take place. This problems lies on the system functionality of the selectionbox in Microsoft Windows.
- Corrected: When an object was deleted, which was marked in an external event, then the data type of the entry was set to *void*. This led to errors in the rule processing because the data types were no longer correct. Now the entry is simply set to the *null* object.

14.3 Editor

- Gnats 9351: Assfail by poptext start when .activeitem=0 (i.e. through .content setting) is now avoided.
- Gnats 9298: The changing of user-defined attributes sometimes led to a crash of the system (only in rare cases). This no longer happens.

15 Version A.04.02.h

15.1 Windows NT

- Gnats 9215: It is possible that an "Access Violation" in the ISA Dialog Manager application can occur. This is true when the applied C-runtime library is not equivalent to Service Pack 4 or higher. It was observed that an "Access Violation" in Microsoft debugger came about, however this was not the case when the application was started outside of the debugger.
  This problem is addressed in the Microsoft Knowledge Base Article Q225099. The "Access Violation" occurred, when a large number of smaller storage areas were present (> 16 MB) and these were then newly allocated. Thus, this problem only occurs with larger dialogs in the ISA Dialog Manager.
  This problem also affects your own applications, when these are linked with an old C-runtime library.
- Gnats 9167: There were OLE controls, which claimed the keyboard entry for themselves, as soon as they became active. This problem occurred only after Gnats 9064 was corrected. Now, an active OLE control is deactivated as soon as another (Dialog Manager) object is clicked on or, when it receives the focus. As a result, the keyboard entry is processed once again by the Dialog Manager.
  A crash sometimes came about when a linked OLE control was prematurely ended (crash). This is no longer the case.
- Gnats 9166: On a poptext where *.style = edittext* no *modified* event was sent, when the user entered something into the content and then pressed the `Return` key to confirm this change. As a result of this, the content was lost. This has now been corrected.
- Gnats 9208: On a poptext with .style = poptext, and which possesses a popup menu with accelerators, the accelerators were not triggered. The accelerators now also works in this situation.

15.2 Motif

- Gnats 9183: No more assfail due to lock overflow by toggling the sensitivity of visible objects with popup menus as children.
- Gnats 9227: When a text field is activated via the left mouse button, then a selection of the entries in the list, which is displayed when the mouse button is kept pressed down, should be possible. This behavior has been implemented.
- Corrected: The text positioning in a poptext under Motif 2.1 has been improved. They are displayed vertically "centered", as is the case under Motif 1.2.
- Gnats 9185: The poptext does not allow itself to be set to sensitive, when it was created as insensitive.

15.3 Java-Windows-Interface

- Gnats 9170: It was not possible to set colors in the Java interface by calling DM_Object.setValue. The result was always an "unconvertible types" error.
- Gnats 9169: The function with anyvalue-parameter was called, as it should have been.
- Gnats 9155: Crash of idmjava, when DM_Object.getVector was called up for an empty tablefield in the Java interface. Similar errors came about when DM_GetVectorValue failed on the dialog side (should be checkable via server-tracefile).

15.4 Core

- Gnats 9140: In the runtime version of IDM existing interface files are now correctly skipped.
- Gnats 9134: The value of the .itemorder attribute (from records, determines the sequence of attributes/child records) is now also saved in the binary file.
- Gnats 9149: Changed-queue has been made robust against recursive flushing/queuing.
- Gnats 9165: The attributes .picture[] and .picture_hilite[] from poptext are now correctly saved in the binary file.
- Gnats 9117: The name of the Tracefile/Logfile/Errwinfile can be called up through the attributes .tracefile, .logfile, .errfile on the setup object.

15.5 COBOL Interface

- Gnats 9211: Memory leak in DMcob_SetVectorValue has been eliminated. Overwritten strings are now released

16 Version A.04.02.g

16.1 Windows NT

- Gnats 9127: The series of events changed during the selection of an image object. The actual series of events is as follows:
  - button down: no event
  - button up: 1. focus 2. activate. If image is able to be toggled and button up is carried out on top of the image, then 3. select.

Caution
The series of events in the IDM are dependent upon the system!

16.2 Kernel

- Gnats 8870: The set events on call-back functions are now properly handled during the reading of binary dialogs having call-back functions.
- Gnats 9084: Dynamic settings on resources were partially not visible so that the object in question had to be switched back and forth from visible to invisible.

16.3 Motif

- Gnats 9124: The values of .startsel and .endsel are not always correct in edittexts and poptexts (in the edittext mode). The values are incorrect when the edit and or poptext in question was not active, i.e. when it did not have the focus.
- Gnats 9118: If an item was previously selected on a poptext, then the list of a different poptext was opened only after a second click of the mouse.
- Gnats 9129: When the attribute .format was set to a poptext, then the arrow on the right side of the object disappeared independent from the style of the poptext.

17 Version A.04.02.f

17.1 Motif

- Gnats 9093: The problem arose when the object edittext was used. When the focus is on the edittext and an accelerator is set off, the edittext sends 2 of the same events. Now, the "select"-event is only set off once.
- Gnats 9100: Select-event is now sent for the entry choice that was already selected. The index of the previously selected entry is in the select event within the attribute thisevent.value. Consequently, it can then be actively determined if the selection of the same entry takes place again.
- Gnats 9094: When a Mnemonic was entered into an edittext field, the focus was not only set to the new text field, but also the Mnemonic letter was erroneously entered into the previous edittext field.
- Gnats 9098: When the attribute .sensitive is set to false on a poptext, the poptext still remains active. This is only true for Motif 2.1. Under Motif 1.2 the insensitive setting works. Now, the insensitive setting also works under Motif 2.1 systems.
- Gnats 9099: The value of .activeitem in an empty poptext was 1 instead of 0.
- Gnats 9102: According to definition when the `Tab` key is pressed the cursor should leave the tablefield. This is not always the case. For example, when a field is edited and is exited by using the `Tab` key, the navigation no longer functions faultlessly.

17.2 Microsoft Windows NT

- Gnats 9087: Problems arose with regard to the representation of images. This was due to the fact that the GDI handles were not released and eventually the handles ran out all together.
- Gnats 9082: When the value of the .language attribute was changed the following error message appeared: [E: THING YIHANDLER CALLED FOR: text <identifier> (task 12)]. This is not a mistake, but rather a false error message.
- Gnats 9064 (A0401h): An OLE control that has been generated with an ATL-Wizard and which works with accelerators was not able to be correctly integrated into the IDM. Keyboard events did not make it to the correct windows. Now the embedded, displayed OLE object gets the first chance for treating the accelerators.
- Gnats 9017: When a top-level window with an activated position raster was partially moved outside of the viewable screen limits and then actively enlarged so that its size surpassed that of the workspace provided, then this sometimes resulted in a crash or the freezing of the program. Now an interactive enlargement beyond the workspace provided is no longer possible.

17.3 Kernel

- Gnats 9073: During the binary reading of text resources without labels, these were then once again correctly recorded in the text search list. The Assfail which appeared now and then with stop ()/destroy() of the text resource did not happen again.
- Gnats 9097: Dynamic settings on resources were partially no longer visible so that the object in question needed to be switched back and forth from visible to not visible. Now, changes on resources are once again propagated on the WSIs even when the resources themselves are not realised.

18 Version A.04.02.e

18.1 Motif

- Gnats 9010: A treeview is positioned at 0,0 even when the following is true: .xleft<>0 and .ytop<>0.
- Gnats 9009: When characters are entered into an edittext via the keyboard, neither charinput or key events were triggered.
- Gnats 9038: An error arose during the initialisation of the active entry of a poptext. When a non-visible poptext is given new values (for example through DM_SetContent), then this is not assumed in the displayed text.

18.2 Microsoft Windows NT

- Gnats 9030: The MDI no longer disappears after minimisation.

18.3 Java-Windows-Interface

- There arose an IllegalAccessException when the Java WSI was used as an applet.

18.4 Kernel

- Gnats 9029: The loading of binary modules with many objects (in the existing case thousands of text resources) in comparison to the older version (A.03.10.f old binary format) took an unreasonable amount of time. Now, the loading of modules (binary and ASCII) has been optimised and improvements in the existing customer dialog have been made.
- Gnats 8973: A format function that is found in a module, which is reloaded when the dialog is loaded, is now correctly tethered together and called up.
- Gnats 9066: No storage access violation when de-loading modules through early releases.

18.5 Editor

- Gnats 9055: The dialog is now correctly written for inherited objects which only (in the editor) contain newly added named rules.

18.6 Network

» Runtime data were not set on the network side. As a result it came to a crash as soon as an application function on the network side was called up.

18.7 COBOL

» Copy routes were generated incorrectly.

19 Version A.04.02.d

19.1 Microsoft Windows NT

It is now possible to open the pop-up menu of an object from the Rule Language with the new method ':openpopup()', provided that this new method is defined to the object in question. The method exists for the following objects: Canvas, Checkbox, Control, Edittext, Image, Layoutbox, Listbox, Notebook, Notepage, Poptext, Pushbutton, Radiobutton, Rectangle, Scrollbar, Spinbox, Splitbox, Statictext, Statusbar, Tablefield, Toolbar and Treeview.

- Gnats 8982: If an invisible pushbutton, upon which a font and a cursor were set, was made visible through .visible := true; then the set font was not displayed. Now, the button is first drawn after the font and cursor settings have been executed.
- Gnats 8955: GIFs are now correctly displayed under 256 color modes. It is possible that some color discrepancies can arise through mapping. The color dispersion is regulated so that most every image is displayed as good as possible.
- Gnats 8969: During the re-docking of a toolbar with an editable child object, i.e. a poptext or an edittext that possessed the focus, the child object lost its content and the focus.
- Gnats 8821: It is now possible to have longer strings in a Treeview (IDM-limited to 32759 characters; previously 512). If the string is longer than the allowed number of characters, then it will be shortened to the allowed number of characters, plus three dots to show that it has been shortened. Before, the entire string was ignored.
- The pop-up menu of the toolbar was only able to be opened by using the right button on the mouse, and not through the menu button that is available on most keyboards. This menu button is now supported for the toolbar.

19.2 Kernel

- Gnats 9012, 9013: The creation of more than 64K text resources free from Assfail.
- Gnats 8965: now parsepath() also returns, according to the module context, non-exported hierarchical children. The forbidden access to non-exported children from "outside" has also been corrected / is no longer possible. This is equivalent to a mapping output.
- Gnats 8977: Reading performance of binary data has been improved.
- Gnats 8596: Cursor position by %-format with positions after the comma and content strings with 0-characters at the end are now correctly determined.

19.3 Editor

- Gnats 8962: Faulty input by parsing :-methods no longer lead to the destruction of the father or to Assfail.

19.4 Motif

- Gnats 9037: When a poptext (or its parents) is set to .visible false, this will lead to a crash if the poptext has the focus and the poptext is the only object in his parent (i.e. Groupbox). This problem only arose under Motif 2.X.
- Gnats 8761: If the attributes .imagefgc and .imagebgc are set to an image, these will be ignored as they are with gif-images. Now the background of the image is portrayed in .imagebgc; the foreground in gif-images is still not portrayed.
- Gnats 8760: When loading certain gif-images IDM crashes with core dump. The reason for this is that during the decompressing of the images the maximum file size was exceeded.
- Gnats 8774: When an edittext was visible and this edittext was given a new content, then start/endsel were set to 0. Now, startsel/endsel is no longer set to 0 but rather remain unchanged.

19.5 Java-Window-Interface

- Gnats 9006: Attribute ".defbutton" was not implemented up to this point in Java WSI. The attribute is now directly presented on a Java class attribute. This means the default-pushbutton behaves in a Java-conformable manner. As a result, the default button is selected when the “Enter” button is activated, but it does not receive a focus. In addition, the default button is solely selected with the “Enter” button. In order to achieve a consistent behavior of Microsoft Windows it is recommended to switch on the Windows Look and Feel option. Unfortunately, this was faultily implemented in the actual Java runtime system (this does function in Java Runtime 1.4.0 BETA). A further consequence arises for one-line edittexts. Since the "Enter" button has no effect on the edittext, but rather on the default button, and because the default button does not obtain a focus, no edittext event is triggered when the “Enter” button is activated.
- Gnats 8978: Java Client does not handle dynamically (during runtime) created objects in a correct manner. These objects were not received in the child vector, whereby in the series of events particular attribute changes were not carried out as planned.
- Gnats 9004: Java Client recorded too many error messages in the console window.

19.6 Java-Interface

The Java Programming Interface has been added on to the presentation side (Client).

Short Tutorial

- The Java package "idmjava.iface" is required.
- For each Dialog Manager data type there is a Java class with the same name. Java class instances are set up and as a rule cannot be changed. Exceptions to this rule are the supplementary classes "DM_ValueBuffer" and "DM_VectorValueBuffer".
- Constants are defined in the mentioned classes and are spelled out in capital letters. For example, DM_Attribut.VISIBLE.
- Interface functions are defined as methods. Almost all important methods can be found in the class "DM_Object". This class replaces the C Data type "DM_ID", which cannot be found in Java.
- Errors are disclosed as "DM_Exception".
- A Java application on the presentation side is derived from the "DM_Display" class (see example below).

Example Canvas.dlg

```
dialog Canvas {}

function java ResetDrawing(object);
function java canvasfunc CanvasFunc();

window {
.title "Canvas Beispiel";
.width 400;
.height 300;

on close { exit(); }

pushbutton {
.text "ResetDrawing";
.xauto 0;

on select { ResetDrawing(C); }
}

canvas C {
.xauto 0;
.yauto 0;
.ytop 40;
.canvasfunc CanvasFunc;
}
}
```

Example Canvas.java

```
import idmjava.iface.*;

import java.awt.*;
import java.awt.event.*;
import java.util.*;
import javax.swing.*;

public class Canvas extends DM_Display
{
static private DM_FuncMap[] funcmap = {
new DM_FuncMap("CanvasFunc", CanvasFunc.class),
new DM_FuncMap("ResetDrawing", Canvas.class, "resetDrawing")
};

static public void main(String[]args) {
try {
new Canvas().bootStrap(args);
} catch(DM_Exception ex) {
ex.printStackTrace();
}
}

public void uiInit(String[] args) {
}

public void bindFunctions(DM_Object object) {
try {
object.bindFunctions(funcmap);
} catch(DM_Exception ex) {
ex.printStackTrace();
}
}

public void uiFinish() {
}

static public void resetDrawing(DM_Object canvas)
{
try {
CanvasFunc func = (CanvasFunc)
canvas.getToolkitData(DM_Attribute.CanvasData);
func.resetDrawing();
} catch(DM_Exception ex) {
ex.printStackTrace();
}
```

```
}

static public class CanvasFunc implements DM_CanvasFunc {
private Vector polygon = new Vector();
private JPanel component;

public void CCM_expose(DM_CanvasFunc.CanvasUserArgs cua,
Graphics graphics) {
ListIterator iter = polygon.listIterator();
try {
Point p1 = (Point) iter.next();
while (true) {
Point p2 = (Point) iter.next();
graphics.drawLine(p1.x, p1.y, p2.x, p2.y);
p1 = p2;
}
} catch (NoSuchElementException ex) {
}
}

public void CCM_resize(DM_CanvasFunc.CanvasUserArgs cua) {
}

public void CCM_reschg(DM_CanvasFunc.CanvasUserArgs cua) {
}

public void CCM_start(CanvasUserArgs cua) {
component = cua.getComponent();
component.addMouseListener (
new MouseAdapter() {
public void mouseClicked (java.awt.event.MouseEvent evt) {
if (component.isEnabled()) {
polygon.add(new Point(evt.getPoint()));
component().repaint();
}
}
}
);
}

public void CCM_stop(DM_CanvasFunc.CanvasUserArgs cua) {
}

public void resetDrawing() {
polygon = new Vector();
component.repaint();
```

```
}
}
}
```

19.7 Expansion of the Edittext under Windows NT/2000

The edittext contains an additional mode for supporting RTF. RTF support makes it possible to format texts just as it is done in Word. RTF formatting is activated on the edittext through the attribute *.options[opt_rtf] true*. All events and attributes, excluding a few exceptions, have the same meaning as those in non-RTF edittexts.

19.7.1 Special Features of the Attributes

The RTF edittext has the following special features in dealing with attributes:

- .startsel /.endsel show the position within the RTF text which conforms to MS-Windows without taking into consideration the formatting symbols. It is also not possible to directly use .startsel/.endsel. Therefor, it is also not possible to edit the content (.content) of the edittext.
- Although .format, .maxchars, .formatfunc is allowed, this unfortunately makes no sense with multiple-lined texts. Above all, the correlation between formatted texts and .content attribute is not possible due to RTF edittext which also deals with modifications. Similarly, the assignment of the cursor position in .content is also not possible.

19.7.2 Formatting

The possible formatting types for text areas are as follows:

Format	Enum-Value for :setformat or :getformat	Value Range
Font	text_font	string, font[1]
Font size	text_size	integer
Foreground color	text_fgc	integer, color[3]
Background color	text_bgc	integer, color[3]
Bold type	text_bold	boolean
Cursive	text_italic	boolean
Underlined	text_underline	boolean

[1]font/color only allowed with :setformat. :getformat does not return an object.

Format	Enum-Value for :setformat or :getformat	Value Range
Indentation left	text_indent_left	integer
Indentation right	text_indent_right	integer
Indentation of other text lines	text_indent_offset	integer
Text-Alignment	text_align	enum (align_left, align_right, align_center, align_justify)

Formatting can be set through the method :setformat(), and can be referenced through the method :getformat.

```
boolean :setformat
(
[integer Start,
integer End, ]
enum Type,
anyvalue Value
)

anyvalue :getformat
(
[integer Start,
integer End, ]
enum Type
)
```

As an option, it is possible to specify a text area with Start/End, upon which the format should be implemented. If this is not specified then the format will be set or called up in the cursor selection (the area between .startsel/.endsel). There are two types of formatting possibilities:

text_font, text_size, text_bgc, text_fgc, text_bold, text_italic and text_underline are character format options; this means that :setformat() actually affects the area that has just been selected (either through .startsel/.endsel or the specification of Start/End parameters).

text_ident_left, text_ident_right, text_ident_offset and text_align are paragraph format options; this means that :setformat() affects the paragraphs that have been touched through the selection (.startsel/.endsel or Start/End).

:setformat() returns the value true if the formatting was correctly carried out. Faulty arguments are signalized through a FAIL. Size or position specifications, as is customary with MS Windows programming, should be stated in “twips”. A “twip” is 1/20 of a point, or 1/1440 inch or in centimeter 1/567.

If :getformat() is carried out in an area which contains various formats of the type in question, then nothing will be returned. In addition, :getformat() does not function properly when the edittext is invisible (.visible = false). This is due to the fact that in this situation the MS-Windows RTF object must

perform a few calculations (formatting) which is usually only done in a visible state. Here, for example, it is not possible to call up the font information.

Special Features

The format sequence text_indent_right is an absolute specification from the right hand. The format sequence text_indent_left is a relative specification which can be used over and over again, but which never goes beyond the margin. The individual values are accumulated from RTF. Therefor, it is also possible to specify negative values (positive number => all lines within a paragraph will be shifted to the right - negative number => all lines within a paragraph will be shifted to the left).

The format sequence text_indent_offset is an absolute specification which shifts all lines, excluding the first line, of the paragraph in question to the right (if the value is positive) or to the left (if the value is negative).

The format sequence text_font expects in general a string containing the name of the font. If the font name is correct and the font is installed in the system, then one will see the effect of :setformat(). In the case that the font is not installed in the system, the modification in RTF-string will still be carried out (it is possible that an RTF-file is to be edited on another computer where the font is installed). The display remains unchanged, and no errors are reported. It is also possible to specify an IDM-font with text_font. In this case, :setformat() extracts the necessary information, such as font name and size and carries out the changes. The return value from :getformat() will always be a string which describes the name of the font.

In the font sequence text_fgc and text_bgc an integer, which describes the RGB value, is expected; i.e., 0xff0000 stands for red, 0x00ff00 stands for green and 0x0000ff stands for blue. Here, an IDM color can also be specified. The return value of :getformat() will always be in the form of an integer value.

19.7.3 Attribute .textwidth

	Rule Language	C	COBOL	Properties
Identifier	.textwidth	AT_textwidth	AT-textwidth	S/G/C/I
Data type	integer	DT_integer	DT-integer	

Value Range: integer

Default Value: 0

Classification: specific attribute

This attribute describes the width of an RTF text in pixel. This is necessary for the alignment of texts (left justified, centred, right justified). A value <=0 means that the width of the text is determined by the text itself. In an edittext without a horizontal scrollbar the width of the text is determined from the width of the edittext itself not including margins. Otherwise, it is determined from the widest line within the

text. A value >0 specifies the width of the text in twips. Scrollbars are made visible when needed. This is carried out with the help of the attribute .hsb_visible.

19.7.4 Text Operations

There are three methods that can be used for manipulating texts:

- To replace a text in RTF edittext with a plain or RTF text.
- Calling up of a text in a particular area of an RTF edittext (as a plain or an RTF text).
- Searching for a particular text in an RTF edittext.

These functions affect the content of the text area which was selected with .startsel/.endsel, when no optional area is specified.

19.7.4.1 Method replacetext

```
boolean :replacetext
(
[integer Start,
integer End, ]
[enum Type, ]
string Text
)
```

This method replaces the entire text or text area with the text given in the parameter. If the given text is an RTF text, then this should be printed with the Type-Parameter in order to bring in the correct formats.

Parameters

Parameter	Type	Value Range	Default Value	Description
Start/End	integer	0 ... #character without Format, -1 for the end	.startsel/ .endsel	Optional specification of the text area. Start should be before the end.
Type	enum	content_plain, content_rtf	content_ plain	Type of the given text parameter.
Text	string/text			Text that is to be copied into the text area.

Return Value

true	Ok: replacement was successful
false	Error: replacement of text area with new text was not possible

19.7.4.2 Method gettext

```
string :gettext
(
[integer Start,
integer End, ]
[enum Type]
)
```

This method delivers the text of the entire text content or a part of the text content. The result is either returned as a “plain” text without formatting or as an RTF text.

Parameters

Parameter	Type	Value Area	Default Value	Description
Start/End	integer	0 ... #character without format, -1 for the end	.startsel/ .endsel	Optional specification of the text area. Start should be before end.
Type	enum	content_plain, content_ rtf	content_plain	Text type in which the text area should be delivered.

Return Value

Text string of the given areas

19.7.4.3 Method findtext

```
integer :findtext
(
string Text,
[integer Start,
integer End, ]
[, boolean CaseSensitive]
)
```

This method searches for the given text string beginning with the selection up through the end of the text, or in the given text area. If the option CaseSensitive=false is given, then the search will be carried out independent of upper or lower case lettering.

Parameters

Parameter	Type	Value Area	Default Value	Description
Start/End	integer	0 ... #character without format, -1 for the end	.startsel/ .endsel	Optional specification of the text area. Start should be before end.

Parameter	Type	Value Area	Default Value	Description
Text	string			Plain text (no RTF) to be searched for.
CaseSensitive	boolean	false, true	true	Optional specification, if the search should be carried out according to upper and lower case lettering.

Return Value

-1	Text could not be found
>=0	Beginning position of the stings found within the text

19.7.5 Text Position

The RTF_Edittext expects/delivers text positions that are independent of the format. The consequence of this is that no directly understandable correlation is possible between .startsel/.endsel or Start/End specifications in the above mentioned methods and .content-String.

On the other hand, the specifications for .startsel/.endsel are consistent with the methods mentioned above.

Position specifications move similarly as they do with .startsel/.endsel from 0...#character, whereby #character stands for the number of characters without formatting (more exact: each character that can be included in a cursor area). Therefor, new lines also count as characters.

20 Version A.04.02.c

20.1 Expansion of the find Method

With the help of this function one is able to search for a specific value within the vector attributes. It is possible to search for the occurrence of a particular value in an array, in a two-dimensional field or in an associative array. Pre-defined as well as user-defined attributes are supported here.

The search can be limited through the specification of a Start/End-Index. It is also possible to locate Case-Insensitive during a search for strings, or after the first appearance as a prefix.

The search is only carried out in the following ranges: [1 ... n] or [1,1] ... [n,m];

0-indices are also not allowed. Since indices of associative arrays are not subject to any order, the index of such attributes is not allowed to be given as the index itself, but rather only as an index of an integer number in the range 1 ... .itemcount[<Attr>] The specification of an index range ([S_y,S_x] bis [E_y, E_x]) by two-dimensional fields (.content[,] on tablefield) affects the limitation of the search area to that of [MIN(S_y,E_y), MIN(S_x,E_x)] ... [MAX(S_y,E_y), MAX(S_x,E_x)]. Therefor, limiting the search to lines and columns is possible without any problems.

The direction of the search is determined according to the specified indices, or rather under the consideration of the direction (only for two-dimensional fields). For example, the given specification of 10,1 instead of 1,10 as the Start/End-Indices causes the search to be run in reverse order. A vertically aligned two-dimensional field is searched through in a lines/columns sequence. In the case of a horizontal alignment the search is carried out in an exactly opposite manner. The index of the first field that matches the value will be returned.

The method :find is called up with the following syntax:

```
anyvalue :find( { attribute Attribute, }
  anyvalue Value
  { , anyvalue StartIdx
{ , anyvalue EndIdx, }}
  { , boolean CaseSensitive }
  { , enum ComparisonForm } );
```

-> attribute Attribute

In this parameter the attribute is indicated, in which the value is to be searched for. If this parameter is not indicated, then for a Poptext/Spinbox from an attribute .text is assumed; concerning all other objects from the attribute .content is assumed. User-defined attributes are also allowed alongside pre-defined attributes.

-> anyvalue Value

After searching for the value in the indicated parameter, the value will be searched for in the vector attribute. If this value is a string or a text, then the parameter CaseSensitive and ComparisonForm will be pulled up for the search since the search is being carried out explicitly for a string. Other

types are also allowed. Principally, only those values will be compared whose types are either “same” or transportable (text is transportable in a string).

-> anyvalue StartIdx (optional)

In this parameter a start index is given to indicated where the search for a value should begin within a vector attribute. If no value is given, then 1 for one-dimensional and [1,1] for two-dimensional attributes will be assumed.

Return Value

0:	Element was not found (vector)
[0,0]:	Element was not found (2dim-field)
nothing:	Element was not found (ass. array)
Other:	Element index, in which the value being searched for exists.

Error Behavior

The calling up of the method will fail if the attribute is not known, or if the Start/End-Index is not within the valid range.

20.2 Motif

- Gnats 8911: After pushing the **right** button on the mouse over an object with a pop-up menu, the X-Server freezes up when the window is set up as insensitive.
- Gnats 8863: If an image object whose view is changed either through .name to employed tiles or directly through the setting of .picture to the file name, then only a portion of the new view will be shown in the same size as the previous view. In the case that the new view is smaller, this will then be shown in the upper left hand corner and no longer in the middle.
- Gnats 8914: After using the Poptext the X-Server freezes up.
- Gnats 8646: If the attribute .picture is not set on an image object, then it will have the wrong size after it has been created (independent of the size raster).
- Gnats 8787: Poptext creates charinput event when setting activeitem. Hence, no charinput event created when setting the activeitem.
- Gnats 8889: The scrollbar is not positioned in the middle when a window with *.dialogbox = true* is dynamically created.
- Gnats 8921: Image looses its width that was set in the dialog with 'sensitive false'.
- Gnats 8887: If one defines `Shift` + `Tab` as an accelerator on a pushbutton, then this will not be set off when this is used.

- Gnats 8926: If the height and width of an image object is set (during its creation), then an image will be assigned. If one then sets the width and height to 0, the object does not re-adjust its size as expected, but rather ends up as the "minimal size".
- Gnats 8937: When executing the IDM-Debugger, an ASSFAIL occurs.

20.3 Microsoft Windows NT

- Gnats 8772: If a graphical object without any set height or width (i.e. the height and width is automatically set by the object itself) is moved through a drag and drop action with the mouse, then the width and the height of the actual size values were assigned. Through the drag and drop action the object lost the characteristic of setting its own size.
- Gnats 8906, Gnats 8910: If an MDI-Child window was maximally created, which also had a maximal size of (.maxwidth>0), then an incorrect maximal size was calculated. This in turn lead to the incorrect size values for the children of this window.
- Gnats 8867: Transparent images in a checkbox style image are now correctly presented.
- Gnats 8859: If external events, that are sent to the IDM from COM-Controls, contain Interface-Pointer as a parameter, then it is now possible that these interfaces be used as sub-controls originating from the Rule Language (analogue to Interface-Pointers as parameter from COM-Control methods.)
- Gnats 8886: The background color of an MDI-Child Window can once again be set.
- Gnats 8638: From now on the handling area of a toolbar will disappear when all .dockable attributes are set to FALSE (i.e. not moveable).
- Gnats 8879: Transparent GIF89a's are now accepted by images in Checkbox-Style.
- Gnats 8860: The listbox should not flicker anymore with dynamic fillings in connection with .topitem = .itemcount.
- Gnats 8901: The height of the Poptext objects are no longer automatically set to 0.
- Gnats 8631: .sysmodal is now defined as follows: When .sysmodal = true, then a dialogbox is emphasised before all other applications. This means that the dialogbox is always in front of the other windows. The actual application, from which the dialogbox is started, is not selectable until this is closed. All other applications are selectable, but remain covered by the dialogbox. If .sysmodal = false, then this will behave like the window MB_TASKMODAL. This means the modality of the dialogbox is limited to that of the father window. All other applications can be laid over the dialogbox. The father window is not selectable.
- Gnats 8902: The height of the Poptext entry fits itself to the selected font size and type. This is also true with regard to dynamical changes in the font and font size.
- Gnats 8767: At the Poptext a new behavior for select-events has been defined as follows: with every change made in the edit area of a Poptext, a select-event will be triggered.
- Gnats 8724: Depending upon the size, the correct Icon variation from an Icon resource is used.

- Gnats 8777: In a Poptext the .activeitem is once again set correctly after editing.
- Gnats 8918: When .width = .height = 0, then the size of the tiles will be calculated automatically.

20.4 Java

- Gnats 8899: The option -IDMjavaserve <port-nr> did not function correctly. The Server crashed due to a memory access violation.
- Gnats 8854: After the contents of a tablefield have been changed in size (.rowcount and/or .col-count), the newly added columns and lines could not be edited.
- Gnats 8852: EM_activate and EM_deactivate events for MDI windows sent when a window was created, or when a setting to the .active attribute took place.
- Gnats 8762: Size and position calculations with Poptexts are now handled in the same way as with other simple objects (edittext, statictext, etc.).

20.5 Kernel

- Gnats 8907: Rules to imported objects are written out binary files according to the texts.
- Gnats 8890: During the resolving of paths of user-defined attributes of strings or index-type strings the resulting string is copied.
- Gnats 8888: By repeated calls to DM_EventLoop(DMF_DontWait) the temporarily created thi-sevent-Objects were not deleted correctly. The result of this was an overflow of object Ids in mod-ule "idm".
- If an edittext was assigned a format resource with a self-defined format function, and directly fol-lowing also a format string, then problems arose.
- Gnats 8868: Expansion of :find() method – here all indexed attributes (i.e. including .content[,] in the tablefield) as well as user-defined attributes are supported. It is also now possible to search for any kind of value and not just strings.
- Gnats 8924: Problems with the format.
- Gnats 8928, Gnats 8929: :index method at poptext now properly functions for all attributes and not only for options.

20.6 Debugger

- Gnats 8892: Up to now it was not possible to set a breakpoint in the IDM-Debugger after the return of the first rule. This was true even when the rule contained other codes and returns.

20.7 MICRO FOCUS COBOL

- Gnats 8913: Memory allocation for output parameter from string types of COBOL functions are now correctly allocated and released.
- Call to cobinit for AIX has been supplemented.

21 Version A.04.02.b

21.1 Motif

- Gnats 8744: After selecting the **right** mouse button on insensitive objects with popup menu, the X-server freezes.
- Gnats 8661: If *.dialogbox := true*, child objects are shown in the wrong order.
- Gnats 8785: Focus handling of poptext does not work when menu is opened.
- Gnats 8786: When poptext (style edittext) is opened, too many events are created. When poptext is opened with `Ctrl` + `Cursor Down`, these events don't appear anymore.
- Gnats 8788: .activeitem can not be set at poptext with format.
- Gnats 8792 When a poptext was opened the first time, a focus event was created on closing it.
- Gnats 8793: If a second poptext has been opened with the mouse without closing the first before, the second one cannot be operated using keyboard commands. The behavior has been changed, so that it is impossible now to leave an open poptext without closing it.
- Gnats 8804: If the poptext (.style:=poptext), is opened the first time with the `Space` key , this poptext can't be controlled by using keyboard commands.
- Gnats 8855: If the menu *"Options"* → *"Configuration…"* in IDMED2 was chosen, an ASSFAIL occurred.
- Gnats 8843: On loading a dialog, in which a window is defined as dialogbox, the IDM crashed.
- Gnats 8754: If the *.picture* attribute of an image was a directly assigned file and not a tile resource, and later the same attribute has got assigned *null*, an assfail occured. Now a correct empty image is shown.
- Gnats 8668: If .topitem in a listbox is placed on the last element of the list, this element vanishes in the shown part of the list.

21.2 Windows NT

- A tile with a bitmap, which was up to 5 pixels smaller (without frame display) than its image object, was not centered correctly in the image.
- Gnats 8844: The problem is actually an error of Word 2000. According to OLE specification a control must not crash, if it is asked for an interface. But in this case it happened if Word 2000 was asked for hr=pCPC->FindConnectionPoint (IID_IPropertyNotifySink, &pCP);. Currently Word 2000 abstains from establishing the IpropertyNotifySink property. The connection comes about, but IDM does not get informed any more about changes!
- The OLE option of IDM now needs an additional system library, the "advapi32.dll". This library now has to be linked to an application when binding!

- Gnats 8397: Certain GIF87a pictures couldn't be read by IDM Win32. The gifreader has been changed so that it is also possible now to read the extensionblock of a gif.
- Gnats 8845: If the DDM server was started with the option -IDMserve, the newly created server process didn't shut down when the net connection to a client was lost. Now this happens after the KeepAliveTime ,which is given from the operating system, runs out. (The default under WinNT4 is about 2 hours.)
- Gnats 8851: The dragging of Drag&Drop of the image object works again now.
- Gnats 8775: If the active entry at the treeview has been removed with :delete, an error message was sent from Windows WSI, so that the the setting on .activeitem could not be done.
- Gnats 8827: When an insensitive scrollbar has been switched visible, it was displayed as sensitive. Now it is displayed as insensitive.
- Gnats 8731: Non-scalable tiles in image objects, which were smaller than the tile, were not shown correctly centered, but rather set off from the center by two pixels in x- and y-direction each.
- Gnats 8754: If the .picture-attribute of an image was a directly assigned file and not a tile resource, and later the same attribute has got assigned *null*, an assfail occured. Now a correct empty image is shown.
- Gnats 8837: When an image, which was not given a size and that already displayed a picture, got a new scalable tile assigned, the image did not calculate its size once more according to the original size of the scalable tile, but rather it scaled the tile to the size of the picture that was shown before. Now the size will be calculated new the correct way.
- If a rule dynamically assigns an image a tile resource other than .picture, it generates an error output [WS didn't process...] in the tracefile.
- In the case of an image with checkbox style, that had no size assignment, the size was calculated by the picture that was active at the moment it became visible. Now, the size of the default picture is always used for the calculating of the image object.
- Gnats 8809: Newer Windows versions and/or service packs are sending still messages during the destruction of a windows object: As a resaul, a focus was set on a already partly destroyed IDM object. The internal access on the recorded, but no longer existing focusobject created an assfail. Now the focus treatment will not be done before the final object destruction.

  Note
  This can change the order of some events at the invisible switching of objects!
- Gnats 8621: Horizontal spinbox buttons are 50% wider now, so that with windows standard settings the arrows can be seen clearly.
- The layoutbox arranges spinboxes without overlapping as well now.
- Gnats 8462: dbselect event for the objects window, groupbox and layoutbox. In thisevent.x/y the coordinates of the mouse cursor at the moment of double click are used.
- Gnats 5578: From now on it is possible, to display GIF89a's transparent as well.

21.3 Java

- Gnats 8875: MDI child windows haven't been activated on initialization.
- The grid calculation out of a referencefont (.reffont) has been changed. The length of the referencestring will be calculated internally and divided by the number of the signs. Therefore grid size will be a little smaller and with that it corresponds more to the size in other WSIs (Windows and Motif). For that reason, porting existing custom dialogs to JAVAWSI will be easier.
- Fonts now support referencestring.
- Gnats 8832: If a dialog with tablefields has been executed as applet in Internet Explorer (VM from Microsoft), the tablefields were not displayed.
- Gnats 8833: If menus were changed dynamically (some items added, .visible switched, and so on), the separators were put to the wrong place. Encoding in Java-WSI: In the encoding class the following variables exist: public Socket socket - which contains the data for the opened socket and public String message – which is an error message. These are outwardly visible and must be set. public String message has to be set at least with a blank.

21.4 Core

- Gnats 8857: Objectleak at the call up of the eventloop has been removed. At every call up of the eventloop a thisevent object was left behind in the storage. Now this object will be destroyed before the loop will be left.
- Gnats 8790: Assfail at the setting of .format at the tablefield with DM_SetVectorValue(), in which a formatfunction is used at the tablefield. Tip for IDM programmers: Attributes such as .formatfunc are actually no longer necessary. The formatfunction can better be set in formatresources and the formatresource than be set at the object in .format. This is not only safer, but also more efficient.
- Gnats 8822: When resolving user defined attributes, 64-bit string pointer are saved now in the correct way.
- Gnats 8567: Assfail at the replacement of invalid object IDs will be avoided now.
- Gnats 8840: .dockable[]-attribute will also be written in the binary file (new binary format).
- Gnats 8856: No assfail any more at the reading of the binary interfaces of modules, which export hierarchical, inherited child records.
- Gnats 8846: Model building with the .model attribute will be copied now correctly in the new binary format. (Children will not be inherited, just attributes).
- Gnats 8858: The nlscat number will be saved now in the new binary format.
- Gnats 8830: Has been solved with Gnats 8754.

21.5 Debugger

» Gnats 8576: Imported modules will be loaded into the list of the named rules within the debugger, even if the rules of the module that the loaded module is importing, have been loaded into the list long before. The error message *“Invalid argument 'r0' (neither line nor rule)”* won't be displayed within the debugger any more if a rule belonging to the newly loaded module is displayed and executed.

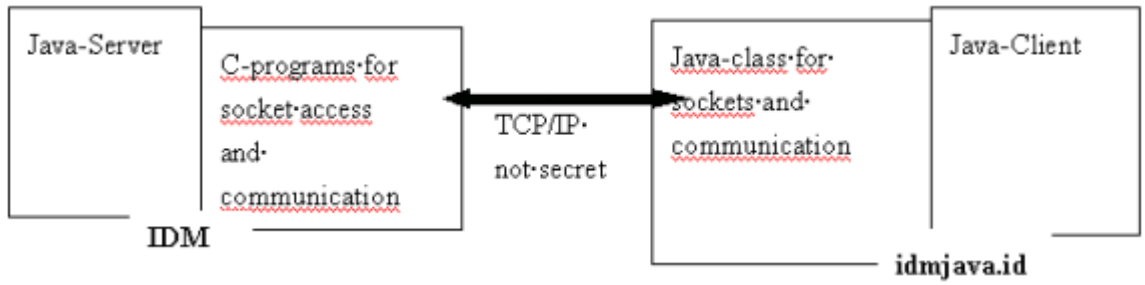
22 Version A.04.02.a

22.1 Encryption in JAVA WSI

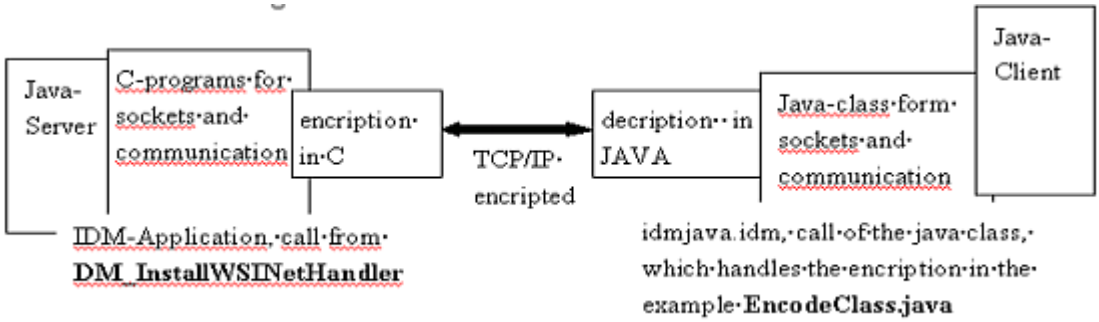
Within the Server-Client architecture of the Java version of IDM, there arises the requirement for encryption of the communication between server and client. Due to the fact that communication between the two parts can take place through an Internet connection, encryption is extremely important in keeping non-discloseable information, i.e. passwords, scrambled so that no unauthorized individuals can access this information.

In the sub-directory *javassl* there is an example of an IDM application which uses communication encryption software.

The IDM Java version consists of a Java server and a Java client. The communication between these two components is carried out through a TCP/IP connection (without encryption). See following illustration.



In the example **javassl** encryption software is switched on from the server side as well as the client side. See following illustration.



The software C/SSL v2.1 and J/SSL v2.0 from Baltimore Technologies was used in the example showing how encryption can be built into a program. For further information see **http://www.baltimore.com/products/jssl/sslintro.html**

C/SSL-API is written in C and is used for communication on Java-WSI server sides. J/SSL-API is available in Java and is used for communication on Java-WSI client sides.

Both software packages correspond to version 3.0 from SSL and offer protection against a PKCS#1 attack.

The software was tested on a UNIX, Sparc Solaris 2.6 operating system.

22.1.1 Functionality of SSL

SSL was developed from Netscape.

SSL (Secure Sockets Layer) is a communication system which guarantees secrecy when communicating with other SSL-capable products. SSL is a protocol which runs above TCP/IP protocols and under top-level protocols such as HTTP. In order to examine and confirm the identity of the communication partner asymmetric encryption and certificates are used. Symmetric encryption is used for the actual communication. An SSL connection can only occur between an SSL-capable client and a SSL-capable server.

A more exact description of the data exchange through SSL can be found under **http://developer-.netscape.com/tech/security/ssl/howitworks.html**.

The following is a description of the steps needed to be taken in order to create an application with the IDM which uses encryption software within the communication of the Java interface within IDM. Software from Baltimore Technologies (C/SSL and J/SSL) was used to test the results.

22.1.2 Encryption on the Server-Side

In order to use the encryption software one must first create an application with the IDM. Within the **AppMain** function, the function **DM_InstallWSINetHandler**, which is used for registering user-defined functions, must first be called up.

In addition, user-defined functions within the application must be defined, which in turn call up the necessary encryption routines. The resulting value and parameter of the user-defined functions are pre-determined.

22.1.3 The Interface Function DM_InstallWSINetHandler

DM_InstallWSINetHandler

This DM-interface function registers the user-defined functions that are called up within the encryption software. This function must be called up in **AppMain** before **DM_Initialize** is called up.

The resulting value and parameter of the user-defined functions are pre-determined.

Syntax

```
DM_Boolean DML_default DM_EXPORT DM_InstallWSINetHandler
(
  DM_WSINetFunctions *wsinetfunctions,
  DM_Uint Operation,
  DM_Options Options
)
```

-> DM_WSINetFunctions *wsinetfunctions

Structure, that the function pointer has on the user-defined functions. This has the following form:

```
DM_WSINetFunctions
{
  DM_AcceptProc,
  DM_SessionProc,
  DM_ShutDownProc,
  DM_OpenProc,
  DM_CloseProc,
  DM_SendProc,
  DM_ExistsMessageProc,
  DM_RecvProc,
  DM_FreeWarningProc
}
```

The specific function types have the following form:

```
int <name of DM_AcceptProc function>
  (int serverfd,void *cliaddr,void *addrlen,char *message)

void * <name of DM_SessionProc function>
  (int clientfd,void *support,char *message)

void<name of DM_ShutDownProc function>()

int<name of DM_OpenProc function
  (int *port,void **supportptr,char *message,char **warning)

int <name of DM_CloseProc function>
  (void *connptr,int *clientfd,char *message)

int <name of DM_SendProc function>
  (void *connptr,char *buffer,int length,char message)

int<name of DM_ExistsMessageProc function>
  (void *connptr,char *message)

int <name of DM_Recv function>
  (void *connptr,char *buffer,int count,char *message)

void <name of DM_FreeWarningProc function>
  (char *warning)
```

-> DM_Uint Operation

One of the two pre-defined constants:

1. *DMF_RegisterHandler* for registering the user-defined functions.
2. *DMF_WithdrawHandler* for de-registering the user-defined functions.

-> DM_Options Options

Is not used and should be preset to 0.

Return Value

TRUE	For *DMF_RegisterHandler*: functions were able to be registered. For *DMF_WithdrawHandler*: return value is always =TRUE.
FALSE	For *DMF_RegisterHandler*: functions were not able to be registered.

22.1.4 User Defined Functions

Description of the specific functions:

- **DM_AcceptProc**
  Accepts a connection to a client.
- **DM_SessionProc**
  Returns the descriptor of the client as a void * return value.
  Sets the parameter for the client session.
- **DM_ShutDownProc**
  Initiates termination actions when disconnecting the connection.
- **DM_OpenProc**
  Configures and opens the socket.
- **DM_CloseProc**
  Terminates a connection.
- **DM_SendProc**
  Sends a message to the client.
- **DM_ExistsMessage**
  Checks if the client has sent a message.
- **DM_RecvProc**
  Reads messages that were sent from the client.
- **DM_FreeWarningProc**
  Frees up storage area which was previously allocated for the parameter *warning* of **DM_OpenProc**.

The normal sequence of the call is:

1. DM_OpenProc function
2. DM_FreeWarningProc function
3. DM_AcceptProc function

4. DM_SessionProc function
5. DM_SendProc function, DM_Recv function, DM_ExistsMessageProc function in variation
6. DM_CloseProc function
7. DM_ShutDownProc function

In the case that your own encryption is to be carried out with the help of the user-defined functions, the pre-given sequence of calls from the IDM must be considered when using the parameter for input and output.

22.1.5 Example

In **encode.c** within the *javassl* directory an IDM application for encryption on the server side can be found. An SSL socket is constructed on the Java server side. The communication is encrypted with the help of the software packets C/SSL and J/SSL from Baltimore Technologies.

22.1.5.1 Encryption on the Client-Side

Encryption on the side of the client is carried out through a Java class which creates a socket with encrypted communication.

The class name can be freely chosen and can be transferred through a parameter when starting the IDM from the client side.

The name of the method used for creating an encrypted socket must be **CreateSocket** with the following Syntax:

```
public void CreateSocket (String host, int port, boolean isapplet, URL url,
String jsslpath, String certificate)
```

Parameters

-> host

Name of the host

-> port

Number of the port

-> isapplet

TRUE, when the client is started as an applet.

FALSE, when the client is started as an application.

-> url

URL of the applet.

-> jsslpath

String, which is given through the parameter idm.jsslpath.

-> certificate

String, which is given through the parameter idm.certificate.

In order to create an IDM application with your own encryption class, the following steps must be taken:

- Compile your own classes with Java.
- From the jar file which is delivered with the IDM, extract the classes with the command `jar -xf idm.jar`. In doing so an *idmjava* directory will be set up in which classes are saved, and a *META-INF* directory will be created in which the Manifest file is saved.
- Pack the compiled classes and the extracted classes from **idm.jar** together into a jar file with the command `jar -cMf <Name of the new jar-files> <Name of your own Class> idmjava/*`.
- When starting the client with Java, the package and the name of your own class must be given with the help of *idm.encode*. The parameters *idm.jsslpath* and *idm.certificate* are able to transfer strings to the method **CreateSocket**. See chapter “Starting the Example for Encryption”.

22.1.5.2 Example

idmuser/EncodeClass.java

This class is an example for creating an SSL socket on the client side with the help of the software (J/SSL) from Baltimore Technologies. This can be found in the *javassl/idmuser* directory. This class is given with the help of the parameter *idm.encode* when starting the Client of the IDM application (see below). The name of the package and the name of the class are not fixed. However, the encryption class alongside the constructor for creating a socket must contain a method with the name CreateSocket.

In the example the program code **Client Authentication**, i.e. the attestation of the client, is contained. This part of the example can be used, but is has of yet not been tested.

22.1.5.2.1 Creating the Example for Encryption

In order to create the example of the IDM application, which consists of **encode.c** and **idmuser-/EncodeClass.java**, a makefile **Makefile** under UNIX is available in the *javassl* directory. For creating an IDM application under Windows the makefile must be adjusted accordingly.

Procedure

1. Adjusting the makefile variables
   - CSSL_PATH
     Path for the directory in which the libraries for the encryption software for the server are stored, i.e. */cssl/debug* or */cssl/release*.
   - CSSL_PATHDATA

Path for the directory in which the certificates are stored and in which data can be temporarily stored, i.e. with editing authority */cssl/data*.

- CSSL_INCLUDE
  Path for the directory in which the Include files are found.
- JAVA_HOME
  Path for the Home directory for the Java library.
- SWING_HOME
  Path for the Home directory for the Swing library.
- JSSL_HOME
  Path for the directory in which the encryption software for the client is stored, i.e. */jssl*.
- CLASS_NAME
  Package and name of the Java class in which the encryption is realized on the side of the Client, i.e. **idmuser/EncodeClass**.

2. IDM_HOMEDIR
   The file **IDM_HOMEDIR** in the previous directory must contain the correct path for the IDM Home directory.
3. Create with Make
   With this command an executable client is created.
   The classes are extracted for the server from the delivered Jar file for the IDM. These own classes – in the example **EncodeClass.java** – are compiled and all classes including the own classes are packed together in a new Jar file **idm_encode.jar**.
   Prerequisite: the directory in which this creation process is carried out must have “write authority”, because a sub-directory named *idmjava* and a sub-directory named *META-INF* have been set up.

22.1.5.2.1.1 Creating Certificates

In order to start the created example it is necessary to have a certificate for the server. For a commercial application it is absolutely necessary to apply for a server certificate with the certification authorities.

Solely for test purposes, keys and certificates can be locally created with the correct software. The UNIX-Shell-Script makecerts.sh or the script makecerts.bat for Windows, which is contained in the C/SSL-Software from Baltimore Technologies, can be used to accomplish this.

22.1.5.3 Starting the Example for Encryption

1. Starting the server
   ```
   cd javassl
   encode <Options> <Dialogfile>
   ```
   The server is started with this command. The options are the same as when starting the Java server with the command `idmjava`.
   For testing purposes the **IDM_HOMEDIR/lib/IDM/bestelldienst.idm** can be used.
2. Starting the Java clients as an application

```
cd demos/javassl
java –classpath ./idm_encode.jar:<JAVA_HOME>:<SWING_HOME>:<JSSL_
HOME>/applet/jssl.jar:<JSSL_HOME>/applet/jcrypto.jar
  <Options>
  -Didm.encode=<Package . Name of the encryption class>
  -Didm.jsslpath <Path for certificate>
  -Didm.certificate <Name of the certificate>
idmjava.idm
```

With this command the client – the class **idmjava.idm** - is started.
JAVA_HOME, SWING_HOME, JSSL_HOME have the same meaning as the variables in the makefile Makefile for creating an example, see above.
The options <*Options*> are the same as when starting the Dialog Manager Java client.
The option

- **idm.encode** states the class path which finishes the encryption. If the class is not found, a non-encrypted socket is created. Example for this parameter is `idm.encode=“idmuser.EncodeClass“`.
- **idm.jsslpath** states the directory path in which the certificates and other temporarily stored data from the encryption software is found. Example /*cssl/data*.

  Attention
  This path must point to the same directory as CSSL_PATHDATA in makefile **Makefile**.
- **idm.certificate** states the name of the Certificate files, example **caCerts.pem**.

When communication between client and server is encrypted, the following message appears on the screen after client has been started: *“Encoded Connection”*.

3. Starting the Java client as an applet
   The following files must be found on the Web server:
   - HTML-page, i.e. **idmapplet.html** (with changes, see below)
   - **caCerts.pem** from C/SSL from Baltimore Technologies, or other identity files
   - **idm_encode.jar**
   - **jcrypto.jar** from the sub-directory *applet* from Baltimore Technologies software J/SSL
   - **jssl.jar** from the subdirectory *applet* from Baltimore Technologies software J/SSL

   The <APPLET>-tag in the HTML page looks exactly the same as the Dialog Manager Java client not including the following exceptions:
   ```
   ARCHIVE=idm_encode.jar; jcrypto.jar; jssl.jar
   <PARAM NAME =“idm.encode“ VALUE=“Package.Name of the encryption class“>
   <PARAM NAME =“idm.jsslpath“ VALUE=“Path for Certificate“>
   <PARAM NAME =“idm.certificate“ VALUE=“Name of the Certificate“>
   ```
   The meaning of the parameter is the same as when starting the client as an application.
4. Command for starting the server and client as an application.
   To start the Java server and the Java client with encrypted communication, one should use the `make run` command.

To start the Java server and the Java client with normal communication, i.e. without encryption, one should use the `make normal` command.

22.2 Changes with Object Referencing

This chapter deals with the relationship between objects and their identifiers, as well as with the referencing of objects.

22.2.1 Hierarchy

The IDM manages objects of a module in a hierarchical manner. Here, the most important aspect is the data which is defined in the module. From the imported modules standpoint, the imported objects are arranged at a lower level (hierarchically speaking) than the imports.

Example: Following Module

```
module M                          => Hierarchy
export model pushbutton MPb {}       M
export window Wi {                   +-MPb
  export MPb Pb {}                   +-Wi
  MPb Pb2 {}                           +-Pb
}                                      +-Pb2
```

is used from a dialog. The hierarchy of the accessible objects is listed to the right.

```
dialog D          => Hierarchy
import I “m.if”;  D
window Wi2        +-I
{                 | +-MPb
  MPb Pb {}       | +-Wi
}                 |   +- Pb
                  +-Wi2
                    +-Pb
```

22.2.2 Identifiers

Each object in a module possesses a symbolic name, otherwise referred to as a identifier. The identifier is either **set** by the user or it is inherited from a model (exception see chapter “Inherited Identifiers and Modularization”).

In the same hierarchy level (objects having the same father) objects having the same identifier can exist, but only one is allowed to be set (this is determined by IDM). See chapters “References to Problems and Problem Areas in Previous Versions” and “Known Errors / Problems within IDM” for exceptions, problems and errors.

The difference between a set and an inherited identifier is the referencing ability. An inherited identifier must be referenced through a **path**. On the other hand, a definite set identifier can only be referenced through the identifier itself.

```
dialog D
model pushbutton MPb {}  // label set, belongs to model
window Wi {
  child MPb Pb1 {}       // label set
  child MPb {}          // label inherited from model
}
window Wi2 {
  child pushbutton Pb1 {} // label set, name Pb1 not longer unique
  child MPb Pb2 {}        // label set
child MPb Pb2 {}      // not possible, because there is an other child
                        // with the same name
  child MPb {}
}
on dialog start {
  MPb.text:="Ok";       // label unique, result is the model
  Wi.MPb.text:="Okay";  // access with a path
  Pb1.text:="Exit";     // Pb1 not unique, result is an error
  Wi.Pb1.text:="Exit";  // access with path
  Wi2.Pb1.text:="Beenden";
  Wi2.Pb2.text:"Start";// access to first pushbutton Pb1
  Wi2.MPb.text:"Stop"; // access to pushbutton.
  Wi2.MPb:[2].text:"Aus"; // access to second object with the same name
  exit();
}
```

Not only objects, but also user-defined attributes are named through a identifier. Please keep in mind that an attribute can not be referenced, however only its context can be accessed. The rule that within a hierarchy level a identifier can only be set once does not only include child objects, but attributes and methods as well!

```
dialog D
window Wi {
  integer Pb := 123;
  // pushbutton Pb {} // Error – label exists at Wi"
window Wi2 {
  pushbutton Pb {} // No ambiguity with attribute Pb at Wi
}
on dialog start {
  print Pb;   // label unique, object pushbutton Pb
  print Wi.Pb; // value of attribute Pb
  print Wi2.Pb; // access to child object
  exit();
}
```

The IDM modularization concept only limits the referencing of objects from outside, and not the access of named attributes and methods. Due to this attributes can not/ must not be exported.

22.2.3 Clarity and Ambiguity

The module identifier is always clear. Ambiguity arises only when various objects within the hierarchy (according to “Hierarchy” including imported objects) are set to the same identifier.

```
module M                                // M is the label of the module
export function string Func(integer);  // label unique in the module
export window Wi {                      // Wi unique
  export pushbutton Pb {}
}
export window Wi2 {                     // Wi2 unique
 export pushbutton Pb {}                // Pb not unique
 window Wi2 {}                          // Wi2 not unique inside module
}
on module start {
  print Func(987);
  M.Wi2.Wi2 := “Wi2 privat”;            // access to Wi2 only with path
}
```

If ambiguity exists, the ambiguous identifiers can be referenced through explicit paths. This must happen in the following example for the pushbutton Pb in the window Wi and Wi2 and WiMain.

```
dialog D                               // D always unique, it is the dialog
import I “m.if”;                       // I unique
export function integer Func(string); // Func not unique
window WiMain {                        // WiMain unique
  pushbutton Pb {}                // Pb not unique(first .Pb from import)
}
on dialog start {
  print D.Func(“Hello”);            // access to Func with path
  print I.Func(123);
  WiMain.Pb:=”Ok”;                 // access to pushbutton with path
  Wi.title:=”Wi”;                  // access to Wi
  Wi.Pb:=”Ok”;
  Wi2:=”Wi2”;

// Pb.text := “PB”;
// Error: Label not unique
  exit();
}
```

22.2.4 Referencing and Referencing Errors

Within a module, i.e. in rules or with static instancing, it is possible to access named objects through their **identifier** or through a **path**. In this case, only objects within the module or imported objects are able to be referenced.

A path is actually a chain of identifiers beginning with an explicit identifier (i.e. through the module name). If a identifier is used more than once within a hierarchy level, then the access to objects with inherited identifiers is possible with the help of a suffix in the form of “:[I]”, whereby 1<I<A (A = number

of objects with the same identifier & father). Principally, a clearly **identifiable** object can be referenced with a path.

Paths are resolved at the end of module loading transactions. If the identifier is not correct, the user will receive an error message such as *“Cannot Resolve…”*.

If an identifier, or the first identifier in a path is ambiguous, the user will receive a message regarding this, for example *“Cannot Resolve … Label is not unique”*.

Such error messages demand that the identifier be corrected accordingly, or that the ambiguity be resolved through an unambiguous path.

When writing dialogs/modules, i.e. from within the editor or in binary format, the smallest unambiguous path is always written out/used.

Alongside these absolute paths it is also possible to state relative paths which for example emanate from this-object a local variable or a parameter. The resolving of the paths can only take place during the time of execution. Therefor, no error messages or warnings for such path specifications can be sent during the loading process. This is true because some of these are only valid during runtime.

```
dialog D
model pushbutton MPb {}
window Wi {
  pushbutton Pb {}
  MPb Pb2 {}
}
window Wi2 {
  MPb Pb2 {}
  MPb {}              // inherits the label from model
  MPb {
     record R1 {}
  }
  checkbox {}         // inherits the name of the default CHECKBOX
}
on dialog start {
  Wi.title:=”Wi”;     // access with unique label
  D.Wi.title:=”WI”;   // access with path possible
  D.WI.title:=”WI”;   // Error: label not unique
  this.Wi.title:=”wi”;// relative path, validation at runtime
  this.WI.title:=”WI”;// relative path, validation at runtime,
                      // result: error
  Wi.Pb2.text:=”Ok1”; // path necessary, lable not unique
  D.Wi2.Pb2.text:=”Ok2”; // path ok
  Wi2.MPb.text:=”Ok1”;
  exit();
}
```

22.2.5 Inherited Identifiers and Modularization

When exported objects possess an inherited identifier, then this heredity is normally carried outwards within the importing module. Concretely, this means than an exported object with an inherited identifier can only be referenced through a path.

Now, the outward bound heredity of the identifier can be stopped since the model of the exported objects is likewise not exported from the module. As a result, such an exported child features a referencing which in turn leads to explicitness. This is true because both model and child have the same identifier, however at the same time they reference different objects.

This constellation can be seen in the following module with model MPb and its instances as hierarchical children.

```
module M
export model pushbutton MPb {}
export model radiobutton MRb {}
model checkbox MCb {}
export model groupbox MGb {
  child MPb {}          // #1
  export child MPb {}   // #2
  export child MRb {}
}
export model MGb2 {
  export .MPb {}    // meaning #1
  export child MCb {}
}
```

The consequence, that the MPb is within the module but not unique from the outside, is only recognizable in the application. The child MCb appears from the outside that an object with an inherited identifier possesses a “set” identifier and is therefor addressable by his name.

```
dialog D
import I “m.if”;
on dialog start
{
  print MRb;  // label ok

  // print MPb;  Error: label not unique
  print I.MPb;  // access to the model
  print MGb.MPb; // access to #2 from model MGb
  print MGb2.MPb; // access to #1 from MGb2

  print MCb; // access to child of MGb2 , model was not exported
  exit();
}
```

Fundamentally, it makes good sense not to use the same names for exported children as those used for the models. This will save you many problems in the future.

22.2.6 Exported Imports

Imports, which in turn are exported again, are handled in a special manner. The identifiers of these imports are handled like inherited identifiers. In addition, an importing module receives all the exported imports directly under the module/dialog of the imported module.

The goal is not to let the import appear as a hierarchy, but rather as direct children of the dialog, even when they stem from an imported module through `export import`.

The following example illustrates this situation.

```
module M1
export import I2 “m2.if”;
export model pushbutton MPb {}
```

M1 also exports the objects from M2.

```
module M2;
export model window MWi {}
```

Access to the exported objects from M2 is also possible in the importing M1 dialog.

```
dialog D
import I1 “m1.if”;
MWi Wi { MPb {} }
window I2 {}       // Warning – label cannot be set,
on dialog start {
  print I1;        // access to import D.I1, label I1 is unique
  print I1;        // access to import D.I2, label I2 is unique
  print D.I1;      // access to import I1
  print D.I2;      // access to import I2
  print I1.I2      // using a path also possible
  exit;
}
```

22.2.7 Overlay through Local Identifier

Within a rule, identifiers from local variables and parameters overlay the object identifier and as a result prevent ambiguity.

```
dialog D
window W {}
rule void Rule(object W) {
  variable integer D:=123;
  print "D="+D;
  print "W="+W;
}
on dialog start {
  Rule(D);  //  => Result  "D=123"
  exit();   //             "W=D"
}
```

22.2.8 References to Problems and Problem Areas in Previous Versions

The following tips should help you in identifying and avoiding problems in existing dialogs. The previous IDM versions had the following problems:

- Functions were able to be defined in an uncertain manner.
- Equivocations were not recognized. When equivocations appeared, the first reference was removed from the module/dialog.
- The read-in sequence is important.
- Faulty writing removal with –writedialog: object referencing is not clearly written out. Even when read-in is possible, in some cases something completely different comes out. This is due to item number 4.

Example

```
dialog D
function string F1();
function integer F1();  // was accepted in version A.04.01.c
window Wi {
  groupbox Gb {
    pushbutton Pb { .text "#1"; }
  }
}
model window MWi {
  groupbox Gb {
    pushbutton Pb { .text "#2"; }
  }
}
on dialog start
{
  print Pb.text;  // result is #1,
                  // after save with –writedialog => #2
  exit();
}
```

22.2.9 Impact on IDM Applications (for the customers)

Because of the problems presented in 2.2.8, dialogs are not able to be loaded for the time being. First of all, all ambiguities **must** be resolved. This can take place either by hand or with the help of the IDM simulation program.

22.2.10 Start Options and Environment

In the case of an emergency, the environment variable `IDM_LABEL_STRICT` is available for a certain amount of time. If this is set to *false*, the recognition of ambiguity is turned off (note: this will only be evaluated once at the beginning). Ambiguous dialogs can be made loadable through this.

In order to migrate an application to an unambiguous referencing within dialogs/modules with existing ambiguity, the option **-slack** in the IDM simulation program can be used. Thereby, the recognition of ambiguity is turned off before the loading of the dialog (old method). After loading, the recognition is either turned off or on according to the value of the environment variable.

Through the combination of **-slack** with **-writedialog** a module with unresolved ambiguity can be transferred to a module with unambiguous referencing.

22.2.11 Known Errors / Problems within IDM

There are operations, such as creating named objects, setting identifiers, or transferring to a different father, whereby it is necessary to ascertain if this identifier already exists in the hierarchy level. This examination does not take into account the inheritance of the identifier.

As a result, as seen in the example below, in case (a) the identifier MPb (#3) cannot be set because the identifier exists through an inherited hierarchical child in the Wi instance. In case (b) a normal exapmpe is shown once again (no inherited identifiers exist), whereby the setting of the identifier is possible.

```
dialog D
model pushbutton MPb {};
model window MWi {
  MPb {}                // #1
}
MWi Wi {
  pushbutton MPb {}
}
window Wi2 {
  MPb MPb {}            // #4
  D.MPb {}              // #5
}
on dialog start {
  // (a)
  print Wi.MPb;         // access to #2
  print Wi.PUSHBUTTON;  // access to #3
  // (b)
  print Wi2.MPb;        // access to #4
  print Wi2.MPb:[1];    // access to #5
  exit();
}
```

22.3 The Layoutbox

Often one is faced with the problem that objects or object building blocks must be arranged.

This is possible to do by hand, however

» it is laborious and

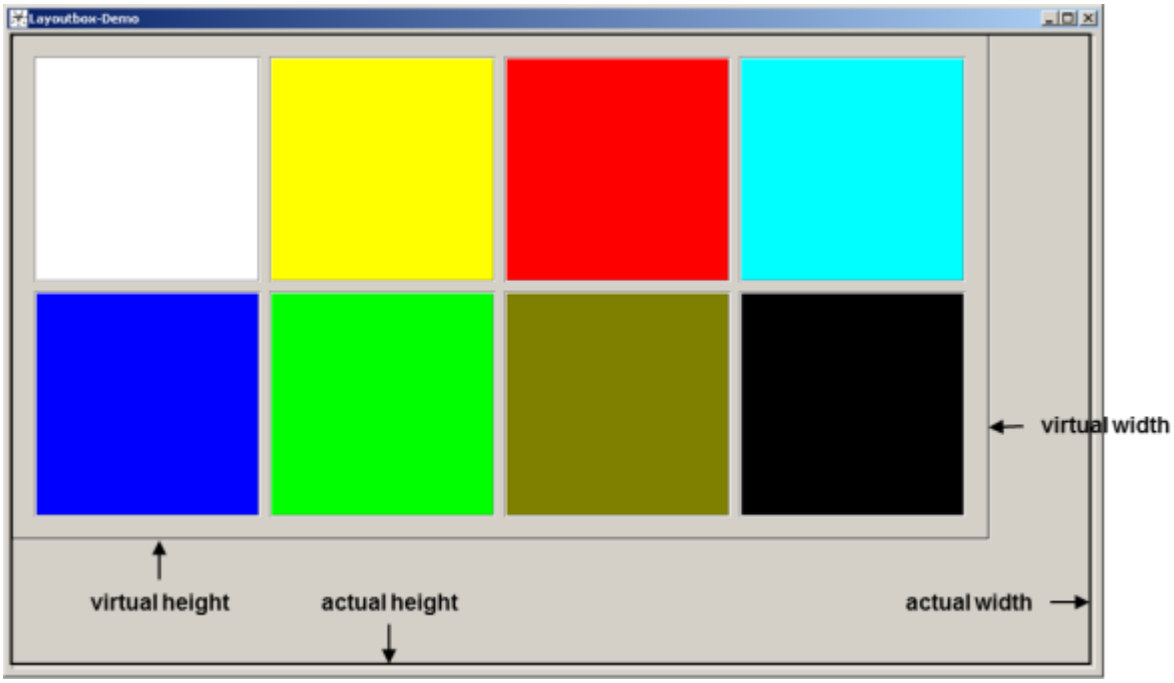
» susceptible

in the case that certain objects change either in their position or size.

The layoutbox can offer assistance in such situations. The layoutbox is a container object in which the objects and building blocks that are to be arranged can be thrown into. They are then automatically arranged.

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

In addition, it makes sure that all objects are accessible, i.e. when an object cannot be visibly displayed (for example if the father window is too small) by providing a scrollbar. The virtual sizes of the layoutboxes are always set (only if wrap is true). They cannot be influenced, however they can be called up in order to establish the working area. This means the virtual sizes can be either smaller or larger than the actual sizes.



The column width or height of a line will always be set according to the largest object in the respective column/line. All other objects are arranged accordingly.

In the case that objects are aligned according to lines, *.yauto* will reference the height of the line. *.xauto* has no meaning, since objects are usually arranged one after the other with the same spacing.

22.3.1 Object Definition

Definition

```
{export} layoutbox {<Identifier>}
{
```

```
  <Attribute classes>
}
```

Events	EM_dbselect
	EM_extevent
	EM_help
	EM_hscroll
	EM_key
	EM_paste
	EM_scroll
	EM_select
	EM_vscroll
Children	Canvas
	Checkbox
	Combobox
	Control
	Edittext
	Groupbox
	Image
	Layoutbox
	Listbox
	Menubox
	Notebook
	Pushbutton
	Radiobutton
	Record
	Rectangle
	Scrollbar
	Spinbox
	Splitbox
	Statictext
	Tablefield
	Treeview

Father	Control Dialog Groupbox Layoutbox Module Notepage Splitbox Toolbar Window
Menu	Pop-up

22.3.2 Attributes

22.3.2.1 New Attributes

Attribute	Data Type	Value Range	Properties	Short Description
direction	integer	1 .. 2	SGCI	The alignment of the objects to be aligned can be selected either line-wise with *.direction = 2* or column-wise with *.direction = 1*. The default value is *2*.
wrap	boolean	true/false	SGCI	Determines if a new calculation of the children must take place when the layoutbox changes in size (i.e. through resize).
mincolwidth	integer	0..65536	SGCI	Is valid for column-wise alignment: indicates the smallest width for an object with *.xauto = 0*, if there is no other object within the column or all other objects in the column are also defined with *.xauto = 0*. This attribute is ignored by line-wise alignment.

Attribute	Data Type	Value Range	Properties	Short Description
minrowheight	integer	0..65536	SGCI	Is valid for line-wise alignment: indicates the lowest height for an object with *.yauto = 0*, if there is no other object within the row or all other objects in the row are also defined with *.yauto = 0*. This attribute is ignored by column-wise alignment..
xmargin	integer	0..65536	SGCI	Indicates the left and right margins in the layoutbox.
ymargin	integer	0..65536	SGCI	Indicates the top and bottom margins in the layoutbox.
xspacing	integer	0..65536	SGCI	Indicates the horizontal spacing between child objects aligned above one another.
yspacing	integer	0..65536	SGCI	Indicates the vertical spacing between child objects aligned next to one another.

22.3.3 Detailed Description of Attributes

22.3.3.1 New Attributes

22.3.3.1.1 direction

	Rule Language	C	COBOL	Properties
Identifier	direction	AT_direction	AT-direction	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 1..2

Default Value: 2 (line wise)

Classification: Object-specific attribute

With this attribute the type of alignment can be set.

There are two variations to choose from:

» Line-wise alignment
The children in the layoutbox are arranged line-wise according to how they stood in the child vector. In each line the maximal height is determined. According to this measurement, the children in the layoutbox are then organized. When the line is full, i.e. when the next object entered into this line is not fully visible, then the next child is entered into the following line.

» Column-wise alignment
The children in the layoutbox are arranged in a column-wise fashion according to how they stood in the child vector. In each column the maximal width is determined. According to this measurement, the children in the layoutbox are then organized. When the column is full, i.e. when the next object entered into this column is not fully visible, then the next child is entered into the following column.

22.3.3.1.2 wrap

	Rule Language	C	COBOL	Properties
Identifier	wrap	AT_wrap	AT-wrap	SGCI
Data Type	boolean	DT_boolean	DT-boolean	

Value Range: true/false

Default Value: true

Classification: Object-specific attribute

With this attribute the page make-up can be switched on and off.

Page make-up means that when a size change or another change of the visible attribute in the layoutbox takes place, then the children also need to be newly sorted so that all children are visible and accessible.

When the page make-up is turned off, no further changes are carried out. (i.e. changes to the attributes in the layoutbox direction, .ymargin, .xmargin... or their children).

In the case that the page make-up has been turned off from the beginning, the objects from the layoutbox will not be aligned.

22.3.3.1.3 xspacing

	Rule Language	C	COBOL	Properties
Identifier	xspacing	AT_xspacing	AT-xspacing	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: Object-specific attribute

With this attribute, the horizontal spacing between the layoutbox and the children can be set.

22.3.3.1.4 yspacing

	Rule Language	C	COBOL	Properties
Identifier	yspacing	AT_yspacing	AT-yspacing	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: Object-specific attribute

With this attribute, the vertical spacing between the layoutbox and the children can be set.

22.3.3.1.5 xmargin

	Rule Language	C	COBOL	Properties
Identifier	xmargin	AT_xmargin	AT-xmargin	G
Data Type	Integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: Object-specific attribute

With this attribute, the right and left spacing between the layoutbox and the children can be set.

22.3.3.1.6 ymargin

	Rule Language	C	COBOL	Properties
Identifier	ymargin	AT_ymargin	AT-ymargin	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: Object-specific attribute

With this attribute, the upper and lower spacing between the layoutbox and the children can be set.

22.3.4 References

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

Attribute	Data Type	Value Range	Properties	Short Description
xauto	short	-1 .. 1		For column-wise alignment: the object is arranged in the respective column whose width is determined by the widest object in it. For -1 from the left edge of the column, for 1 from the right edge of the column, for 0 the object is widened to the column width. If in a column only objects with .xauto = 0 are present, these are assigned the value of .mincolwidth as width. For line-wise alignment this attribute has no meaning.
yauto	short	-1 .. 1		For line-wise alignment: the object is arranged in the respective row whose height is determined by the highest object in it. For -1 from the bottom edge of the row, for 1 from the top edge of the row, for 0 the object is heightened to the row height. If in a row only objects with .yauto = 0 are present, these are assigned the value of .min-rowheight as height. For column-wise alignment this attribute has no meaning.

22.3.4.1 Position Rastering

The positioning of children from the layoutbox is supported, but does not make much sense.

It is only supported for dialog editing purposes.

If one would like to try with the help of the editor to see if the layoutbox is better suited as a container than a groupbox, the posraster attribute would be deleted by each child. If the groupbox, after the fact, then appears to be the better choice for the user, the user must then re-set the attributes to all the

children per hand. When posrastering is used for children within the layoutbox, the result can be quite different (worse) to what one expects, due to the fact that the rounding up or down during the positioning of the various objects (with and without borders) can lead to non-uniform alignment of objects.

22.3.4.2 Virtual Sizes

The layoutbox computes the virtual width and height by itself. This can be set and selected by the user. However, settings from the user will be ignored.

The virtual sizes always specify the work area, i.e. the rectangle area which the children encompass with their outer margins in the layoutbox.

22.3.5 Example

A simple use of layoutbox objects can look like the example below:

```
dialog Dialog { }

color Black “black”

window W
{
    .title "Layoutbox-Demo";
    .width  400;
    .height 400;
    child layoutbox Lb {
        .xauto  0;
        .yauto  0;
        .xleft  5;
        .xright 5;
        .ytop   5;
        .ybottom 50;
        .xspacing 10;
        .yspacing 10;
        .xmargin 20;
        .ymargin 20;
        .wrap true;

        child pushbutton Pb1 {
            .text "Pushbutton1";
        }

        child statictext St1 {
            .text "Statictext";
        }

        child groupbox Gb1{
```

```
   .bgc Black;
             .xauto  1;
             .yauto  1;
             .width 100;
             .height 100;
          }

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

22.4 New Object Combobox for Motif

The poptext now supports the ".style" attribute with all its values ("poptext", "edittext" and "listbox") and the attribute ".showitem" for the Motif 2.1 platform.

Further details pertaining to the poptext can be found in the chapter “Poptext (Combobox)” of the “Object Reference” The notes for Motif in this chapter are only valid for Motif 1.2.

The implementation is based on a new interface (Widget), which is available starting with MOTIF 2. Due to this, slight modifications resulted which in turn affected ".style := poptext", for example:

- A changed appearance of the arrow used for opening the list.
- Changed selection color.
- A size-modification of the poptext. It is now slightly higher and not so wide. This can lead to problems such as having letters cut off at the bottom (i.e. "gÖ@"), if a height is preset. If a height has not been preset, the poptext will then be presented in a slightly larger format.
- "Select" - this event already emerges when another text is chosen or is taken on in another text field. Up to now, this event appeared when the selection list was closed. Now, the "select" event is only available when an entry (the attribute ".activeitem") was actually changed.

The poptext under Motif 2.1 can now be opened or closed by using the keyboard with the following key combinations: `Ctrl` + `Cursor Down` or `Ctrl` + `Cursor Up`.

22.5 Togglable Image for Motif and Microsoft Windows NT

The image object has been enhanced with an ".style" attribute and it can have two different modes. The .style attribute can also have two values:

- .style : pushbutton style
- .style : checkbox style

By the ".style" setting of 'Checkbox', a switching between modes (active/inactive) takes place. A graphic can be assigned to each mode. For this purpose the image attribute ".picture" was induced:

- picture[tile_default] <Tile> - picture in an inactive mode
- picture[tile_active] <Tile> - picture in an active mode

With the pushbutton setting, the switching between modes is not possible. The object receives the graphic assigned to picture[0]. This behavior is equivalent to the behavior in previous versions.

The image object now supports GIF-graphics (89A format), which possess transparent colors.

22.6 Microsoft Windows NT

- Empty messagebox: empty messageboxes have appeared in the past. This was only the case when the error messages exceeded the allowed length.
- Gnats 8724: Icons now appear in the desired size. If an Icon, i.e. 16x16, is requested from an Icon resource with various possible resolutions (for example 16x16 or 32x32), the best size will be chosen first and eventually scaled to size.
- Gnats 8394: the support for wheel mice was added for the following objects:
  - Tablefield: By using the wheel one is able to scroll up and down. In combination with the `Shift` key one is able to scroll page by page. In combination with the `Ctrl` key one is able to scroll horizontally.
  - Poptext: In the open mode it is possible to scroll vertically. The active entry remains unchanged. In the closed mode the poptext will open.
  - Groupbox: Here, scrolling up and down takes place line by line. Here too, in combination with the `Shift` key one can scroll page by page; in combination with the `Ctrl` key one can scroll horizontally. The height of the line can still be set with the attribute *.vsb_linemotion*.
  - Treeview: Here, scrolling up and down takes place line by line. With the `Shift` key one can scroll page by page; in combination with the `Ctrl` key one can scroll horizontally.
  - Edittext multi-line: Here, scrolling up and down takes place line by line.
  - Listbox: Here, scrolling up and down takes place line by line. With the `Shift` key one can scroll page by page; in combination with the `Ctrl` key one can scroll horizontally.

  With Logitech drivers it is important to set the mouse features to *“Only use MS Office compatible scroll”*, otherwise the driver does not stick to the Windows standard.

22.7 JAVA Interface

- It is now possible to encrypt the communication between server and client by using SSL. In order to do this, it is necessary to create an IDM application, and to be in possession of a software encryption license.
- Java cursors can now be assigned to IDM objects.
- Tablefields can be edited now. This means that the following attributes are now supported: .edittext, .editable, .edittype, .editpos, .format[][], .maxchars[][]. As a result, the following events are now also supported: EV_charinput, EV_modified.

Restrictions

- Row headers are not able to be edited.
- Multi-line contents are not supported, i.e. .editpos=true, .sizraster=true and .rowheight>1 => for editing purposes a single line edittext is used.

22.8 DM-Kernel

- Gnats 8673: Exported functions from non-exported applications did not make it to the interface section of a binary module. This is why the function was not able to be set as exported during the reading of the binary module (without interface data).
- Gnats 8769: Setvals on dynamic resources (font, tile, cursor, color) showed an effect.
- Gnats 8765: Initial values during parsing were corrected.
- Gnats 8741: Export-Imports do not receive an allocated label reference in order to guarantee their uniqueness.

22.9 Debugger

The debugger under Motif now possesses the same interface as that under Windows.

Restrictions

- Motif in general: no statusbar.
- Motif 1.2: on account of this, no combobox:
  - Instead of a poptext, an edittext is used.
  - The short list of the last expressions entered can be obtained through the menu.
  - A short list does not exist for the last entries of a rule search.

22.10 Motif

- Gnats 6776: If a window with a groupbox is closed through a select rule (a select rule which hangs on a groupbox), the image, which is situated directly under the groupbox in another window, will trigger no further events.
- Gnats 3940: If a text is set to an image during runtime, the image will increase in size, and the text under the image will be displayed. If the image text is deleted (Image.text = null), the image will return to its previous size.
- Gnats 8538: The following errors in the treeview object have been corrected: It is now possible to set .sensitive to true, even if .sensitive was set to false when the treeview was created. This also holds true in the case that .visible was set to false and then back to true after the switching of attributes. When selecting in an up and down motion with the arrow button, the core attribute .activeitem is updated and the activate event is sent. After setting the .activeitem attribute from the

Rule Language, the focus frames are also automatically updated. The .topitem remains visible after the switch -> not visible -> visibility correctly displayed.

» Gnats 8646: If the attribute .picture is not set to an image object, then it will not have the correct size after it has been created (independent of the size raster).

23 Version A.04.01.I

23.1 Core

» Gnats 9271: The double release (detaching), which is triggered when opening a popup menu, of a text that was just deleted through a repeated deletion in a recursive event loop is now avoided.

24 Version A.04.01.k

24.1 Microsoft Windows NT

» Gnats 9167: There were OLE controls, which claimed the keyboard entry for themselves, as soon as they became active. This problem occurred only after Gnats 9064 was corrected. Now, an active OLE control is deactivated as soon as another (Dialog Manager) object is clicked on or, when it receives the focus. As a result, the keyboard entry is processed once again by the Dialog Manager.

24.2 Core

» Gnats 9084: Dynamic settings on resources were partially invisible. As a result, the objects affected by this had to be switched back and forth from invisible to visible. Now, changes on resources are propagated on the WSIs, even when the resources themselves are not realized.

25 Version A.04.01.j

Skipped in order to avoid confusion with A.04.01.i.

26 Version A.04.01.i

26.1 Core

» Gnats 9097: Dynamic settings on resources were partially invisible. As a result, the objects affected by this had to be switched back and forth from invisible to visible. Now, changes on resources are propagated on the WSIs, even when the resources themselves are not realized.

27 Version A.04.01.h

27.1 Microsoft Windows NT

» Gnats 9064: An OLE control, which was generated with ATL-Wizard and which also works with accelerators, could not be linked correctly in IDM. Keyboard events did not appear in the correct windows. Now, the OLE object, which is displayed as embedded, is the first to receive the chance to handle the accelerators.

27.2 Core

» Gnats 8977: The reading performance of binary data has been improved.

28 Version A.04.01.g

28.1 Microsoft Windows NT

- Gnats 8969: Editable children of a toolbar (poptext, style edittext and edittext) no longer lose their content after the toolbar has been redocked.
- Gnats 8821: Longer strings are now possible in a treeview (IDM limit is 32759 characters, previously 512). When a string is longer than the allowed length, the string is shortened and is marked with three dots at the end to show that it has been altered.

28.2 Core

- Gnats 8965: parsepath() now returns non-exported hierarchical children according to the module context. The unauthorized access to non-exported children from "outside" has also been corrected, i.e. this is no longer possible. This is equivalent to a path phrase.

28.3 Editor

- Gnats 8962: Faulty entries by parsing from :-methods no longer leads to a destruction of the father, i.e. to an assfail.

29 Version A.04.01.f

29.1 Microsoft Windows NT

» Gnats 8859: When external events, which are sent from COM controls to the IDM, contain interface-pointers as a parameter, then these interfaces can be used as subcontrols through the Rule Language (analog to interface-pointers as parameter from COM control methods).

29.2 Core

» Gnats 8765: The problems that came about when reading binary file have been corrected.

» Gnats 8868: Extension of the :find() method – all indexed attributes (i.e. also .content[,] on the tablefield), as well as userdefined attributes, are supported. In addition, it is possible to search for any value, and not just for strings.

» Gnats 8928, Gnats 8929: The :index method now functions for all attributes in the poptext, and no longer only for .options.

29.3 Micro Focus COBOL

» Now cobinit is also called on AIX.

30 Version A.04.01.e

30.1 Microsoft Windows NT

- If a maximized MDI window was created or made visible together with a maximized MDI child and a toolbar or a statusbar, the toolbar (and sometimes the statusbar) was covered up by the MDI child.
- Gnats 8625: The allocation of selection buffers for the multi-selection in the filereq-object has been corrected.

30.2 DM Core

- Gnats 8708: Optimization of the destroy if the object is used only in the current module and the object is passed as a parameter to other rules.
- Annotation from "on Attr changed"- rule are now noticed.
- Gnats 8736: Memory, that was allocated through calls from the DM functions for return strings, will be released at the very latest when exiting the function (the utmost function by recursive function calls).
- Gnats 8737: :insert-methods on a poptext with no content functions correctly again.
- Gnats 8742: Detaching from inherited labels, so that used counters do not overflow.
- Gnats 8670: Assfail with this version is no longer comprehensible.
- Gnats 8454: Assfail with this version is no longer comprehensible.
- Gnats 8673: Assfail with this version is no longer comprehensible.
- With D&D from listbox/treeview as the source, type_text & type_file are now occupied with the correct values.
- Gnats 8752: Label translation is nonexistent in the rule code by binary reading.
- Gnats 8753: The 0-field of the induced attributes [row,col] of the tablefield are written in the new binary format.
- Gnats 8673: Exported functions from non-exported applications did not make it into the interface section of the binary modules. Because of this, the function could not be labeled as exported when the binary module (without interface data) was read.
- Gnats 8769: Setvals dynamic resources (font, tile, cursor, color) show an effect.

30.3 COBOL-Interface

- Byte corrections in DMcob_LSetVectorValueBuff were forgotten. This lead to problems with Compaq equipment.

31 Version A.04.01.d

31.1 Expansion with Drag & Drop Under Microsoft Windows NT

Up to now, the values of drag & drop operations which were transferred from source to target arose either from the types that were given on the source-resource of the source object, or directly from the text content of the object.

From the over-definition of pre-defined methods :setclip on the source object, it is now possible in the Rule Language to freely set the values to be transferred.

The method :setclip is immediately and implicitly called up as soon as the mouse is set into action by a drag & drop operation (i.e. pressing or moving the mouse). The explicit call of :setclip() in a rule is not allowed.

The “invisible” parameter *clipboard* is used for setting the data values that were moved from the source to the target object. This refers to an object of the ***clipboard*** class. In this class, the data values can be set or read with the help of type enum induced *.value[E]* attribute. Allowed are only those types that are declared in the source resource used by the source object. Value assignments to .value [E], with other types as the index create errors.

In an over-defined :setclip() rule, or when no other model methods are available, the :setclip() model methods can be accessed in the standard behavior by calling up *this:super()*.

Execution Example

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

model pushbutton MPb
{
  .source Src;
  :setclip()
  {
    this:super();  // Standard-Methode fuer Objektklasse
    Clipboard.value[type_file]:=”FILE: “+
        Clipboard.value[type_file]
  }
}
window Wi
{
  child MPb PbSource
  {
```

```
    .text “Source”;
    :setclip()
    {
      // Ueberdefinition der :setclip-Methode durch
      // eine benutzerdefinierte Methode
      // Ein Parameter mit dem Namen “Clipboard” enthaelt
      // die ID des Clipboard-Objektes, dem man die
      // Werte im .value[]-Attribut zuweisen soll.

      this:super();   // :setclip-Methode des Modells aufrufen

      // Wir setzen einen neuen Wert fuer den Text-Typ
      Clipboard.value[type_text] := “MY TEXT DATA”;
    }
  }
  on close
  {
    exit();
  }
}
```

31.2 Microsoft Windows NT

- Gnats 8686: After receiving the keyboard focus, a child object was unnecessarily scrolled into the visible area even though it was already visible.
- Gnats 8732: When copying (`Ctrl` + `C`) multi-lined strings, some characters were not copied.
- Gnats 8730: The MDI windows with size rasters were not correctly (Windows-conform) changed in size when maximized. This lead to problems with MDI Icons. Maximized MDI windows are no longer aligned to their size raster or posraster.
- Gnats 8718: The error message [E: WSI error processing listbox .topitem] appeared again and again, even though topitem was never set. The error message appeared when .itemcount was transacted.
- Gnats 8693: The value of the .function attribute is now also saved in the new binary format.
- Gnats 8703: Module-Start now appears before Dialog-Start. As a result, imported modules will now be started before the dialog.
- An over-definable method :setclip was added to allow the setting of various values for drag & drop operations.
- Gnats 8702: The scrollbar slider was not displayed when the listbox was set to .sensitive = false during the creation process. Now the slider is displayed in the case of insensitivity.
- Gnats 8691: The user no longer receives a fatal error message when event rules to an imported object were defined in another module.
- During the binary writing of a module, it should no longer come to a fatal error in dealing with an inherited exported child with an inherited name.

31.3 DM-Core

- Gnats 8717: The reading-in of binary modules and dialogs, which uses resources such as accelerators, is now correctly carried out.
- Gnats 8674: Write trampoline module for COBOL functions that use records with sub-records and .parameter (obsolete), is now correctly carried out.
- When resolving user-defined attributes, paths to objects, which disagree with the data type of the attributes, are recognized.
- The .refstring attribute on fonts is now also considered in the new binary format.
- Gnats 8722: Even rules to implicitly created defaults are now written in the binary data.
- Gnats 4733: The "find" method of the objects poptext, listbox, treeview and spinbox behave undefined, as long as not all possible parameters are entered. Now, optional parameters must not be included.
- Gnats 7210: Ambiguous label must be resolved through the path.
- Gnats 7211: Event rules to imported ambiguous objects will now be written out with an unambiguous path.
- Gnats 7224: Named top-level rules no long have a special status with regard to the naming process. They behave like normal objects, i.e. the name must be absolutely clear when one plans on addressing the object directly by name.
- Gnats 6732: Rules and methods are allowed to have the same name.
- Gnats 8696: Unambiguous paths will now be correctly written out with binary writing with the help of Label-Module-Revision.
- Gnats 8695: During binary writing (new format) from a dialog/module, in which the number 2147483648 appeared, the IDM fell into an endless loop.
- Gnats 8711: Correction for hierarchical, non-exported children in binary format.
- Gnats 8705: The answer to diverse problems with the new binary format:
- The use of inherited hierarchical children lead to assfail. The error message "Interface not up-to-date" came about when an exported child was derived from a non-exported default, and when the child did not receive a name of its own (child could not be set as exported). The value from .edittext on a tablefield was always written out, even when it was inherited. The sequence by binary writing between application and import was not correct. This lead to a resolve error when setting .application to an import. No imported format function could be given to the format since this reference was not released. Runtime library did not release resolve paths on attributes in the binary module.

31.4 Java-Interface

- Format resources are now supported in edittexts and comboboxes (.style = edittext). The following have not yet been implemented: override mode, numerical formats and format functions.

» General optimizations. Performance improvement when transferring strings to the client side.

» Gnats 8706: the active item attribute was not updated after selecting a tablefield with the selection type sel_row or sel_column.

31.5 MICRO FOCUS COBOL

» Swapping a parameter value in DMcob_LGetVectorValueBuff has been corrected.

32 Version A.04.01.c

32.1 New Data Format

A new data format has been added to the Dialog Manager which combines the advantages of the previous available formats (Binary/ASCII), while at the same time avoiding their disadvantages. This new format does not allow for the saving of imported information in a module, but nevertheless the data can be read into the module efficiently without the use of supplementary data (interface files). In addition, this new format can be used with any version and architecture type. This means that when using this new format it is not necessary to re-load the old modules with the new program. The format is very similar to the GNT format which is supported by the MICRO FOCUS COBOL Compiler.

The advantages of the new format are listed below:

- Since dependency between modules no longer exists in the files, it is only necessary that the manipulated modules be compiled in the intermediate format when changes have been made. Modules that use this module need not be newly compiled.
- Since only modified modules need to be compiled, the number of modules to be allocated in the software distribution can be greatly reduced. Especially those PC's that are not connected to a LAN are able to reduce software allocation expenses. In turn, this new scheme will also have a positive "saving" effect on network capacity, whereby this can then be used for other important tasks.
- Since this format is no long dependent upon the version, changes in the actual runtime system can be released independent of the modules. This means that a new runtime system with new functions can be set into a productive operation, but this does not mean that they need to be switched over to productive systems. In this way it is possible to implement releases in which some modules are already using the new functions. These are then released at the same time and the others remain unchanged on the production processor. Here too, one can see an immense reduction in operating expenses tied to software allocation.
- Since the modules are computer hardware independent, they can be compiled in the temporary format on any computer and then used on all architectures without having to be specially created there.
- Developers, who work on central modules, can compile modules during test phases in this temporary format since the dependent modules in this module do not contain any information about the actual module. In this case it is not necessary to compile the modules again, rather they can be used exactly how they are at this time. With this process, developers can also save time since they can test their modules for complex applications in the efficient temporary format and do not have to use the much slower ASCII format.

This new data format will completely replace the old binary format beginning with version A.04.01.c. Due to compatibility reasons, both formats will continue to be supported in version A.03.10.m and higher.

Through this new form of saving data, it is possible that inconsistencies arise between modules (when these show forms of dependency). These inconsistencies are ignored and gone around when ever possible. Depending upon the mode, a protocol will be carried out or not carried out. Due to this, it is important to check protocols regularly to ensure that errors be detected quickly.

Example

Module 1 defines a color and exports it.

Module 2 imports module 1 and uses this color in an object.

Both modules are compiled in the temporary format. Afterwards, the color is deleted in module 1 and once again compiled in the temporary format. Now, when the system is turned on, the color from module 1 will be used when module 2 is called up. Since this color is no longer available, the command is ignored and according to the mode, an error message will either be written in a protocol or not. In contrast to the ASCII version, the system will continue to run. This is true because it is only the color allocation which is not contained in the code.

Beginning with version A.03.10.m there is a new binary format which is version and computer-architecture independent.

- Version independent: The binary files are upwardly compatible. This means that a binary file which was written with version A.03.10.m can also be read in version A.03.10.n.
- Architecture independent: A binary file which was written with MS-Windows can also be read on a UNIX machine.
- No more dependency between imported binary files and imported binary modules. When changing a module export, the dialogs/modules which use this module must no long be re-written. The only exception: When a path or search symbol, which has a reference to a module within the interface file, has been changed, then all binary files must be newly created.
- Robustness against faulty exports. When a reference is missing, i.e. when an exported object has been deleted, the loading process is not stopped but rather one receives an error message and the loading process continues with the next object or attribute.

In order to use the new binary format, all dialogs and modules must be re-written in the new binary format. A combination of the old and new format is not compatible. However, a combination of binary and ASCII modules present no problem whatsoever.

32.1.1 Changed Options

32.1.1.1 -writebin

This well-known option writes in the old binary format (in version A.03.10.m up to version A.04.01.b). Beginning with version A.04.01.c it writes in the new binary format.

32.1.1.2 -IDMbinerror

When this binary option is set to false (-IDMbinerror=false), error messages are suppressed in the binary reading process.

32.2 Microsoft Windows NT and Microsoft Windows 3.1

» Gnats 8595: When the focus was set to a poptext, and the F4 key was activated, this resulted in an Assfail. Since the F4 key on a poptext with *.style = listbox* should not result in any action, this event will not be reacted to in the future.

32.3 Microsoft Windows NT

» Gnats 8628: If the attributes .connect and .active were set to a control and directly after that a method from OLE-control was called up, in some situations the method call was activated before the settings to the attributes were carried out which then lead to a system shut down. Now, it is important to check that all attribute settings have been carried out before the OLE object is accessed.

32.4 Core

» A new binary format has been introduced

32.5 Editor

» Gnats 8589: Poptexts that display the application and import lists in the editor are once again updated.

» The user data side for the module/dialog is now visible.

32.6 Network

» Gnats 8590: rexec now correctly detects the port number for exec-service under Linux.

32.7 Dialog Documentator IDD

» Gnats 8574: The IDD crashes by unauthorized storage access when the number of levels within the hierarchy exceeds 10.

32.8 MICRO FOCUS COBOL

» Gnats 8581: Avoid double definitions, for example .editable and .Editable.

33 Version A.04.01.b

33.1 Microsoft Windows NT

- Gnats 8602: Just by entering the first letter of a word, the appropriate entry in a listbox can be selected.
- Gnats 8595: If the focus was set to a poptext (.*style* = *listbox*), and the `F4` key was pressed at the same time, the result was an assfail.
- There is a new enum-indexed boolean attribute 'options' on the treeview object. Currently, only one index exists: 'opt_rightclick_selects'. If the following is true, '.options[opt_rightclick_selects]=true', then entries can be selected with the right mouse button; default is false.
- Gnats 8568: The .dockable attribute in the toolbar is now correctly supported. If a docking area is set to false, the toolbar can no longer be docked there.
- Gnats 7266: In scrolling within a groupbox, it sporadically caused update problems with regard to the children in the groupbox. Although these children were correctly positioned, unfortunately some messages (WM_PAINT, WM_NCPAINT, WM_ERASEBKGND) were not shown.
- Gnats 4069: The index of the cell, in which the entry takes place, is now transferred along in thisevent.index by a charinput-event in the tablefield.
- Gnats 8583: Multi-selection should now correctly mirror the selected data in .value[].

33.2 Core

- Gnats 8587: By larger dialogs it sometimes came to an overflow of Reference-Counter-Variable which then lead a storage access error.
- Gnats 8217: After the creation of a treeview with a set content .activeitem = 0 occurred even though the first entry was activated. Now, .activeitem = 1, when the treeview is created and the content is not empty.
- Gnats 3314: Variant shifting during runtime is now carried out.
- Gnats 8572: Assfail occurred during the loading of a binary dialog. When a format function was assigned to an edittext within a dialog (whereby at this time no format string was given), this resulted in an assfail when trying to load a dialog with a binary format. The correlation between the implicitly created format resources and the edittext now functions in binary dialogs as well
- Gnats 8582: Garbage collection of Ole-Interfaces does not happen anymore after a potential drop; this prevents a computer crash by “close” in the editor.
- Gnats 8554: Label-Detach from stat.Variables in expressions takes into account the compile status of the rule.

33.3 Motif

- Gnats 4069: The index of the cell, in which the entry takes place, is now transferred along in thi-sevent.index by a charinput-event in the tablefield.
- New object splitbox: The Motif version of the splitbox is now supported.
- The behavior of the tool “help” was modified. The tool “help” now appears directly next to the mouse pointer after it has not been moved for a long time.

33.4 Editor

- Gnats 8589: Poptexts that display the application and import lists in the editor are now updated.
- The user data side for the module/dialog is now visible.

33.5 Network

- Gnats 8590: Rexec now correctly detects the port number for exec-service under Linux.

33.6 Dialog Documentor IDD

- Gnats 8574: The IDD no longer crashes by unauthorized storage access when the number of levels within the hierarchy exceeds 10.

33.7 Debugger

- Gnats 8527: In the debugger labeled rules within modules were partially not displayed.

33.8 COBOL-Interface

- Gnats 8581: Double definitions such as AT-editable and AT-Editable should be avoided.

34 Version A.04.01.a

34.1 An Overview of Important Changes

- New object Filerequester
- New object Splitbox
- New object Subcontrol
- New object **Notebook** for Motif
- New object **Notepage** for Motif
- New object **Treeview** for Motif
- New methods
- New attributes (graphics) for listboxes, comboboxes

34.2 New Object Filerequester

The object filereq is a modal file selection window. It differentiates itself between three modes:

- choosing a file to open
- specification for saving a file and
- choosing an index.

In addition, selectable files/indexes can be restricted by an example, or the selection of existing files/indexes can be limited.

34.2.1 Object Definition

Definition

```
{export} filereq {<Identifier>}
{
  <Attribute Class>
}
```

Events	EM_extevent
Children	record
Father	dialog, module
Superclasses	object
Menu	none

34.2.2 New Attributes

Attribute	Data Type	Value Range	Properties	Short Description
changedir	boolean	false/ **true**	SGCI	By a successful selection, the last index in the .directory attribute should be assumed.
directory	string		SGCI	Initial directory ("" is the working directory of the application)
extension	string		SGCI	The data suffix which is to be attached to the chosen data name, in the case that the file does not possess an suffix. The default value "" means that no suffix should be attached.
style	enum	**fr_load** fr_save fr_directory	SGCI	The data selection mode (load/save file, select index)
multisel	boolean	**false**/true	SGCI	Allows multi-selection of data (only WIN32)
mustexist	boolean	false/**true**	SGCI	Selection is allowed only when files / indexes exist. For WIN32 this is only limited to fr_load style.
options[]	boolean	[fro_createprompt] **false**/true [fro_overwriteprompt] **false**/true [fro_relativepath] false/**true**	SGCI	The setting of window-system-specific features. fro_createprompt and fro_ overwriteprompt are only effective under WIN32, fro_relativepath only effective under Motif2.1.
pattern	string		SGCI	Motif: A sample displaying selectable files / indexes. WIN32: Sample lists for the selection and entry of data.

Attribute	Data Type	Value Range	Properties	Short Description
text[]	text	[frt_ok frt_cancel frt_help frt_update frt_path frt_directories frt_file, frt_pattern frt_nomatch]	SGCI	Only Motif: Texts for labeling dialog elements.
title	text		SGCI	Name of the selection window
value	string		G	Result of the selection: complete data or index name. In the case of multi-selection, this attribute must be indexed during the retrieving process.
value[]	string		G	

34.2.3 Inherited Attributes

Attribute	Data Type	Value Range	Properties	Short Description
font	font			only Motif
cursor	cursor			only Motif
visible				Turned off
sensitive				Turned off
real_visible				Turned off
real_sensitive				Turned off
fgc				Incomplete support for Motif
bgc				Incomplete support for Motif

34.2.4 Description

The data selection window is opened in the Rule Language with the help of the built-in function **query-box** or from the application side with the function **DM_QueryBox**. The attribute .visible is not available for this object.

The beginning position of the data selection window can be influenced through the indication of the parent parameters. Please refer to the table at the end of this chapter.

During selection, all other data entry possibilities of the application are blocked. The user can navigate himself through the index tree and select wanted files with the mouse or the keyboard. By confirming the selection with the OK button, or by double clicking the selection, the user receives the following return value:

- button_ok, when the user has selected a file. By a multi-selection, the chosen file name then stands as the complete path indication in the .value or .value[]. If the attribute .changedir possesses a true value, then the attribute .directory will obtain the path of the index from which the file/index was chosen.
- button_cancel, when the user has aborted the choice.
- nobutton, when an error occurs which then prevents the opening of the selection window.

The selection possibility can be configured through the indication of an initial index (attribute .directory) and through a pattern (.pattern), which is applied to the shown, selectable data. When a data suffix is given in the attribute .extension, then this will be hung onto the chosen name if no other suffix exists.

According to the mode (attribute .style) and window system, the selection window is presented in various forms:

Microsoft Windows	Motif

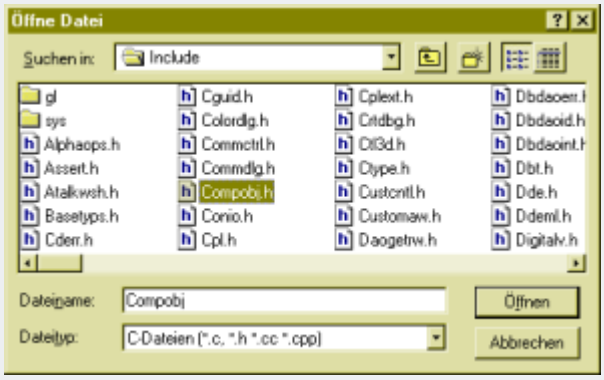


Figure 1: *Selecting file to be loaded*

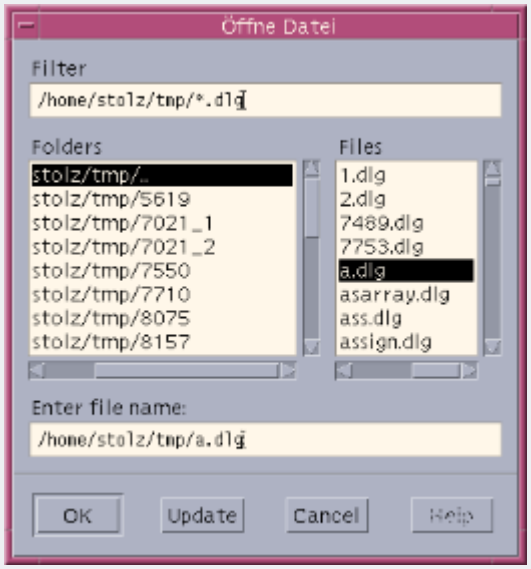


Figure 2: *Opening the file to be loaded*

Microsoft Windows	Motif

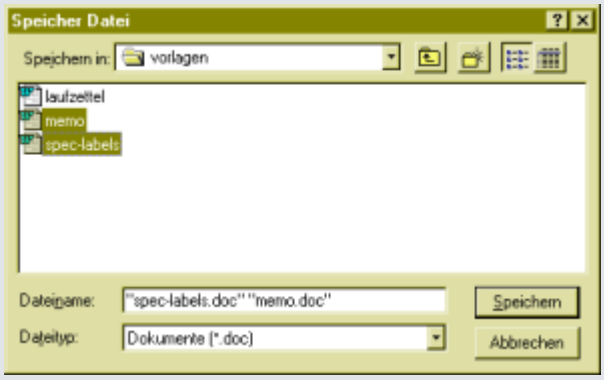
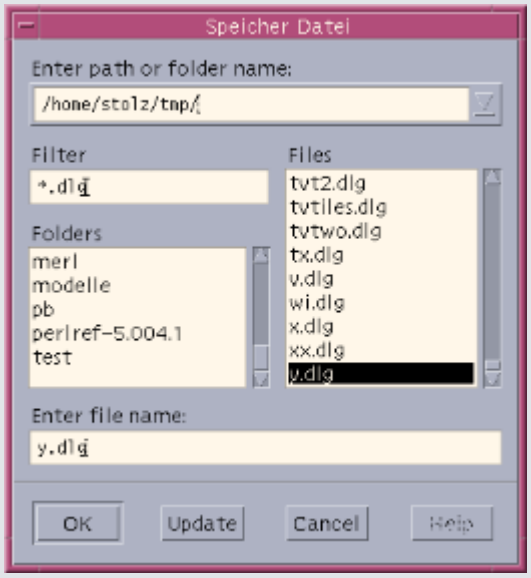

	

Figure 3: *Selecting the files to be saved (with multi-selection)*	***Figure 4:*** *Selecting file to be saved (relative path, Motif 2.1)*

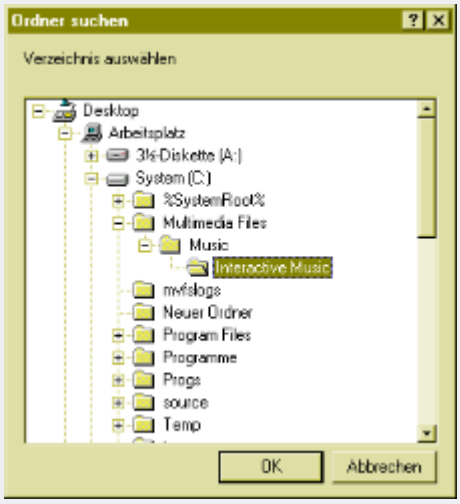
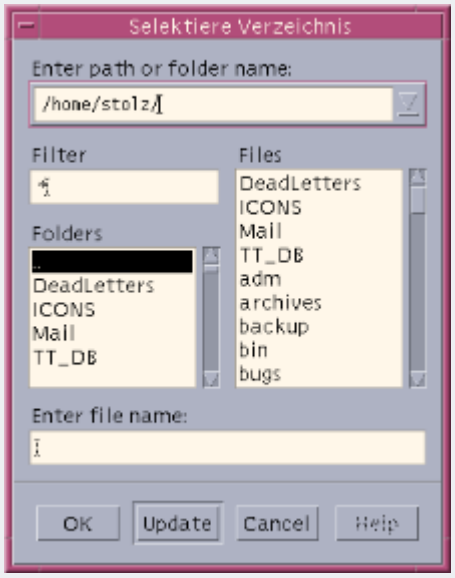

	

Figure 5: *Selecting a directory*	***Figure 6:*** *Selecting a directory*

Some attributes (.options, .multisel, .text) allow for system-specific functions to be used, offer additional functions or configure the selection. At the same time these attributes are not available on all platforms, plus not all attributes have an impact.

The initial position of the selection window is dependent upon the window system in use. The following table shows the effect of the parent parameter through the querybox call on the position.

	Initial Position of the Selection Window		
Parent-Para-meter	Motif	WIN32	
		File Selection	Index Selection
No spe-cification or null	Upper left corner of the desktop	Upper left corner of the desktop when no application window is visible, or in the upper left corner of the last window in the application.	Upper left corner of the desktop
Window Object	Centered in the window	Upper left corner of the desktop	

34.2.5 Detailed Overview of New Attributes

34.2.5.1 changedir

	Rule Language	C	COBOL	Properties
Identifier	changedir	AT_changedir	AT-changedir	SGCI
Data Type	boolean	DT_boolean	DT-boolean	

Value Range: boolean

Default Value: true

Classification: object-specific attribute

This attribute controls the acceptance of the index in which the user found himself by a successful selection in a data selection window within the attribute .directory. If it is set to true, the index path will be accepted. Thus, the next time the file selection is opened, one can continue where one left off.

References and Dependencies

filereq, .directory, querybox()

34.2.5.2 directory

	Rule Language	C	COBOL	Properties
Identifier	directory	AT_directory	AT-directory	SGCI
Data Type	string	DT_string	DT-string	

Value Range: string

Default Value: ""

Classification: object-specific attribute

This attribute, when opened via the built-in function querybox(), contains the path of the initial directory for data/file selection.

When the attribute has the value "" or an invalid path name, then the work directory of the application will appear as the initial directory when called up via querybox(). Please note that the path indication must correspond to the notation of the system.

For MS-Windows this is *<drive>:\<dir 1>\…\<dir n>* and for Unix */<dir 1>/…/<dir n>*. Be aware that "\" within strings must be written "\\" in a Dialog Manager file.

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

Special Features of MS-Windows

If the attribute *.style* is set to *fr_directory* on the filereq object then the .directory attribute will define the first path. Only those directories under this one can be selected. Changing to another drive is not possible. The attribute value "" will be interpreted as "workplace" and not as working directory.

References and Dependencies

filereq, .changedir, querybox()

34.2.5.3 extension

	Rule Language	C	COBOL	Properties
Identifier	extension	AT_extension	AT-extension	SGCI
Data Type	string	DT_string	DT-string	

Value Range: string

Default Value: ""

Classification: object-specific attribute

This attribute allows for the definition of a data suffix. After the selection through the querybox() call, and if the file does not have a suffix, then "." plus its attribute value will be appended to the file.

Special Features of MS-Windows

Only the first three characters of the suffix string will be attached to the file. Extensions are only used on files.

References and Dependencies

filereq, querybox(), .style, .value

34.2.5.4 multisel

	Rule Language	C	COBOL	Properties
Identifier	multisel	AT_multisel	AT-multisel	SGCI
Data Type	boolean	DT_boolean	DT-boolean	

Value Range: boolean

Default Value: false

Classification: object-specific attribute

This attribute controls the user's selection of a filereq object. When the value is true, the user can choose more than one file at a time. The selected files are then indexed in the attribute .value. Normally, the chosen file/index can be found in the scalar attribute .value.

Special Features

This attribute, used for the selection of files, is **only** supported by MS-Windows for fr_load and fr_save modes.

References and Dependencies

filereq, .value, .style, querybox()

34.2.5.5 mustexist

	Rule Language	C	COBOL	Properties
Identifier	mustexist	AT_mustexist	AT-mustexist	SGCI
Data Type	boolean	DT_boolean	DT-boolean	

Value Range: boolean

Default Value: false

Classification: object-specific attribute

This attribute makes sure that only existing files/indexes can be selected (value true). Otherwise, no other check takes place to establish if the file or the entered path is correct.

Special Features of MS-Windows

This attribute only works in the fr_load mode.

References and Dependencies

filereq, querybox()

34.2.5.6 options[I]

	Rule Language	C	COBOL	Properties
Identifier	options	AT_options	AT-options	SGCI
Index Type	enum	DT_enum	DT-enum	
Data Type	boolean	DT_boolean	DT-boolean	

Index Area	fro_createprompt, fro_overwriteprompt, fro_relativepath
Value Range:	boolean
Default Value:	false, only [fro_relativepath] has the value true
Classification:	object-specific attribute

This attribute allows for the controlling of system-specific settings and influences the look and other functions of the filereq object.

Index Value	Default Value	Limitations	Definition
fro_createprompt	false	MS-Windows in the fr_load mode. Attribute .mus-texist must be set to true.	In selecting an existing file, a dialog will appear for safeguarding reasons when setting up a file. Please refer to “Figure 7”.
fro_overwriteprompt	false	MS-Windows in the fr_save mode.	When selecting an existing file, a dia-log appears and asks if the file should be written over or not. Please refer to “Figure 8”.
fro_relativepath	true	Motif beginning with version 2.1	The example and the path at that moment are represented separately and the selected file is only shown as a file name without an index. Please compare “Figure 2” and “Figure 4”.

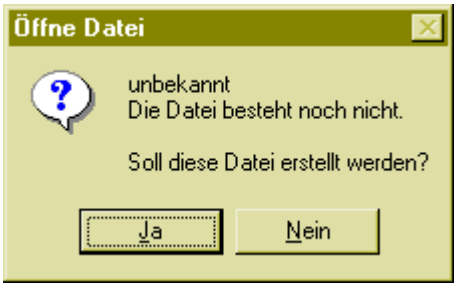


Figure 7: Setting up a dialog (fro_createprompt)

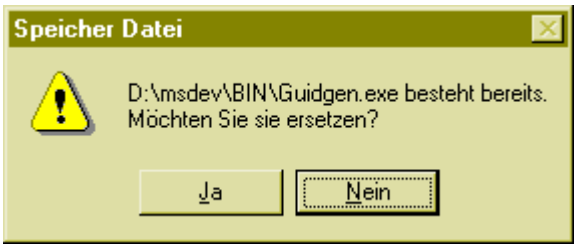


Figure 8: Overwriting a dialog (fro_overwriteprompt)

References and Dependencies

filereq, .directory, querybox()

34.2.5.7 pattern

	Rule Language	C	COBOL	Properties
Identifier	pattern	AT_pattern	AT-pattern	SGCI
Data Type	string	DT_string	DT-string	

Value Range: string

Default Value: ""

Classification: object-specific attribute

This attribute allows for the selection possibilities of a filereq object to be set through a pattern. Only those files/indexes that fit in this pattern will be presented. In the case that "" is given, all files (Motif: *, MS-Windows *.*) will be shown.

Special Features of Motif

In this case, the pattern is the regular term that can be changed by the user himself. The scope of functions is predefined by Motif. Thus, below is just a short insight into the most important special characters:

Special Characters	
*	None or an undefined number of various characters.
?	Any character
[a-z]	The characters a-z, as an area
[0123]	Numeration 0, 1, 2, 3

Special Features of MS-Windows

The pattern is only effective in the fr_load and fr_save modes. MS-Windows expects to receive a list of file extensions with pattern lists. The user is only offered the file extensions; he is not allowed to enter his own pattern. The division between file extensions and patterns comes about with the help of a tabulator character “\t”. In order to list a number of patterns, one can use “;” as a separator.

Example

```
.pattern "IDM-Dateien\t*.idm;*.dlg;*.mod\tDokumente (*.doc)\t*.doc;\tAlle
Dateien (*.*)\t*.*";
```

References and Dependencies

filereq, querybox()

34.2.5.8 style

	Rule Language	C	COBOL	Properties
Identifier	style	AT_style	AT-style	SGCI
Data Type	enum	DT_enum	DT-enum	

Value Range: fr_load, fr_save, fr_directory

Default Value: fr_load

Classification: object-specific attribute

This attribute defines the mode of the filereq object, and thereby one of the following three applications:

- fro_load: opening a file to load
- fro_save: opening a file to save
- fro_directory: selecting a directory

Special Features of MS-Windows

The different modes are represented on various dialog objects of the system. In most cases, the means of presentation, i.e. labels, color, font, are set to each application (i.e. button labels) but are not over-definable.

The fro_directory mode is more restrictive here in the handling of the attributes .directory and .pattern. In addition, this directory does not allow for the entry of a non-existing index.

References and Dependencies

filereq, .directory, .multisel, .directroy, .pattern, querybox()

34.2.5.9 text[l]

	Rule Language	C	COBOL	Properties
Identifier	text	AT_text	AT-text	SGCI
Index Type	enum	DT_enum	DT-enum	
Data Type	text	DT_text	DT-text	

Index Area	frt_ok, frt_cancel, frt_help, frt_update, frt_path, frt_directories, frt_file, frt_pattern frt_nomatch
Value Range:	text, string
Default Value:	""
Classification:	object-specific attribute

This indexed attribute allows for the labeling of dialog elements of a filereq object to be taken up.

Special Features

This attribute is only supported by Motif. “Figure 9” shows the assignment of the separate entries of the window elements. The assignment applies to the file selection and the index selection.

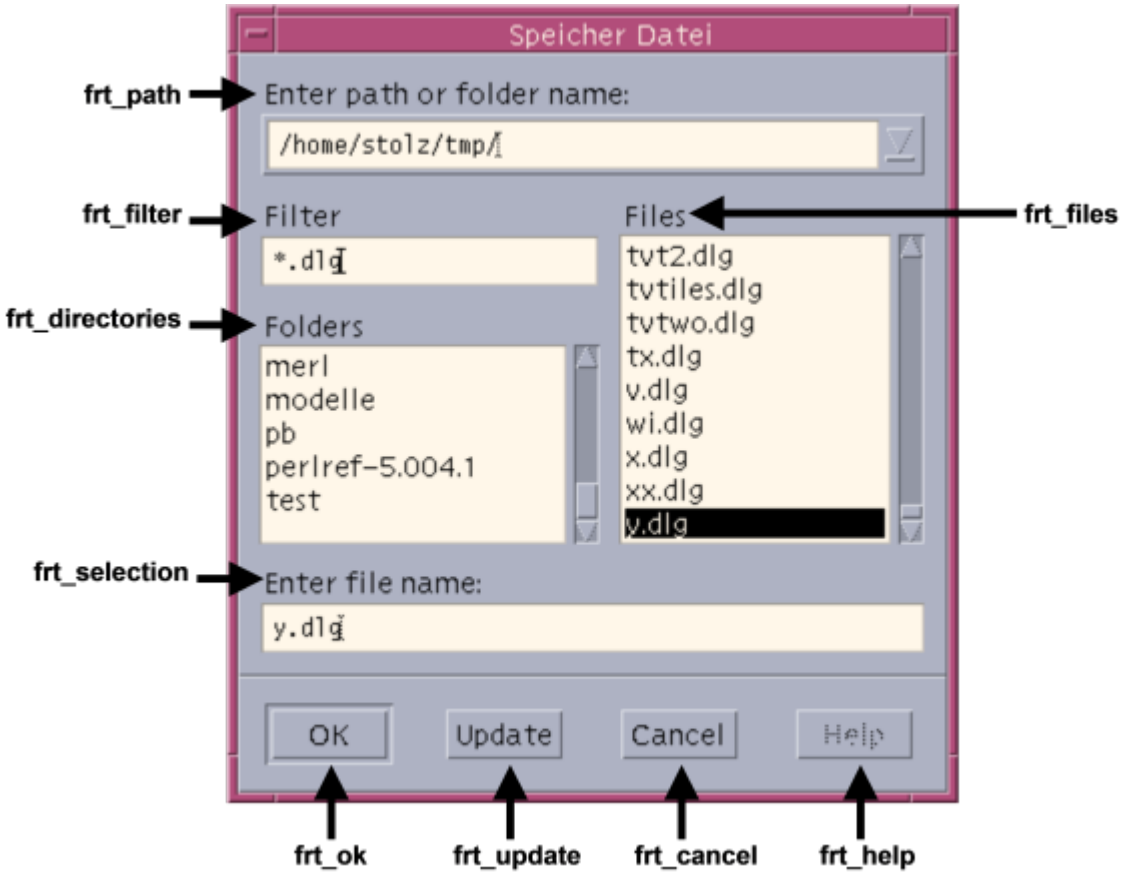


Figure 9: Text Assignment

References and Dependencies

filereq

34.2.5.10 title

	Rule Language	C	COBOL	Properties
Identifier	title	AT_title	AT-title	SGCI
Data Type	text	DT_text	DT-text	

Value Range: text,string

Default Value: ""

Classification: object-specific attribute

This object defines the title of the selection window (filereq object).

Special Features of MS-Windows

By the selection of the directory (fr_directory mode) the title appears only as a heading and not as a title of a window.

References and Dependencies

filereq, querybox()

34.2.5.11 value

	Rule Language	C	COBOL	Properties
Identifier	value	AT_value	AT-value	G
Data Type	string	DT_string	DT-string	

Value Range: string

Default Value: ""

Classification: object-specific attribute

This attribute, after a successful selection (return value from querybox() was button_ok), contains a file or a directory with its complete path. Please take note that this path contains different data delimiters depending upon the system being used. When the filereq object has switched on the multi-selection option (.multisel = true), the list of the chosen files will appear in an indexed fashion.

References and Dependencies

filereq, .directory, .style, .value[I], .multisel, querybox()

34.2.5.12 value[I]

	Rule Language	C	COBOL	Properties
Identifier	value	AT_value	AT-value	G
Data Type	string	DT_string	DT-string	

Value Range: string

Default Value: ""

Classification: object-specific attribute

This attribute, after a successful selection (attribute .multisel = false, return value from querybox() was button_ok), contains a complete list of path information. Please take note that this path contains different data delimiters depending upon the system being used.

In the case of single selection (.multisel = true), the list of chosen files in this attribute will not be indexed.

The number of selected files can be determined by accessing *.count[.value]*.

Example

```
rule void PrintValues(object Fr)
{
  variable integer I;
  for I:= 1 to Fr.count[.value] do
    print Fr.value[I];
  endfor
}
```

Special Features of MS-Windows

The amount of memory for the entire list of selected paths is limited to 64000 bytes.

References and Dependencies

filereq, .directory, .value, .style, .multisel, querybox()

34.2.6 References

The filereq object is reproduced with the data/file selection possibilities that are offered by the system. Due to this, the appearance and the function scope of the filereq object can vary according to the system being used. Improvements made to the system can be beneficial for everyone.

The possibility of offering a help button for all systems is momentarily not available.

The work directory, in which the application can be found, is not influenced by the filereq object.

34.2.6.1 Motif

The representation (font, color, text) is normally defined by the Window Manager. Only texts and fonts are over-definable by the application programmer.

The selection of a file can take place either by clicking the OK-button or by double clicking the file itself.

Independent of the mode of the filereq object, this deals with the same type of file selection window. Thus, it is important that the title or the text bring across the actual content of the file to the user.

Checking to see if a file or a directory exists, as well as attaching a file suffix to a file, is done by the IDM.

34.2.6.2 MS-Windows

Please note that the presentation (font, color, texts), not including the title, are defined through the settings in your MS-Windows system.

Here too, a selection is either carried out by double clicking the item or by clicking the OK-button. MS-Windows offers a multi-selection with regard to the selection/entry of data. A key combination (either with the `Shift` key or the `Ctrl` key) allows for the selection of a range of items or the toggling of the activation status.

The directory selection is, in contrast to the Motif implementation, limited to the selection of an existing directory and does not offer the possibility of working with a pattern. Furthermore, by the indication of an initial directory, the directory selection is limited to only lower level directories.

An additional feature of MS-Windows is the inquiry dialogs which ask the user if a file should be created or written over.

34.2.6.3 Portability References

In order to maintain consistency with the functions and their looks, the following points should be taken to heart:

- Do not use multi-selection.
- Set the attribute .title so that the resulting action is clear (Loading/Saving/Selecting).
- In order to maintain consistency in the language, the languages set in your system must correspond to your texts.
- Use various patterns (attribute .pattern). Avoid using a pattern for the directory selection.
- Use the directory selection without the initial start index.
- UNIX and MS-Windows have different path characters. This needs to be taken into account when working with path indications.
- When querying the system, the attribute .opsys_type can be used on the setup-object.

34.2.7 Example Dialog

The opening of a file selection window (filereq) can appear as follows:

```
!! Example for MS-Windows
filereq Fr
{
  .style fr_load;
  .title “Zeige GIF-Bild an”;
  .directory “c:\”;  /* Motif: “/” */
  .pattern “GIF-Datei\t*.gif\tAlle\t*.*”; /* Motif: “*.gif”; */
  .extension “gif”;
}
rule void OpenFile()
```

```
{
  if button_ok = querybox(Fr) then
    ViewGif(Fr.value);
  endif
}
```

34.3 New Object Splitbox

The object splitbox is a help object (similar to a groupbox) for splitting a computer's monitor display area into smaller sections. Each area is then separated by a splitbar which can be interactively moved around with the mouse by the user in order to optimally size the areas individually. Every visible child in a splitbox is assigned to a specific area, and in this area only this child can be found. The size and the visibility of the child is set according to the size of the area it is assigned to. If one wants to place more than one child in a split area, then a groupbox as child must be implemented here, which can then incorporate the desired number of children wanted. A groupbox set up as a child allows for split areas to be configured with virtual sizes.

For each split area the user can set the size according to his needs. In addition, minimal and maximal values can be set whose higher and lower limits cannot be interactively (moving the splitbar with the mouse) exceeded.

The splitbars can be either horizontally or vertically aligned.

(Exception: with Motif 1.2 the splitbars can only be horizontally aligned.)

Definition

```
{export} splitbox {<Identifier>}
{
  <Attribute Class>
}
```

Events	EM_cut
	EM_extevent
	EM_key
	EM_help
	EM_resize
	EM_paste
	EM_select

Children	Canvas
	Checkbox
	Combobox
	Control
	Edittext
	Groupbox
	Image
	Listbox
	Menubox
	Notebook
	Pushbutton
	Radiobutton
	Record
	Rectangle
	Scrollbar
	Spinbox
	Splitbox
	Statictext
	Tablefield
	Treeview
Father	Dialog
	Groupbox
	Module
	Splitbox
	Toolbar
	Window
Superclass	Aggreg
Menu	Popup

34.3.1 New Attributes

Attribute	Data Type	Value Range	Properties	Short Description
direction	integer	1 .. 2	SGCI	The alignment of the splitbars can be changed vertically with *.direction = 1* or horizontally with *.direction = 2*. The default value is *1*. Attention: this attribute has no meaning under Motif 1.2.
barwidth	integer	0..32576	SGCI	Determines the width of the splitbar in a pixel value. The default value is 5.
size[I]	integer	0..65536	SGCI	The enlargement of a split area with index I (one-based) is set with this attribute (pixel- or rast value). A default value can be defined with index = 0. This can then be transferred to all elements of .size[...], for which no explicit value momentarily exists.
maxsize[I]	integer	0..65536	SGCI	Sets the maximal size that a split area can be interactively changed to with the mouse.
minsize[I]	integer	0..65536	SGCI	Sets the minimal size that a split area can be interactively changed to with the mouse.
real_size[I]	integer	0..65536	G	Here one can call up the actual size of a split area (pixel value).

34.3.2 Inherited Attributes

Attribute	Data Type	Short Description
font	font	Has no meaning for the splitbox.
cursor	cursor	The cursor which belongs to the object. This is the same in every split area.

Attribute	Data Type	Short Description
visible	boolean	Determines the visibility of an object. Only the object can be switched to visible or invisible, not the individual split areas.
sensitive	boolean	Selectability of an object in its entirety. Every split area will be sensitive/insensitive.
real_visible	boolean	The actual level of visibility of an object.
real_sensitive	boolean	The actual level of selectability of an object.
fgc	color	Has no meaning.
bgc	color	Determines the background color of the object. This is the same for all split areas.
bordercolor	color	Splitbars without a 3D-look are shown with this color, if this is set accordingly.
…		All other attributes of the superclass “Aggreg”.

34.3.3 Detail Description Events

34.3.3.1 EM_cut

Only Valid for MS-Windows

When a split area acts as a source of a drag & drop operation, then this event will be triggered. It is left to the IDM programmer to program out the semantics of such an action.

In addition to the usual information that is delivered in thisevent (see drag & drop), information about the child index to which the split area belongs is also spitbox-specific delivered.

34.3.3.2 EM_paste

Only Valid for MS-Windows

If something is dropped over a split area during a drag & drop operation, then an EM_paste-event will be sent. The IDM programmer can react to this with his *on paste* rule and can carry out the needed actions.

In addition to the usual information delivered in thisevent, information concerning the child index which belongs to the spilt area is also splitbox-specific delivered.

34.3.3.3 EM_select

When a splitbox is selected with the mouse within the split area, the EM_select event is triggered. The child index (one-based), which belongs to the area in question, is found in thisevent.index. If the selection took place on a splitbar or on the border of a splitbox, then *nothing* will appear in thisevent.index.

34.3.3.4 EM_resize

MS-Windows Behavior

In order to change the size of a splitbox interactively, the mouse cursor must be placed on one of the splitbars. As shown in the example below, the look of the cursor will change when the positioning is correct.

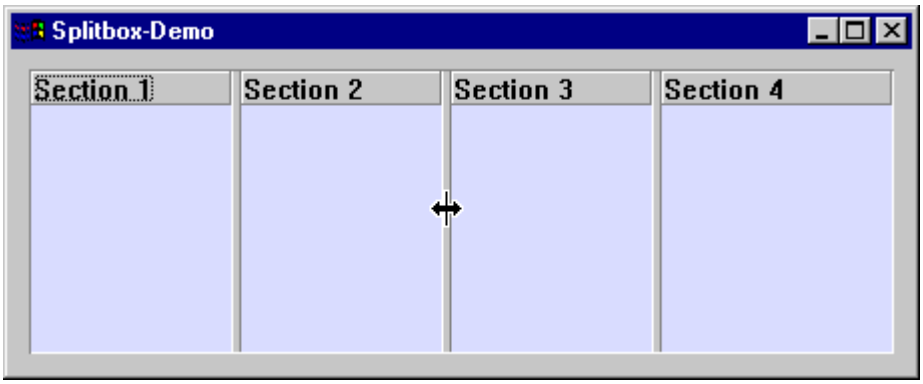


Figure 10: *Mouse cursor above a splitbar*

After correctly positioning the cursor on the splitbar, press down the far left button of the mouse and move the splitbar either to the right or the left (in this exercise) depending on the size you want to achieve. In the following example one can see how a transparent splitbar appears, which moves together with the cursor, in order to help direct the user when changing the size of an area within a splitbox. If the splitbar is moved too far to the right or left (i.e. if the minimal and maximal values are violated), then the bar being moved will remain at the last valid position.

On MICROSOFT WINDOWS the resizing action can be aborted with the `Escape` key.

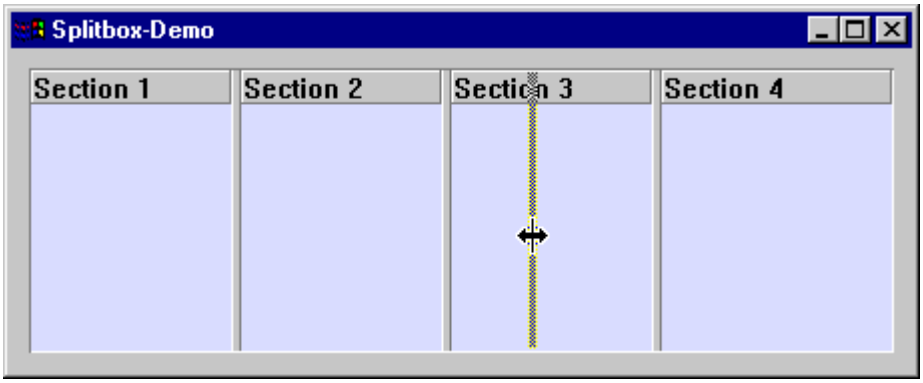


Figure 11: *Moving the splitbar*

When the desired size has been reached, the left mouse button can be released. In doing so this sends a message to the splitbox that a new size is to be set. Example 12 documents the end result of such an action.

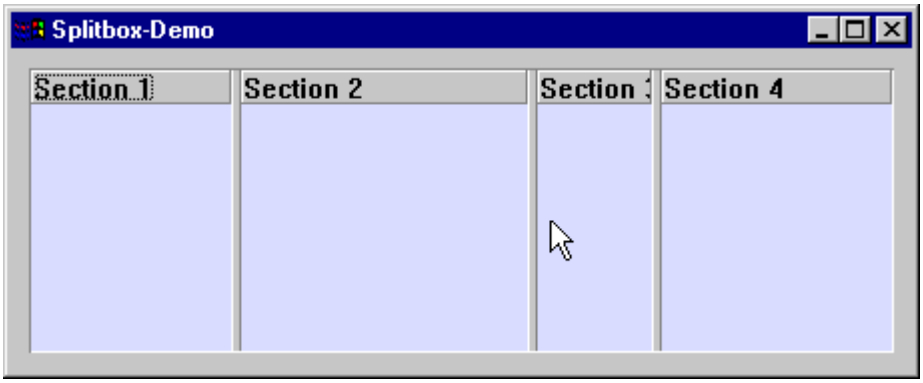


Figure 12: *Result of the resizing*

Now, no more than two EM_resize-Events are sent. One, which shows that the size of the second split area has been changed (relevant here is the new value in size[I]-vector; if this value remains unchanged, no event will be triggered). In thisevent.index the child index (one-based) of the affected split area is registered. The second EM_resize-Event is devoted to the change in the third area. Accordingly, the index of the third child is entered in thisevent.index.

Special Features

When .sizeraster is set to true on an object, this has no initial effect during the area sizing action. The splitbar is simultaneously moved with the mouse cursor. Only after the splitbar is relinquished will it snap over to the proximate upper left raster position so that the neighboring area can be brought back to raster size.

For Motif: this must be specified.

34.3.4 New Attributes

34.3.4.1 size[I]

	Rule Language	C	COBOL	Properties
Identifier	size[I]	AT_size	AT-size	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: object-specific attribute

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, falls 0<=I and I<=childcount

If the size [I]-vector is set before the child is created (typical for the widespread notation in the Rule Language, whereby the object attribute is set before the children), then the index can only be increased from one setting to the next by one.

Example

```
child splitbox {
  .size[0] 50;
  .size[1] 100;
  .size[2] 80;

  child .. Ch1 {}
  child .. Ch2 {}
}
```

size[0] is the so-called zero-element, that no split area is assigned to, and which is only there for convenience. The value of the zero-element is transferred to the other elements of the size[I]-vector for which no explicit setting exists. This allows for all default values for every element in the size [I]-vector to be changed by just one setting.

The size[]-vector shrinks each time a child is destroyed. Therefor, when the number of children is reduced, the elements in .size[I]-vector are deleted. This is true for: I>.childcount. In the case that new children are created, new elements will be added at the end of size[I]-vector. These elements inherit their values from the default element size[0].

Please note: The sum of the dilations of all split areas plus the sum of the widths of all splitbars should equal the width (if .direction=1) or the height (if .direction=2) of the splitbox. In reality, this is rarely the case. This is why it is sometimes necessary to ignore the values in size[I]-vector in some split areas (see Section "Calculating the Size of Split Areas"). The actual dilation of a split area can be called up with the attribute .real_size[I].

References and Dependencies

splitbox, minsize[I], maxsize[I], real_size[I], childcount

34.3.4.2 minsize[I]

	Rule Language	C	COBOL	Properties
Identifier	minsize[I]	AT_minsize	AT-minsize	SGCI
Data Type	integer	DT_integre	DT-integer	

Value Range:	0..65536
Default Value	0
Classification	Object-specific attribute

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. The index is zero-based. The valid values for this are:

minsize[I] – valid, if 0<=I and I<=childcount

If the minsize[I]-vector is set before the children are created (typical for the popular notation in the Rule Language whereby the object attribute is set first and then the children), then the index can only be increased from one setting to the next by one.

Example

```
child splitbox {
  .minsize[0] 50;
  .minsize[1] 100;
  .minsize[2] 80;
  child Ch1 {}
  child Ch2 {}
}
```

minsize [0] is the so-called zero-element, that no split area is assigned to, and which is only there for convenience. The value of the zero-element is transferred to the other elements of the minsize[I]-vector for which no explicit setting exists.

The minsize[]-vector shrinks each time a child is destroyed. Therefor, when the number of children is reduced, the elements in .minsize[I]-vector are deleted. This is true for: I>.childcount. In the case that new children are created, new elements will be added at the end of minsize[I]-vector. These elements inherit their values from the default element minsize[0].

Please note: The sum of the dilations of all split areas plus the sum of the widths of all splitbars should equal the width (if .direction=1) or the height (if .direction=2) of the splitbox. In reality, this is rarely the case. This is why it is sometimes necessary to undershoot the values in minsize[I]-vector in some split areas (see Section "Calculating the Size of Split Areas"). One only needs to think of the special case that the sum of all minimum values is already larger than the dilation of the splitbox itself. In order for all split areas to be visible, the minimum values must be undershot. The minimum values can not be undermined by the interactive movement of the splitbar.

References and Dependencies

splitbox, size[I], maxsize[I], real_size[I], childcount

34.3.4.3 maxsize[I]

	Rule Language	C	COBOL	Properties
Identifier	maxsize[I]	AT_maxsize	AT-maxsize	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: Object-specific attribute

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. The index is zero-based. The valid values for this are:

maxsize[I] – valid, if 0<=I and I<=childcount

If the maxsize[I]-vector is set before the children are created (typical for the popular notation in the Rule Language whereby the object attribute is set first and then the children), then the index can only be increased from one setting to the next by one.

Example

```
child splitbox {
  .maxsize[0] 50;
  .maxsize[1] 100;
  .maxsize[2] 80;

  child .. Ch1 {}
  child .. Ch2 {}
}
```

maxsize[0] is the so-called zero-element, that no split area is assigned to, and which is only there for convenience. The value of the zero-element is transferred to the other elements of the maxsize[I]-vector for which no explicit setting exists.

The maxsize[]-vector shrinks each time a child is destroyed. Therefor, when the number of children is reduced, the elements in maxsize[I]-vector are deleted This is true for: I>.childcount. In the case that new children are created, new elements will be added at the end of maxsize[I]-vector. These elements inherit their values from the default element maxsize[0].

Please note: The sum of the dilations of all split areas plus the sum of the widths of all splitbars should equal the width (if .direction=1) or the height (if .direction=2) of the splitbox. In reality, this is rarely the case. This is why it is sometimes necessary to undershoot the values in minsize[I]-vector in some split areas (see Section "Calculating the Size of Split Areas"). One only needs to think of the special case

that the sum of all minimum values is already larger than the dilation of the splitbox itself. In order to avoid the emergence of empty spaces, the last split area must be lengthened accordingly.

References and Dependencies

splitbox, minsize[I], size[I], real_size[I], childcount

34.3.4.4 real_size[I]

	Rule Language	C	COBOL	Properties
Identifier	real_size[I]	AT_real_size	AT-real-size	G
Data Type	integer	DT_integer	DT-integer	

Value Range: 0..65536

Default Value: 0

Classification: object specific-attribute

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:

real_size[I] – valid, when 1<=I and I<=childcount

This attribute, as well as all other real_...-attributes, can first be called up after the object is created and is visible on the screen. If real_size[I] is called up from an invisible child (because of this whose section has not yet been created), the value 0 will be returned.

References and Dependencies

splitbox, .childcount, minsize[I], maxsize[I], size[I]

34.3.4.5 direction

	Rule Language	C	COBOL	Properties
Identifier	direction	AT_direction	AT-direction	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range: 1..2

Default Value 1

Classification object-specific attribute

This attribute controls the alignment of the splitbars. If .direction = 1, then the splitbars will be displayed vertically and can be moved either to the left or the right. If this attribute is set to 2 then the

splitbars are displayed horizontally and can be moved either up or down. The attribute may be transferred at run-time, whereby then a reorientation of the splitbars and thus the split areas will take place.

Under Motif 1.2 the splitbars can only be aligned horizontally. On account of this the value in .direction is ignored.

References and Dependencies

splitbox

34.3.4.6 barwidth

	Rule Language	C	COBOL	Properties
Identifier	barwidth	AT_barwidth	AT-barwidth	SGCI
Data Type	integer	DT_integer	DT-integer	

Value Range:	1..65536
Default Value	5
Classification	object-specific attribute

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 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.

References and Dependencies

splitbox

34.3.5 Reference

With regard to splitboxes, special attention should be paid to the layout, geometric management, and the use of boundary conditions such as activated size rastering or the employment of minimal and maximal values.

34.3.5.1 Size Rastering

In contrast to other objects, which allow for the use of attributes such as .xraster, .yraster and .reffont, the raster grid of the splitbox is not applied to the entire object but rather to the individual split areas. This helps in avoiding problems with rounding differences which would be unavoidable if one had to align separate split areas on global raster grids while at the same time having to situate splitbars

(whose values are given in pixel values) in the pattern. Having to adjust the width of the splitbars in raster sizes is not so attractive (it just looks really bad). The use of the raster grid locally only for the split area in question seems to solve this problem. In addition, the calculation rule used up to now for determining pixel values can be simplified. In general: pixel value = (raster value – 1)*raster. For the size calculation of the split area: pixel value = raster value*raster.

34.3.5.2 Calculating the Size of Split Areas

For calculating the actual size of a split area the following algorithm is used.

1. The first step is to try to assign each and every area the size that is indicated in the attribute .size [I].
2. If after step one it is determined that more space (split area width plus splitbar width) was used as was provided for through the definition of the geometric attributes such as .width, .height, .xauto, .yauto, etc., then the split areas were cut down to size under consideration of the minimal values allowed beginning from the bottom/right side until the space restrictions were filled. If this can not be achieved, for example because the sum of all minimal (minsize[I]) values already surpasses the scope, then the split areas will be gone through again beginning with the right side. This time these will be cut without considering the minimal values. On the lower/righ side a row of split areas having the size 0 can emerge. Here, one only sees the leftover splitbars. At the end of this process it is possible that although all split areas set to 0 have been reduced, that there still is not enough space (the sum of the width of the splitbars is larger than the dilation of the splitbox). In this case, the split areas including their splitbars will be moved to the bottom/right outside of the visible area (the same amount of space that is lacking).
3. If after step one it is determined that less space was used as was provided for through the definition of the geometric attributes, then the split areas will be enlarged so that the entire area in the splitbox is utilized (but not larger than the maximal size allowed). If this goal is not attained on the first try, for example because the sum of all maximal values and the width of the splitbars is already smaller than the dilation of the splitbox, then the last split area will be enlarged as needed to fill up the space regardless of what size values it was originally given.

Keep in mind that during this process the attributes .size[I], .minsize[I], and maxsize[I] are not changed. This means that the attributes still have the values given to them by the IDM programmer, which were only partially considered due to the averse conditions. This property allows for a new calculation of the actual split area widths after each manipulation of the allowed dimensions. If, for example, a splitbox is attached to a window with .xauto=0 and .yauto=0, then one can observe when interactively reducing the window size how the split areas, one after the other, are reduced with and then without consideration of the minimal values. If the window is enlarged again, then the splitbars will return to their original positions.

The values in .size[I] are only changed when these do not agree with the minimum/ maximum values in minsize[I]/maxsize[I]. The values in minsize[I] have a higher priority for the IDM as the values in maxsize[I].

In the case that the minimum and maximum values are violated, as described in steps 2 and 3 above, the affected splitbars can no longer be interactively moved since they are already in a position that is not permitted. Through the grasping of and letting go of the various splitbars, more clearly those in .size[I]-vector, the size of the splitbar in the split area in question will receive new values. This results in the triggering of EM_resize events, although optically nothing has changed; except for the attribute in .size[I]. In any case, the IDM programmer is informed that the values he had set in size[I]-vector have been changed by the end user. When the end user tries to move the same split bar again, optically speaking nothing will change since the minimum and maximum values originating from the set restrictions are still valid. This time, no further EM_resize events will be sent since the values in size[I] were suitable set after the first try.

A special feature which should be taken into account is that dealing with size rastering. Since it is basically impossible to perfectly fit all split areas to the raster size, according to the alignment process described above, as a rule only one split area exists whose size does not fit to the multiple raster values. This is true for the split area that was lastly changed according to the algorithm described above.

The following is an example of this situation:

A splitbox exists with the following attributes and values: .direction=1, .size[0]=10, minsize[0]=2. Size rastering is activated and the raster setting on the father object is: .xraster=10. The splitbox is tied to the father window with the following values: .xauto=0 and .yauto=0 and is connected to the father window. The splitbox can be seen in “Figure 13”. One can observe that not all values from .size[I]-vector were filled due to external circumstances. The two split areas to the right have already reached their minimum size values (20 pixel). The first split area has the desired size of 100 pixel. The second split area from the left was the last one to be changed by the algorithm and therefor no longer fits in the raster grid.

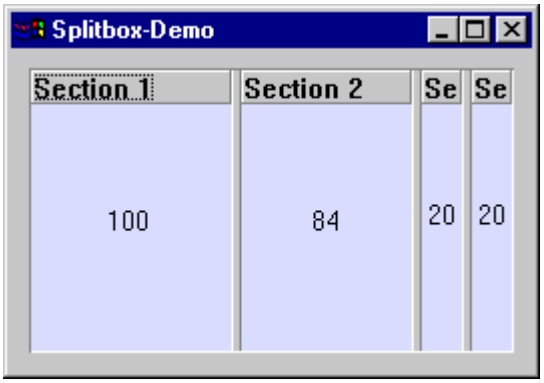


Figure 13: *Size calculation of the split areas*

34.3.6 Example Dialog

A simplified use of a splitbox object can be described as seen below:

```
dialog Dialog { }
window W
{
.title "Splitbox-Demo";
```

```
.width  400;
.height 400;
child splitbox Splbox {
.xauto  0;
.yauto  0;
.xleft  5;
.xright 5;
.ytop   5;
.ybottom 50;
.size[0] 50;
.minsize[0] 20;
.maxsize[0] 100;

child pushbutton Pb1 {
.text "in Section 1";
}

child statictext St1 {
.text "in Section 2";
.yauto -1;
.xauto -1;
}

child groupbox Gb1{
.xauto  0;
.yauto  0;
.xleft  4;
.xright 4;
.ytop   4;
.ybottom 4;
}

child treeview Tv {
.xauto  0;
.yauto  0;
.xleft  4;
.xright 4;
.ytop   4;
.ybottom 4;
.content[1] "Eins";
.content[2] "a";
.content[3] "b";
.content[4] "c";
.content[5] "Zwei";
.content[6] "d";
.content[7] "Drei";
```

```
.content[8] "Vier";
.content[9] "e";
.content[10] "f";
.content[11] "g";
.level[2] 2;
.level[3] 2;
.level[4] 2;
.level[6] 2;
.level[9] 2;
.level[10] 2;
.level[11] 2;
.firstchar 1;
.editable true;
.style[style_lines] true;
.style[style_buttons] true;
.style[style_root] true;
.selstyle multiple;
}
}
on close {exit();}
}
```

34.4 New Object Subcontrol

OLE-Objects that can be used as servers within IDM with the help of IDM ***Control*** (please refer to the documentation IDM OLE Interface), have as a rule a rather complex structure and can consist of a series of further child objects. For example, OLE-Control “Word.Application” has a collection of “Documents”, where the documents are kept. In turn, these documents have “Words”, “Sentences”, “Ranges” etc. as their own children. In order to be able to access these sub-objects from the Rule Language, i.e. to call up methods or attributes, the ***Subcontrol*** object was introduced. Thus, the ***Subcontrol*** represents an OLE child object which can either be directly or indirectly administered from an OLE server. Therefor, the starting point for accessing such an object is always the ***Control***.

Definition

```
{export} subcontrol {<Identifier>}
{
  Object-specific attribute
}
```

Events	none
Children	subcontrol
Father	control

	subcontrol
Menu	no

34.4.1 New Attributes

Attribute	Data Type	Value Range	Properties	Short Description
connect	boolean	true, false	S,G/D/C	This prompts the subcontrol to establish a connection to an OLE server (true), or to disconnect an existing connection (false). Default is false.

The default value for the attribute is false. This means that there is no current connection to an OLE server. If the attribute is set to true, the subcontrol will try to establish a connection to an OLE server (normally a collection). If the attribute is changed to true the subcontrol will try to connect to an OLE server (normally a Collection) that the father object belongs to. The identifier of the subcontrol must comply with the attribute name of the OLE server that was named first.

In the following example, the document collection of Word OCX is connected:

```
dialog WordClient

window WnFenster {
  .title "WordClient";

  on close {
  exit;
  }

  control CoWord {
    .mode    mode_client;
    .name    "Word.Application";
    .visible true;
    .connect true;

    child subcontrol Documents {
    }
  }
}

on dialog start {
  CoWord.Visible := true;
  Documents.connect := true;
  print Documents.connect;  // setpoint: true
}
```

A connection can only be successfully established when the father is realized and connected, and the identifier of the subcontrol agrees with the name of the OLE server attribute.

If the attribute .connect is set to false, the connection to the OLE server will be terminated. With the current implementation the children of a subcontrol will still remain connected!

	Rule Language	C	COBOL	Properties
Identifier	connect	AT_connect	AT-connect	SGCI
Data Type	boolean	DT_boolean	DT-boolean	-

Value Range:	true, false
Default Value	false
Classification	object-specific attribute

The attribute .connect can only be set to true when the father control or the subcontrol is connected. If the father looses his connection when the attribute has already been set to true, this will have no further effect.

References and Dependencies

control, visible

34.4.2 References

34.4.2.1 Implicit Creation of Subcontrols

The most prominent feature of the subcontrol is that these objects can be implicitly created without having initially defined the object as static in a dialog or dynamically created it with **create()**. Supposedly, the implicit employment of this object is used the most in actual practice. Thus, for example, one can access separate documents by using Word as an OLE server:

Example

Let's say that ***control*** "Co" is defined somewhere in the Dialog.

```
...
child control Co
{
    .mode mode_client;
    .name "Word.Application";
    .visible true;
    .connect true;
}
...
```

If this is true, then the following could be found in a rule:

```
rule OLETest ()
{
    variable object Doc;
    Doc := Co.Documents:Add();
    // Word has a Collection that is entitled Documents.
    // Co.Documents is used to access this OLE object. In order to
    // address the object in IDM, a subcontrol with the identifier
    // "Documents" is implicitly created which is then registered
    // from Control Co as a child. The :Add()-method is a member of
    // the collection entitled "Documents". Therefor, it is
    // possible to apply it to the recently created subcontrol.
    // Thereby, a new document will be created in Word, and as a
    // return value a cursor on IDispatch-Interface of this
    // document is delivered. In order to have an equal counterpart
    // on the IDM side, here too a subcontrol is created as a child
    // of the first subcontrol, which itself is connected to the
    // Word document. This object will ultimately be assigned to
    // the variable "Doc".

    // Now, the subcontrol in Doc can be used for calling up the
    // methods of the OLE documents or to set or call up its
    // properties.
    print Doc.FullName;
}
```

In general, the following is true:

If a method or a property returns a pointer to an IDispatch interface, or if this is transferred to the IDM as a parameter, then the IDM will create a subcontrol here that is connected to the IDispatch interface. This means that the *.connect* attribute of such a subcontrol is already set to *true*.

As a rule the identifier (label) of such a created object is not set so that something similar to *“sub-control Co.SUBCONTROL[2]”* will appear in the tracefile. One exception to this rule concerns the accessing of properties which OLE objects return. Since these usually represent Collections and are therefor well-known to the IDM, an implicitly created subcontrol, in this case an identifier, then receives the name of the property.

Example

```
Doc1 := Co.Documents:Add();
Doc2 := Co.Documents:Item(1); // Assumption: No other Word documents
                             // were open beforehand.
```

In this example, two subcontrols are created in the first line: One with the identifier “Documents”, another with the identifier “SUBCONTROL”. The identifier of the second subcontrol thereby explains itself that the IDM, in working with the Add() method, has no further available information. The only thing that is created in the second line is a subcontrol with “SUBCONTROL[2]” as its identifier. The creation of a subcontrol for Documents does not apply since the IDM itself realizes that the object already

exists. On account of this the following situation comes about. On the OLE side we have two objects: one is the Collection “Documents”, and the other is an blank document. On the IDM side we have three objects: a subcontrol for the Collection “Documents” and two subcontrols in the variables Doc1 and Doc2, which are connected to the blank document on the OLE side. Therefor it is left up to the IDM programmer to avoid a flood of unnecessary subcontrols through creative programming.

A concerned reader might ask himself at this point what happens with all these subcontrols, if they are no longer needed. This will be discussed in the following section.

34.4.2.2 Garbage Collection for Subcontrols

All implicitly created subcontrols must also be deleted again. For this purpose, the IDM makes a note of how many subcontrols were just referenced from an object. If it is then established that a subcontrol is no longer is use, either from a variable (local, global, static) or a user-defined attribute, then the subcontrol will be deleted. There are two strategies concerning this topic which are explained below. Important: The following statements are written with reservation because it is not foreseeable if other strategies will be used in the future.

1. If an implicitly created subcontrol is assigned to a variable (or something similar), and if this variable is overwritten at a later date with another value, then the subcontrol can once again be released provided that it has no children.

   Example

   ```
   Doc := Co.Documents:Add();  // 2 Subcontrols (A for documents and B for a
   document that
                               // is created from Add()) will be implicitly
   created.
   Doc := Co;                  // The subcontrol B is no longer used from
   variable Doc
                               // and can therefor be destroyed. Now subcontrol
   A
                               // can determine that it no longer has children
   and
                               // it can also destroy itself.
   ```

2. By unfavorable constellations it can happen that a subcontrol does not notice that it can be destroyed. This happens, for example, when a newly created subcontrol has not been assigned anywhere. In this case, the triggering moment is lacking in order to throw away the object. For this reason the IDM occasionally will start a so-called cleaning action. Here, implicitly created subcontrols are tested if they are no longer needed. If this tests true, they will be destroyed.

Important

Since the IDM garbage collection only takes into consideration the references from the Rule Language, great care should be taken, if ObjectIDs of subcontrols are managed from the C interface (this is also true for COBOL and C++). If such an ID is temporarily stored in C, IDM does not realize this. After the calling up of the C function, the IDM was able to realize that the subcontrol was no longer needed and then disposed of it accordingly. In the case that the control is once again transferred to

the C side, then it is no longer presumable that the previously saved ObjectID is still valid. Therefor, one should try to avoid saving the subcontrols in C. Although, if one still wishes to do this, then it is important to prove that the object is still referenced in the Rule Language (for example in a global, in a static variable or in a user-defined attribute). Only then can there be a guarantee that the subcontrol will not be thrown away.

34.4.3 Example Dialog

The following example shows how subcontrols can be employed.

```
dialog Word

window WnWindow
{
  .active false;
  .width 400;
  .height 100;
  .title "Word.Document.8";

  on close
  {
      if Co.connect then
        Co:Quit();
      endif
      exit();
  }

  child control Co
  {
    .visible true;
    .active false;
    .xauto 0;
    .xleft 20;
    .xright 20;
    .yauto 0;
    .ytop 20;
    .ybottom 40;
    .mode mode_client;
    .name "Word.Application";
    .connect true;
  }
  child pushbutton PbOpen
  {
    .xauto 1;
    .xleft 100;
    .width 76;
```

```
  .yauto -1;
  .height 27;
  .ybottom 10;
  .text "Open Doc";
  on select
  {
      variable object Doc:=null;

      // Prerequisite: the file must exist
      Doc := Co.Documents:Open("d:/tmp/altesdokument.doc");

      if Doc <> null then
          print "DocName: " + Doc.FullName;
          print "DocSaveStatus: " + Doc.Saved;
      endif
  }
}
child pushbutton PbAdd
{
  .xauto 1;
  .xleft 20;
  .width 76;
  .yauto -1;
  .height 27;
  .ybottom 10;
  .text "New Doc";
  on select
  {
      Co.Documents:Add("d:/tmp/neuesdokument.doc");
      // Prerequisite: the file must exist
  }
}
child pushbutton PbClose
{
  .xleft 180;
  .width 92;
  .yauto -1;
  .height 27;
  .ybottom 10;
  .text "Close Doc";
  on select
  {
      // First (regarding the document list) close the document
      Co.Documents:Item(1):Close();
  }
}
```

```
  child pushbutton PbQuit
  {
    .xauto -1;
    .width 85;
    .xright 20;
    .yauto -1;
    .height 27;
    .ybottom 10;
    .text "Quit";
    on select
    {
        // close Word
        Co:Quit();
    }
  }
}
on dialog start
{
    Co.Visible := true;
}
```

34.5 Motif

- An object Notebook with Notepage was included with the following limitations:
  - Beginning with version Motif 2.1
  - .binding: in addition to bind_solid and bind_spiral there is a new value
    - bind_none: the binding is completely omitted
  - Under Motif one can control the number of notepages (XmNbackPageNumber). This is done with the attribute .binding:
    - .bind_solid 3 pages will stick out,
    - .bind_spiral 3 pages will stick out,
    - .bind_none no page will stick out.
  - .tabshape: is not supported
  - .font: this is set at the same time with the notepages.
  - .majortabheight not supported
  - .majortabwidth not supported
  - .minortabheigth not supported
  - .minortabwidth not supported
  - .multiline not supported
- Notepage:

- .tabtype: under Motif it is distinguished between major_tab and minor_tab.
- .multiline is not supported under Motif.
- .tabalignment is not supported under Motif.

- Object Treeview with the following limitations:
  - style_root is always set. By style_lines and style_buttons the superior father is always taken into consideration.
  - .picture can use tiles with various heights and widths. The tiles are not scaled to a basic size.

34.6 Microsoft Windows NT

- Gnats 8302: An OLE-Control WangImage.EditCtrl.1, of which no properties were set, generated a messagebox with an error message when this was set .visible = true. The reason for this was that the IDM asked for the metafile of this control.
- Gnats 8326: When a combobox with poptext style was opened up, and the user ran the mouse pointer over the various items in the list, the marking moved with the pointer but when the `Return` key was pressed, the marked item was not presented in the title of the combobox. This has now been changed, now the marked item is transferred to the combobox header.
- Event open when opening a menubox. Important: When processing the event, neither the menubox nor any other superordinate object within the box can be changed! Sometimes this can randomly work, but this is not always the case.
- The attributes .dockable and .docking of the toolbar can now be set from the Rule Language independent of one another. If the values are in contradiction of one another, then the attribute .docking has priority.
- New Attributes
  - `.picture[I]  identifier  tile  S,G/D/C`
  - `.picture_hilite[I]  identifier  tile  S,G/D/C`
  - `.picwidth  integer  integer  S,G/D/C`
  - `.picheight  integer  integer  S,G/D/C`

  on the object listbox and poptext. Pictures having the size .picwidth multiplied by .picheight pixel are represented next to the listbox entries and poptexts, but not in the edittext of an editable poptext. Transparency is supported. Marked (usually accented with a blue background) entries display pictures from .picture_hilite[I], and unmarked pictures from .picture[I]. The attributes .picture [I] and .picture_hilite[I] are inherited by poptexts, but not by listboxes (analog entry texts). Both attributes can only be indexed with such values to which an entry already exists!
- The following changes were made to the drag & drop functionality:
  - Scrolling of objects: groupbox, notepage, treeview, listbox, edittext(multiline), tablefield, window during drag & drop (wish from gnats 6739, 8155). Visual feedback of objects: treeview, listbox, edittext.

- Spinbox (with child Edittext), tablefield, when these act as a goal of a drag & drop operation.
- In the on paste rule for edittext and spinbox, thisevent.index now contains a two dimensional index with: first(thisevent.index) = the character position in which it is dropped. It is one-based. If one drops at the beginning of an edittext, it is position 1. second(thisevent.index) = the line number in which it was dropped. This too is one-based. => It is always on 1 by single-line edittexts.
- In the on paste rule for groupbox, notepage, and window, thisevent.index now contains a two-dimensional index: first(thisevent.index) = the X pixel coordinates, in which it was dropped. second(thisevent.index) = the Y pixel coordinates, in which it was dropped.
- (Wish from gnats 5820) In the resources source and target type_file can also be used alongside type_object and type_text. Thus, it is possible to transfer external file names per drag & drop (i.e. from Explorer) into an IDM-application or to transfer the file names out of IDM.

- Gnats 7777: Copy & paste was not available by an edittext that was set to .target. `Ctrl` + `C` and `Ctrl` + `V` also did not function. Reason: faulty implementation of AT_index-Tasks in the Class Editable. Copy & paste is now in working order.
- Gnats 7778: A repeated paste in an object that is used for drag & drop (.source/.target set) lead to a crash. This was due to a storage leak of the transfer structure when this was put into the event queue (EM_Paste). Another reason was due to an insufficient amount of storage for the transfer string, and the strings were then not terminated with NULL when the transfer string contained newlines (EM_Cut or EM_Copy).
- Gnats 7937: A tablefield containing .selstyle = single and .selection[sel_column] or .selection[sel_row] was not able to be dragged. This is once again in working order. As in the past it is still necessary that a rectangular, interrelated area is copied or dragged.
- Gnats 7999: Notepage and groupbox can be used as a source of the drag & drop operation.
- Attribute *.borderwidth* on the ***image***. If *.borderwidth = 0*, then the image will appear in a non-pressed state without any borders. This is true if it is sensitive. If *.borderwidth <> 0* then the behavior will remain unchanged. Default is *1*.
- Interactive docking and positioning of toolbars.

34.7 Core

- When the attribute .parent is set, it is now tested if such a setting is valid, i.e. if cycles in the object tree come about. Only after this has been checked is it possible to hang the object on a new father.
- A number of new methods have been added and some existing methods have been corrected:
  - :create() cannot be redefined; is equivalent to the built-in function.
  - :destroy() cannot be redefined; is equivalent to the built-in function.
  - :init() is now carried out directly after a module is started; but still before the "on start" rule. Naturally, this method is called up for this object after the creation of an object. All further properties remain the same. A method can be over-defined by a user. The pre-defined method

doesn’t do anything other than call up this:super(); this is the same as: :init() { this:super(); }. If this:super() is always called up in the entire inheritance hierarchy within the :init() methods, then this:super() makes sure on the highest level that :init() methods of the children are also called up. If this chain is somehow broken by :super()-call up, then the children will remain non-initialized.

- :clean() is new and can be over defined by the user. The predefined methods do nothing more than call up this:super(); this is the same as: :clean() { this:super(); }. If the :clean() method is always called up in the entire inheritance hierarchy within this:super(), then this:super() makes sure at the highest level that :clean() methods of the children will also be called up. If this chain is somehow broken by :super()-call up, then the children will not be "cleaned". :clean() This method is only implicitly called up: Directly before the destruction of an object and before the unloading of a module/dialog, but after the "on finish" rule has been run.
- The :set() method can be over defined by the user. The predefined methods only carry out the setting of the affected method through this.X := value;. Recursive calls of :set() for the same object are prevented. :set() can be called up once again for the same object with :recall(). Methods can be explicitly called up. They are implicitly called up when a setting of an attribute of an object is carried out by the Rule Language or the DM interface functions (similar to on changed).
- The :get() method can be over defined by the user. The predefined method only returns the value of the attribute that was called up. Recursive calls of :get() for the same object are prevented. A recursive call can take place with :recall(). Methods can be explicitly called up. They are implicitly called up when an attribute is requested from the Rule Language or the DM interface functions.
- :setinherit(): the method can be over defined by the user. The predefined method only carries out the setting on the affected attribute this.X := inherited value; and Inheritbit is then set appropriately. Recursive calls of :setinherit() for the same object are prevented. Methods can be called up recursively with :recall(). Methods can only be explicitly called up (:setinherit() or built-in function setinherit()).
- :recall(): the method cannot be over defined. This is basically equivalent to the semantic of :call (). This method can force :set(), :get(), :setinherit() to be recursively called up for the same object.

- Toolbar, notepage, and groupbox accepted the statusbar as a child. Notepage, groupbox accepted the toolbar as a child.
- The porting version no longer contains "idm"-program, but rather only a "pidm". Now, dialogs can no longer simulate since porting versions are not intended for developing applications.
- Gnats 8516: Getvalue on .groupbox only delivered a superordinate groupbox and not a superordinate window/notebook/notepage as before.
- Gnats 8509: consistency has been established between access and issuance from ambiguous, inherited identifiers.

34.8 Editor

- The statusbar was able to be placed freely in the window with the right button on the mouse, and was then corrected by the first resizing of the window. Now a statusbar is always placed at the bottom of the window.
- Multi-selection in the filerequester for loading dialogs under NT is possible. In addition, a drag & drop target for files is now possible in the file list
- It is possible to get to the "referencing window" either through the menuitem "show referencing" in the edit menu in the object attribute window, or through the pop-up menu of the object browser in the main window. This then shows all references for this particular object from the dialog/module, and from the loaded submodules. This "object list" is sub-grouped according to the incidences in the files. If a referencing appears in a sub-module, which has not already been edited, then this file is marked as a "sub-module" of the main file. Referencing refers to the use of objects in an attribute, or as an absolute or unambiguous path rule code. Thus, the window shows, non-relative accessing of the object and as a result indicates which other objects are affected when this object is deleted or if the exportation is taken away. The right side of the window shows in detail in which attributes the referencing takes place. The edit button allows for the changing over to the secondary window which corresponds to the object (object attribute window, rule window, etc...). In order to make clear to the user that the removal of exports can have far-reaching consequences, the export flags in the secondary window and in the object browser are secured through a warning/question. This warning can be turned off in the editing page of the configuration window in "export warning" field.
- Gnats 8517: Extension .dlg/.mod is hanged automatically when saving a dialog or module. The problem of having incorrect characters hung onto selected files when using the filerequester has been solved.
- Gnats 8491: Selection in the rule selection list has been debugged, so that "save as" correctly saves the selected rule.
- Gnats 8498: The editor no longer renames write-protected files. The overwriting of or deletion of write-protected files from the editor is no longer possible.
- Gnats 8502: The labeling of the filerequester patterns has been corrected.
- Gnats 8489: Update problem with variable & application has been corrected. No further warnings by the selection of a variable or when creating an application.
- Gnats 8492: The menu "save as" in the source text/model editor/attribute window is no longer visible.
- Gnats 8494: The attribute window cuts paths that are too long. This is indicated by..., so that more important information can be displayed in the title. The spacing is automatically fitted to the window size.
- Gnats 8504: The location in the child vector remains the same by re-parsing in the source text window.
- Gnats 8505: The geometry side for the control object is switched to visible.

- Gnats 8506: The code window no longer allows for the editing of defaults. Inherited identifiers are not transformed during re-parsing in the code window.
- Gnats 8511: When objects/resources/applications are deleted with the delete button in the secondary window, then the file will be marked as changed.

34.9 IDD

- Gnats 8321: See Debugger section.

34.10 Debugger

- The status files of the debugger now have the default extension "dbg". The filter files of the profiler now have the default extension "idp".
- Gnats 8321: The idmdbg and the idd filerequester memorizes the directory in which he was last present (by the last OK). Here, the new filereq object is used; the old filerequester was either deactivated or deleted.