

ISA Dialog Manager

RELEASE NOTES IDM 5

A.06.03.b

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

Microsoft, Windows, Windows 2000 bzw. NT, Windows XP, Windows Vista, Windows 7, Windows 8, Windows 10 and Windows 11 are registered trademarks of Microsoft Corporation

UNIX, X Window System, OSF/Motif, and Motif are registered trademarks of The Open Group

HP-UX is a registered trademark of Hewlett-Packard Development Company, L.P.

Micro Focus, Net Express, Server Express, and Visual COBOL are trademarks or registered trademarks of Micro Focus (IP) Limited or its subsidiaries in the United Kingdom, United States and other countries

Qt is a registered trademark of The Qt Company Ltd. and/or its subsidiaries

Eclipse is a registered trademark of Eclipse Foundation, Inc.

TextPad is a registered trademark of Helios Software Solutions

All other trademarks are the property of their respective owners.

© 1987 – 2024; ISA Informationssysteme GmbH, 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 derivatives, 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> | either <A> or

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>

$\langle \text{digit} \rangle ::= 1 \mid 2 \mid 3 \mid \dots 9 \mid 0$
 $\langle \text{lowercase} \rangle ::= a \mid b \mid c \mid \dots x \mid y \mid z$
 $\langle \text{uppercase} \rangle ::= A \mid B \mid C \mid \dots X \mid Y \mid Z$

Table of Contents

Notation Conventions	3
Table of Contents	5
1 Version A.05.02.o	17
1.1 Windows	17
1.2 Motif	18
1.3 Core	18
1.4 Debugger	18
1.5 USW	19
1.6 COBOL	20
2 Version A.05.02.n	21
2.1 Windows	21
2.2 Motif	21
2.3 Documentation	22
3 Version A.05.02.m	23
3.1 Windows	23
3.2 Motif	23
3.3 Core	23
3.4 USW	23
3.5 ISA Form Designer (IFD)	23
4 Version A.05.02.l	24
4.1 Windows	24
4.2 Motif	25
4.3 Core	26
4.4 COBOL	26
4.5 Editor	26
4.6 Debugger	27
4.7 ISA Form Designer (IFD)	27

5 Version A.05.02.k	28
5.1 ISA Formular Designer (IFD)	28
5.2 New C Interface Function DM_StrCreate()	28
6 Version A.05.02.j	30
6.1 Windows	30
6.2 Motif	30
6.3 Core	30
6.4 COBOL	30
6.5 Reexport – Inheritance of export with hierarchical children	31
6.5.1 New keyword reexport	31
6.5.2 Supplementation of the Editor	33
7 Version A.05.02.i	35
7.1 Important Changes	35
7.2 Windows	35
7.3 Motif	36
7.3.1 New option .options[opt_scroll_on_focus]	37
7.4 Core	39
7.5 Network	39
7.5.1 IPv6 support	39
7.6 Debugger	40
7.7 COBOL	40
7.8 USW	40
8 Version A.05.02.h	41
8.1 Important changes	41
8.2 Windows	41
8.3 Motif	42
8.4 Core	43
8.5 Unix	44
8.6 Editor	44
8.7 New Tablefield Attribute .preeditssel	45
9 Version A.05.02.g	47
9.1 Windows	47
9.2 Motif	49
9.3 Core	50
9.4 Unix	51

9.5 Editor	51
9.6 USW	51
9.6.1 Extensions of the USW interface	52
9.6.1.1 Information on the use of resource with USW widgets	52
9.6.1.2 Preferred size and raster categories	52
9.6.1.2.1 Geometry calculation	52
9.6.1.3 Formatter functionality	52
9.6.1.3.1 Format handling	52
9.6.1.3.2 Functions for the use of formats for editable content	53
9.6.1.3.2.1 Use of the functions	53
9.6.2 New or changed data types and structures	54
9.6.2.1 Data Types	54
9.6.2.1.1 UC_ControlName (new)	54
9.6.2.1.2 UC_FormatFlags (new)	54
9.6.2.1.3 UC_FormatNaviMode (new)	54
9.6.2.1.4 UC_FormatStatus (new)	54
9.6.2.1.5 UC_RasterType (new)	55
9.6.2.2 Structures	55
9.6.2.2.1 UC_API (changed)	55
9.6.2.2.2 UC_ControlArg (new)	56
9.6.2.2.3 UC_Dimension (new)	56
9.6.2.2.4 UC_FormatDesc (new)	57
9.6.3 New Functions	58
9.6.3.1 applyFormat	58
9.6.3.2 control	59
9.6.3.3 fmtContent	59
9.6.3.4 fmtDisplay	60
9.6.3.5 fmtFocus	62
9.6.3.6 fmtModify	63
9.6.3.7 fmtNavigate	64
9.6.4 New Constants	66
10 Version A.05.02.f4	68
10.1 Motif	68
11 Version A.05.02.f3	69
11.1 Motif	69
11.2 Core	69
11.3 USW	69
12 Version A.05.02.f2	70
12.1 Windows	70
12.2 Motif	70
12.3 Unix	70

13 Version A.05.02.f	71
13.1 Compatibility	71
13.2 Windows	71
13.3 Motif	73
13.4 Core	74
13.5 COBOL	75
13.6 IDM Builder Process	75
13.6.1 The builder process mode	75
13.6.1.1 Example	76
13.6.1.2 Usage instructions	77
13.6.1.3 Usage information for the IDM Eclipse Plugin	77
13.6.1.4 Particularities	77
13.6.2 New parameters for IDM and PIDM	78
13.6.3 New IDM library parameter	78
14 Version A.05.02.e4	80
14.1 Motif	80
14.2 Core	80
15 Version A.05.02.e3	81
15.1 UNIX	81
15.2 Motif	81
15.3 Editor	81
16 Version A.05.02.e2	82
16.1 Windows	82
16.2 Core	82
17 Version A.05.02.e	83
17.1 Windows	83
17.2 Motif	84
17.3 Core	85
17.4 Compatibility	87
17.5 New Functions for Error Analysis	87
17.6 New Interface Function DM_GetToolkitDataEx()	87
18 Version A.05.02.d2	90
18.1 Windows	90
18.2 Motif	90
18.3 Core	90

19 Version A.05.02.d	91
19.1 Important Changes	91
19.2 Windows	91
19.3 Motif	94
19.4 Core	95
19.5 Editor	96
19.6 Java	96
19.7 Installation Guidelines Unix	96
20 Version A.05.02.c6	100
20.1 Core	100
21 Version A.05.02.c5	101
21.1 VMS	101
22 Version A.05.02.c4	102
22.1 Windows	102
22.2 Motif	102
23 Version A.05.02.c3	103
23.1 Windows	103
23.2 Motif	103
23.3 Core	103
23.4 Editor	103
24 Version A.05.02.c2	104
24.1 Windows	104
25 Version A.05.02.c	105
25.1 Important Changes	105
25.2 Windows	105
25.3 Motif	106
25.4 Core	107
25.5 Editor	107
25.6 Debugger	108
25.7 COBOL	108
26 Version A.05.02.b4	109
26.1 Windows	109
26.2 Core	109

27 Version A.05.02.b3	110
27.1 Core	110
28 Version A.05.02.b2	111
28.1 Core	111
29 Version A.05.02.b	112
29.1 Important Changes	112
29.2 Windows	112
29.3 Motif	114
29.4 Core	114
29.5 Editor	114
29.6 Net	115
29.7 COBOL	115
30 Version A.05.02.a	116
30.1 Important Changes	116
30.2 Validity Range for Better Type Checking	117
30.2.1 Restrictions	118
30.2.2 Access and Value Tests	119
30.2.3 Derivation of the Validity Range	120
30.2.4 Enhancement of the IDM Syntax	120
30.2.5 New Attributes	122
30.2.5.1 scope[attr]	122
30.2.5.2 indexscope[attr]	123
30.2.5.3 typescope	123
30.2.5.4 typescope[I]	124
30.3 New Interface Function DM_ErrorHandler	124
30.3.1 New Tracing Codes	127
30.4 Addition of a Value Resolution in the Path Referencing	127
30.4.1 New Attribute .constant	129
30.5 Parameter Enhancements within the Dialog Manager	130
30.5.1 Increase of Available Parameters to 16	131
30.5.2 Optional Default Values for Parameters	131
30.5.3 Expansion Editor	132
30.5.4 Compatibility	132
30.6 Borderless Geometry (only for MS Windows)	132
30.6.1 Attribute .borderraster	132
30.6.1.1 The Meaning of true	135
30.6.1.2 The Meaning of false	135

30.6.2 References and Dependencies	135
30.7 New Additions to the Image Object (only Windows)	138
30.7.1 Attribute alignment	138
30.7.2 New Attribute spacing	138
30.7.3 Attribute tilestyle	139
30.8 Keyboard Navigation Adjustments for Motif	140
30.9 Support for on open on Menuboxes under Motif	142
30.10 Windows	144
30.11 Motif	146
30.12 Core	146
30.13 Editor	147
30.14 COBOL	147
30.15 Java-Platform	147
30.16 Debugger	149
31 Version A.05.01.h	150
31.1 Java	150
31.2 New Functions for Error Analysis	150
32 Version A.05.01.g4	151
32.1 Core	151
33 Version A.05.01.g3	152
33.1 New Functions for Error Analysis	152
33.1.1 Overview	152
33.1.1.1 Safety Tracing	153
33.1.1.2 Dumpstate	153
33.1.1.3 Exception Catcher	153
33.1.2 Dumpstate Output	154
33.1.2.1 Process	155
33.1.2.2 Errors	155
33.1.2.3 Callstack	155
33.1.2.4 Events	156
33.1.2.5 Usage	156
33.1.2.6 Memory	158
33.1.2.7 Slots	158
33.1.2.8 Visible Objects	159
33.1.2.9 Example	159
33.1.3 Noteworthy for Windows/Threads	161
33.1.4 Safety Tracing	161
33.1.5 Exception Catcher	162

33.1.6 New Start Options	162
33.1.6.1 -IDMdumpstate <enum>	162
33.1.6.2 -IDMdumpstateseverity <string>	162
33.1.6.3 -IDMcallstack <boolean>	163
33.1.6.4 -IDMcatchexceptions <boolean>	163
33.1.6.5 -IDMstrace	163
33.1.6.6 -IDMstracefile <filepath>	163
33.1.6.7 -IDMstraceopts <string>	163
33.1.7 New Built-in Function dumpstate	164
33.1.8 New Interface Function DM_DumpState	166
34 Version A.05.01.g2	167
34.1 Network	167
35 Version A.05.01.g	168
35.1 COBOL	168
35.2 Editor	168
36 Version A.05.01.f	169
36.1 Important Clarification	169
36.2 Windows NT/Windows XP	169
36.3 Motif	171
36.4 Core	171
36.5 Editor	172
37 Version A.05.01.e2	173
37.1 Windows NT / Windows XP	173
38 Version A.05.01.e	174
38.1 Important Modifications	174
38.2 New Features of the IDM Editor	174
38.2.1 Tools	174
38.2.2 File Modification Check	175
38.3 New Attribute .navigation	176
38.3.1 Attribute .navigation	176
38.4 Windows NT / Windows XP	177
38.5 Core	183
38.6 Editor	184
38.7 Network	185
38.8 COBOL interface	185
38.9 Unix	186

39 Version A.05.01.d	187
39.1 Important modifications	187
39.2 Change of Network Interface (DDM)	187
39.2.1 Consequences for the existent dialogs	188
39.2.2 Application example	188
39.2.3 Further Information	194
39.2.4 Changes to the Application Object	194
39.2.4.1 New Attributes	194
39.2.4.1.1 .errorcode Attribute	194
39.2.4.1.2 .systemerror Attribute	195
39.2.4.2 start/finish Event	196
39.2.4.3 Erratic Behavior in Application Functions	196
39.2.5 Network Application Side	196
39.3 Windows NT / Windows XP	197
39.4 Motif	200
39.5 Core	202
39.6 Editor	204
39.6.1 Changes in the Editor on Microsoft Windows	205
39.7 Network	206
39.8 COBOL interface	206
39.9 Unix	207
40 Version A.05.01.c3	208
40.1 Windows NT / Windows XP	208
41 Version A.05.01.c	209
41.1 Important Modifications	209
41.2 Multiscreen Support Under Motif	209
41.2.1 New attributes	210
41.2.1.1 screencount	211
41.2.1.2 screen or screen[]	211
41.2.1.3 real_screen	212
41.2.1.4 display	213
41.2.1.5 Attribute extensions for the setup object	214
41.2.2 Display Resource	214
41.3 Extension Repository Identifier	216
41.3.1 New attributes .repos_id and .repos_id[]	218
41.4 New options for grouping objects under Windows	218
41.5 Windows NT/Windows XP	219
41.5.1 Wheel mouse and wheel mouse support with Microsoft Windows	225
41.5.2 Installation of the IDM 5 versions, modification	226

41.5.2.1 Further options on installed previous versions	231
41.6 Motif	231
41.7 Java Interface	232
41.8 Core	232
41.9 Editor	233
41.10 Network	233
41.11 COBOL interface	233
42 Version A.05.01.b5	234
42.1 Motif	234
43 Version A.05.01.b2	235
43.1 Windows NT / Windows XP	235
43.2 Motif	235
44 Version A.05.01.b	236
44.1 Windows	236
44.2 Unix	239
44.3 Motif	239
44.4 Core	240
44.5 Editor	241
44.6 Network	241
44.7 COBOL Interface	241
45 Version A.05.01.a	242
45.1 New Improvements to the Graphical Editor (Editor III)	242
45.1.1 Overview	242
45.1.2 In General	243
45.1.3 Menus	244
45.1.4 Function Bar	247
45.1.5 Browser	248
45.1.6 Toolbox	250
45.1.7 Design Area	251
45.1.8 Properties	252
45.1.9 Other Changes	255
45.1.10 Writing a Dialog into a File	255
45.1.11 Motif	255
45.2 Windows	256
45.2.1 Changes Resulting from the Introduction of the Windows XP Platform	257
45.2.1.1 Dialog Manager Objects and Visual Styles	257

45.2.1.2 Calculating the Raster Size	258
45.2.1.3 Object Scrollbar and Scrollbars of Other Objects	258
45.3 Motif	259
45.4 Core	259
45.5 Editor	260
Index	261

1 Version A.05.02.o

1.1 Windows

- » Gnats 12253: The **tablefield** avoids unnecessary repainting of fields, which could cause the fields to flicker when hovering the mouse over the **tablefield**. The issue was observed on Windows 7 with “Aero Style” disabled.
- » Gnats 12152: Drag-and-drop at the **edittext** works correctly again. It could happen that drag-and-drop transferred the text that was selected when the **edittext** received the focus. It could also happen that no drag-and-drop was initialized at all (*.startsel* = *.endsel*) or that the selection could not be changed using the mouse pointer. These problems are resolved.
- » Gnats 12091 and 12012: In a maximized and visible **window**, the arrangement of child objects is now correct even if all position and size attributes of the window are changed one after the other. In certain situations, it happened that the children were positioned in the maximized window according to the geometry of the non-maximized window.
- » Gnats 12059: **Pushbuttons** that have a pop-up menu and are made dynamically visible now observe the set font.
- » Gnats 12019: The attached **edittext** is now displayed by the **tablefield** even if the assignment to the *.edittext* attribute takes place after the **edittext** had been made invisible.
- » Gnats 12013: An **MDI child window** with *.mapped* = *false* will not be displayed when creating it, even if another MDI child window is maximized at the same time. The problem arose because windows with *.mapped* = *false* were activated, which generally did not cause the window to be made visible.
- » Changed (Gnats 11878): No error is reported anymore if the attribute *.format* of the **tablefield** is set and the attribute *.format* of the attached **edittext** is also set.

Note

Since the *.format* attribute is controlled by the **tablefield**, the attribute of the attached **edittext** should **not** be used.

- » Gnats 11870: The values of *.open[]* are now updated if *.topitem* is set to an entry within a closed sub-tree in an invisible **treeview** (resulting in this sub-tree to be opened). Corresponding *on open* events are triggered.

Notes

- » Since the *.topitem* entry is always visible, the sub-tree containing this entry cannot be closed programmatically in a visible **treeview**. This sub-tree may be closed interactively because a higher-level entry must be accessible for this purpose. This ensures that the current *.topitem* is always above the set *.topitem*.

- » The value of *.topitem* is corrected if the entry set as *.topitem* cannot be scrolled to the top of the treeview.
- » Fixed: Since WINDOWS VISTA the width of the window frame is made up of the width of the “sizing border” and an additional “padding”. This led to wrong assumptions for size calculations if the Windows subsystem version was set to 6.00 when linking the application (default as of VISUAL STUDIO 2012). The problem could be observed in a maximized MDI child window, where a stripe was still visible.

1.2 Motif

- » Gnats 12275: Minimum and maximum **window** sizes are now set correctly and consistently considering the *.borderraster* attribute.
- » Gnats 12135: If *.startsel* and *.endsel* were not set for an **edittext**, changing *.content* could cause solid cursors (carets) to be drawn in several places. This does not occur anymore.

Remarks

- » Modifying *.content* or *.navigable* of an **edittext** may influence whether the cursor (caret) is drawn solid in another edittext. It should be noted that on MOTIF no input is possible in an edit-text with *.navigable = false*.
- » The solid cursor does **not** allow conclusions about navigation order or focusing of an **edittext**.

1.3 Core

- » Gnats 12228: A crash is now avoided that could occur when deleting the entire content of an input. The crash could occur if the format of the input contained the symbols “/ptz” and there were special number constellations.
- » Gnats 12067: For **user-defined associative arrays**, *.itemcount* is now correctly calculated even in the special constellations where the default value is set again and the inheritance chain contains models without set values. This avoids a crash of the **:index()** method, which resulted from accessing an index outside the index range.
- » Gnats 11087: Multiple loading of binary modules that are imported by a dialog or module as well as by their imported modules is now avoided.

1.4 Debugger

- » Gnats 12251: The rule search (“Find...”) has been comprehensively revised:
 - » Resuming the search (“Find...”) after the first hit works again.
 - » Changes to the search string are taken into account correctly.
 - » The search now starts with the rule selected in the Rule Browser.

- » When the end (search direction “*Down*”) or start (search direction “*Up*”) of the dialog or module is reached, the search is continued at the beginning or at the end.
- » The search can now be triggered by pressing `Return` in the input field “*Find what*”.
- » The search window now stays in the foreground.

1.5 USW

- » New (Gnats 12264): The callback function **UC_Inquire()** has been extended on MICROSOFT WINDOWS with the query *UC_I_querykey*, so that the USW can process navigation keys, which are interpreted by the IDM by default.

The data type *UC_InqName* has been extended:

```
typedef enum {
    UC_I_undef = 0,
    ...
    UC_I_querykey    // since IDM version A.05.02.o
} UC_InqName;
```

The structure **UC_InqArg** has been extended:

```
typedef struct {
    DM_InqName name;
    union {
        ...
        UC_Replace replace;
        UC_Querykey querykey;    // since IDM version A.05.02.o
    } arg;
} UC_UC_InqArg;
```

The structure **UC_Querykey** was newly introduced:

```
typedef struct {
    #if defined(WIN32)
        MSG msg;
        DM_Boolean wantit;
    #else
        DM_Int msg;
        DM_Boolean wantit;
    #endif
} UC_Querykey;
```

The structure **UC_Querykey** is used on MICROSOFT WINDOWS to query whether a keyboard message is needed by the USW. It is left to the WSI, when and for which keyboard messages to call **UC_Inquire** with *UC_I_querykey*.

The *msg* element contains the Windows message.

If the element *wantit* is set to *DM_TRUE*, then the message is not processed by the IDM. As a result, among other things, the keyboard navigation is turned off.

An example of usage can be found in **demos/usw/uctext.c**. In it, only those keys are filtered that the IDM would process by default.

- » New (Gnats 11558): The attribute *.userdata* can now be defined as a vector or matrix attribute. The IDM now allows the definition of the attribute *.userdata[]*. The following restrictions apply to the attribute descriptor:
 1. It should only appear once.
 2. The data type must be *DT_anyvalue* and the value of the element *indexed* must be *UCI_VECTOR* or *UCI_MATRIX*. The elements *rowcount* and *colcount* should be controlled by a different attribute.
 3. The default data value should be (*UC_Data*) 0. Because *.userdata* attributes are reserved for non-visual information only, the **Update** function of the USW will **not** be called for these attributes.

- » New: For USW's now the interface functions **DM_SetContent()** and **DM_GetContent()** can be used.

For full usability, the USW should use the attributes

```
string .content[]
boolean .active[]
boolean .sensitive[]
anyvalue .userdata[]
```

with the same indexing. For this the redefinition of the attributes *.sensitive[]*, *.active[]* and *.userdata[]* in indexed form (as *UCA_VECTOR* or *UCA_MATRIX*) is allowed.

- » New: For attributes of a USW, now the data type *DT_text* may also be used in the attribute descriptor.

Calling **attrGet()** of the USW interface returns the text as string.

The assignment of a string to an attribute of type *DT_text*, e. g. using the API function **attrSet()**, is rejected by the WSI with a TypeClash error.

This also applies to the format handling functions if it is attempted to assign the new content string to an attribute of type *DT_text*. The format functions **fmt*()** then return a *UC_FS_FAILURE*.

- » Changed: In the USW examples, some attributes now have the new data type *DT_text*.
- » Fixed: A USW now observes the *.borderraster* attribute.

1.6 COBOL

- » Gnats 11767: The COBOL examples can be built again.

2 Version A.05.02.n

2.1 Windows

- » Gnats 12000: A content (*.content*) of a **poptext** with *.style = edittext*, which matches the beginning of a list entry, is no longer deleted when closing the list. The cause of the error was that in this case the Windows poptext completed its content when the list was closed and *.activeitem* was incorrectly determined when the *.content* attribute was updated.
The attributes *.content* and *.activeitem* are now calculated correctly and if necessary, *select* and *activate* events are generated. However, the behavior of the Windows object used for the poptext has not been changed.

Notes for the Poptext on Windows

- » When opening and closing the list, a poptext with *.style = edittext* completes the content of the editing field (*.content* attribute) if it matches the beginning of a list entry. This is a default behavior of the Windows object. If the completion changes the active entry, an *activate* event is generated. If there is no completion, no event is generated.
- » When moving the mouse over the open list of a poptext, the entry under the mouse pointer is highlighted. However, the entry is not activated, which can be recognized by the fact that the text is not transferred to the editing field. Therefore, no *activate* event occurs and when the list is closed, the index in the *select* event corresponds to the entry actually activated and not to the entry under the mouse pointer. This is a behavior of the Windows object.

2.2 Motif

- » Gnats 12031: Setting *.activeitem* of a **poptext with format** now causes a correct update of the displayed content string. The display is now also correct for subsequent interactive selections.
- » Gnats 11995: From a focused **notepage**, the **focus** now again moves correctly to the children or to the next object when navigating with the **Tab** key. In some previous versions, the error occurred that the next notepage was focused and activated.

Notes for Notebook and Notepage on Motif

Similar to Windows, when focusing a notepage tab, a *focus* event is triggered on the notebook object.

- » Gnats 11985: There is no longer an assfail when changing the font of an **edittext with format** while the edittext gets the focus.

2.3 Documentation

In the **English** documentation the chapters “Functions” and “Global Variables” (previously titled “Variables”) are no longer in the “Resource Overview” but in the manual “Rule Language” (according to the German documentation).

3 Version A.05.02.m

3.1 Windows

- » Gnats 11846: The window “*Funktionen auswählen*” in the installation routine of the German IDM has been enlarged so that the buttons are no longer truncated.

3.2 Motif

- » Gnats 11950: On the **treeview** the heuristic to determine the maximum number of treeview entries has been corrected. With this correction, for treeviews that can be built up completely, no more error messages should occur that the maximum number of entries has been reached.
- » Gnats 11942: The workaround on the **treeview** to handle the Motif error that opened nodes do not show their child nodes when opening a parent node has been improved. Dynamic setting of `.open [!] := true` if one of the parent nodes is not open no longer results in a faulty representation of the child nodes.

Note

Similar to the Windows version, setting `.open [!] := true` on a child node does **not** automatically open the parent node of this node.

3.3 Core

- » Gnats 11938: The cause of an error message or warning of the GNU C++ compiler when including the file **IDMuser.h** has been eliminated.

3.4 USW

- » Gnats 11915: Destroying a **USW widget** with a **matrix or vector attribute** will now properly free the memory allocated for the attribute. A crash in this situation is avoided.

3.5 ISA Form Designer (IFD)

- » Gnats 11900: Sporadic IFD crashes caused by overwriting memory in an internal function no longer occur. The crashes were observed when starting the printing of the form.

4 Version A.05.02.I

4.1 Windows

- » Gnats 11822: Destroying a **menuitem** with an image that could not be created does not cause an asfail anymore.
- » Gnats 11821 (no change): A **font** resource with the *bold* font attribute will result in differently wide grids on different systems if the font definitions of the systems are different.
When calculating the grid width, the IDM combines the average width and the maximum width from the font description. In the case described, the maximum width is different. To avoid such problems, a reference string can be specified at the font (suffix *r*: = "<RefString>", dynamically changeable via the font attribute *.refstring*), from which the width of the grid is calculated.
- » Gnats 11796: If a **child window** (MDI window) with *.maximized = true* was made visible or when *.maximized = true* was set for it, then child objects tied to the right or bottom were positioned incorrectly. This issue has been resolved.
- » Gnats 11791: In a multi-line **RTF edittext**, line breaks now are correctly taken into account when querying *.startsel* and *.endsel*.
In addition, an error has been resolved that may cause the selection to be incorrect. This occurred, for example, if *.content* still contained plain, unformatted text with line breaks.

Notes on the Methods of the RTF Edittext

- » When the parameters *Start* or *End* of the methods **findtext**, **getformat**, **gettext**, **replacetext** and **setformat** are greater than the text length, then the text length is used in the RTF edittext.
- » If no characters are marked (*.startsel = .endsel*) or *Start = End* when calling the **getformat** method, then the format **to the left** of the specified position is returned.
- » Fixed: The attributes *.startsel* and *.endsel* can now be set individually for the **RTF edittext**.
- » Changed: The **setformat** method of the **RTF edittext** can now set a font that is specified as a string in the form <Size>.<Fontname> (e.g. "12.Courier", analogous to the **font** resource on MICROSOFT WINDOWS). The size and font name are evaluated and set.
- » Gnats 11729: The visibility state of an **edittext** is now restored if it was attached to a **tablefield** and is detached from it. The problem was noted when editing a dialog in the IDM Editor. There an edittext, which was assigned to a tablefield, became invisible, after releasing it from the tablefield.

Note

If an edittext is bound to a tablefield, this changes attribute values of the edittext. Therefore, all attribute values of the edittext should be checked and updated when it is released from the tablefield again.

The modified attributes include *.content*, *.format*, *.multiline* and *.sensitive*, as well as all geometry attributes, if *.editpos = true* at the tablefield.

Attention

The change of attribute values of the `edittext` by the `tablefield` may change in every IDM version.

- » Gnats 11709: If an attribute that affects the size of the window (`.width`, `.xleft`, `.reffont`, etc.) is modified on a **maximized window** (`.maximized = true`) then the maximized state will not be cleared anymore. In addition, size changes in the minimized state are now executed again.
- » Changed: When querying `.startsel` and `.endsel` of an **edittext**, `.endsel` may now be less than `.startsel`. In this case, the cursor is located left of the selection. However, if the user selects text with the mouse, `.startsel` is always less than `.endsel` and it can not be recognized where the cursor is located.
Until now `.startsel` was always less than or equal to `.endsel`, except that it was set differently from the Rule Language.

4.2 Motif

- » Gnats 11840: The heuristic that dynamically determines the critical limit of visible entries in a **treeview** has been corrected (see “Version A.05.02.f”, “Motif”, [Gnats 10742](#)). The heuristic is now more reliable for mixed indents and reverse-indents of the entries and more consistent in terms of the determined maximum number in a treeview with pictures.
- » Gnats 11809: For the objects **pushbutton**, **checkbox**, **radiobutton** and **statictext** the use of **foreground** (`.fgc`) and **background color** (`.bgc`) is now more consistent. When a widget is used for **radiobutton** and **checkbox** by explicit `.options[opt_use_widget] = true`, then unassigned colors are determined by the widget and not inherited from the parent object as it is usual for a gadget. When setting a color to `null` it is attempted to establish the initial state.

Explanation and Notes

For the object classes **pushbutton**, **checkbox**, **radiobutton**, **menuitem** and **statictext** the ISA Dialog Manager uses either “lightweight” gadget classes or “heavyweight” widget classes from Motif. The use of the widget classes can be enforced explicitly by `.options[opt_use_widget] := true` or implicitly by certain attributes or conditions. The use of widget classes is necessary if the mentioned objects have a pop-up menu or at least one of the attributes `.cursor`, `.statushelp` und `.tool-help` is set.

Gadgets and widgets differ in the use of colors. Gadgets usually use the corresponding colors of the father for unassigned colors. Occasionally, Motif will adjust unassigned colors, for example, a set black background will usually result in a white foreground (text color) if the foreground color is not set, to ensure readability. In contrast, widgets have their own default colors, which are used for unassigned colors.

Basically there is no possibility in Motif to remove a set color. The IDM's dynamic setting of a color to `null` is therefore problematic.

The IDM tries to take the standard behavior of Motif into account. For `.options[opt_use_widget] = false`, or if widget usage is implicitly caused, the IDM uses for unassigned colors the respective current color of the parent. That is, in these cases the behavior corresponds to the gadget classes of Motif, deliberately ignoring the later setting of another color at the father. For explicitly enforced

use of widgets by `.options[opt_use_widget] = true`, the use of colors is like for the widget classes of Motif.

- » Gnats 11757: When an **edittext** that is attached to a **tablefield** loses the focus, its content is only written to the `.content[l,J]` attribute of the tablefield, if it has actually changed. This prevents writing the content of the edittext to the wrong `.content[l,J]` attribute when the focus is set to another tablefield entry within a *select* rule.

4.3 Core

- » Gnats 11855: Truncation of strings starting approximately at the 3rd character no longer occurs in the following constellation: `.format = ""` and a format function calling **DM_FmtDefaultProc()** is used and *AppFormatCodepage* is UTF16. The cause of the problem were inconsistent buffer sizes for the conversion between the internal code page and *AppFormatCodepage* through the **DM_FmtDefaultProc()** function.
- » Gnats 11773: In the *treeview*, **DM_SetVectorValue()** for a user-defined attribute no longer triggers unnecessary redrawing.
- » Gnats 11658: At the **tablefield** no more assfail occurs in the special situation that in `.userdata[0,*]` a reference inherited to other cells is stored and the referenced object as well as other objects are destroyed.

4.4 COBOL

- » Gnats 11717: The COBOL functions **COBOLMAIN**, **COBOLAPPINIT** and **COBOLAPPPFINISH** now return a correct exit status as well on architectures where the number representation of COBOL and C is different.

4.5 Editor

- » Gnats 11872: The event list for callback functions is correctly defined again.
- » Gnats 11871: Re-definable methods can now be edited again.
- » Gnats 11726: Default values for **function parameters** are now correctly initialized at the first setting in the Editor so that a correct function declaration is written when the dialog or module is saved.

4.6 Debugger

- » Gnats 11685: Several issues related to **static array variables** have been fixed:
 - » For uninitialized static array variables, “*nothing*” is now displayed on the Variables page.
 - » Dynamically initialized static array variables no longer cause an error message from the debugger.
 - » Parsing the initial values of static array variables works again.
- » Fixed: The field width of the “*Depth*” indicator in the status bar has been increased. On Motif, a narrower margin now ensures that the text is completely visible when editing tablefields.

4.7 ISA Form Designer (IFD)

- » Gnats 11853: After an error correction on the IFD, the *.EntryType* attribute is no longer copied from the template. Forms that have exploited the wrong behavior therefore no longer function as in older versions of the IFD. To facilitate the correction of forms, an “*IFD WARNING*” warning is now written to the log file if *.EntryType* is not explicitly set.

5 Version A.05.02.k

5.1 ISA Formular Designer (IFD)

The ISA Form Designer option now can be used with the Windows XP version of ISA Dialog Manager on Windows XP, Windows Vista and Windows 7.

To launch the editor of IFD select the menu command “Extras | Form Designer ...” in the IDM editor.

Note on Character Encoding

The IFD receives texts in the encoding of the application code page. Therefore only characters included in this code page can be processed. When an application using the IFD is linked with “startup.obj” the application code page is *CP_iso8859*. When the application is linked with “ustartup.obj” the application code page is *CP_utf8*. Other code pages may be set with the function *DM_Control()*.

5.2 New C Interface Function *DM_StrCreate()*

With the function ***DM_StrCreate()*** a text with a given character encoding can be created.

```
DM_String DML_default DM_EXPORT DM_StrCreate
(
    DM_String str,
    DM_UInt1 strCP,
    DM_UInt1 toCP,
    DM_Options options
)
```

Parameters

-> ***DM_String str***

Source text to initialize the newly created text.

-> ***DM_UInt1 strCP***

Character encoding (code page) of the source text.

-> ***DM_UInt1 toCP***

Character encoding (code page) of the newly created text.

-> ***DM_Options options***

Currently unused, should be set to 0.

Parameter values

Besides the already known code page constants (*CP_ascii*, *CP_iso8859*,...) the following code page constants can be used for the parameters *strCP* and *toCP*:

Constant	Meaning
CP_appl	currently set application code page
CP_format	currently set format code page
CP_input	currently set input code page
CP_output	currently set output code page
CP_display	currently set display code page
CP_system	currently set system code page

These constants may **only** be used with DM_StrCreate.

Return Value

The newly created text.

The text has to be freed with **DM_Free** when it is not needed anymore.

6 Version A.05.02.j

6.1 Windows

- » Gnats 11710: An assfail is avoided if the garbage collection of a **subcontrol** object is called recursively. The problem was observed during a drag & drop procedure in a control object. The cause of the error was that during a recursive call of the garbage collection, it was attempted several times to destroy the same father object.
- » New (Gnats 11667): The IDM can now also load bitmap files with the newer file headers BITMAPV4HEADER or BITMAPV5HEADER.

6.2 Motif

- » Gnats 11698: **Treeview** objects can once again be focused using a key command, depending on the value of the attribute .options[opt_scroll_on_focus] and the visibility of the active entry.

6.3 Core

- » Gnats 11745: The integrated function **destroy()** now calls the clean method of the object to be destroyed. With this, it now acts as the interface function DM_Destroy() if it is called with the option DMF_ForceDestroy.

Please Note

- » The doit parameter of the function and method destroy as well as the option DMF_ForceDestroy of the interface function DM_Destroy are **not** transferred to the method :clean. Rather, :clean is always called with Doit = true.
- » When calling the function or the method destroy with doit = false or the interface function DM_Destroy without DMF_ForceDestroy for a model or a default, the clean method is called although the respective object is **not** destroyed.

6.4 COBOL

- » Gnats 11721 and 11715: With the functions **DMcob_CallRule** and **DMcob_CallMethod** terminations and crashes are now avoided whose cause could be attributed to the fact that numerals transferred as arguments were not converted from COBOL in C representation.

6.5 Reexport – Inheritance of export with hierarchical children

6.5.1 New keyword reexport

By using the new keyword **reexport** instead of **export**, the **export** property of the child objects of a model is inherited. For this, **reexport** must be specified with the model on the child and father object. With an object that was inferred from the model, **reexport** only has to be specified on the father. Unlike the **export**, there is no **reexport** necessary for the inherited child of the inferred object.

When **reexport** is indicated on the father of an inferred object, all children that are exported with the model will automatically be exported with this object. Here it does not matter if their export results from the keywords **export** or **reexport** or if it is inherited already. When **export** is indicated on the father of an inferred object, this interrupts the inheritance of the **export** property for the child objects. This holds true even if they are declared with **reexport** in the model or if they inherited the property themselves. A child object whose father is not exported in any kind will not be (re)exported.

The keyword **reexport** can be used wherever **export** can also be used. This means that **reexport** can only be used in modules and not in dialogs. On child objects that have inherited their export properties, the keywords **reexport** and **export** are ignored. Therefore in existing modules, **export** can be replaced with **reexport** everywhere in order to use the changed inheritance behavior.

Example

```
reexport model window MwiBase {
  reexport child edittext Et {}
}
reexport model MwiBase MwiDerived {
}
```

The inherited child *MwiDerived.Et* can be accessed from the outside as it inherits the **export** property of *MwiBase.Et*.

If **export** is used instead of **reexport**, *MwiDerived.Et* must be explicitly exported if you want to access it from the outside.

```
export model window MwiBase {
  export child edittext Et {}
}
export model MwiBase MwiDerived {
  export .Et {}    // inherited child
}
```

This also applies if **export** is used with the inferred object and the inheritance is interrupted by this. In the following example **export** on *MwiOne* prevents *MwiOne.Et* from being exported although with the model *MwiRe* **reexport** is indicated at the father as well as at the child. Conversely for an inferred object a **reexport** of the father causes all children that are exported in the model to be exported in the inferred object. In the example *MwiTwo.Et* inherits its **export** property through the **reexport** of *MwiTwo* although there is **export** indicated on *MwiEx* and *MwiEx.Et* and not **reexport**.

```
reexport model window MwiRe {
```

```

    reexport child edittext Et {}
}
export model window MWiEx {
    export child edittext Et {}
}
export model MWiRe MWiOne {
    // No access from outside to the inherited child MWiOne.Et
    // as export on MWiOne interrupts the inheritance.
    // To make MWiOne.Et accessible from outside
    // an explicit export .ET {} is required.
}
reexport model MWiEx MWiTwo {
    // The inherited child MWiTwo.Et can be accessed from outside
    // because MWiEx.Et has export and MWiTwo has reexport.
    // Therefore MWiTwo.Et inherits the export property from MWiEx.Et.
}

```

The next example illustrates that the inheritance of the export can only occur with inherited hierarchical children. As with export, with reexport there is no inheritance for the instantiable object at the root of a father-child-hierarchy.

```

export default pushbutton PUSHBUTTON { }
reexport default image IMAGE { }

model pushbutton MPbOne {} // does not inherit export from PUSHBUTTON
model image MImOne {}      // does not inherit reexport from IMAGE either
export model pushbutton MPbTwo {} // Explicit export or reexport are
reexport model image MImTwo {}    // necessary to enable access to MPbTwo
                                   // and MImTwo from outside the module.

// In the code below the pushbuttons and images are not
// inherited hierarchical children of the groupbox or layout models.
// Hence they do not inherit export or reexport from their defaults
// and must be explicitly exported to publish them outside the module.
export model groupbox MGb {
    pushbutton PbOne {} // not accessible from outside
    export pushbutton PbTwo {} // export needed for access from outside
    image ImOne {} // not accessible from outside
    reexport image ImTwo {} // reexport needed for access from outside
}
reexport model layoutbox MLy {
    pushbutton PbOne {} // not accessible from outside
    reexport pushbutton PbTwo {} // reexport needed for access from outside
    image ImOne {} // not accessible from outside
    reexport image ImTwo {} // reexport needed for access from outside
}

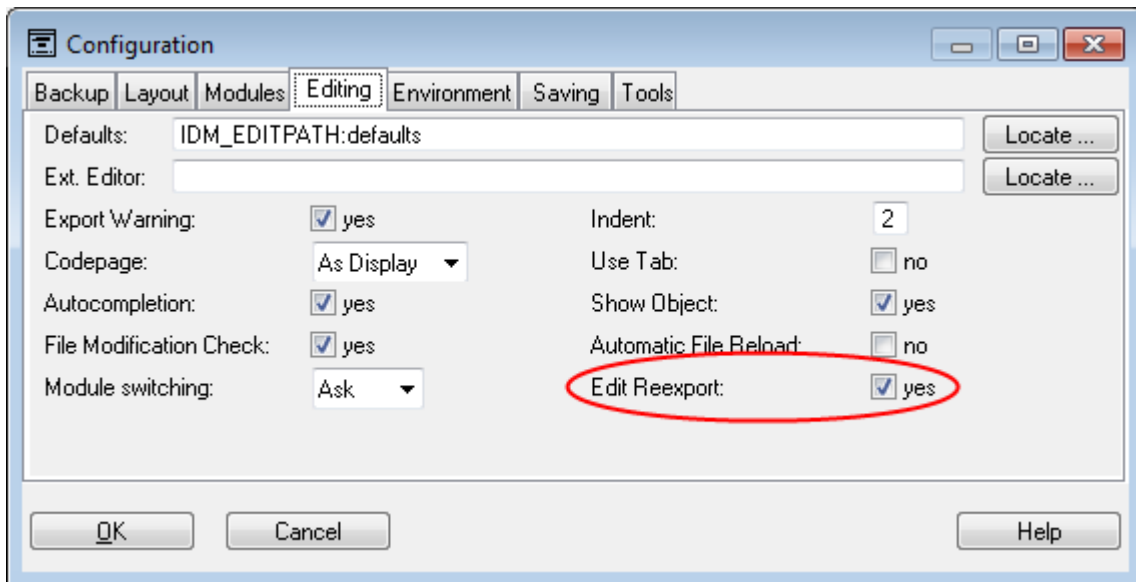
```


Comments

- » If you replace export with reexport in your modules, you should – as with all other source text changes – newly create all interface and binary modules so that the changed export conditions are taken into consideration.
- » We recommend using either export or reexport consistently in order to prevent access restrictions that are difficult to trace.

6.5.2 Supplementation of the Editor

In the "Configuration" dialog (menu command "Extras | Configuration ...") on the "Editing" page, you can activate and deactivate on the "Edit Reexport" checkbox the display and edit of the reexport properties of objects. If the checkbox is activated, reexport can be edited in the properties area and in the context menu of the object browser. Otherwise export is shown and edited there.



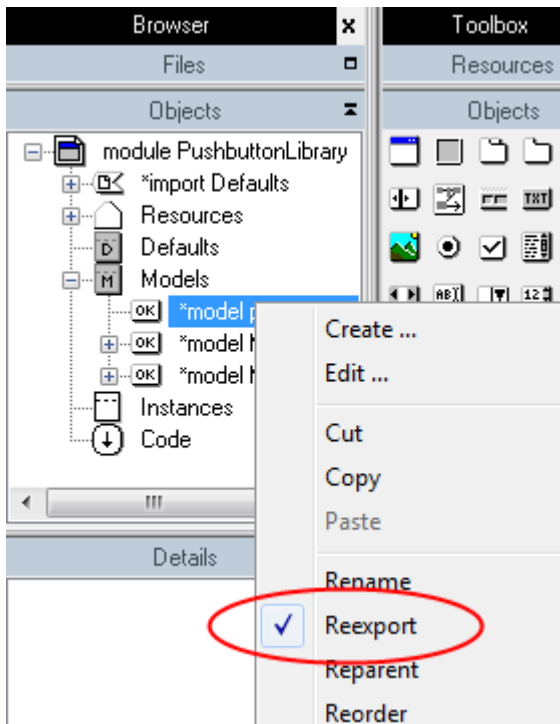
In the properties area, if "Edit reexport" is activated, a "Reexport" checkbox is shown instead of the "Export" checkbox.



In this case, it can assume three states, however only the states "active" and "inactive" can be set.

Status	Meaning
active	Object is reexported
inactive	Object is neither reexported nor exported
undefined	Object is only exported

With activated "Edit Reexport" the reexport property can be defined over the context menu of an object in the object browser as well.



In the object browser, exported as well as reexported objected are marked with a leading *.

7 Version A.05.02.i

7.1 Important Changes

- » The network option of the ISA Dialog Manager (DDM, Distributed Dialog Manager) now supports the IPv6 protocol on all architectures that support IPv6 natively (see chapter “IPv6 support”).
- » Objects with virtual size (groupbox, layoutbox, notepage, window) and treeview under Motif have the new option `.options[opt_scroll_on_focus]` which can be used to specify whether a child object, that is focused, is scrolled into the visible area (Gnats 11607, 11614 and chapter “New option `.options[opt_scroll_on_focus]`”).
- » The functions **DM_SetVectorValue()** and **DM_GetVectorValue()** are now also supported with USW classes for indexed attributed with indices types UCI_VECTOR and UCI_MATRIX (Gnats 11389).
- » The function **sendEvent()** of an USW class can only be called with events that are valid for USW classes and that have been registered for a widget (Gnats 11144).

7.2 Windows

- » Gnats 11669: An **edittext** with `.options[opt_rtf] = true` no longer loses any character formatting (font, bold, italics, ...) when it is made visible. When setting the content in invisible state, the formatting of the content is now correctly presented when making the edittext visible again.
- » Gnats 11629: A problem was remedied by which in an **edittext** object with a small-sized font, the content was scrolled to the left although it could actually fully fit in the edittext.

Note

The problem was only observed in an IDM version that was not generally distributed.

- » Gnats 11598: The scroll position is now adapted if a large-sized font has been assigned to an **edittext** object. Therefore the cursor is no longer moved out of the visible area or content that fits in the edittext is no longer cut off on the right. Previously, this could happen when statically and dynamically setting the font, whereby the problem did not occur when subsequently setting `.startsel` and `.endsel`.
- » Gnats 11588: **GIF files** containing an "application extension" block are now correctly imported and displayed.
- » Gnats 11517: In **poptext** an error was corrected which with `.style <> poptext` in certain constellations (e.g. decreasing `.itemcount` with `.activeitem` on the last list element) could lead to `.content` having an incorrect value or an old value shown in the entry field.
- » Gnats 11503 (no change):

Information on Using Colors and “Visual Styles” With Tablefield

If no colors are set for the **tablefield**, IDM uses the system colors of the respective state. With `.fieldshadow = true` the “Visual Styles” of the Windows object “pushbutton” are used. They define the allocated colors. If `.rowheadshadow` or `.colheadshadow` is activated, the colors of the “Visual Styles” of the Windows object “headeritem” are used.

If colors are defined, then these colors are used. In an active field, `.fgc` determines the background color and `.bgc` foreground color. If **only one** of the two colors are set, the respective system color is used for the **other** color. Therefore if only one color is defined, foreground and background color are not exchanged, but the defined color is allocated differently. There is no own color assignment for an active **and** insensitive field. It is depicted just as an active field.

Recommendation

Either define **both** colors – `.fgc` and `.bgc` – or **none** of the two colors.

Note

In the ISA Dialog Manager for Windows 2000, the colors were already assigned as described. However, for active objects foreground and background colors were exchanged, which in the case of just one defined color usually produced a good result. First the “Visual Styles” (in particular the Windows pushbuttons) in part do not exchange the colors, but merely use a different background.

- » Gnats 11474: Placing the **focus** on a **notebook**, whose active notepage does not have any focusable children, now works. The problem was detected by a mnemonic on an insensitive statictext, which only functions when the notepage besides the statictext has further child objects.
- » Gnats 11418: With activated “Visual Styles”, the **standard background color** of an insensitive **spinbox** object with a sensitive statictext as child, has changed. Now the “Visual Styles” of the edittext background are used, so that the appearance no longer differs from the other objects.
- » Gnats 11401: A **GIF image** with **transparency** is now also shown correctly if the RGB value of the transparently defined color index corresponds another color index. The transparent color is now automatically allocated to an unused RGB value.
- » Corrected: The **debug versions** of the included **examples** could not be executed as an incorrect C-Runtime was defined in their manifests. Now there are two manifests that can be used depending on the configuration.

7.3 Motif

- » Gnats 11608 (no change, also see Gnats 11607 and 11614 below):

Information on Object Position and Keyboard Focus

Under Motif, only objects that are visible or that can be scrolled into the visible area may obtain **keyboard focus**. This is the typical behavior of Motif applications, unlike MS Windows.

For compatibility reasons between the platforms, the ISA Dialog Manager also enables the positioning of child objects with negative positions in non-visible areas under Motif. As Motif actually does not allow this, some check mechanisms have to be bypassed. However, this usually rules out that the objects are reachable via keyboard navigation and that they can be focused.

Recommendation

Always position objects fully in the visible area in order to ensure that they can receive keyboard focus.

- » Gnats 11607 and 11614 (also see Gnats 11608 above):
Objects with virtual size (**groupbox**, **layoutbox**, **notepage**, **window**) and **treeview** have the new option `.options[opt_scroll_on_focus]`. This option is used to define whether a child object is scrolled into the visible area when it gets focused.
See chapter "New option `.options[opt_scroll_on_focus]`".
- » Gnats 11589: **GIF files** containing an "application extension" block are now correctly imported and displayed.
- » Gnats 11578 and 11539: Objects keep **focus** also with attribute changes, which require an internal destruction and new create ("clobbering").
- » Gnats 11566: If a **notebook** or a **notepage** has been switched to insensitive, an insensitive notepage can no longer be selected or activated.
- » Gnats 11569: Placing the cursor (caret) in an edittext deletes the selection again. This affected Open Motif 2.3 where selection deletion after [Error remedy #1279 in Open Motif](#) no longer worked.
- » Gnats 11574: With **tablefield** the horizontal alignment of multi-line texts now works with `.xalignment` or `.colalignment`.

Note

With edittext and also the object, which is used for editing texts in a tablefield, it is still not possible to align right or center.

- » Gnats 10927: The attribute `.toolbar` is now – analog to the other hierarchy attributes such as `.groupbox` and `.notepage` – accessible for all object classes and via the attribute list `.attr[]`.

7.3.1 New option `.options[opt_scroll_on_focus]`

Objects with virtual size (groupbox, layoutbox, notepage, window) and treeview have the new option `.options[opt_scroll_on_focus]`. This option is used to define whether a child object is scrolled into the visible area when it gets focused.

Purpose of the option

The option `.options[opt_scroll_on_focus]` primarily serves to improve the accessibility of objects via keyboard navigation and to enable that they are focused in certain constellations via keyboard or mouseclick. This achieves more consistent behavior between Motif and Windows applications.

However, the option – due to the differences between platforms in focus handling – cannot achieve full consistency, nor that objects can be focused by key command in all situations.

Relation Between Object Position, Visibility, Keyboard Navigation and Focusability Under Motif

Under Motif, only objects that are visible or that can be scrolled into the visible area may obtain focus. This is the typical behavior of Motif applications, unlike MS Windows. Please note that a child object may be fully visible in a grouping object, but is still not accessible and reachable via keyboard, as the grouping object is not fully visible.

For compatibility reasons between the platforms, the ISA Dialog Manager also enables the positioning of child objects with negative positions for the x and y coordinates of the left upper corner in the non-visible areas under Motif. Motif does not actually allow this, so that the IDM must bypass check mechanisms. Consequentially, this usually rules out that the objects are reachable via keyboard navigation and that they can be focused.

Recommendations

- » The best way to ensure the accessibility of objects via keyboard navigation and their ability to be focused with keyboard and mouse is to always position them fully in the visible area and to avoid negative values for their x and y coordinates.
- » If this is not possible, you can attempt to improve key navigation and focusability with the help of `.options[opt_scroll_on_focus]`.

Description of Option

Attribute	<code>.options[opt_scroll_on_focus]</code>
Data type	boolean
Standard value	true
Access	get, set
Allowable with	groupbox, layoutbox, notepage, treeview, window
Platform	Motif (Unix)
IDM version	from A.05.02.i

Values

true (Default)

With grouping objects (groupbox, layoutbox, notepage, window) with virtual area or for treeview, the IDM attempts to scroll a child object or an entry that is to be focused in the visible area.

false

The IDM does **not** attempt to scroll child objects or entries that are to be focused into the visible area.

Although not or partially visible child objects or entries with deactivated scrolling may now not receive focus, Motif however at least allows for child object and entries to be focussed that already are completely in the visible area.

Note

» **Treeview**

With `.options[opt_scroll_on_focus] = false` a treeview may be inaccessible by keyboard if the active entry is outside the visible area. It can still be focused by mouseclick.

7.4 Core

- » Gnats 11678: Changing an attribute of a **document model** no longer triggers the inheritance of the `.version[]` attribute. The error can lead to the deletion of an existing DOM tree, as when changing `.version[]` another version the XML interface was loaded.
- » Gnats 11592: Crashes no longer occur when setting **code page** `CP_ucp` (iconv) under MS Windows. A conversion to `CP_ucp` is still **not** possible as there is no iconv under MS Windows.

7.5 Network

- » Gnats 11660: The network display side no longer terminates with a **protocol error** if a net handler has been installed that changed the data. The cause of the problem was that the application side sent an inquiry not marked as `DM_NET_CONNECT` before `Applnit` was called.
- » New: **Tracing** on the application side can now receive texts in the **application code page** and in the **internal code page**. This change remedies problems that may occur in certain situations from the conversion between code pages (e.g. illegible format strings).
- » New: Texts can now be sent and received in the **application code page** and in the **internal code page**. This change remedies problems that may occur in certain situations from the conversion between code pages (e.g. failed dynamic function calls, as the function name was falsified during a conversion).

7.5.1 IPv6 support

The network option of the ISA Dialog Manager (DDM, Distributed Dialog Manager) now supports the **IPv6** protocol on all architectures that support IPv6 **natively**.

When an IPv6 address is defined in the dialog script, it has to be written in brackets ('[' and ']'), as is the case with URLs (e.g. "[::1]").

Note

The prerequisite of "native" support means e.g. that IPv6 is **not** supported on MS Windows 2000, as the "Microsoft IPv6 Technology Preview for Windows 2000" makes use of different DLLs.

7.6 Debugger

- » Gnats 11555: When evaluating variables and expressions in complex model structures and load constellations, the **Debugger** no longer causes an assfail.

7.7 COBOL

- » Gnats 11593: **Unnecessary conversions** between **internal** and **application code page** and back are now avoided, so that there is no more risk of falsifying strings with this conversion. The problem was detected as functions of a COBOL application could no longer be called on the application side of the DDM. They should have been called dynamically (DMufcob_BindThruLoader), where an application codepage was defined that changed the ASCII area (in the case observed CP_ucp under the use of iconv).

7.8 USW

- » New (Gnats 11389): The functions **DM_SetVectorValue()** and **DM_GetVectorValue()** are now also support for USW objects with indexed attributes. The index (element "indexed" of the attribute descriptor) can have the type UCI_VECTOR or UCI_MATRIX.

Data Types for Object Referencing in the Attribute Descriptor

In the attribute descriptor, only data types of the resources (DT_color, DT_font, DT_tile, etc.) can be used for object referencing, as the USW-API does not allow access to any other arbitrary objects. For other values, DT_anyvalue can be used or referencing can be deferred to user-defined attributes.

- » Change (Gnats 11144): **sendEvent()** can only be called with events that are valid for the USW classes and that are registered for the respective class. Faulty sending of external events (extevents) via **sendEvent()** is now no longer possible.

Change in the Functions registerClass() and sendEvent()

In the events parameter (UC_N_events) of **registerClass()** only the following events can be registered and are therefore allowed in the event mask (*evmask* in UC_EventInfo) of the *event* parameter of **sendEvent()**.

EM_activate	EM_changed	EM_charinput	EM_close
EM_cursor_move	EM_dbselect	EM_deactivate	EM_deiconify
EM_deselect	EM_deselect_enter	EM_finish	EM_help
EM_hscroll	EM_iconify	EM_modified	EM_move
EM_open	EM_resize	EM_scroll	EM_select
EM_start	EM_vscroll		

8 Version A.05.02.h

8.1 Important changes

- » Under Windows, with **poptexts** having style = edittext or .style = listbox the text cursor in the entry field is always placed at the end of the content when the poptext loses the **focus**. The behavior of poptexts with and without format is therefore more consistent. (Gnats 11121)
- » Under Motif, objects with **virtual size** (window, groupbox, notepage) now support the scrollbar attributes .hsb_arrows and .vsb_arrows, .hsb_linemotion and .vsb_linemotion, .hsb_pagemotion and .vsb_pagemotion. (Gnats 11456)
- » New option .options[opt_old_select] of **tablefields** under Motif, with which a selection-oriented or action-oriented dispatch of Select events **can be set**. This enables the consistent behavior of the Motif and the Windows version of the IDM. (Gnats 11319)
- » New attribute .preeditssel of the **tablefield** with which the initial text selection for the entry is controlled. With this attribute, the consistent behavior of direct character entry and entry after activation of the edittext (e.g. by double clicking) can be achieved. (Gnats 10021, chapter “New Tablefield Attribute .preeditssel”)

8.2 Windows

- » Gnats 11477: If the content of a **tablefield** is changed by a **method**, all changes are now directly shown. Before, e.g. the line height was not directly updated during changes with :exchange().
Note
With tablefield, the use of a method that changes the content (e.g. insert, exchange) leads to the deactivation of the connected edittext without saving the content. In earlier IDM versions, the edittext remained active, however not all changes were shown immediately.
- » Gnats 11471: With a **statictext** or **image**, if the text is changed in the visible condition, tools such as screen readers or Visual Studio's Spy no longer return the old value but the changed value.
- » Gnats 11431: A **mnemonic** change in the visible condition, now also works with **statictext** and **image** directly following the change. Before, the mnemonic was shown directly, but only began to work after further actions (e.g. making invisible and then visible again).
- » Gnats 11428: In the **XML interface**, access to the attributes .idispach and .ixmldomdocument2 of a document object now returns correct values. The error led e.g. to a memory access violation when calling the OLE method AddRef.
- » Gnats 11355: With the XP version of IDM, the old text no longer remains in the poptext if the list was emptied by setting :itemcount = 0. Before, the text was removed only once the mouse was moved across the poptext, for instance. Plus, now the size of the popup list is properly updated.

Comment

With `.style = edittext` the text remains as `.content` is an independent attribute.

- » Gnats 11122: A docked **toolbar** now generates a *resize* event if its size is interactively changed. In addition, *move* events of other toolbars are sent, which are moved due to the change in size. A move event is only generated if the attributes `.dock_offset` or `.dock_line` change. There is e.g. no move event if a toolbar docked at the bottom is moved down but stays in the same toolbar line. When indirectly changing `.dock_offset` or `.dock_line` – for example as a result of enlarging another toolbar – the change is only executed virtually so as to not change the defined position permanently. With this, toolbars resume their defined position as soon as space for this is available.
- » Gnats 11124: With **OLE controls** that are created with MFC, it may occur that **no connection** could be established, as the OLE control immediately canceled during a connection attempt. The cause of this was that the MFC application terminated when releasing the OLE ClassFactory object, which it actually shouldn't have done. The problem was remedied by releasing the OLE ClassFactory object not before the IDM control object disconnects the connection.
- » Gnats 11121: With the **poptext** having `style = edittext` or `.style = listbox`, the text cursor in the entry field is now always placed at the end of the content when the poptext loses the **focus**. This was already the case when the poptext had a format, so that the behavior is now more consistent. It does however depart from regular Windows behavior, which places the cursor at the beginning. Plus, no marking is shown in the entry field when the poptext does not have focus.
- » Corrected: The content of an **edittext** which is connection to a **tablefield** with `.editpos = false`, is now marked correctly during activation, if the content of the focused cell was changed in the tablefield after it had been focused. The problem was only visible when the new text was longer than the old one.
- » Corrected: When an **edittext** with format is focused when it becomes visible, the values of the attributes `.startsel` and `.endsel` are now observed. Problems were detected with one edittext which was already in a visible window and was made visible subsequently as the only focusable object.
- » Corrected: In the XP version of the IDM, the attribute `.showitem` of a visible **poptext** can now be changed dynamically.

8.3 Motif

- » Gnats 11456: Objects with **virtual size** (window, groupbox, notepage) now support the scrollbar attributes `.hsb_arrows` and `.vsb_arrows`, `.hsb_linemotion` and `.vsb_linemotion`, `.hsb_pagemotion` and `.vsb_pagemotion`.
- » Gnats 11455: A workaround prevents crashes with the **poptext** under Motif from version 2.1 that could occur if certain attribute changes to the poptext are triggered when mouse-clicking the arrow to open the list.

Note

Making changes to a poptext while opening the popup list must be avoided **in all cases** (e.g. in a deselect rule). Certain changes can only be executed by destroying and re-creating ("clobbering") the poptext. This is delayed starting with IDM version A.05.02.h in order to enable the regular

procedures within the toolkit and to prevent crashes. However, because of this, the old content of the poptext may still be visible. Plus, the poptext in this constellation does not send any events until the list has been closed. In such a situation, a warning is written into the log file.

- » Gnats 11441: **Listboxes** in a **dialogbox** are initially shown in the **right size**. Before, it could occur that they were too large by the width and the height of the scrollbar area.
- » Gnats 11415: When changing .text and .activeitem of a **poptext** with the style edittext or listbox, the **content string** now also remains if an internal destroy or re-create ("clobbering") of the popup text is necessary. Finally, the change of .text and .activeitem functions correctly again and changes the content string.
- » Gnats 11339: **Pushbuttons** with .defbutton = true in a toolbar within a **dialogbox** do not cause assfails anymore.
- » Gnats 11321: With **treeview**, in the event of a **ButtonPress event** (mouse click with any keys) outside of an entry, no **select** event without index is sent.
- » New (Gnats 11319): The **tablefield** was extended with the option **.options[opt_old_select]**, which is used to set selection-oriented or action-oriented dispatch of **select** events. This enables the consistent behavior of the Motif and the Windows version of the IDM.
.options[opt_old_select] has the data type boolean with the following value meanings
 - » **true** (default value): Select events are dispatched when changes are made to the activation status (selection-oriented). This corresponds to previous behavior including IDM version A.05.02.g.
 - » **false**: Select events are sent at mouseclicks or when pressing the selection button (action-oriented). This corresponds to behavior under Microsoft Windows. With this, the select event does not show changes to the activation status anymore. The events activate and deactivate can be used in order to also respond to changes in selection with **.options[opt_old_select] = false**.

Note

- » The standard value will change in a future major release of the IDM. This is why it is recommended to explicitly set the desired value value for **.options[opt_old_select]**.
- » The option has no consequences under MS Windows.
- » Gnats 11030: At internal destroys and re-creations ("clobbering") of objects that are necessary for some attribute changes, the focus is kept at the respective object, as far as possible. Plus focus, activate and deactivate events of other objects are suppressed that occur in this connection.

8.4 Core

- » Gnats 11454: The overwritable **:get** methods are now also called for attributes that were previously delivered by WSI in the visible area.
- » Gnats 11403: With **USW object attributes**, **referencing** of objects and resources from other modules now also functions in the runtime version of IDM error-free, without the error message "Failed set indexed value" or "IDM-E-RefEvalError".

- » Gnats 11397: Changing the cursor within **numeric formats** now correctly and sets the display position of the cursor behind the comma, making it consistent with **cursor navigation**. This prevents an assfail with entries.
- » Gnats 11396: The IDM allows just one definition of the same overwritable methods per inheritance level. If multiple definitions are defined, the parser now returns the error "Method already exists".
- » Gnats 11370: Reading of the attribute `.options[opt_center_toolhelp]` on the edittext by getvalue no longer returns an error.
- » Gnats 11343: The generation of the *activate* and *deactivate* event in a **tablefield** with `.selstyle = extended` was optimized. When changing the selection there is now **one** activate event of the element that is newly selected and **one** deactivate event of the previously selected element.
- » Gnats 11253: A **user-defined attribute** set to *null* is now no longer exported as `.unknown` but once again correctly as `null`.
- » Gnats 10192: For the objects **module** and **dialog** the unused scalar attribute `.options` was removed from the integer type. There are no more duplications of `.options` in the attribute list.
- » New (Gnats 10021): The **tablefield** was extended with the attribute `.preeditssel`, with which the initial text selection for entry is controlled. With this attribute, the consistent behavior of direct character entry and entry after activation of the edittext (e.g. by double clicking) can be achieved. Also see chapter "New Tablefield Attribute `.preeditssel`".
- » Gnats 9319: When calling `DM_CallMethod()` the error code `DME_UnknownMethod` is saved on the error stack for unknown or unfound methods, to differentiate between errors caused by incorrect parameter number or -types.

8.5 Unix

- » Gnats 11205: A redundant symbolic link to the document directory was removed from the **distributed archives**, so that no more warnings are issued when unpacking with "tar".

8.6 Editor

- » Gnats 10232: When converting an instance to a model, the original coordinates now also remain if the instance is selected by clicking in the design area.
- » Gnats 10066: The editor now also recognizes file changes with activated "file modification check" if they occur during the simulation and asks the user to reload the file once the simulation is complete.
- » Gnats 10063: When adding a user-defined attribute in the user data area, the data type "anyvalue" is automatically defined when entering the descriptor. Therefore it can no longer happen that the attribute is deleted with all its entered information during "Assign" if no data type was entered by mistake.
During the course of this change we also remedied that error that the data type field could no longer be edited as soon as an "Assign" was made. As earlier, user-defined attributes in the

descriptor field are not preceded by a period so as to more clearly distinguish them from pre-defined attributes. Plus, under Motif that texts in the entry fields of the table should be better legible.

- » Remedied: With the representation of **dialogboxes** in the **IDM Editor** the children of the dialogboxes are no longer offset by the position of the window.

8.7 New Tablefield Attribute .preeditssel

Identifier:

.preeditssel

Classification: object-specific attribute

Definition

Argument type: enum

C definition: AT_preeditssel

C data type: DT_enum

COBOL definition: AT-preeditssel

COBOL data type: DT-preeditssel

Access: set, get

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

This attribute only exists for tablefield.

The .preeditssel attribute controls the initial selection of the text, if the edittext of a tablefield is explicitly or implicitly activated. An implicit activation of the edittext takes place by entering a character, explicit activation for example takes place by double clicking.

With the aid of .preeditssel the same initial text selection can be set for both activation forms and the same behavior can be achieved. Entry behavior refers to the replacement of existing text through entry, as well as to inserting entered characters before or after the content of the edittext.

The attribute can assume the following values

Value	Behavior
<i>preeditssel_default</i> (default value)	Corresponds to the previous behavior: <ul style="list-style-type: none">» With explicit activation, an entered character replaces the content.» With implicit activation, an entered character is appended to the content.

Value	Behavior
<i>prese_ all</i>	For explicit as well as implicit activation, the content is always entirely selected – from .startsel = 0 to .endsel = length(content). An entered character replaces the content.
<i>prese_ begin</i>	No selection, the cursor is set at the beginning for explicit as well as implicit activation. An entered sign is placed before the content.
<i>prese_ end</i>	No selection, the cursor is set at the end for explicit as well as implicit activation. An entered sign is appended to the content.

From IDM version A.05.02.h the attribute is supported on Unix platforms with Motif and on Microsoft Windows.

9 Version A.05.02.g

9.1 Windows

- » Gnats 11272: Loading data from a bitmap file no longer results in a crash when the file size and the size of the bitmap data differ from one another.
- » Gnats 11219: When changing sizes, the content of an **edittext** is now scrolled into the visible area as far as possible. Before it could happen that the edittext would no longer scroll its content, if it fit in terms of length in the edittext but, e.g. was placed by a **format** so that it wasn't fully displayed.
- » Gnats 11203: When changing the *.toolhelp* attribute the toolhelp was displayed immediately once the mouse hovered above the respective object. Now it is shown following a mouse movement with a delay. If the toolhelp is **visible** while changing *.toolhelp*, the text changes. If it independently blinds out, the mouse has to be moved out of the object and back in again.

Note

"WM_MOUSEMOVE" events that are provoked from another application can interrupt the showing and hiding of the toolhelp. For instance, *.toolhelp* may open after its change, although the mouse was not moved. It may also happen that a visible toolhelp may not be blinded out again or a toolhelp may not be shown.

An application-driven open process of toolhelps is not planned. If this is desired, the function **DM_GetToolkitData** can be used to query the attribute *AT_toolhelp* of the **setup** object to get the Windows handle of the "tooltip control" that the ISA Dialog Manager uses for display.

The Toolhelp can be opened with the following example code:

```
#include <windows.h>
#include <commctrl.h>
#include IDMuser.h

void DML_default DM_ENTRY OpenToolhelp __0() {
    DM_ID idSetup = DM_DialogPathToID(
        (DM_ID) 0, (DM_ID) 0, "setup");

    if (idSetup != (DM_ID) 0) {
        HWND hwndToolhelp = (HWND) DM_GetToolkitData(
            idSetup, AT_toolhelp);

        if (hwndToolhelp != (HWND) 0) {
            SendMessage(hwndToolhelp, TTM_POPUP,
                (WPARAM) 0, (LPARAM) 0);
        }
    }
}
```

The "OpenToolhelp" function must be defined in the dialog respectively. The Windows message "TTM_POPUP" is available from version 6 of the Common Controls Library (comctl32.dll), i.e. from the XP version of the ISA Dialog Manager.

- » Gnats 11163: For an **edittext with format** the display of new and longer content is now no longer cut to the length of the previous, shorter content. The problem was not visible if, following the assignment of the new and longer content, further "actions" were performed (e.g. focus setting on the edittext).
- » Gnats 11109: A group of **pushbuttons** is no longer exited when **navigating** with the **cursor keys** (arrow keys). Since the introduction of the *.navigation* attribute (in IDM version A.04.04.o or A.05.01.e) it could happen that the next object was jumped to in the tab sequence.
- » Gnats 11089: When opening windows that have a menu, focus is now placed on the first object. This was previously not the case as the window was activated when setting up the menu by mistake.
- » Gnats 10968 (no change):

Information on Using Symbol Fonts

When using fonts that contain symbols (e.g. "Wingdings") the following should be observed:

1. The name must correspond to the actual name of the font (no alias). If Windows executes a font exchange, IDM recognizes this and forces the replacement to a font that supports the WIN ANSI character set. This is necessary to prevent the unintentional selection of an illegible font.
2. A font that contains symbols is allocated to the Windows SYMBOL character set. This is mapped on the Unicode range \uF000–\uF0FF (user-defined range in Unicode). In order to display the character 0xFE, \uF0FE must be defined.

Comment

This is generally not necessary on a western European Windows system as the system code page CP1252 is almost identical to the lower Unicode range and contains almost all characters. This is why this does not lead to a conversion and the desired character is shown – quasi by chance. This however is not the case on other Windows language versions.

- » Gnats 10971 (no change): When making a **window with toolbar** visible, no unnecessary *resize* events occur any longer since version A.05.02.f.

Comment

The problem was remedied by correcting another error (probably Gnats 10799). Gnats 10971 has been added in the release notes for version A.05.02.f.

- » Gnats 10778: The status of a **window** not always corresponded to the settings of the attributes *.iconic* and *.maximized*. Problems could occur when other attributes, such as *.titlebar*, *.sizeable* and *.borderwidth*, had unusual and infrequently used values or value combinations.

» Change

With

```
.maximized true;  
.iconic true;
```


a window is now shown minimized and can be restored to maximum size. Up to now it was sometimes not shown minimized and was often restored to its regular size.

» **Patch**

With

```
.titlebar false;  
.sizeable true;  
.borderwidth 0;
```

a window is now created with the correct width – and not 2 pixels too wide.

» **Patch**

With

```
.titlebar false;  
.sizeable true;  
.borderwidth 0;  
.iconic true;
```

a window is now correctly display minimized.

- » New: WinRunner support was extended so that the focus element of the tablefield can be queried (see IDM_TF_GETFOCUSCELL in IDMuser.h).
- » Corrected: The start of the IDM Editor no longer leads to an assert– and the respective entry in the log file – by an editbox object.

9.2 Motif

- » Gnats 11333: Layoutboxes with children that were initially made visible now obtain scrollbars – if necessary.
- » Gnats 11300: Objects in a **layoutbox** are now correctly positioned if **scrollbar attributes** and **virtual sizes** (.vwidth, .vheight) are defined in the dialog for the layoutbox. The values of the named attributes are now deleted before visualizing the layoutbox, as it calculates these attributes itself.
- » Gnats 11285: **Dialogboxes** in which the default pushbutton (.defbutton = true) is in a **notepage**, no longer generate an error message.
- » Gnats 11259: So as to not interrupt text entry, with **edittexts** having a **popup menu**, in which **mnemonics** are defined, the popup menu is no longer opened automatically when entering a mnemonic character (as is common from Motif 2.1).
For the other focusable objects **without** text entry, such as pushbutton, checkbox, radiobutton, image, scrollbar and spinbox with statictext, the popup menu opens automatically when entering a mnemonic character, as before.

Note

As with all object classes, the pop-up menu in the edittext from Motif 2.1 can be opened by default with the key combination **Shift + F10**.

As an alternative to mnemonics, accelerators can be used.

- » Gnats 11214: In the Motif-2 version of the IDM the **cursor control keys** can be used to navigate forward **and backward** through **edittext** objects.

- » Gnats 11209: When hiding a **window** or a **poptext** the **focus** is not passed on. The faulty focus- and deselect events are prevented with this.
- Attention! Change in Behavior**
- If the focus changes when hiding an object hierarchy in which an edittext was focused to an externally located edittext, it will retain the focus even when the object hierarchy is made visible again.
- » Gnats 11202: Dynamic setting of the **toolhelp text** of a **tablefield**, **poptext** or **treeview** now affects the regular appearance of the toolhelp, even if the .toolhelp had previously been assigned with *null*.
 - » Gnats 11199: Dynamic instancing of a **menubox** as a child of a visible splitbox does not cause an **assfail**.
 - » Gnats 11172: The **scrollbar sliders** of a **tablefield** are now displayed at the correct position, even when the content is wider and higher than 32,000 pixels.
 - » Gnats 11165: IDM could crash if a model with a **notebook** and **notepages** was instantiated in the visible area. The problem has been remedied. The cause was due to the failure of XtWid-getDestroy() when inheriting the .font attribute from the notebook to the notepages.
 - » Gnats 11161: When editing an **edittext** in a **tablefield** the **tabulator key** is now used to jump to the **next cell**. With this, the behavior now once again corresponds to the IDM version A.05.01.
 - » Gnats 10898: Destroying the active **notepage** now makes the next (or the first) notepage of the notebook visible without causing an **assfail**.
 - » Gnats 10633: The sensitivity status of **poptexts** in a **toolbar** is now correctly updated if the sensitivity of a window changes. The problem was noticed in the IDM editor when opening the object history with the most recently selected objects could induce the desktop to no longer respond.
 - » Gnats 9707: An **image** object with .width = 0 and .height = 0 (prefsize) now changes its size correctly when .text or .picture is changed.

9.3 Core

- » New (Gnats 11266): The output of the **dumpstate()** function can now also be redirected to a file.

Syntax

```
void dumpstate(
    {anyvalue Filename input
    {, enum State input}}
)

void dumpstate(
    {{anyvalue Filename input},
    enum State input}
)
```

Parameters

anyvalue Filename input

File in which the status information is output (optional parameter).

Default value: If the parameter is missing, the status information is output to the trace or log file.

enum State input

This parameter influences the sections of the status information that are output (optional parameter). If the parameter is missing, the default value *dump_error* applies.

- » Gnats 11247: When using **DM_SendEvent()** and **DM_QueueExtEvent()** in threads and with strings, which are coded in the ACP code page there are no more interactions with string transfers during C function calls. These interactions for instance led to the random change of transferred strings.
- » Gnats 11223: **Associative arrays** can now be filled with a maximum of 65,528 entries. If it is attempted to create more entries, this is logged as "[SV] FAILED (IDM-E-IndexRange)". This way there are no more crashes when attempting to create more than 65,528 entries.

9.4 Unix

- » Gnats 11170: A problem was remedied that disables the IDM from being installed as the **install-ation** was **terminated** with error messages. This error was observed under Solaris 10 with the Distributed Dialog Manager. The cause was attributed to the fact that during the generation of serial numbers, the options to be activated were not processed correctly.

9.5 Editor

- » New: On the page with the specific properties of the **toolbar**, the attributes *.autosize* and *.auto-align* can now also be set.
- » New: For **images** the *menubox* style can now also be set on the page with the specific properties.

9.6 USW

- » Gnats 11218: If a **label** of a **user-defined attribute** is queried and this label is 31 characters long (without the period), this could lead to subsequent crashes. This problem has been remedied. The cause of the problem was that a part of the USW attribute table was overwritten. This table for instance, is also accessed when a USW class has been registered and an unknown attribute is being searched for.
- » Gnats 11143: Reducing the **line index** of a **matrix attribute** no longer leads to a crash. In the example in which the problem was detected, *.itemcount* was defined as the "rowindex" of *.content*. Reducing *.itemcount* led to a crash.

9.6.1 Extensions of the USW interface

The USW interface now also supports a "preferred size" (size definition 0) for USW widgets as well as the raster categories "border object" and "singleline object". There is now also a formatter functionality for USW widgets.

9.6.1.1 Information on the use of resource with USW widgets

It must be ensured that dynamic changes to resources (e.g. of the image file path or the font size) that a USW class uses are not passed on. It is therefore recommended to use only immutable (static) resources.

9.6.1.2 Preferred size and raster categories

9.6.1.2.1 Geometry calculation

Position and size of objects that are implemented as USW, are principally set by IDM on the basis of their geometry attributes (*.xleft*, *.yauto*, etc.) and may not be changed. If no width or height was set (*.width*, *.height*) a default width or height is set for the widgets. This default width or -height can be pre-defined by the USW with the new function **UC_Control**. 100 is used if nothing was defined.

Raster categories are used when calculating the geometry based on raster coordinates, which can also be set with the function **UC_Control**:

- » An object can have a frame for which special rules apply in terms of converting raster coordinates to pixel coordinates. The calculated position (*.xleft*, *.xright*, *.ytop* and *.ybottom*) is increased by half a raster unit in this process. The size (*.width* and *.height*) is reduced in turn by a whole raster unit. The raster positions of child objects are also increased by half a raster unit.
- » An object that is typically just one line high, can obtain the default height and be centered vertically in the raster line by defining the height with 0.

9.6.1.3 Formatter functionality

9.6.1.3.1 Format handling

The USW interface supports the use of format and format resources in two ways.

1. Application of a format with the new function **applyFormat** to obtain the resulting string to be displayed from a string (see chapter "applyFormat").
2. The use of formats for editable content with the interaction of content string, format resource, cursor- and selection position as well as the format-based length restriction (maxchars).

9.6.1.3.2 Functions for the use of formats for editable content

The following functions are available for the interplay and the transformation between the content string and display string:

<code>fmtDisplay</code>	Initializes the format and returns the display string.
<code>fmtNavigate</code>	Moves the cursor or the selection in the display string.
<code>fmtModify</code>	Changes the string at the specified location in the display string.
<code>fmtFocus</code>	Reports a focus change to the format.
<code>fmtContent</code>	Returns the content string as well as the cursor and the selection position in the content string. Is only used when changes to the attributes themselves are forwarded.

This functions are explained in detail in chapter “New Functions”.

With most functions, the attributes involved in format handling are defined via the **UC_FormatDescr** structure. In addition, almost all functions return the information (display string as well as cursor and selection position in the display) necessary for USW class implementation.

The functions above administer the additionally required content and display information and invoke the respective formatter tasks. Administration is designed so that one editable, formatted content with content, display and additional information is stored per widget. Here, indexed content is allowed for content, format and cursor position. When using indexed format therefore involves the exchange of additional information with a respective formatter change.

To facilitate the use of content with formats, the functions have the following properties:

- » Also when defining null ID as format resource, the respective action (cursor movement or content change) is executed.
- » If desired, changes made to content and cursor information is written back to the attributes and an event is sent.

9.6.1.3.2.1 Use of the functions

The use of functions for the format handling of editable content should be as follows:

- » In USW-Callback **UC_Create** initialization is executed with **fmtDisplay** and the first query after the display string.
- » In Callback **UC_Update** the attributes involved in formatting are updated with the function **fmtDisplay**.
- » The receipt or loss of keyboard focus is signalized with the function **fmtFocus** via the parameter *enter* = *TRUE* or *FALSE*.
- » For entries and changes to the cursor positions the functions **fmtModify** and **fmtNavigate** should be called.

» In Callback **UC_Destroy** the saved display information is released by calling **fmtDisplay** with parameter *formatDescr* = *NULL*.

Example

You can view a code example in the USW class `uctext`, encoded in **uctext.c** of the USW example provided with the ISA Dialog Manager.

9.6.2 New or changed data types and structures

9.6.2.1 Data Types

9.6.2.1.1 UC_ControlName (new)

This type is a name for the task that is to be executed by the control call.

```
typedef enum {  
    UC_C_undef = 0,  
    UC_C_prefsiz,  
    UC_C_rastertype,  
    UC_C_resize  
} UC_ControlName;
```

9.6.2.1.2 UC_FormatFlags (new)

This type defines settings that influence format handling.

```
typedef enum {  
    UC_FF_OMITAPPLY = 1  
} UC_FormatFlags;
```

9.6.2.1.3 UC_FormatNaviMode (new)

This type defines potential navigation types for the **fmtNavigate** function.

```
typedef enum {  
    UC_FN_MOVECUR = 1,  
    UC_FN_MOVESEL = 2,  
    UC_FN_SETABS = 7  
} UC_FormatNaviMode;
```

9.6.2.1.4 UC_FormatStatus (new)

This type defines the potential return statuses of the functions designated for format handling. Depending on the function, combinations of individual values are also possible.

```
typedef enum {
```

```

UC_FS_FAILURE = 0,
UC_FS_SUCCESS = 1,
UC_FS_CONTENT = 2,
UC_FS_CURSOR = 4,
UC_FS_EVENT = 8
} UC_FormatStatus;

```

9.6.2.1.5 UC_RasterType (new)

This type is a name for the raster category.

```

typedef enum {
    UC_RT_normal = 0,
    UC_RT_hasborderraster,
    UC_RT_centersvertical
} UC_RasterType;

```

9.6.2.2 Structures

9.6.2.2.1 UC_API (changed)

Structure of the USW interface with the necessary function pointers. The structure may not be changed.

Definition

```

typedef struct {
    DM_UInt1    version;
    DM_UInt1    codepage;

    UC_QueryAPIFunc    queryAPI;
    UC_AttrGetFunc     attrGet;
    UC_AttrSetFunc     attrSet;
    UC_SendEventFunc   sendEvent;
    UC_AttrNumFunc     attrNum;
    UC_CreateStringFunc createString;
    UC_FreeStringFunc  freeString;
    UC_EditorRunningFunc editorRunning;
    UC_RegisterClassFunc registerClass;
    UC_RegisterWidgetFunc registerWidget;
    UC_GetToolkitDataFunc getToolkitData;
    UC_ApplyFormatFunc  applyFormat; //new
    UC_FmtDisplayFunc   fmtDisplay;  //new
    UC_FmtModifyFunc    fmtModify;    //new
    UC_FmtNavigateFunc   fmtNavigate;  //new
    UC_FmtContentFunc    fmtContent;   //new

```

```

    UC_FmtFocusFunc    fmtFocus;    //new
    UC_ControlFunc     control;      //new
} UC_API;

```

Meaning of the Elements

Element	Meaning
version	Version of the interface structure.
codepage	Code page used for the strings.
queryAPI...control	Function pointer to interface functions.

9.6.2.2.2 UC_ControlArg (new)

This structure is used for data interchange via the UC_Control function. It contains a name that defines the task. The task could be additional data allocated in a union element. Chapter “New Constants” defines which union element must be used.

Definition

```

typedef struct {
    UC_ControlName name;
    union {
        UC_Dimension prefsiz;
        UC_RasterType rastertype;
        ...
    } arg;
} UC_ControlArg;

```

Meaning of the Elements

Element	Meaning
name	Name of the task
arg	Argument of the task

9.6.2.2.3 UC_Dimension (new)

General definition of the size of a rectangular area.

Definition

```

typedef struct {
    DM_UInt2 width;
    DM_UInt2 height;
} UC_Dimension;

```


Meaning of the Elements

Element	Meaning
width	Width of the area
height	Height of the area

9.6.2.2.4 UC_FormatDesc (new)

This structure is used to define the format handling.

Definition

```
typedef struct {
    UC_AttrIndex    formatAttr;
    UC_AttrIndex    contentAttr;
    UC_AttrIndex    curposAttr;
    UC_AttrIndex    selposAttr;
    UC_AttrIndex    maxcharsAttr;
    DM_UInt4        contentEvmask;
    DM_UInt4        cursorEvmask;
    UC_FormatFlags  flags;
} UC_FormatDescr;
```

Meaning of the Elements

Element	Meaning
formatAttr	Defines the attribute that contains the ID of the format resource. The attribute must have the data type <i>DT_format</i> .
contentAttr	Defines the attribute that contains the content that is to be formatted. The attribute must have the data type <i>DT_string</i> or <i>DT_text</i> .
curposAttr	Defines the attribute that contains the cursor position. The attribute must have the data type <i>DT_integer</i> . It is not taken into consideration with an element value of 0.
selposAttr	Defines the attribute that contains the selection position. The attribute must have the data type <i>DT_integer</i> . It is not taken into consideration with an element value of 0.

Element	Meaning
maxCharsAttr	Defines the attribute that contains the maximum character length that should be passed on to the format. The attribute must have the data type <i>DT_integer</i> . It is not taken into consideration with an element value of 0.
contentEvmask	Event mask to signalize content changes.
cursorEvmask	Event mask to signalize changes to the cursor or the selection position.
flags	Addition settings for format handling in the form of flags that can be combined (added) bit-wise.

9.6.3 New Functions

9.6.3.1 applyFormat

This function returns the formatted display string for a content string.

```
typedef DM_String (DML_default DM_EXPORT *UC_ApplyFormatFunc)
__((DM_ID id,
   DM_ID fmtorfuncid,
   DM_String fmtstring,
   DM_String content,
   DM_Options options));
```

Parameters

id	Identifier of the USW object is required to supply the display string in the code-page which was for the widget intended
fmtorfuncid	ID if a format resource or a format function The following information is possible in combination with <i>fmtstring</i> : <ul style="list-style-type: none"> » Format resource in <i>fmtorfuncid</i> or » format function in <i>fmtorfuncid</i> with format string in <i>fmtstring</i> or » formatstring in <i>fmtstring</i>
fmtstring	Formatstring (see <i>fmtorfuncid</i>)
content	String on which the format is to be applied
options	For future extension, currently only use with 0

Return Value

Display string	Formatted string (content string on which the defined format was used)
NULL	Error in the specified format, the parameters or the call situation (run state)

9.6.3.2 control

This function is used to set control information such as preferred size or raster category.

```
typedef DM_Boolean    (DML_default DM_EXPORT *UC_ControlFunc)
__((DM_ID dmid,
   UC_ControlArg *pArg,
   DM_Options options));
```

Parameters

dmid	Identifier of the USW object
pArg	Pointer to a structure with a task, which sets control information (see chapter “UC_ControlArg (new)”)
options	For future extension, currently only use with 0

For the tasks *UC_C_prefsize*, *UC_C_rastertype* and *UC_C_resize* the following must be observed:

- » The tasks may only be used in the callbacks **UC_Create** and **UC_Update**.
- » The tasks are not executed directly but are buffered and executed at the end of **UC_Create** or **UC_Update**.

Return Value

TRUE	The task was executed successfully and the control information was set.
FALSE	Error, the tasks could not be executed and the control information was not set.

9.6.3.3 fmtContent

This function returns the last buffered content information, i.e. the content string with additional information most recently used by format handling.

```
typedef UC_FormatStatus    (DML_default DM_EXPORT
   *UC_FmtContentFunc)
__((DM_ID id,
   DM_FmtContent *fmtContentResult,
   DM_Options options));
```

Parameters

id	Identifier of the USW object
fmtContentResult	Pointer to the most recently buffered content information with content string, cursor and selection position, string length, etc. (see structure DM_FmtContent in IDMuser.h).
options	For future extension, currently only use with 0.

Format handling of a USW widget is initialized by the **fmtDisplay** function, which is to be called before **fmtContent**. **FmtDisplay** as well as the functions **fmtModify** and **fmtNavigate** can modify the content information (content string, cursor and selection position). These modifications can be simply queried with **fmtContent**.

FmtContent is only necessary in the case of delayed return of changes (the respective format handling functions contain **UC_FF_OMITAPPLY** in the *flags* element of the **FormatDescr** structure). In other cases, changes of content information are automatically passed on to the attributes that are defined in the **FormatDescr** structure.

Return Value

UC_FS_SUCCESS	Function was executed error-free.
UC_FS_FAILURE	An error occurred while executing the function.

9.6.3.4 fmtDisplay

This function is used to initialize format handling, to administer the information necessary for this and to supply the formatted display string. If the widget contains several pieces of editable content, which has to be shown by different specifications in the **FormatDescr** structure, the change should take place via **fmtDisplay**.

```
typedef UC_FormatStatus (DML_default DM_EXPORT
    *UC_FmtDisplayFunc)
    __((DM_ID id,
    UC_FormatDescr * formatDescr,
    UC_Index row,
    UC_Index col,
    DM_FmtDisplay *fmtDisplayResult,
    DM_UInt4 *dpymaxcharsResult,
    DM_Options options));
```

Parameters

id	Identifier of the USW object
formatDescr	Description structure for format handling (see below). If this parameter is <i>NULL</i> , the buffered display structure information is released. Format handling administration information is cleared when hiding or destroying the USW object.
row	1. index value for indexed attributes.
col	2. index value for indexed attributes.
fmtDisplayResult	Upon successful execution of the function, with this pointer you receive the display information (formatted string, cursor and selection position, string length and its size in bytes) for further use. If no format has been set, (the attribute defined in the element <i>formatAttr</i> of <i>formatDescr</i> contains the identifier <i>NULL</i>) <i>fmtDisplayResult</i> in the element <i>string</i> of the DM_FmtDisplay structure (see IDMuser.h) contains the unchanged content string. Therefore no different treatment of content with and without set format is necessary.
dpymaxcharsResult	With this pointer you get the maximum string length for the display string administered by format.
options	When specifying <i>DMF_ShipEvents</i> dialog events are additionally sent with attribute changes. These must be specified in the <i>contentEvmask</i> and <i>cursorEvmask</i> elements of the <i>formatDescr</i> , whereby <i>EM_changed</i> and <i>EM_extevent</i> are not allowed.

formatDescr

For the determination of the display string, the function queries the content of the content attribute of the **FormatDescr** structure (element *contentAttr*), the format resource from the format attribute (element *formatAttr*) as well as the information necessary for format initialization such as cursor or selection position or the maximum character length (elements *curposAttr*, *selposAttr* and *maxcharsAttr* of the **FormatDescr** structure). The parameters *row* and *col* are used for all attribute accesses. For index specifications ≤ 0 for scalar attributes the scalar value is used, so that a mixture of indexed content attributes and scalar format attributes is possible.

The content attribute should have the data type *DT_string*. The data type *DT_text* is also allowed, yet leads to highly inefficient use and should **only** be used in the event of a delayed return transfer to the content attribute (with *UC_FF_OMITAPPLY* in the *flags* element of the **FormatDescr** structure and restoring of the content, e.g. only in the event of loss of focus and not for every key entry).

The format attribute should have data type *DT_format* and return a format resource or the *NULL* identifier. The other attributes should have the type *DT_integer* in order to be taken into consideration.

If the `UC_FF_OMITAPPLY` bit is not set in the *flags* element of the **FormatDesc** structure, the changes to the content information (string, cursor position or selection position) made by format handling are immediately stored within the function call in the respective attribute. The changes can then also be consistently queried in the Rule Language.

Return Value

UC_FS_
SUCCESS

If successful.

bitwise connected with

UC_FS_
CONTENT or
UC_FS_
CURSOR

Signalizes a change in content information (**not** display information). This means that the content string or the cursor or selection position have been changed and the changes are set in the respective attributes. Only if `UC_FF_OMITAPPLY` is **not** set in the flags element of the **FormatDesc** structure.

and

UC_FS_EVENT

Signalizes the sending of events (which were defined in the **FormatDesc** structure). Only if `DMF_ShipEvents` has been set in the *options* parameter.

UC_FS_
FAILURE

In the case of an error.

9.6.3.5 fmtFocus

This function signalizes format handling via its *enter* parameter on the receipt and loss of keyboard focus.

This function ensures that the respective format task *FMTK_enter* or *FMTK_leave* are only called once.

```
typedef UC_FormatStatus    (DML_default DM_EXPORT
    *UC_FmtFocusFunc)
    __((DM_ID id,
    UC_FormatDesc * formatDescr,
    UC_Index row,
    UC_Index col,
    DM_Boolean enter,
    DM_FmtDisplay *fmtDisplayResult,
    DM_Options options));
```

Parameters

id	See chapter “fmtDisplay”.
formatDescr	
row	
col	
fmtDisplayResult	
options	
enter	After calling the function this parameter defines whether the USW object has keyboard focus or not. <i>TRUE</i> : The object has keyboard focus. <i>FALSE</i> : The object does not have keyboard focus.

Return Value

UC_FS_ SUCCESS	Function was executed error-free.
UC_FS_ FAILURE	An error occurred while executing the function.

9.6.3.6 fmtModify

This function, with the aid of the formatter functionality, replaces a part of the display string with another string.

```
typedef UC_FormatStatus    (DML_default DM_EXPORT
    *UC_FmtModifyFunc)
__((DM_ID id,
    UC_FormatDescr * formatDescr,
    UC_Index row,
    UC_Index col,
    DM_String string,
    DM_UInt4 curpos,
    DM_UInt4 selpos,
    DM_FmtDisplay *fmtDisplayResult,
    DM_Options options));
```

Parameters

id	See chapter “fmtDisplay”.
formatDescr	Compatible behavior for unset format is also supported.
row	
col	
fmtDisplayResult	
options	
string	A string that is to be inserted in the display string between <i>curpos</i> and <i>selpos</i> .
curpos	Area in the display string that is replaced with a new string.
selpos	

Return Value

UC_FS_	If successful, see chapter “fmtDisplay”.
SUCCESS	
UC_FS_	
CONTENT	
UC_FS_	
CURSOR	
UC_FS_EVENT	
UC_FS_	In the event of an error or rejection of the change by the format.
FAILURE	

9.6.3.7 fmtNavigate

With the aid of the formatter functionality, this function changes the cursor position or selection position in the display string.

```
typedef UC_FormatStatus      (DML_default DM_EXPORT
    *UC_FmtNavigateFunc)
    __((DM_ID id,
    UC_FormatDescr * formatDescr,
    UC_Index row,
    UC_Index col,
    UC_FormatNaviMode mode,
    DM_Int4 xoffset,
    DM_Int4 yoffset,
    DM_UInt4 curpos,
    DM_UInt4 selpos,
    DM_FmtDisplay *fmtDisplayResult,
    DM_Options options));
```


Parameters

id	See chapter “fmtDisplay”.
formatDescr	Compatible behavior for unset format is also supported.
row	
col	
fmtDisplayResult	
options	
mode	Defines the functionality of the cursor or selection movement. Possible values are: <i>UC_FN_SETABS</i> : absolute positioning in the display string, on the position defined in <i>curpos</i> and possibly in <i>selpos</i> . <i>UC_FN_MOVECUR</i> , <i>UC_FN_MOVESEL</i> or a combination of both: relative offset in the display string by <i>xoffset</i> in horizontal direction and by <i>yoffset</i> in vertical direction.
xoffset	Relative offsetting of cursor or selection in X direction (horizontal) if <i>mode</i> = <i>UC_FN_MOVECUR</i> , <i>mode</i> = <i>UC_FN_MOVESEL</i> or a combination of both. Positive values are offset forwards (to the right) and negative values backwards (to the left).
yoffset	Relative offsetting of cursor or selection in Y direction (horizontal) if <i>mode</i> = <i>UC_FN_MOVECUR</i> , <i>mode</i> = <i>UC_FN_MOVESEL</i> or a combination of both. Positive values are offset forwards (down) and negative values backwards (up).
curpos	New, absolute position of the cursor in the display string, if <i>mode</i> = <i>UC_FN_SETABS</i> .
selpos	New, absolute position of the selection (together with <i>curpos</i>) in the display string, if <i>mode</i> = <i>UC_FN_SETABS</i> .

Return Value

UC_FS_SUCCESS	If successful, see chapter “fmtDisplay”.
UC_FS_CONTENT	
UC_FS_CURSOR	
UC_FS_EVENT	
UC_FS_FAILURE	
UC_FS_FAILURE	In the event of an error, if the cursor or selection movement is rejected by the format (typically the return value of the format tasks <i>FMTK_setcursorabs</i> or <i>FMTK_keynavigate</i>).

Note

The formats offered by IDM do not support multiple lines.

9.6.4 New Constants

Constants	Data type	Meaning
UC_C_prefsize	UC_ControlName	Preferred size should be set, the element <i>arg.prefsiz</i> e contains the value.
UC_C_rastertype	UC_ControlName	The raster category should be set, the element <i>arg.ras-</i> <i>tertype</i> contains the value.
UC_C_resize	UC_ControlName	A new calculation of position and size should be triggered, no element required.
UC_FF_omitapply	UC_FormatFlags	Bitflag for the flags element of the UC_FormatDescr struc- ture to signalize to format handling that content and cursor changes are not to be passed on to the respective attribute.
UC_FN_MOVECUR	UC_FmtNaviMode	A relative movement of the cursor position is desired.
UC_FN_MOVESEL	UC_FmtNaviMode	A relative movement of the selection position is desired.
UC_FN_SETABS	UC_FmtNaviMode	An absolute setting of cursor and selection position is desired.
UC_FS_CONTENT	UC_FormatStatus	Flag for signaling that content has been changed.
UC_FS_CURSOR	UC_FormatStatus	Flag for signaling that the cursor or the selection position has been changed.
UC_FS_EVENT	UC_FormatStatus	Flag to signalize that an event has been sent
UC_FS_FAILURE	UC_FormatStatus	The format handling function could not be executed suc- cessfully.
UC_FS_SUCCESS	UC_FormatStatus	The format handling function was executed successfully.
UC_RT_centervertical	UC_Rastertype	The object receives the standard height and is centered ver- tically in the raster line, if 0 has been defined for its height. See chapter “Geometry calculation”.

Constants	Data type	Meaning
UC_RT_has-borderraster	UC_Rastertype	The object has a frame. Special rules apply for the conversion of raster coordinates to pixel coordinates. See chapter “Geometry calculation”.
UC_RT_normal	UC_Rastertype	When converting raster coordinates to pixel coordinates, no special raster categories are used. See chapter “Geometry calculation”.

10 Version A.05.02.f4

10.1 Motif

- » Gnats 11259: So as to not interrupt text entry, with **edittexts** having a **pop-up menu**, in which **mnemonics** are defined, the popup menu is no longer opened automatically when entering a mnemonic character (as is common from Motif 2.1).

For the other focusable objects **without** text entry, such as pushbuttons, checkbox, radiobutton, image, scrollbar and spinbox with statictext, the popup menu opens automatically when entering a mnemonic character, as before.

Note

As with all object classes, the pop-up menu in the edittext from Motif 2.1 can be opened by default with the key combination **Shift + F10**.

As an alternative to mnemonics, accelerators can be used.

11 Version A.05.02.f3

11.1 Motif

- » Gnats 11214: In the Motif-2 version of the IDM the **cursor control keys** can be used to navigate forward **and backward** through ***edittext*** objects.
- » Gnats 11199: Dynamic instancing of a ***menubox*** as a child of a visible splitbox does not cause an **assfail**.

11.2 Core

- » Gnats 11223: **Associative arrays** can now be filled **with a maximum of 65,528 entries**. If it is attempted to create more entries, this is logged as "[SV] FAILED (IDM-E-IndexRange)". This way there are no more crashes when attempting to create more than 65,528 entries.

11.3 USW

- » Gnats 11218: If a *label* of a **user-defined attribute** is queried and this label is 31 characters long (without the period), this could lead to subsequent crashes. This problem has been remedied. The cause of the problem was that a part of the USW attribute table was overwritten. This table for instance, is also accessed when a USW class has been registered and an unknown attribute is being searched for.
- » Gnats 11143: Reducing the **line index** of a **matrix attribute** no longer leads to a crash. In the example in which the problem was detected, `.itemcount` was defined as the "rowindex" of `.content`. Reducing `.itemcount` led to a crash.

12 Version A.05.02.f2

12.1 Windows

- » Gnats 11163: For an **edittext with format** the display of new and longer content is now **no** longer **cut** to the length of the previous, shorter content. The problem was **not** visible if, following the assignment of the new and longer content, further "actions" were performed (e.g. focus setting on the edittext).

12.2 Motif

- » Gnats 11165: IDM could crash if a model with a **notebook** and **notepages** was instantiated in the visible area. The problem was remedied. The cause was due to the failure of XtWidgetDestroy() when inheriting the *.font* attribute from the notebook to the notepages.
- » Gnats 11161: When editing an **edittext** in a **tablefield** the **tabulator key** is now used to jump to the **next cell**. With this, the behavior now once again corresponds to the IDM version A.05.01.
- » Gnats 10633: The sensitivity status of **poptexts** in a **toolbar** is now correctly updated if the sensitivity of a window changes. The problem was noticed in the IDM editor when opening the object history with the most recently selected objects could induce the desktop to no longer respond.

12.3 Unix

- » Gnats 11170: A problem was remedied that disables the IDM from being installed as the **install-ation** was **terminated** with error messages. This error was observed under Solaris 10 with the Distributed Dialog Manager. The cause was attributed to the fact that during the generation of serial numbers, the options to be activated were not processed correctly.

13 Version A.05.02.f

13.1 Compatibility

As internal structures were changed, C-, C++- and COBOL source texts of IDM applications have to be recompiled and COBOL copybooks need to be regenerated.

13.2 Windows

- » Gnats 11128: With the **edittext** object, it might occur that the text was pushed out of the visible area to the left, although there was sufficient space. This behavior was observed with an edittext connected to a tablefield with `.editpos = true`. The error has been remedied as the text position is now updated when a new text is assigned or the size of an object is changed.
- » Gnats 11051: The text does not remain selected in a **poptext** object if it loses focus.
- » Gnats 10958: The text is not scrolled in an **edittext** with format if the edittext is activated or deactivated.

Note for Developers Who Use an IDM Object Monitor or Windows Subclassing

The ISA Dialog Manager changes the text of **edittext** objects now also with EM_REPLACESEL messages and no longer just with WM_SETTEXT messages.

- » Gnats 10918: With the **pop-up menu** now also all **menuitem** objects are shown if they have been relocated in the child vector while the object was visible to which the pop-up menu belongs.
- » Gnats 10902: From WINDOWS VISTA with active "Visual Styles", the title of a **notepage** is moved to the right if it contains a **mnemonic**. This behavior has **not** been changed as it is caused by the "Visual Styles".

Note

To prevent the effects, use the following common Windows settings for **notebooks**

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

If **mnemonics** are used, **all** notepages of a notebook must have mnemonics so that the same offset occurs for all titles.

- » Gnats 10885: The **popup menu** of a **poptext** with open selection list is only opened if the cursor is **in** the poptext (incl. selection list) when right-clicking the mouse. With this the first right-click now opens the popup menu on another object.

Note

As it cannot be prevented that the selection list collapses when opening the popup menu, the popup menu is **moved to the edge** of the poptext. The alternatives, not opening the popup menu in the selection list or at the position of the cursor, seem to be a poorer choice.

- » New (Gnats 10874): With the **64-bit version** of the ISA Dialog Manager, the **OLE interface** can now also process **64-bit data types**. Here, a 64-bit data type is converted to the IDM data type

DT_pointer, as the ISA Dialog Manager is principally designed for 32-bit integer values.

Note

DT_pointer is converted by a 32-Bit-IDM-OLE **server** into the OLE data type VT_UI4, while a IDM-OLE-**client** interprets VT_UI4 as *DT_integer*.

- » Gnats 10865: For **OLE events**, the **output parameters** are now returned correctly if input parameters have been defined before the output parameters. Processing the event rule **no** longer leads to an error message [E: Error occurred while copying values into output-parameters.].
- » New (Gnats 10861): In the **XP version** of the ISA Dialog Managers a path (name) HAND can now also be defined for a **cursor** resource. This loads the **system cursor** HAND.

Note

- » If the path HAND is used for a **self-defined** cursor, **now** the system cursor is loaded and **no longer** the self-defined cursor.
- » The system cursor HAND is not available **before** Windows 2000, instead the standard cursor is reverted to. If older Windows version have to be supported, an own cursor can be defined with the HAND path.
- » Gnats 10857: If a **tablefield** line which was beneath the mouse cursor was hidden, its old content was drawn as soon as the mouse was moved. The cause was a faulty end condition in a loop, which has now been corrected. The error may have also possibly been the cause for sporadic crashes.
- » Gnats 10842: In the **edittext** the *.xmargin* setting now works correctly again. Problems could occur from version A.05.02.d. In some cases, this caused the content of an edittext to be scrolled to the left. The IDM versions before A.05.02.d are not affected by the problem.
- » Gnats 10831: A problem was remedied with the **tablefield** that could lead to crashes (assfail in general) when the *.edittext* attribute was switched in the visible state. The cause of the problem were inconsistently restored internal references or internal references restored at the wrong time (during destroy).
- » Gnats 10825: The *.topitem* attribute of the **treeview** objects is not reset to 1 when the object is destroyed. The error was observed while changing the *picture* attribute with the function **DM_SetVectorValue()**, as the treeview object is destroyed and re-created here, if the index is 0 or not defined (corresponds to 0).
- » Gnats 10811: With active “Visual Styles” the **focus frame** of an **image** object is now directly drawn on the frame defined by the “Visual Styles”. Therefore it is not drawn further to the inside than without “Visual Styles”. In addition, the calculation of the preferred size has been corrected so that an image object is not set up too low, when it only has an image and no text.

Note

Since IDM version A.05.02.c images with active “Visual Styles” are drawn up to the predefined frame. In this process, the calculation of the preferred size has slightly changed.

- » Gnats 10799: For **windows** with *.xauto* = 0 or *.yauto* = 0 **toolbar** objects are now shown when windows are made visible and not after changing the size of the window.

- » Gnats 10795: If the **Return** key is pressed in a **multi-line edittext with format**, a “\r\n” was transferred to the content by mistake. Plus with an “\n” no linebreak was added to the content if a format had been defined. This error has been remedied. Regarding the maximum number of characters, it must be observed that only one character is inserted when pressing the **Return** key.

Note

The codepage "utfwin" is not suitable as an application codepage if character strings have to be processed that contain <carriage return> characters. The reason is that <Linefeed> characters (linebreak) with this codepage are converted into the character sequence <Carriage-Return><Linefeed>. These are overread to prevent duplicate <Carriage-Return> characters. Instead of "utfwin" the codepages "utf16" or "utf16l" should be used as they do not execute this special treatment.

- » Gnats 10971: When making a **window** with toolbar visible, no more unnecessary *resize* events occur.

Comment

This entry was supplemented retroactively as the problem was remedied with the correction of another error (probably Gnats 10799). This was first detected during the development of version A.05.02.g.

- » Gnats 10606: A **C#-Assembly** could crash when attempting to send OLE events to the ISA Dialog Manager. The cause was that the OLE method **IDispatch::GetIDsOfNames** – which is neither required nor necessary for an event interface – returned the error E_NOTIMPLEMENTED. The problem has now been remedied by fully implementing the **IDispatch interface** of the **IDM-OLE interface**.

Note

It is recommended to **not** use any events with return values as an unprocessed event may suffice to cause C# to crash.

13.3 Motif

- » Gnats 11080: When making **treeview** objects visible, the stacking order is now correct, so that the following objects in the child vector appear in front of the treeview.
- » Gnats 11066: Changing focus by hiding a **grouping objects** while on of its children was focused could lead to the application freezing. This was caused by an endless loop when searching for the next object to be focused, which has now been removed.
- » Gnats 11045: The dynamic change of *.sensitive* now also works with a **spinbox** if it was declared with *.sensitive = false*.
- » Gnats 11004: It could occur that an application failed to respond if a loss of focus was forced with a **spinbox** (or the respective edittext). An endless loop, which has now been removed, was the cause of this.
- » Gnats 10973: When making an editable **poptext** visible the *.editable* attribute is now correctly observed.

- » Gnats 10895: With some Window Managers, it was observed that the background color changed when the foreground color (text color) in an **edittext** was changed in the visible state. Now the default background color for text and list objects (edittext, listbox, poptext) is consistently reset to the original color.
- » Gnats 10821: A **notepage** is now correctly made visible after relocating it in the child vector of a visible notebook.
- » Gnats 10819: When focus is lost, the **edittext** no longer generates any false modified and duplicate select events.

Note for Motif Platforms

Depending on the **Window activation mode** of the Window Manager, repeated *activate*, *focus* and *deselect* events may occur with a **poptext** if the mousekey is released above another window, after it had been pressed to open the popup list and to select an entry. The focus status of poptext and window however is correct in terms of X-events. The behavior usually occurs in the mode "Window activation on click" when the Window Manager activates the other window when releasing the mouse key.

- » Gnats 10743, 10742: The **treeview** object can cause freezes and crashes if the number of entries reaches the critical limit of the Motif **XmContainer** widget (ca. 1,500 entries, a respective message was output in the log file). The problem has been remedied. This also results in better performance during expanding and collapsing if `.style[style_buttons] = false`.

13.4 Core

- » Gnats 11016: **beep()** no longer crashes if the object was destroyed before.
- » Gnats 10984: With the **dumpstate** output an application no longer crashes if events are output for an object that has been destroyed in the meantime.
- » Gnats 10904: During the conversion of strings in codepage *CP_utfwin* a "\r" in the string no longer leads to a crash of the conversion, but rather the character is merely skipped. This is why strings with linebreaks are now correctly displayed when they are inserted into a **multi-line edittext**.
- » Gnats 10880: The resolving of object paths, e.g. when calling the function **DM_PathTold()** no longer incorrectly reports a possible recursion, after it has been called with many unknown or invalid object paths.
- » Gnats 10883: The definition of the interface function **DM_DumpState()** is now contained in the IDMuser.h.
- » Gnats 10836: An error was remedied in the **poptext** object that could lead to incorrect selections of the content (*.content*) under Windows with a poptext with **pattern format** and `.style = edittext`. The error could be found in the incorrect interpretation of *startsel*- or *endsel* values.
- » Gnats 10806: The function **DM_GetVectorValue()** no longer returns any temporary (internal) format resources for the attribute *.format*.

Note

- » Please note that a *.format* query can only return the format definition when using format resources. Null is returned for format definitions via *formatstring* or *format* function. This is why it is recommended to use format resources.
- » The options *DMF_GetMasterString* and *DMF_GetLocalString* provide, as before, a format-string as with a format definition via *formatstring*.
- » Therefore the returned values of the function **DM_GetVectorValue()** now correspond to those of function **DM_GetValueIndex()** or a **getvalue()** in the Rule Language.
- » Gnats 10756: The internal function **DM_GetRecord()** – which is used by other interface functions to transfer **records** from the dialog to a C-application – no longer overwrites structure elements if an attribute access fails.
- » Gnats 10755: Dynamic change of *.canvasfunc* on a **canvas** object when visible now causes the cancelation of the old function and the registration of the new function.
- » Gnats 10720: The built-in function **querybox()** no longer generates a changed event if the dialogbox is destroyed within the call. This also removes the cause for the crashes which could occur in this situation.
- » Gnats 10572: The *pass* and *throw* instructions now also return a FAIL if they are used in a **set** method. In addition, fail instructions are now also correctly observed in the simulation rules for functions to catch errors.

The following therefore applies for the forwarding of errors with *pass* and *throw* as well as *checking* and *catching* with *fail* instructions:

 - » Errors are forwarded to the caller from redefined *get* and *set* methods as well as user-defined rules, methods and simulation rules.
 - » Errors from the methods *init*, *clean*, *setclip* and *action* are not forwarded to the caller.
- » Gnats 10057: **DM_ShutDownProc()** is now defined with the argument type *void* to prevent warnings in the case of restrictive compiler settings.

13.5 COBOL

- » Gnats 10889: For the file *IDMcobws.cob* it is now ensured that no line has more than 72 columns (characters). This avoids the error message "Period missing" as COBOL compilers do not interpret any more signs beyond column 72.

13.6 IDM Builder Process

13.6.1 The builder process mode

For complex, modular IDM applications, the time requirement for the creation of the application with **make** or other build management tools are significantly determined by the loading times of the modules and dialogs.

Every call of the simulation (IDM or PIDM) for the creation of an interface, binary, funcmap- or trampoline file necessarily loads the specified dialog or module file. If the application consists of several modules with many deep import dependencies, long loading times are primarily caused by reloading the imported modules.

A clear reduction of this reloading time can be achieved by using the IDM or the PIDM in the **builder process mode**.

Here an IDM simulation program runs in the background in the builder process mode as **server** and waits for requests to create interface, binary, funcmap and trampoline files. Hereby, this IDM builder process runs in the **shared modules mode** which is normally activated by setting the environment variable `DM_SHARED_MODULES`. In this process, the reuse of imported modules is based on the equality of the interface name and not, as is usually the case, on the import descriptor and the super-ordinate import. This leads to a clear reduction of loading time for modules with many imports.

The IDM or PIDM calls required during the creation of the application merely connect as **client** on the IDM builder process. The necessary arguments are sent to the server and the return status is passed on.

The IDM builder process is not started manually, but starts automatically when correctly used (option: **+builder**). After a definable time-out period (option: **-buildertimeout <secs>**) it stops automatically.

With this, the previous use of the IDM or the PIDM should be converted to the use in the builder process mode with relative ease and without any problems.

13.6.1.1 Example

Makefile with a simple modularized dialog:

```
PIDM=pidm
IDMARGS=-IDMenv MODPATH=if;.
IDMARGS_WRITEBIN=$(IDM_ARGS) +searchsymbol MODPATH -writebin
IDMARGS_WRITEIF=$(IDM_ARGS) -writeexport
all:: bin/dialog.dlg
bin/dialog.dlg: dialog.dlg if/defaults.if
$(PIDM) $(IDMARGS_WRITEBIN) @$ dialog.dlg
if/defaults.if: defaults.mod
$(PIDM) $(IDMARGS_WRITEIF) @$ dialog.dlg
bin/defaults.mod: defaults.mod
$(PIDM) $(IDMARGS_WRITEBIN) @$ defaults.mod
```

Converted makefile, the activation of the builder process mode is underlined:

```
PIDM=pidm
IDMARGS=-IDMenv MODPATH=if;. +builder
# Windows:
# IDMARGS=-IDMconsole -IDMenv MODPATH=if:. +builder
IDMARGS_WRITEBIN=$(IDM_ARGS) +searchsymbol MODPATH -writebin
IDMARGS_WRITEIF=$(IDM_ARGS) -writeexport
all:: bin/dialog.dlg builderstop
bin/dialog.dlg: dialog.dlg if/defaults.if
```

```
$(PIDM) $(IDMARGS_WRITEBIN) @$ dialog.dlg
if/defaults.if: defaults.mod
$(PIDM) $(IDMARGS_WRITEIF) @$ dialog.dlg
bin/defaults.mod: defaults.mod
$(PIDM) $(IDMARGS_WRITEBIN) @$ defaults.mod
builderstop:: # optional
$(PIDM) -builderstop
```

13.6.1.2 Usage instructions

For the reuse of the same IDM builder process (i.e. the same IDM or PIDM in the builder process server mode), the process number of the father process of the IDM or PIDM client must be identical. Otherwise, the option **-builderid** can be used and the IDM builder process can be manually started and stopped. It is advisable that IDM specific construction steps take place successively to enable the reuse of the IDM builder process within the adjustable time-out period.

No IDM builder process runs initially. It is automatically started with the option **+builder** and contains the environment of the process being started. During the communication between the builder client and the server, the environment variables and the work directory of the builder client set with **-IDMenv** are passed on. These variables therefore overlay the variable set in the IDM builder process. Changing the work directory ensures an identical situation for client and server. If no connection can be established between client and server, this leads to the typical error messages such as “... *can't open/connect builder pipe* ...”.

The IDM builder process is not intended to parallelize the build. Only one client at a time can connect to the IDM builder process.

If the maximum number of modules has been reached (ca. 4,000 modules can be loaded simultaneously) in the IDM builder process during loading, the IDM builder process is restarted and the message [I: restart build server] is output. Here, the loaded, reusable import modules are lost.

13.6.1.3 Usage information for the IDM Eclipse Plugin

For the use of the builder process mode with the MAKEGEN plugin, the options **-IDMconsole** and **+builder** should be used. They enable Eclipse CDT (C/C++ Development Tooling) for makefile projects, to check IDM error messages from the IDM or PIDM error parser of the IDM Eclipse plugin.

13.6.1.4 Particularities

- » The builder process mode is available for Microsoft Windows and Unix or Linux, but not for VMS.
- » If an IDM builder process has to be started first, this is followed by a short delay as the process start must be waited for.
- » During the build run, waiting times may arise that are caused by the delayed completion of the builder following a time-out or by the make process waiting for the termination of all child pro-

cesses or the closing of the output channels. In this case, the IDM builder process can simply be stopped by calling IDM or PIDM with the **-builderstop** option.

- » Communication between the IDM or the PIDM in the builder process client server mode is performed via a **named pipe** (under Unix, typically in the /tmp directory). The respective safety settings and access authorizations should also be made. A forced termination of IDM or PIDM – e.g. with `kill -9` – should be avoided in order to ensure the cleanup of the pipe files.
- » When using **+builder** the forwarding of the output of the trace or the logfile (e.g. via the options – `IDMerrfile` and `-IDMtracefile`) is not effective as the IDM build process is started without redirection.
- » Under Microsoft Windows the error messages of the IDM or PIDM are normally displayed as messageboxes and other outputs are written to the IDM logfile. To receive error messages in one console, the option **-IDMconsole** may be used. This option is passed on when starting the IDM builder process. This should enable the build processes to pass on its relevant error messages and outputs to stdout or stderr.

13.6.2 New parameters for IDM and PIDM

+builder

This option causes the IDM and PIDM to run in the builder mode. If it has not already happened, it is started as IDM builder process in the background. All requests for writing interface, binary, fun-cmap or trampoline files are passed on to this IDM builder process. This process works in the **shared modules mode** to save time when reloading imported modules,

-builderid <Builder-Identifikator>

Normally the IDM builder process and the IDM or PIDM, that use the process in the builder mode, have the same father process. This can be bypassed by specifying an own builder ID as a file name **without** the preceding path.

-buildertimeout <secs>

With this option, the waiting time of the IDM or the PIDM in the building process mode is defined in seconds. After this time has been reached, the IDM builder process terminates. After the defined waiting period, connection attempts and calls fail.

-builderstop

With this option, a running IDM builder process can be stopped before reaching its time-out period.

13.6.3 New IDM library parameter

-IDMconsole

This option causes logfile and error messages to be written in the standard output channels (STDOUT) under MICROSOFT WINDOWS – analog to Unix. If the process does not run within the

console, a new console is opened for potential outputs. In the predecessor versions of Windows XP, a new console is **always** opened.

14 Version A.05.02.e4

14.1 Motif

- » Gnats 11066: Changing focus by hiding a grouping object while one of its children is focused could lead to the application freezing. This was caused by an endless loop when searching for the next object to be focused, which has now been removed.
- » Gnats 11027: In the **MCard** option of the IDM no more crashes take place if a keyboard sequence begins with several modifier keys.
- » New: In a C-program, tracing of the **MCard** module can be activated with
`DM_Trace_Mcard = TRUE`
and the string conversion between display and application codepage with
`DM_Mcard_ApplCodepage = CP_iso8859`.

14.2 Core

- » Gnats 10984: With the **dumpstate** output an application no longer crashes if events are output for an object that has been destroyed in the meantime.
- » Gnats 10904: During the conversion of strings in code page *CP_utfwin* a "\r" in the string no longer leads to a crash of the conversion, but rather the character was merely skipped. This is why strings with line breaks are now correctly displayed when inserting them into a **multi-line edittext**.
- » Gnats 10880: The resolving of object paths, e.g. when calling the function **DM_PathTold()** no long incorrectly reports a possible recursion, after it has been called with many unknown or invalid object paths.

15 Version A.05.02.e3

15.1 UNIX

- » New: Support for the **RHEL 5.5** platform (Red Hat Enterprise Linux with kernel version 2.6.18, GCC 4.1 and OpenMotif 2.3.1). The 32 and 64-bit versions are now supported.

15.2 Motif

- » Corrected: Now when a **Spinbox** is made invisible the error “*X Toolkit Error: Object "(null)" does not have windowed ancestor*” no longer occurs under Motif 2.3.1.

15.3 Editor

- » Corrected: In the editor it is now possible to create **image** and **progressbar** objects as children of a **statusbar**.

16 Version A.05.02.e2

16.1 Windows

- » Gnats 10857: When a **tablefield** row situated under the mouse cursor was made invisible, its old content was marked as soon as the mouse was moved. The reason for this was a faulty end condition of a loop, which has since been corrected. This error also led to sporadic system crashes.
- » Gnats 10842: The setting of *.xmargin* within the **edittext** is fully functional again. In version A.05.02.d some problems arose that caused the edittext to be scrolled to the left. This problem does not exist in the IDM versions previous to version A.05.02.d.
- » Gnats 10831: A problem in the **tablefield**, that eventually led to a crash (general assfail) when the edittext attribute in a visible state was altered in a visible state, has been corrected. The problem occurred due to inconsistent internal links or rather restoring them at the wrong time (e.g. during destroy).
- » Gnats 10795: When the **Return** key was pressed within a **multi-line edittext with format**, then “\r\n” falsely appeared in the content. In addition, when a format was defined no line break was inserted for the characters “\n” in the content. This error has since been corrected. With respect to the maximal number of characters allowed please take note that when the **Return** key is pressed now only one character is inserted.

Note

The codepage "utfwin" is not suitable to be used as an application codepage when character strings that contain <Carriage-Return> need to be processed. The reason for this is that the <Linefeed> mark (line break) in this codepage is converted to a <Carriage-Return><Linefeed> string. In order to avoid a double <Carriage-Return> mark, these are read over or rather ignored. Instead of using "utfwin" as codepage it is better to use "utf16" or "utf16l" since they do not carry out this special treatment.

16.2 Core

- » Gnats 10836: An error that in Windows sometimes led to the false marking of the content (*.content*) of a **poptext with pattern format** and *.style = edittext* has been corrected. The error was due to a false interpretation of *.startsel* or *.endsel* values.

17 Version A.05.02.e

17.1 Windows

- » Gnats 10763: The new help functions for error analysis in the IDM version A.05.01.g3 and A.05.01.h used the system function **OpenThread** which is not available under Windows NT 4.0. This has been changed so that the help functions can also be used when the IDM is run under Windows NT 4.0.
- » Gnats 10749: The background of the editing area within a **poptext** object is now displayed in the correct color and also reacts to changes made to the background color.
- » Gnats 10699: The **control** object (.mode = mode_client) now finds the type information of OLE controls that are implemented with the IDM (.mode = mode_server). In addition, now the character information of superclasses in the interfaces is also found.
- » Gnats 10698: Often the parameters of an OLE method call on a **control** object were not correctly treated when the methods were found in a type library. As a result the following error occurred: *“One or more of the arguments could not be coerced”*. Now, from the type library only is ascertained if a parameter is to be transferred as a reference or pointer (**VT_BYREF**). The remaining the parameter treatment corresponds to the IDM versions up to version A.05.02.c.
- » Gnats 10692: A **control** model with .mode = mode_server could not be instantiated when an OLE client wanted to connect. For this reason no connection could be made even after the server application was started. This problem has been corrected.
- » Gnats 10684: An IDM application is no longer blocked when setting a cursor that beforehand was positioned over a blocked application.

Note

The cursor is set each time an object is switched to visible (this is also true when starting a module).

- » Gnats 10676: If a window did not have a title bar (.titlebar) or borders (.borderwidth) and was also not able to be enlarged (.sizeable), then its width ended up being 2 pixels too wide. This error has been corrected.
- » Gnats 10670: Due to an unwanted change to A.05.02.c maximized windows observed the raster. The original behavior has once again been restored.
- » Gnats 10662: The automatic size calculation of the **statictext** and **spinbox** objects took place as if they were set to insensitive even though the objects were set to sensitive. This error has been corrected in version A.05.02.d. As a result these objects can end up being 2 pixels bigger in size than normal. This error was detected on a groupbox, whereby part of the frame at the bottom right side was not shown. This problem only occurred when setting objects to visible (setting .visible to true) and not with active “Visual Styles” (Windows XP, Vista).

- » Gnats 10648: An error that occurred when maximizing the size of a window with raster has now been corrected. The window status changed when these windows were maximized but their actual size remained unchanged.
- » Corrected: In the IDM version A.05.02.d the attributes *.real_width* and *.real_height* were miscalculated on the **statusbar** object. Now these are calculated correctly.
- » New (Gnats 10504): The image type (*type*) **DM_GFX_BMP** of the interface function **DM_PictureHandler()** can now be given a transparent mask (*trans_mask*) under Windows. Please note that for the mask must be a monochrome Windows bitmap (data type HBITMAP) that has the same size as the picture (image).

Note

A transparent mask is only possible for a **DM_GFX_BMP** image. For other image types the parameter *trans_mask* is ignored.

17.2 Motif

- » Gnats 10796: When **changing the size of the client area** of a window by using *.sizeraster = true* it is possible that objects in the client area are slightly cropped. This is due to the fact that the client area was increased by a multiple raster size. This problem has been corrected.
On account of this there has been a change with respect to **changing window sizes** with *.sizeraster = true*: Window sizes can be changed according to the step sizes within the raster when this is supported by the Window Manager.
- » Gnats 10793: As far as this can be influence by the IDM, it is guaranteed that modal windows (dialogbox, filerequester or messagebox) are displayed in the foreground of an IDM application, e.g. in front of the “normal” (eventually insensitive) windows.

Note

The IDM tries to keep the top modal window in front of the other IDM application windows. The IDM is however not able to guarantee a specific sequence of other windows and dialogboxes because the “stacking order” is determined by the Window Manager.

- » Gnats 10776: A **statusbar** now assumes the correct background color of the parent object.
- » Gnats 10772: By the initial depiction of an empty **treeview** no **scrollbars** were displayed. This was a problem with Motif that has now been able to be worked around.
- » Gnats 10764: The visibility of **scrollbars** depending on *.vwidth/.vheight* and *.hsb_visible/.vsb_visible* is now consistent with that of the IDM Windows version and as a result corresponds to the documented behavior. This means that *.vwidth* is only taken into account when *.hsb_visible* is set and *.vheight* is only considered when *.vsb_visible* is set. Prior to this under Motif both *.vwidth* and *.vheight* were considered as soon as one of the attributes, either *.hsb_visible* or *.vsb_visible*, was set.
- » Gnats 10758: A **tablefield** that is connected with *.xauto = 0* and *.yauto = 0* now changes its size correctly when the size of the parent object is adjusted.

- » Gnats 10707: When making a titled **dialogbox** visible sometimes a crash in **YiWindowCreate** occurred. This problem has been corrected.
- » Gnats 10686: When resizing a window sometimes the frame of the **groupbox** was not displayed. This problem has been corrected.
- » Gnats 10675: Under Motif 1.2 the height of a **poptext** changed when the *.visible* attribute was changed. As a result parts of previously visible text were cut off. The height calculation (when *.height = 0*) was corrected so that the height remains the same when switching over to *.visible*.
- » New (Gnats 10535): The *.borderraster* attribute is now supported under Motif.
- » New (Gnats 10527): The thickness of the focus frame of the entries in **tablefield** objects is no longer fixed to 3 pixels but rather is controlled by the X-resource value **highlightThickness** of the **XmTextWidget** class. The thickness of the focus frame can be overwritten through the defining of **itemHighlightThickness** of the **IdmTablefield** resource (e.g. by entering *“*IdmTablefield.itemHighlightThickness: 1”* in the X-resource file).
- » New (Gnats 10504): The IDM now supports the defining of a transparent mask (*trans_mask*) to an image by GFX handler. In addition, now Pixmap images can be used.
- » Corrected: A **progressbar** with *.height = 0* is now – analog to the Windows version – centered within the raster.
- » Corrected: The **scrollbar** now correctly takes into account the thickness of the shadow frame (2 pixels) in the arrow mode and when calculating the height when *.height = 0*. Now the thickness of the focus frame is determined via the desktop settings (X-resource).
- » Change: The transparent mask is now calculated for **tile** resources that are loaded from Motif image cache via **XmGetPixmap()**. The foreground and background colors are no longer only black and white, but are rather the same as the foreground and background colors of the screen.
- » Change: In order to ensure a consistent shadow width on an image object with respect to the desktop settings, it is now possible to give the attribute *.borderwidth* a negative value. In this case the object assumes the **shadowThickness** of the **XmPushButtonWidget** class as its frame thickness.

17.3 Core

- » New (Gnats 10723): For the numerical format of **format** resources there are new format modifiers **c** and **c0** to control the modification to an empty string when deleting digits.
Format modifiers **c** and **c0** for numerical formats:
 - c** Digit deleting mode:
The deletion of the very last digit in a string leads to an empty string and not to a string with the value *0*.
 - c0** 0-deletion-mode:
When a string represents the numerical value *0* after an entry or a deletion, then the string will be replaced with an empty string.

Both modifiers have no impact when they are combined with the z-format modifier (zero-mode).

Notes

- » Given a one-digit content string, this digit may not be situated rightmost within the formatted string. Rather it can be, e.g. with respect to technical digit entries, the first digit left of the comma.
- » In 0-deletion-mode a content string can change to an empty string when only one digit not equal to 0 exists and this digit is deleted.
Example: 7000.00 will be changed to an empty string when the 7 is deleted by using the **Del** or **Backspace** key.
- » Gnats 10722: The re-hanging of an exported object no longer triggers an erroneous warning for a non-existent attribute `.itemcount[.count]`.
- » Gnats 10714: The attribute `.defbutton` is now correctly written out by the IDM Editor. This is also true for **menuitem** objects.
- » Gnats 10690: It sometimes occurred that read access on inherited default values from associative arrays failed to work. Now these read accesses deliver the correct value.
- » Gnats 10682: False parameter data types, when positioned in the area of mandatory parameters, were not reported when calling up built-in functions.
- » Gnats 10666: At the trampoline generation, a shortened signature is used for records having multiple entries in order to bypass the length limit of the C preprocessor.
- » Gnats 10644: The decline in performance when destroying objects or freeing memory has been corrected.
- » Gnats 10628: Several problems with string conversions, for example occurring when changing the string format of a visible edittext, have been corrected. When calling a format routine, the display string adjusts the size of the display buffer used for converting between format code pages and internal code page. Now faulty conversions due to the display buffer being too small no longer occur. Double releases of the same memory blocks are now also avoided.
In addition, the problem of a defined pattern format of the display string sometimes not being correctly completed has also been corrected. This problem led to conversion errors being reported even though the conversion of the visible string was carried out without error. The reason for this was that memory areas behind the actual string end were converted.
Finally, conversion and length mismatch errors now generate different error messages.
- » Change: The interface function **DM_Initialize** can no longer be called using with the undocumented option `0x1000`. Now, **DMF_DontInitWSI** should be used.

Note

The incorrect use of **DMF_DontInitWSI** can lead to a system crash.

- » New (Gnats 10528): In order to allow access to the coordinates and sizes of the tablefield cells, the function **DM_GetToolkitDataEx** was added to the C interface (see chapter “New Interface Function **DM_GetToolkitDataEx()**”).

- » New: Trace codes (.tracing[]-indices) can now be requested on the **setup** object via *.itemcount* and *:index()*.

17.4 Compatibility

As a result of having to change internal structures (which is sometimes necessary), C, C++ and COBOL source texts must also be newly compiled by the IDM applications. In this case, COBOL copy lines must also be newly generated.

17.5 New Functions for Error Analysis

The new functions for error analysis (see “Version A.05.01.g3”) are also available beginning in version A.05.02.e.

Compared to version A.05.01.g3 the following changes have been made

- » New: The number and size of memory allocated by values (value memory of the rule interpreter) has been added to the usage section of the dumpstate (see chapter “Dumpstate Output”).
- » Change: Under UNIX a complete dumpstate is no longer returned when an INT signal is caught by the exception catcher. Only the stack, errors, events and process sections are written out.

17.6 New Interface Function DM_GetToolkitDataEx()

This function is an enhanced form of DM_GetToolkitData and allows for the transfer of additional values and structures via the *data* and *options* parameters.

```
FPTR DML_default DM_EXPORT DM_GetToolkitDataEx
(
    DM_ID objectID,
    DM_Attribute attr,
    FPTR data,
    DM_Options options
)
```

Parameters

-> DM_ID objectID

This parameter is the object identifier whose window system-specific data shall be requested.

-> DM_Attribute attr

This parameter names the attribute to be requested.

<-> FPTR data

For a compatible use of DM_GetToolkitData, use data = NULL.

-> DM_Options options

Currently unused, should be set to 0.

Return Value

Depending on the type of the requested values, this function delivers the corresponding values converted to FPTR.

The following data structures and defines are available for the communication

```
typedef struct {
    DM_Int4    x;
    DM_Int4    y;
    DM_UInt4   width;
    DM_UInt4   height;
} DM_Rectangle;

#define DM_TKAM_index      (1 << 0)
#define DM_TKAM_data      (1 << 1)
#define DM_TKAM_handle    (1 << 2)
#define DM_TKAM_widget    (1 << 3)
#define DM_TKAM_rectangle (1 << 4)

typedef struct {
    DM_UInt4   argmask;
    DM_Value   index;
    DM_Value   data;
#if defined(WSIWIN) || defined(WIN32)
    HANDLE     handle;
#endif
#if defined(MOTIF)
#   ifdef _XtIntrinsic_h
        Widget widget;
#   else
        FPTR   widget;
#   endif
#endif
    DM_Rectangle rectangle;
} DM_ToolkitDataArgs;
```

The **DM_ToolkitDataArgs** structure serves as a way of communicating with the **DM_GetToolkitDataEx** function. It allows for the transfer and return of various value types. **DM_TKAM_*** defines combined by logical OR within the *argmask* field of the structure serve as a tag for a set value. The **DM_Rectangle** structure is a help structure for the coordinates and size of a rectangular “object form”.

The following Motif and Windows functions are available via **DM_GetToolkitDataEx**

» **Parameter Value**

ObjectID	Attribute (attr)	Data
(DM_ID) 0	AT_ObjectID	Pointer to DM_ToolkitDataArgs structure with set widget field and argmask = DM_TKAM_widget.

Function

Determine the corresponding Object-ID for a widget. When successful, the data pointer is returned and in the DM_ToolkitDataArgs structure the ID is saved in the data substructure.

» **Parameter Value**

ObjectID	Attribute (attr)	Data
ID of a tablefield	AT_CellRect	Pointer to DM_ToolkitDataArgs structure with a set index field and argmask = DM_TKAM_index

Function

This determines the coordinates and the size of a cell (in pixel) for a tablefield. These are stored in the *rectangle* field of the transferred **DM_ToolkitDataArgs** structure. The cell refers to the rectangular area within a tablefield in which shadow, focus and activation frames and text are marked. The lines drawn between cells do not belong to the cells themselves.

The delivered coordinates are relative to the upper left-hand corner of the tablefield.

When the position and size of a cell can be determined, then **DM_GetToolkitDataEx** returns a pointer to the indicated **DM_ToolkitDataArgs** structure. In all other cases it returns NULL. Position and size can only be determined when the tablefield, lines and columns of the cell are switched to visible and the cell is either completely or partially visible in the tablefield. A concrete, absolute invisibility (e.g. if the window is situated outside of the viewable area of the screen or is otherwise covered) can not be ruled out and is dependent on the windows system. This may be true even if values for position and size are returned.

18 Version A.05.02.d2

18.1 Windows

- » Gnats 10670: Due to an unwanted change to A.05.02.c maximized windows observed the raster. This has now been corrected and the original behavior of the windows has been restored.
- » Gnats 10662: The automatic size calculation of the **statictext** and **spinbox** objects took place as if they were set to insensitive even though the objects were set to sensitive. This error has been corrected in version A.05.02.d. As a result these objects can end up being 2 pixels bigger in size than normal. This error was detected on a groupbox, whereby part of the frame at the bottom right side was not shown. This problem only occurred when setting objects to visible (setting *.visible* to *true*) and not with active “Visual Styles” (Windows XP, Vista).
- » Gnats 10648: An error that occurred when maximizing the size of a window with raster has now been corrected. The window status changed when these windows were maximized but their actual size remained unchanged.
- » Corrected: In the IDM version A.05.02.d the attributes *.real_width* and *.real_height* were miscalculated on the **statusbar** object. Now these are calculated correctly.

18.2 Motif

- » Gnats 10686: When resizing a window sometimes the frame of the **groupbox** was not displayed. This problem has been corrected.
- » Gnats 10675: Under Motif 1.2 the height of a **poptext** changed when the *.visible* attribute was changed. As a result parts of previously visible text were cut off. The height calculation (when *.height = 0*) was corrected so that the height remains the same when switching over to *.visible*.

18.3 Core

- » Gnats 10682: False parameter data types, when positioned in the area of mandatory parameters, were not reported when calling up builtin functions.
- » Gnats 10644: The decline in performance when destroying objects or freeing memory has been corrected.

19 Version A.05.02.d

19.1 Important Changes

- » The distribution of the IDM for UNIX platforms has been changed. In the future, the distribution will take place in the form of tar/gzip archives. A serial number is necessary to install the IDM (see chapter “Installation Guidelines Unix”).
- » The XML IDM interface is now available for UNIX platforms beginning with this version. Here, the **libxml2** and **libxslt** libraries are necessary.
- » The **document** object (XML Document) now contains the new attributes `version[enum]` and `real_version[enum]`. With `.version` the XML toolkit version can be specified under MS Windows. With `.real_version[enum]` the actual loaded version of the XML toolkit can be accessed. When querying `real_version` the system will try to load the toolkit that is stated in `.version`.
- » The USW interface is now available under Microsoft Windows. The User Supplied Widgets (USW) can be integrated in the IDM via this interface. As a result, a few changes were made to the USW interface.
- » In connection with the USW interface two new attributes were implemented for all objects: `external` and `external[I]`. `.external` is a Boolean type and indicates if the object in question is a USW. On the other hand, `.external[I]` delivers the registered USW object class at position I.
- » As a result of the implementation of the USW interface under Windows, two additional elements were added to the **DM_Value** structure, which caused the static initialization to be changed as seen below:
Old: `DM_Value valVoid = { FALSE, FALSE, DT_void };`
New: `DM_Value valVoid = { FALSE, FALSE, FALSE, 0, DT_void };`
In addition, signatures in the C++ interface were changed.

19.2 Windows

- » Gnats 10587: Changing the size of MDI child windows works correctly again. When child windows are maximized in size the title and menu bars are no longer covered by the client area.
- » Gnats 10520: The interactive docking of toolbar objects when they are dragged at their title bar now also works correctly under Windows Vista.
- » Gnats 10497: The **window** object falsely recognized the child object of a toolbar object as a focused object, when at the time the child object was made visible it did not possess a focused object. This no longer occurs.
- » Gnats 10489: When an **image** object was deleted while at the same time the left mouse button was pressed (e.g. in a *focus* rule), then no more *select* events were created for other **image** objects. This problem has been corrected.

- » Gnats 10470: A **scrollbar** object is no longer implicitly disabled when both arrows are set to be disabled in this area in question. Now the **scrollbar** object once again accepts the focus. Furthermore, the keyboard navigation no longer remains stuck on the **scrollbar** object. (This problem only arose under the Windows 2000/NT version of the IDM.)
- » Gnats 10451: When a **messagebox** is closed, then the object that had the focus before the messagebox was opened in the window owning the messagebox receives the focus again. The window is no longer set to insensitive when opening the messagebox which prevented the focus from being returned.
- » Gnats 10440: When triggering the mnemonics of an insensitive **statictext**, the system menu is no longer erroneously opened.
- » Gnats 10413: The forwarding of an explicit set-up of an **edittext** in a **tablefield** is now carried out correctly. Now it is avoided that an **edittext** object in a **tablefield** is made visible after a parent object of the same tablefield is switched to invisible. This could happen when transferring parent objects. In the trace file this problem is documented by the following message: "[IM] Cannot create <object>, because it was destroyed".
- » Gnats 10401: The changing of the *sensitive* attribute of a **window** object to a visible status is now communicated correctly to the **toolbars** and **statusbar** of the window.
- » Gnats 10373: The icons of a window are now correctly set so that the dialog for switching between applications (key combination **Alt + Tab**) under Windows XP no longer shows the standard icons in the window classes ("IDM Eye") and under Windows Vista no longer shows an enlargement of the small icon (from the title bar).
- » Gnats 10363: Starting the IDM editor via the button in the start menu (created during installation) sometimes caused an error to occur when the installation path of the IDM contained blanks. This error message no longer occurs. Now, when installing the IDM, the paths for the start menu entries are set correctly even when the path contains blanks.
- » Gnats 10362: The problem that resulted from a different error correction, namely that context menus (pop-up menus) without the *on open* rule did not open as they should have, has been corrected.
- » Gnats 10361: Since each version of the MSXML control object is different with respect to the supported standards, the **document** object now has a new attribute *.version* for specifying the necessary version of the MSXML control. For example, the MSXML control (starting in version 4) supports the validation of an XML Document against an XSD (XML Schema Definition). The setting of the attribute *.version* does not necessarily lead to the loading of the required MSXML control version. When querying the attribute the set value is returned. To test dynamically if the desired MSXML control can be loaded, the other new attribute *.real_version* must be queried.
- » Gnats 10356: A memory leak when accessing the *.xml/* attribute of a **document** or cursor object has been corrected. This was caused when text, that was stored in a temporary buffer, was not released.

- » Gnats 10347: Crashes sometimes occurred (mostly an assfail in an object/slot) when an IDM window was opened or manipulated on a Citrix system with multiple monitors. The reason for this is that Citrix intercepts the Windows messages WM_GETMINMAXINFO. This brought the IDM into an endless loop. The crashes are now avoided but the problem remains in principle. The ISA Dialog Manager needs these WM_GETMINMAXINFO messages in order to display the window in its correct size. When these are intercepted the correct display of these windows can not be guaranteed. Windows without raster defined and fastened at top/left cause the least problems.
- » Gnats 10338: Now an **edittext** object is correctly assigned to a **tablefield** when a hierarchical model in a visible state is dynamically instantiated. This problem occurred when an **edittext** object was created after a **tablefield** object and the setting of the attribute **.edittext** on the **tablefield** was not relayed to the window system.
- » New (Gnats 10136): The **image** object can automatically create an image that is displayed in an insensitive state. If the new option **.options[opt_sim_insensitive]** is set to **true** for an image without a picture for the insensitive presentation (**.picture[tile_insensitive] = null**), then a faded version of the **image** object is shown. Note: This function is not available for Windows versions before Windows 2000 or Windows 98.
- » Gnats 10104: The OLE control (**.mode = mode_client**) follows the optimal OLEMISC_SETCLIENTSITEFIRST. Through this the depiction problem of a Java Bean OLE object within the control object has been solved.
- » Gnats 10032: The set minimal height of a window was relatively high when many **menubox** objects existed. This was due to the fact that the calculation of the minimal height of the window was set to the minimal width, whereby most every **menubox** object demanded its own line. In addition to this, the window was not able to be dragged to the maximal height when it possessed the minimal width. This problem came about because in order to calculate the maximal height of a window, the maximal width was also used in the calculation, whereby the menu line was usually a single line. The same was true when a window possessed multiple toolbar objects, whereby the problem arose not only in connection with the calculation of the height but also with the width of the window. The calculation of minimal and maximal sizes happens dynamically so that all sizes can be attained. In the calculation, the window is brought to the desired size (when possible) and then from this state the minimal and maximal height is calculated.
With respect to docked toolbar objects it is possible that the minimal and maximal height and width cannot be interactively attained. This problem arose when a further adjustment to the width or the height was made which caused the toolbar objects to become completely rearranged that in turn led to a violation of the minimal and maximal height and width. In this case, it is enough to just change the height and width (please note the transposition).
- » New (Gnats 9534): By using the start option **-IDMno_yi_monitoring** the activation of monitoring functions, which are installed with **YiRegisterUserEventMonitor**, can be suppressed or stopped. This option can be of help when looking for errors that are suspected of being caused by monitoring functions. In place of the start option also the environment variable **IDM_NO_YI_MONITORING** can be specified or the option **.options[opt_yi_monitoring]** of the **setup** object can

be set to false. If monitoring functions were switched off via start options or environment variables, they cannot be turned on again by setting `.options[opt_yi_monitoring] = true`.

- » Corrected: Inactive **notepage** objects were in some cases not adjusted to the size of the surrounding **notebook** objects. This problem arose in **notebook** objects that were tied to their parents using `.xauto` or `.yauto = 0`.
- » Corrected: A docked **toolbar** object with an active raster reacted to changes made to `.xleft` or `.ytop` when a raster was active. In addition, a docked **toolbar** object displaced by the raster offset.
- » New: The OLE interface now analyzes type information. As a result, OLE controls can be called with the data types expected by them and the IDM interface is no longer depending on them to convert the data (as usually required).

19.3 Motif

- » Gnats 10536: When a window only contains a single focusable object (no grouping object) and a grouping object with navigable children, then an endless loop occurred when the only focusable object was made invisible. This problem has been corrected.
- » Change (Gnats 10466): Grouping objects on which only one virtual size (width and height) is set now behave analog to the Windows version. This means the size that is not set results dynamically from the size of the grouping object and is no longer constant with regard to the original size.
- » Gnats 10465: When setting `.yorigin` on **groupboxes** or **windows**, whose size is virtual, then the value for the vertical scrollbar is now correctly set.
- » Gnats 10408: Setting the size of windows with menus no longer leads to constant *resize* events.
- » New: The **poptext** object under Motif 2.x supports the attributes `.xmargin` and `.ymargin` in influencing the horizontal and vertical distance between the text and the outside frames. Since the **poptext** under Motif is a compound object, the attributes influence the distance between the text and the textfield frames and therefore only have an indirect affect on the distance between the textfield and the outside frames. The `.ymargin` attribute influences the height of the pop text when no height is specified (`.height = 0` and `.yauto <> 0`). Too high of values can lead to the text not being visible by predetermined heights. The `.ymargin` attribute counteracts this when a width is specified which in turn adjusts the displayed text area and distance between the frames. For negative attribute values, the standard values of the system are used.

Noteworthy under Motif

- » The use of `.xmargin` is not recommended because Motif displays the cursor within the frame area when it is left or right of the text.
- » The textfield frame under Motif is normally only visible for the **styles edittext** and **listbox**.
- » The value -2 for `.xmargin/.ymargin` is a compatibility mode to the precedent version which had a slightly different height calculation and text positioning for `.height = 0`.
- » The default value is 5 and therefore different from the Windows platforms.

- » New: On the Motif platforms beginning with version 2.1 the tile resources now also loads images via the **XmGetPixmap()** function. Through this predefined pixmaps and the loading of images in XPM format are now possible. Under OpenMotif 2.3 the Motif library also supports the image formats JPEG and PNG.

Note

The `\(..|\..\\)` pattern only functions under the LINUX platforms for which an extended pattern was set with `*.\(gif|png|...)`.

- » Change: The calculation of the width of a poptext, whereby `.width = 0`, was corrected so that the longest text in the poptext is now completely visible.
- » Change: The attributes `.xmargin` and `.ymargin` of the edittext object are only set by the IDM when values are `>=0`. Otherwise the system standard values are used.

19.4 Core

- » Gnats 10615: The optional parameter (*boolean destroy input*) of the `stop()` function is now treated correctly again.
- » Gnats 10563: Default values of static variables are once again written out correctly (e.g. from the editor).
- » Gnats 10592: The blocking of input when setting `.startsel` or `.endsel` of an edittext containing a number format is now avoided.
- » Gnats 10549: An error that occurred when saving or reverting the temporary memory for a *string* parameter has been corrected. This error happened only under certain circumstances, for example, when exactly the limit of a page (64 KB) was reached.
- » Change (Gnats 10458): WSI error messages that occur when setting invalid attribute values are now only emitted in the development version because these are regarded by the user as disturbing and not seen as an error in the dialog script.
- » Gnats 10522: An inconsistency between the allocation and transferring of strings (without a reference to a text resource) in comparison to the use of local variables has been corrected. These are now transferred as pure strings.
- » Gnats 10494: The C interface functions **DM_GetValueIndex()**, **DM_SetValueIndex()** and **DM_ResetValueIndex()** now carry out a code page conversion of the index parameters where necessary.
- » Gnats 10487: In the Windows 2000/NT version no more negative results occur when using the function **random()**.
- » Gnats 10462: When saving dialogs and modules, blank spaces in local event rules and inheritable objects without an identifier are now avoided.
- » Gnats 10452: Regarding the administration of identifiers within modules, the previous limit of 256KB for the identifier space has been raised. Now it is possible to have at least 65,500 identifiers with random lengths (max. 31 characters).

- » Gnats 10420: When a parameter return value (e.g. from a method call) cannot be saved because the term is not an LVALUE, a clear error message is sent.
- » Gnats 10395: Annotations on static attributes with indexed strings are now correctly written out and stored.
- » Gnats 10382: The **create** method now correctly identifies object IDs that may have become available after the error message *“no more free object ID for module”* occurred.
- » Gnats 9466: The trampoline generation now better supports the re-use of common parameter records. The generated header files can now be easily included as long as there is no redefinition of records having a different structure. As was before, please note that module and function identifiers used during the generation must be unique.

19.5 Editor

- » Gnats 10419: The copying and pasting of child objects in the object browser as well as changes in the code window no longer lead to a syntax error *“**** Error in line 1: syntax error at or near >child<”*.
- » Gnats 10233: The execution of the **delete** method of an object no longer leads to the attribute *.visible* being set to *true* for the window to which the object belongs.
- » Gnats 9731, Gnats 9545 Gnats 9060: In the *“object”* section within the properties area it is now possible to assign and make multiple changes in the attribute groups at the same time. By user actions, which lead to an actualization of the object area, the user is specifically asked if these changes should be made or not. This is not true for interactions within the design area.
- » Corrected: A defaults file is now also found under MS Windos when the PATH environment variable is analyzed at start-up of the editor.

19.6 Java

- » Gnats 10346: When a dialogbox (**window** with *.dialogbox = true*) is open, dialog events are now created.
- » Gnats 10345: Visible objects can once again dynamically be made invisible.
- » Gnats 10343: Examples for the Java interface are now also installed during installation.

19.7 Installation Guidelines UNIX

The following installation guidelines can also be found in the README.txt file.

The Dialog Manager comes packed in archives which where themselves crypted or contain crypted files. Your licensed options will automatically install and decrypt the necessary files when using the serial number with the delivered installation script.

Archives required for installation:

<architecture>.tar.gz	for each supported architecture
doc-de.tar.gz.crypt	german documentation
doc-en.tar.gz.crypt	english documentation
IDM.EXAMPLES.tar.gz.crypt	C and IDM dialog examples
COBDEMO.tar.gz.crypt	COBOL examples

For a successful installation the utilities tar and gzip are required.

To install IDM follow steps 1 to 3:

1. Extracting the architecture dependent files:

Extract the architecture archive under a non-superuser (root) account with tar&gzip utility. You may create an empty directory first, for example with the commands

```
mkdir new
cd new
gzip -d < <path to architecture.tar.gz> | tar xf -
```

Copy the *.crypt files in the same directory. These files have to match the extracted architecture directory.

2. Installing the ISA DIALOG MANAGER software:

Change to the directory for the binary machine type, e.g.

```
cd sparc.solaris-100.motif-21
```

In that directory you will find the shell script **install.sh** which provides easy installation.

In case you want to install the DIALOG MANAGER software under an account different from root/bin, use the option **-uid <username>** or **-gid <groupname>**.

To install the Dialog Manager in the system, type **as root sh ./install.sh**.

In this case you do not need to modify the \$PATH variable and you do not have to give any special include and link flags for compilation.

The script request the serial number. For an installation without interaction use the **-serial <serial-number>** option.

In order to keep the system tree clean from additional software or if you want to install more than one DIALOG MANAGER version at the same time, type **as root**:

```
sh ./install.sh -homedir "/directory_you_want_for_idm"
```

To skip the installation of the documentation or the examples use the **-nodoc** or **-nosamples** option.

3. Setting the environment for using DIALOG MANAGER:

For csh Users

Put in your **~/.cshrc** file the following lines:

```
setenv IDM_HOMEDIR "/directory_you_installed_idm"
set path=($IDM_HOMEDIR/bin $path)
source $IDM_HOMEDIR/lib/IDM/idmenv
```

For sh and ksh Users

Put in your **\$HOME/.profile** file the following lines:

```
IDM_HOMEDIR="/directory_you_installed_idm"
export IDM_HOMEDIR
PATH="$IDM_HOMEDIR/bin:$PATH"
export PATH
. $IDM_HOMEDIR/lib/IDM/idmenv.sh
```

It is necessary to set IDM_HOMEDIR to the home directory of Dialog Manager, which normally is done in the **.profile** or **.cshrc** file (see step 3).

Online documentation is available in english or german.

To read the online documentation after installation open the start page **doc-de/html/IDM.htm** or **doc-en/html/IDM.htm** in a HTML browser or press the help key **F1** inside the editor "idmed".

If you have any problems with the installation, please call +49-711-22769-24 or send e-mail to idsupport@isa.de

Explanation of the Installed Files

? stands for the installation directory

?/bin/idd	Dialog Manager Dialog Documentator for Motif
?/bin/idm	Dialog Manager Simulator for Motif
?/bin/idmdbg	Dialog Manager Debugger for Motif
?/bin/idmhelp	Script to start the reader of the IDM documentation
?/bin/idmnet	Dialog Manager Simulator for Network application side
Motif	
?/bin/idmndx	Dialog Manager Simulator for Network display side Motif
?/bin/idmed	Dialog Manager Editor
?/include/IDMco*.cob	Include file for COBOL-applications
?/include/IDMuser.h	Include file for C-applications
?/include/IDMmcard.h	Include file for the mcard option
?/lib/libIDMinit.a	Dialog Manager base library (not shared)
?/lib/libIDM.*	Dialog Manager development library for Motif
?/lib/libIDMrt.*	Dialog Manager runtime library for Motif
?/lib/libIDMnet.*	Dialog Manager network library application side
?/lib/libIDMndx.*	Dialog Manager network library display side
?/lib/libIDMusw.*	User supplied widget support library (USW)
?/lib/*cob*.o	COBOL modules
?/lib/IDM/*	Directory containing configuration and dialog files
?/lib/IDM/Version	Version file
?/lib/IDM/MakeDefs	Include file for the Make utility
?/lib/IDM/idmed	Editor specific files
?/lib/IDM/startup.c	Startup module containing main routine (C source)
?/lib/IDM/*startup.o	Startup object modules (normal/net/unicode) containing
main	
?/lib/IDM/idmenv.sh	Environment setup file for sh and ksh

?/lib/IDM/idmenv	Environment setup file for csh and tcsh
?/lib/IDM/gencobfx	Program to build COBOL functioncall-module
?/lib/IDM/idd	Dialog Documentator specific files
?/lib/IDM/idd/idd.sty	LaTeX style file for the Dialog Documentator
?/lib/IDM/idmdbg	Directory for specific files of the Dialog Debugger
?/lib/IDM/doc/*	Directory containing documentation
?/lib/IDM/edextra	Files of the edextra option
?/lib/IDM/usw	USW specific files
?/lib/IDM/main.o	main card object file
?/lib/IDM/mcard.o	mcard object file
?/IDM.EXAMPLES	Directory with dialog and C examples
?/COBDEMO	Directory containing cobol samples

20 Version A.05.02.c6

20.1 Core

- » Gnats 10628: Several problems with string conversions, for example occurring when changing the string format of a visible edittext, have been corrected. When calling a format routine, the display string adjusts the size of the display buffer used for converting between format code pages and internal code page. Now faulty conversions due to the display buffer being too small no longer occur. Double releases of the same memory blocks are now also avoided.
In addition, the problem of a defined pattern format of the display string sometimes not being correctly completed has also been corrected. This problem led to conversion errors being reported even though the conversion of the visible string was carried out without error. The reason for this was that memory areas behind the actual string end were converted.
Finally, conversion and length mismatch errors now generate different error messages.
- » Gnats 10563: Default values of static variables are once again written out correctly (e.g. from the editor).

21 Version A.05.02.c5

21.1 VMS

Several problems in dealing with the path information in user-specific configuration data of the editor (**idmed.prf**) have been corrected.

22 Version A.05.02.c4

22.1 Windows

- » Gnats 10451: When a **messagebox** is closed, then the object that had the focus before the messagebox was opened in the window owning the messagebox receives the focus again. The window is no longer set to insensitive when opening the messagebox which prevented the focus from being returned.

22.2 Motif

- » Gnats 10465: When setting `.yorigin` on **groupboxes** or **windows** with virtual sizes the value for the vertical scrollbar is now set correctly.

23 Version A.05.02.c3

23.1 Windows

- » Gnats 10440: When triggering the mnemonics of an insensitive **statictext**, the system menu is no longer erroneously opened.
- » Gnats 10373: The icons of a window are now correctly set so that the dialog for switching between applications (key combination **Alt + Tab**) under Windows XP no longer shows the standard icons in the window classes (“IDM Eye”) and under Windows Vista no longer shows an enlargement of the small icon (from the title bar).

23.2 Motif

- » Gnats 10408: Setting the size of windows with menus no longer leads to constant *resize* events.

23.3 Core

- » Gnats 10395: Annotations on static attributes with indexed strings are now correctly written out and stored.
- » Gnats 10382: The **create** method now correctly identifies object IDs that may have become available after the error message “*no more free object ID for module*” occurred.

23.4 Editor

- » Gnats 10233: The execution of the **delete** method of an object no longer leads to the attribute *.visible* being set to *true* for the window to which the object belongs.

24 Version A.05.02.c2

24.1 Windows

- » Gnats 10363: Starting the IDM editor via the button in the start menu (created during installation) sometimes caused an error to occur when the installation path of the IDM contained blanks. This error message no longer occurs. Now, when installing the IDM, the paths for the start menu entries are set correctly even when the path contains blanks.
- » Gnats 10362: The problem that resulted from a different error correction, namely that context menus (pop-up menus) without the *on open* rule did not open as they should have, has been corrected.

25 Version A.05.02.c

25.1 Important Changes

- » The ISA Dialog Manager now supports Windows XP x64 platform for Intel and AMD x86-64 processors. In connection with this, support for Microsoft Visual Studio 2008 was also added.

Note

This is not to be confused with the IA64 platform (Intel Itanium) for which no Dialog Manager Version for MS Windows exists.

- » Resulting from an error correction on the **poptext** object (combobox) the attribute *.activeitem* within the static area can no longer be set to a value larger than the number that is defined in the attribute element *.text[[]]*.
It is also no longer possible to set *.activeitem := 1; before* the definitions of *.text[[]]*. This leads to error messages when loading the dialogs/modules because here the sequence of attributes that are to be imported are observed. When *.activeitem := 1;* is set **after** the definitions of *.text[[]]* attribute then this problem is eliminated.

Note

The reestablished behavior now equates the behavior of the other list objects.

25.2 Windows

- » Gnats 10342: As a result of some changes made so that the **image** object would only be displayed if the object status was changed, the focus frames were not correctly refreshed. This problem has been corrected.
- » Gnats 10336: While deactivating a window an active OLE control, which was attached via a **control** object, was not properly deactivated. As a result sometimes the OLE control did not take the focus when reactivated.
- » Gnats 10335: In the process of making an empty **treeview** object visible, *.activeitem* is no longer incorrectly set to 1.
- » Gnats 10331: When setting an icon to a window object, which was not (or is no longer) existent, an Assfail could occur. This is no longer the case.
- » Gnats 10328: By non-activated "Visual Styles" (only XP version) or rather in the W2k version, the slide control is no longer switched to insensitive when scrolling page by page through grouped objects. This sometimes occurred when an arrow was insensitive before the scrolling took place.
- » Gnats 10325: An Assfail on a tablefield object has been eliminated. The Assfail was observed when a line was deleted by a click in the last line of a tablefield object in an on modified rule.

- » Gnats 10307: Arguments of OLE events are no longer incorrectly set to NULL. This mistake sometimes occurred when an OLE event possessed a **subcontrol** object as argument and the garbage collection took place before the processing of the event.
- » Gnats 10129: When dynamic changes are made to a font the line height and the slide control are also changed accordingly in the **listbox** and on the **poptext** object.
- » Gnats 9952 and 8605: In Microsoft Windows the **image** object now has a new menubox style (attribute `.style`, value `menubox`).

See Also

Attribute style

Object image

- » Gnats 9680: A memory leak that occurred due to this error correction has been eliminated.
- » Changed: The handling of the attribute `.spacing` (**image** object) has been changed, so that now a consistent use under Motif is possible.

See Also

Attribute spacing in the “Attribute Reference”

- » Eliminated: The `scale` option of the **tile** resource was not always correctly observed by the **image** object. Problems occurred when one size of the **tile** resource already was set “correctly” so that the **tile** resource only had to be adjusted in one direction.
- » Eliminated: Set colors in the **progressbar** object are now no longer shown in the progressbar area because this can result in an unpleasant appearance of active “Visual Styles”. Otherwise, the set color is still observed.

Note for Active “Visual Styles” (Dialog Manager XP Platform Under MS Windows Versions Beginning With XP)

The coloring of active “Visual Styles” is determined to the greatest extent by this. A set color in the Dialog Manager will not always be shown in this case. To avoid unpleasant color combinations it is better to avoid color settings.

- » Eliminated: Regarding the keyboard navigation within grouped objects, the focus object is now scrolled into the viewable area even when a size raster is set to the father object.

25.3 Motif

- » Gnats 10341: The `Enter` key on a keypad is handled by the Dialog Manager in an edittext object just like the `Return` key:
 - » single line: the focus is set to the next object
 - » multi line: a new line is added (new line)

Caution

Since the conversion of a new line action is carried out via translations and Motif virtual `KP_Enter` is mapped on `osfActivate`, the `Enter` function in multi line edittexts only works when the binding of `Enter` to `osfActivate` is broken or when a special translation is set up.

Both of these possibilities should be avoided due to the potential side effects that can arise. An alternative solution is to change the keyboard mapping of the X-Server.

- » Gnats 6858: The resize line on the **tablefield** object is displayed in the correct place during a user action.
- » Changed: Negative values for the *.spacing* attribute on the **image** objects are now supported.

25.4 Core

- » Gnats 10349: It is again possible to indicate valid areas beyond module limits.
- » Gnats 10344: A memory error that occurred in the **splitbox** object has been eliminated.
- » Gnats 10329: Call back functions are called again for each set event.
- » Gnats 10327: Problems with the internal management of objects (or rather IDs in general) have been eliminated. Now when objects are destroyed occupied slots become freed up again.
- » Gnats 10324: A memory leak in the rule interpreter has been eliminated.
- » Gnats 10321: User defined attributes and functions within an application object are now again correctly written out in the IDM Editor.
- » Gnats 10317: Several problems that arose when loading modularized dialogs in the Dialog Manager Editor have been eliminated. In the case above an Assfail occurred during the loading of a module.
- » Gnats 10314: A string parameter (output) in C occupied with the value NULL is now again delivered to the IDM as an empty string.
- » Gnats 10306: Output parameters that need information such as index/object or rather value/attribute in order to be set, are now handled correctly.
- » Gnats 10301: The possible indentation space within a trace file has been greatly increased.
- » Gnats 9272: A possible Assfail in connection with a **tablefield** object is avoided in the Dialog Manager core through prior consistency checks.
- » Eliminated: The generated C++ file of the C++ interface contained multiple carriage return symbols. This problem has been eliminated.

25.5 Editor

- » New: The reference string of font resources can be changed in the editor.
- » New: The editing of the attributes *.alignment*, *.spacing* and *.tilestyle* in the object/attributes area is now possible.
- » Gnats 10342: The color fields in the color toolbox are now displayed correctly under MS Windows.
- » Gnats 10065: An error message now only occurs when the user explicitly sets *.activeitem* to 0.
- » Gnats 9883: The maximum number of colors that can be displayed in the color toolbox has been increased to 32.

25.6 Debugger

- » New: The Dialog Manager debugger now supports an instant display of all the local variables and arguments of the actual executed rules in the main view. In addition, static local variables are supported.

It is now possible to switch between the existing display of expressions and the new display of local variables via the **Tab** key.

In addition, it is also possible to change the value shown.

- » Gnats 10330: The value of static variables within the rules can now be queried and changed accordingly.
- » Gnats 10332: CR+LF are now correctly changed to LF when text data is read in (i.e. for the debugger status, debugger status data...). As a result, this resolved an error that appeared when a saved debugger status was loaded.

25.7 COBOL

- » Eliminated: The COBOL copy route can now be bypassed with the increased number of arguments to 16.

26 Version A.05.02.b4

26.1 Windows

- » Gnats 10331: When an icon was set to a window object, which did not exist or no longer existed, an Assfail was possible. This problem has now been eliminated.
- » Gnats 10328: By non-activated "Visual Styles" (only XP version) or rather in the W2k version, the slide control is no longer switched to insensitive when scrolling page by page through grouped objects. This sometimes occurred when an arrow was insensitive before the scrolling took place.
- » Gnats 10325: An Assfail on a tablefield object has been eliminated. The Assfail was observed when a line was deleted by a click in the last line of a tablefield object in an *on modified* rule.

26.2 Core

- » Gnats 10327: Problems with the internal management of objects (or rather IDs in general) have been eliminated. Now when objects are destroyed occupied slots become freed up again.

27 Version A.05.02.b3

27.1 Core

- » Gnats 10324: A memory leak in the rule interpreter has been eliminated.

28 Version A.05.02.b2

28.1 Core

- » Gnats 10321: User defined attributes and functions within an application object are now again correctly written out in the IDM Editor.
- » Gnats 10314: A string parameter (output) in C occupied with the value NULL is now again delivered to the IDM as an empty string.
- » Gnats 10306: Output parameters that need information such as index/object or rather value/attribute in order to be set, are now handled correctly.
- » Gnats 10301: The possible indentation space within a trace file has been greatly increased.

29 Version A.05.02.b

29.1 Important Changes

- » Beginning with this version the ISA Dialog Manager no longer supports Microsoft Windows 95. If the Dialog Manager still shall be used under Windows 95, the shell extensions of the Internet Explorer 4.01 (or newer) need to be installed.

29.2 Windows

- » Gnats 10292: The slide controls of a grouped object now again are sensitive when the grouped object is set to sensitive. Problems only did arise when switching to sensitive with active “Visual Styles”.

Note

If the ISA Dialog Manager is executed on the Windows 2000 platform under Microsoft Windows Vista, then the slide control of grouped objects can be displayed in part in the Windows 2000 appearance and in the new appearance. This problem occurred in some places in which a child of a grouped object overlapped with a slide control.

In order to bypass this problem the use of “Visual Styles” needs to be explicitly deactivated in the application properties (built with IDM for Windows 2000):

- » right mouse click on the application, select “*Properties*”
- » switch to tab “*Compatibility*”
- » set a check mark next to “*Deactivate visual designs*”
- » close the dialog by clicking “*OK*”
- » Gnats 10290: When re-arranging viewable objects in the child vector of the father object, it no longer comes about that the child objects of the transferred objects become invisible. Furthermore, no entries in tracefile (“*[IM] cannot create <object>, because it was destroyed*”) appear.
- » Gnats 10240: The treeview object no longer triggers additional *on .activeitem changed* events after an user action that raises an *activate* event.
- » Gnats 10219: When an application opened a modal window directly after closing a messagebox or a filerequester object, this could have lead to two windows of an application being simultaneously activated and blinking in turn. This problem has been corrected.
- » Gnats 10157: The statusbar object now again is activated when a change to the font has been made.
- » Gnats 10149: When changing the content of a poptext object (*.style <> listbox*) under “Visual Styles” while the list was open (upwards), a part of the list remained on the screen. This problem was observed when less than 30 of the 30+ defined entries were displayed.

- » Gnats 10130: Sometimes a focus frame was not displayed when switching out of one OLE object in to another via a mouse click (i.e. pushbutton). In addition to the error correction in relation to the UI states that were implemented with Windows XP (see Gnats 10074) it has now been assured that the UI states in an OLE environment are changed when the control object has the focus.

Note

As long as the window in question is operated only with the mouse no focus frame will appear.

- » Gnats 10126: When changing the attribute `.picture[0]` on the treeview object in a non-visible state, the layout of the treeview object will be adjusted to the new size of `.picture[0]`. However, making such a change in a visible state should be avoided. This can lead to flickering as a result of the necessary reconstruction of the treeview object.
- » Gnats 10124 and 10119: A statusbar object without a frame (`.borderwidth = 0`) no longer overlaps with the toolbar area or the work area. Resulting unclean effect no longer occur.
- » Gnats 10107: The file **readme.txt** on the installation CD has been updated.
- » Gnats 10074: Some mnemonic letters were not shown as underscored and some focus frames were not fully displayed. This was due to the UI status that was introduced with Windows XP: when a window is opened by a mouse click then no focus frames or mnemonics will be shown until the keyboard is used for the next step. This last problem did not always occur in IDM. In Microsoft Windows XP it is possible to determine if the focus frame and mnemonics (underscored letters) can be displayed by the click of a mouse or not. The latter is the default setting in Windows XP.

The Dialog Manager for Windows XP follows this same approach so that neither underscoring nor focus frames appear. As soon as the **Alt** key is pressed, the underscores reappear.

As a result, how the focus frames appear is implicitly influenced. This is the same in the Dialog Manager. After the first time a navigation key is used (e.g. **Tab**) the focus frame appears again. These settings are only valid for one window. When a new window is opened Microsoft Windows XP checks the last entry. Depending on the system settings, in some cases underscores and focus frames are not shown by the click of the mouse. If underscores and focus frames reappear after certain keyboard activity, these settings will remain until the window has been newly constructed.

Under Windows XP this system setting can be changed as follows:

- » open “*Display settings*” (Desktop)
“*Start*” → “*System Control*” → “*Display*”
- » go to “*Display*” tab
- » click on “*Effects...*”
- » remove the check mark next to “*Hide underscored letters for keyboard navigation (reverse with Alt key)*”
- » close all dialogs by clicking “*OK*”.
- » Gnats 9639: Rework: When the `.activeitem` attribute was changed while a poptext object had been opened, no event occurred when `.style = edittext`. This problem always happened when a poptext object contained multiple entries that had the same beginning. Now, when opening the list the first

element in the list is marked and not the element that corresponds to the *.activeitem*. This is the standard behavior of a poptext under Microsoft Windows and therefore cannot be changed.

- » Eliminated: The statusbar object now again displays text in systems that do not support Unicode (i.e. Windows 98). The actual Windows versions support Unicode so that these are not affected by this problem.
- » Changed: Support for Microsoft Visual Studio 2005 has been changed to Microsoft Visual Studio 2008. The sample projects have the number 9 instead of the number 8 in their name.
- » **For your information**
It was possible to choose a list element in a poptext object by doing the following:
By keeping the left button pressed over the arrow or the static text while at the same time moving the mouse releasing the mouse button at the desired position.
As soon as “Visual Styles” are active this no longer works; here you need to let the mouse button go before you choose and then click the mouse again.

29.3 Motif

- » New: The attributes *.alignment*, *.spacing* and *.tilestyle* on the image object are now also supported under Motif. The default value for *.spacing* on Motif platforms is 2.
- » Gnats 10014: A corrupted initial string on the poptext object where *.style = edittext* no longer occurs.
- » Changed: Conform to the standard behavior of a Motif combobox with a permanent list, the pop-text object (*.style = listbox*) does not automatically get the focus by the click of the mouse; this happens only after clicking in the text field area. The *focus* and *deselect* events of the poptext object only appear after the focus has been changed either towards or away from the text field.

29.4 Core

- » Gnats 10297: A crash was possible when writing bootstrap arguments to a tracefile. This could happen if a configfile containing only a single line was read in. This problem has been eliminated.
- » Gnats 10289: The negation of values is working correctly again.
- » Gnats 10276: No more *on extevent* rules are called up when *on changed* events are triggered.

29.5 Editor

- » Eliminated: The deletion of event rules from the object popup menu or via the delete button in the toolbar is up and running again.

29.6 Net

- » Gnats 10264: The decrease of performance on the application side, caused in part by the introduction of Unicode, has been eliminated as well as possible.
- » Gnats 10061: A protocol error could occur when calling a network function while **Applnit** was still running. In this case a restart of the application, including **Applnit**, is avoided and a trace entry is written.

29.7 COBOL

- » Gnats 10166: In the COBOL interface examples a parameter that was needed to call up the **BindFuncs** was missing. This has been corrected.

30 Version A.05.02.a

30.1 Important Changes

- » New: This Dialog Manager version allows user defined attributes, variables, local variables (in rules and methods), parameters and return-types of rules, user defined methods, event rules and application functions to be enhanced with a range of validity. This has been introduced to enable a stricter type checking during the initial uploading process enabling errors in the dialog to be detected early on.
Sample use case: restriction of attributes/variables on objects that are derived from a certain model.
See chapter “Validity Range for Better Type Checking”.
- » New: It is now possible to set up a handler function that is called when errors are detected by the rule interpreter.
See chapter “New Interface Function DM_ErrorHandler”.
- » New: More possibilities for path referencing in the static part of Dialog Manager files.
See chapter “Addition of a Value Resolution in the Path Referencing”.
- » New: The number of parameters for rules within the Dialog Manager has been increased from 8 to 16. It is also now possible to define default values.
See chapter “Increase of Available Parameters to 16”.
Changes in the binary format result from this, see also chapter “Compatibility”.
- » New: The geometric behavior of objects that possess a frame can now be configured through the attribute *.borderraster* (only MS Windows platforms).
See chapter “Borderless Geometry (only for MS Windows)”.
- » New: **Processor symbols were changed** in order to allow for a query of the windows version that will still work in later window versions. The symbol WSIWIN was introduced which is defined by the Windows version number.
Now the following applies:
WSIWIN32 correlates to 0x400
WSIWINXP correlates to 0x501
(for Windows Vista: 0x600)
- » New: Enhancement of the image objects by allowing, for example, the display of non-dockable menu lines.
See chapter “New Additions to the Image Object (only Windows)”.
- » Conversion of the **keyboard navigation under Motif** in order to solve several problems with it.
See chapter “Keyboard Navigation Adjustments for Motif”.
- » New: The *open* event on the *menubox* object is now supported under Motif.
See chapter “Support for on open on Menuboxes under Motif”.

30.2 Validity Range for Better Type Checking

This Dialog Manager version allows user defined attributes, variables, local variables (in rules), parameters and return-types of rules, user defined methods, event rules and application functions to be enhanced with a range of validity. This has been introduced so that a stricter type checking can take place during the initial uploading process enabling errors in the dialog to be detected early on.

Sample use case: restriction of attributes/variables on objects that are derived from a certain model.

The validity range is an add-on to a “value container” e.g. an attribute or a variable (incl. parameters) and does not have a type of its own. A defined range of validity ensures that only settings that fit within this range are allowed – so the “value container” always has a consistent value. The corresponding tests for this happen during the loading and execution of rule code.

Secondly, the range of validity can also limit further access from the “value container”. Access in this sense refers to the access to a child, attribute or method. Such access can lead to the return of a value with a derived range of validity.

Example

```
dialog D
// Basic model with a child
model window MWi {
    statictext StHeader {}
}
// Derived and expanded model
MWi Wi {
    .StHeader {
        integer Width := 10;
    }
    edittext Et {
    }
}
// A second model
model window MWi2 {}
// Entity from the second model
MWi2 Wi2 {
    // Attribute that can only be contained in MWi2-derived objects
    object[MWi2] Ref := Wi; // Syntax error due to violation of validity range
}
// Rule that can only be used on objects derived from MWi
rule void SetHeader(object[MWi] O, string Header) {
    O.StHeader.text := Header; // O can never be Wi2!
    O.Et.xauto := 0; // Syntax error due to access violation
    O.StHeader.Width := 20; // Syntax error due to access violation
}
on dialog start {
    variable object[MWi] O; // Variable only for MWi-derived objects
    SetHeader(Wi,"First");
```

```

SetHeader(Wi2,"Second"); // Syntax error due to violation of validity range
0 := D.child[2]; // Dynamic violation of the validity range
}

```

30.2.1 Restrictions

The following restrictions are currently possible for the data types listed below

Data Type	Validity Value	Value and Access Restrictions, Derivation of the Validity Range
string	integer value (>0)	No restrictions. As before this validity value serves to represent the string size that is necessary for the generation of trampoline data for COBOL.
object	object	The value can be 0 or it must be identical with the validity value or rather be derived from it (comparable to instance_of-Method). Access restrictions to user defined attributes, children and methods that exist on the object given as validity value. Derivation of the validity range when accessing a child object.
object	class	The value can be 0 or it must be an object of the corresponding class. No access restrictions.

A validity range can be defined in the following cases:

1. Data type of a user defined attribute
2. Index data type of a user defined associative field
3. Data type of a global variable
4. Return type of an application function, named rules or user defined methods
5. Parameter types of application functions, named rules and user defined methods as well as event rules
6. Data types of local variables in named rules and user defined methods as well as event rules

Example

```

dialog D
model window MWi {} // Following numbers
MWi Wi { // refer to the
record Rec { // previous list
    object[MWi] Ref := Wi; // 1)
    boolean AssArray[object[MWi]]; // 2)
    .AssArray[Wi] := true;
}
variable object[MWi] GlobalVariable := Wi; // 3)
function object[color] GetBgc(object[MWi] Window); // 4), 5)
rule object[MWi] GetModel(object[Wi] P) { // 4), 5)
    return P.model;
}

```

```

}
on Wi extevent 123 (object[MWi] P) {                               // 6
    variable object[MWi] V := P;
}

```

30.2.2 Access and Value Tests

Static definitions contained in the validity range are always tested during the loading process. This is also true for assignments and accesses in rules that are recognizable as constants. In this sense object paths (i.e. “Wi” in the example below) are to be seen as a “constant” value that can be tested during the loading process.

A dynamic test always takes place in references to value containers with a set validity range or rather in value calls or accesses from within these, and as a result create a **Fail** and can be intercepted within the parenthesis of a fail(...) construct. Call parameters and returns from user defined methods are sometimes only dynamically tested. This is due to the validity information not necessarily being available at loading time when using attributes and methods of imported objects.

As a rule predefined “variables” such as this or thisevent are not bound to a validity range. This is also true for return types and parameters of predefined attributes, methods and built-in functions. Therefore the usage should be carried out with “special cautiousness”.

Since the validity range has no type of its own, no type checking or type conversion takes place. A “casting”, as is used in other programming languages, is not necessary most of the time. This can take place via the intermediate storage on a variable of the data type *object*.

The access testing only takes into consideration the actual status (objects, existing children and attributes).

The following short example contains correct and incorrect applications and shows the expected time at which the test is carried out (load time/run time).

```

dialog D
:
model window MWi { child edittext Et{} }
MWi Wi { child pushbutton Pb {} }
model window MWi2 { }
MWi2 Wi2 { checkbox Cb {} }
:
variable object[MWi] I1 := Wi
variable object[MWi2] I2 := Wi2
variable object O;
:
O := I1; // Validation test not necessary
I1 := O; // Validation test during loading process
I2 := I1; // Validation test during loading process - VALIDATION ERROR!
I2 := W1; // Validation test during loading process - VALIDATION ERROR!
:
// Valid/allowed access to child object:
print I1.Et;

```

```

print O.Pb;

// ACCESS ERROR to child object and its test time period:
print I1.Pb;      // Access test during loading process because I1 possesses
validity range
print O.Unknown;  // Access test during loading process
print I2.Cb;      // Access test during loading process
print I2.Unknown; // Access test during loading process
print getvalue(I2,. Unknown); // Access test during loading process

```

30.2.3 Derivation of the Validity Range

The validity range is derived further when children/attributes are accessed.

This happens when an attribute with a validity range is accessed. This is also true when access to a child objects occurs from a value container with a validity range.

```

dialog D
model window MWi {
  child groupbox Gb {

    child edittext Et {
      rule void Apply() {}
    }
  }
}
MWi Wi {
  object[MWi] Ref := MWi;
  .Gb {
    checkbox Cb { rule void Apply(); }
  }
  rule void Apply() {
    variable object[MWi] This := this;
    This.Gb.Et:Apply();
    This.Gb.Cb:Apply(); // Access error because Cb is not in MWi.Gb
    this.Ref.Gb.Cb:Apply();// Access error because Cb is not in MWi.Gb
  }
}

```

30.2.4 Enhancement of the IDM Syntax

Syntactic changes to the Rule Language are as follows (highlighted red):

General Definitions

```

<DataType> ::= anyvalue | attribute | boolean | class | enum | event |
index | integer | method | object | pointer

```



```

<ParameterType> ::= { input } { output }
<ReturnType> ::= void | <DataType>
<Validity> ::= '[' <ValidityValue> ']'
<ValidityValue> ::= <Class> | <Object> | null | <IntegerValue>
<Object> ::= <Identifier> [ .<Identifier> { ':'<Identifier> ']' } ]

```

Definitions for Named Rules and Methods

```

<Rule> ::= { export } { public } { extern } rule
    <ReturnType> { <Validity> } <Identifier>
    '{' { <Parameter> [ , <Parameter> ] } '}' { <ReposId> }
    <OptionalProgramBlock>
<Parameter> ::=
    <DataType> { <Validity> } <Identifier> { ':' <Value> } <ParameterType>
<OptionalProgramBlock> ::= ';' | <ProgramBlock>

```

Definitions for External Events

```

<ExternalEventRule> ::= on <Object> extevent <ConstantValue>
    '{' { <Parameter> [ , <Parameter> ] } '}'
    { <RuleCondition> }
    <ProgramBlock>

```

Definitions for Local Variables

```

<VariableStatement> ::= { static } variable <Variable> [ , <Variable> ] ';'
<Variable> ::= <DataType> { <Validity> } <Identifier> { ':' <Value> }

```

Definitions for Global Variables

```

<GlobalVariable> ::= { export } { config } <VariableType> <DataType>
    { <Validity> } <Identifier> { <ReposId> } { ':' <Value> } ';'
<VariableType> ::= constant | variable

```

Definitions for User-defined Attributes

```

<AttributeDefinition> ::=
    <DataType> { <Validity> } <Identifier> <AttributeType> ';'
<AttributeType> ::= <ScalarAttribute> | <IndexedAttribute> |
    <AssociativeAttribute> | <ShadowAttribute>
<ScalarAttribute> ::= { ':' <Value> } ';'
<IndexedAttribute> ::= '[' <IntegerValue> ']' { ':' <Value> }
<AssociativeAttribute> ::= '[' <DataType> { <Validity> } ']' { ':' <Value> }
<ShadowAttribute> ::=
    shadows { instance } <Object> <Attribute> { '[' <Index> ']' }

```

Definitions for Application Functions

```
<ApplicationFunction> ::= { export } function { <Language> }  
    <ReturnType> { <Validity> } <Identifier>  
    '(' { <FunctionParameter> [ , <FunctionParameter> ] } ')'   
    { alias <String> } { <RepoId> } <OptionalProgramBlock>  
<FunctionParameter> ::= <Parameter> | <RecordParameter>  
<RecordParameter> ::= record <Object> { := <Value> } <ParameterType>
```

30.2.5 New Attributes

The attributes scope[attr], indexscope[attr], typescope and typescope[] have been added.

30.2.5.1 scope[attr]

Identifier:

.scope[attr]

Classification: object-specific attribute

Definition

Argument type: attribute
C definition: AT_scope
C data type: DT_anyvalue
COBOL definition: AT-scope
COBOL data type: DT-anyvalue
Access: get

This attribute allows the validity range of user defined attributes to be obtained.

For access to the validity range of user defined attributes one must use the index attribute name via the data type *attribute*.

This attribute is available for all objects that are allowed to contain user defined attributes.

See Also

Chapter “Validity Range for Better Type Checking”

Attributes indexscope[attr].typescope, typescope[]

30.2.5.2 indexscope[attr]

Identifier:

.indexscope[attr]

Classification: object-specific attribute

Definition

Argument type: attribute

C definition: AT_indexscope

C data type: DT_anyvalue

COBOL definition: AT-indexscope

COBOL data type: DT-anyvalue

Access: get

This attribute allows for the validity range of the index to be read from user defined associative attributes.

To access this attribute the attribute name is entered as the index of the data type *attribute*.

This attribute is available for all objects that contain user defined attributes.

See Also

Chapter “Validity Range for Better Type Checking”

Attributes scope[attr], typescope, typescope[]

30.2.5.3 typescope

Identifier:

.typescope

Classification: object-specific attribute

Definition

Argument type: void

C definition: AT_typescope

C data type: DT_anyvalue

COBOL definition: AT-typescope

COBOL data type: DT-anyvalue

Access: get

This attribute allows for the validity range of return types to be read by user defined return types.

This attribute is available for the object classes *function* and *rule*.

See Also

Chapter “Validity Range for Better Type Checking”

Attributes scope[attr], indexscope[attr], typescope[I]

30.2.5.4 typescope[I]

Identifier

.typescope[I]

Classification: object-specific attribute

Definition

Argument type: integer

C definition: AT_typescope

C data type: DT_anyvalue

COBOL definition: AT-typescope

COBOL data type: DT-anyvalue

Access: get

This attribute allows the validity range of parameters to be read.

Access to the validity range of named rules, event rules and functions takes place via the indexing with the data type integer in the range from 1 ... *count[.typescope]*.

See Also

Chapter “Validity Range for Better Type Checking”

Attributes scope[attr], indexscope[attr], typescope

30.3 New Interface Function DM_ErrorHandler

DM_ErrorHandler

This new function allows for handler functions to be set up that are then called when an error, which is recognized by the rule interpreter, occurs.

Syntax

```
DM_Boolean DML_default DM_EXPORT DM_ErrorHandler
(
    DM_ErrorHandlerProc funcp,
    DM_UInt operation,
    DM_Options options
)
```

Parameters

->DM_ErrorHandlerProc funcp

This is a pointer to the function that should be installed as an error handler.

-> DM_UInt operation

This parameter defines the action that should be carried out. The following values are possible:

DMF_RegisterHandler Handler function should be registered

DMF_EnableHandler Handler function should be enabled

DMF_DisableHandler Handler function should be disabled

DMF_WithdrawHandler Handler function should be withdrawn

-> DM_Options options

This must be 0 when not being used.

Return Value

TRUE Action was successful.

FALSE Action could not be carried out.

This can happen due to the following reasons: when an action creates two handlers, when the function pointer is NULL or when the handler that is being addressed cannot be found.

A non-intercepted error in the rule code of the application (i.e. faulty parameter types, dynamic access to a non-existing attribute or relative child, each of them not caught by fail()) is noted by the rule interpreter, as EVAL ERROR in the log file or the trace file. Messages in [] that begin with "W:", "E:" or "I:" such as a module/interface loading error do not belong to the rule interpreter and are therefore not passed on to the error handler.

The error handler serves as a way of informing the IDM applications in advance of these application errors. Numerous, not identical, error handlers are possible, which are carried out in reverse order of registration. Only the activated error handlers are called and receive the same information structure

(DM_ErrorInfo) as parameter. These calls are recorded in the trace file. A recursive call of the error handler is not allowed. When the DM_ErrorHandler function is called within an error handler it is not allowed to install or de-install a (new) handler.

Bear in mind that the error handler is synchronously called and insofar the design and coding must take place with great care and under consideration of the special constellations and restrictions! Calling the handler before starting the dialog is possible (i.e. when errors in :init()-rules are present) as is by improper use in an unsuitable run state (i.e. forced loading of a faulty rule via DM_CallRule() within a format function). Avoiding such constellations is recommended. Furthermore, when the IDM is used to display an error box then an independent error dialog, which is started before the application, should be used.

The error handler function to be installed must be defined as follows:

```
typedef void (DML_default DM_CALLBACK *DM_ErrorHandlerProc) __((DM_ErrorInfo
*info));
```

The delivered information structure provides information about the error:

```
define EITK_rule_engine 1

typedef struct {
    /* user info, not changeable */
    DM_UInt1 task; /* task/component which detects the error */
    DM_ErrorCode errcode; /* error code */
    DM_ID object; /* this object or null-ID */
    DM_ID rule; /* rule where error occurred */
    DM_String file; /* module/dialog filename or NULL */
    DM_String message; /* error message or NULL */
} DM_ErrorInfo;
```

Under no circumstance this information must be changed by any error handler!! Principally, strings are coded using the application codepage.

The task entry reveals information about the IDM components that report the error. These should be checked now and again for possible future additions/changes. Momentarily, only errors from the rule interpreter as EITK_rule_engine are passed on.

These entries reveal information about the error code, about the rule in which the error occurred as well as the error text that belongs to it. When possible, the this-object and the file name of the module/dialog, in which the faulty rule can be found, are made available.

Example

```
#include <IDMuser.h>
void DML_default DM_CALLBACK ErrorHandler __1((DM_ErrorInfo *, info))
{
    if (info->task == EITK_rule_engine)
    {
        DM_TraceMessage("ERROR: errcode=%d message=\"%s\" rule=\"%I\"",
        DMF_LogFile|DMF_Printf,
```

```

info->errcode, info->message,
info->rule);
    }
}
int DML_c DM_CALLBACK AppMain (int argc, char **argv)
{
    DM_ID dialogID = (DM_ID)0;
    if (DM_Initialize (&argc, argv, 0) == FALSE)
        return (1);
    if (argc>1)
    {
        if (!(dialogID = DM_LoadDialog (argv[1], 0)))
            return (1);
        DM_ErrorHandler(ErrorHandler, DMF_RegisterHandler, 0);
        DM_StartDialog (dialogID, 0);
        DM_EventLoop (0);
        return (0);
    }
    return 1;
}

```

30.3.1 New Tracing Codes

When error handlers are called they are recorded with the following tracing codes:

[EC] Error handler is called

[ER] Return from the error handler

30.4 Addition of a Value Resolution in the Path Referencing

Up to now, paths in the static part of IDM files only allowed access to an object or rather an exported object (please refer to chapter “Referencing & Referencing Errors” in the release notes for “Version A.04.02.a”). Now, the value resolution is also supported when accessing an attribute or variables so that static paths now behave very similar to rule paths. This means that path sections, which do not describe a child object but rather a variable or an attribute, use their value for the further path resolution. Of course, the same access limitations for exported objects are also valid for additional path resolutions.

Basically, only static settings to user defined attributes and variables support the complete value resolution. It is absolutely necessary that pre-defined attributes receive the specification of a child object in the final path section, in which a value resolution in the previous path section is carried out.

In order to guarantee compatibility with previous IDM applications, the referencing of variables only returns the value as the final path section, if the target variable or target attribute is not of type any-value or object.

Please keep in mind that the path resolution takes place before the initialization of the objects (call of the :init method); the attribute or rather the variable value must also be previously set. Accordingly, no predefined :get-methods for the value resolution will be called during the attribute access.

Important

Despite the assimilation attained between “static” paths and paths within the dynamic rule section, they behave much differently especially with respect to global variables. Value resolution in paths within the rule section has priority; variable referencing is possible by using .self. This is different when working with static initializations. In this case, and according to the target type, the reference has a higher priority than the value.

When using the IDM editor one should avoid using static initializations that function by means of value resolution because during the saving process these paths cannot be written out or restored (analog to string attributes that can be initialized with text resources).

The following example illustrates the possible forms of usage and also contains cases that show a path resolution error.

Example

```
// -----  
// File: bsp.mod  
module M  
export record ERec {  
  object Ref := ERec.Sub2;  
  export child record Sub1 {  
    integer X := 2001;  
  }  
  child record Sub2 {  
    integer Y := 1998;  
    child record Sub3 {  
      integer Z := 2010;  
    }  
  }  
}  
}  
// -----  
// File: bsp.dlg  
dialog D  
  
import I1 "~:bsp.if";  
  
variable integer V1 := 1918;  
variable object V2 := Defs;  
  
text Tx "Hello World";  
  
// Definition of attributes for a static initialization
```



```

record Defs {
    integer D1 := 640;
    integer D2;
    string D3 := "Hello!";
    integer D4[4];
    .D4[2] := 666;
    :init() {
        this:super();
        this.D2 := 2;
    }
    record SubDefs {
    }
}

record Rec {
    integer A := V1;          // Path value resolution for a variable
    integer B := Defs.D1;    // Path value resolution for an attribute
    object C := D.V1;        // Path closure to an ID/object-Ref.
    // integer D := Defs.D2; // ERROR: Value at time of termination unseeded!
    string E := D.V2.D3;     // Multiple value resolution via object-Ref.
    integer F := ERec.Sub1.X; // Termination via exported child object
    // integer G := ERec.Sub2.Y; // ERROR: Sub2 not exported!
    integer H := ERec.Ref.Y;  // Detour allowed via object attribute
    // integer I := ERec.Ref.Sub3.Z; // ERROR: Sub3 not exported!
    // integer J := Rec.K; // ERROR: Mutual paths are not terminable!
    // integer K := Rec.J; // ERROR: Mutual paths are not terminable!
    integer L := Defs.D4:[2]; // Path termination on separate field elements
}

window Wi {
    // .title Defs.D3; // ERROR: Path end piece is not an object
    // .width Defs.D1; // ERROR: Path closure only pre-def. object attributes
    // .userdata Defs.D1; // ERROR: Path end piece is not an object
    .userdata V2.SubDefs; // Path value resolution OK at the beginning
    edittext Et {
        .content Tx; // Value resolution for text resources has always worked
    }
    on close {
        exit();
    }
}

```

30.4.1 New Attribute .constant

Identifier:

.constant

Classification: object-specific attribute

Definition

Argument type: boolean
C definition: AT_constant
C data type: DT_boolean
COBOL definition: AT-constant
COBOL data type: DT-boolean
Access: set/get

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

This attribute is only available for the **variable** object.

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

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

Example

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

30.5 Parameter Enhancements within the Dialog Manager

Now, 16 different parameters and the definition of default values of parameters are possible in the Dialog Manager.

This also has an effect on the graphic editor and the compatibility with other Dialog Manager versions.

For further information please refer to chapters “Increase of Available Parameters to 16”, “Optional Default Values for Parameters”, “Expansion Editor” and “Compatibility”.

30.5.1 Increase of Available Parameters to 16

Beginning with this version of the Dialog Manager it is now possible to use up to 16 parameters:

- » User defined named rules and methods
- » Predefined methods and built-in functions. This includes: **:super**, **:recall**, **:call**, **sprintf**, **sendevent** and **execute**
- » Parameter list by event rules for external events
- » Calling up OLE server methods (via a control object)
- » DM functions for calling up rules, methods and functions. This includes: **DM_CallRule**, **DM_CallFunction**, **DM_CallMethod**

30.5.2 Optional Default Values for Parameters

The Dialog Manager Rule Language has been enhanced with the possibility of consigning default values to parameters. The preservation of the assigned value for all parameters allows for rule-, method- and function-calls with less arguments.

The declaration is possible in the following areas:

1. When declaring named rules and methods

```
{export} rule <Returntype> <ID> ([<Datatype> <Name> {:= <Default-Value>}  
<Type>])
```

2. When declaring application functions

```
{export} function <Language> <Returntype> <ID>  
    ([ <Datatype> {<Name of the Parameter>} {:= <Default-Value>} <Type>]  
    ) ;
```

3. And for event parameters of external events

```
on {<Objectpath>} extevent <Number/Value> {[<Datatype> <Name> {:= <Default-  
Value>} <Type>]}
```

The default values are only permitted for parameters of input types and must be provided consecutively at the end of the parameter list. Furthermore, they need to match the data type of the parameter.

Calling methods, rules and functions as well as sending external events can be done without specifying the parameters for which default values are given. The argument values will be assigned from left to right of the parameter variables according to their position. Therefore it is recommended to place the parameters that are most frequently omitted at the end of the parameter list.

The parameter variables contain default values not only when called directly from the Rule Language, but also when they are called indirectly via **:call/:recall/:super** or a DM function on the application side (e.g. via **DM_CallRule**, **DM_CallMethod**, **DM_CallFunction**, **DM_QueueExtEvent** or **DM_SendEvent**).

Attention

Optional parameters are not supported by the Dialog Manager control object or by the message resource; the use of default values when programming an OLE server or an OLE client with the Dialog Manager OLE interface is currently not possible.

30.5.3 Expansion Editor

It is now possible to declare up to 16 parameters for application functions. It is also possible to specify a default value.

30.5.4 Compatibility

The binary format has been changed. A forward compatibility exists, this means that binary data from older IDM versions are executable, but conversely newly created binary files cannot be read by older IDM versions.

Due to changes made to the Dialog Manager Header file, a new compilation of the application is necessary.

30.6 Borderless Geometry (only for MS Windows)

The new attribute *.borderraster* has been added so that the geometrical behavior of an object with borders can be configured.

30.6.1 Attribute *.borderraster*

Identifier:

.borderraster

Classification: geometry attribute

Definition

Argument type: boolean

C definition: AT_borderraster

C data type: DT_boolean

COBOL definition: AT-borderraster

COBOL data type: DT-boolean

Access: set/get

This attribute is only available on MS Windows platforms.

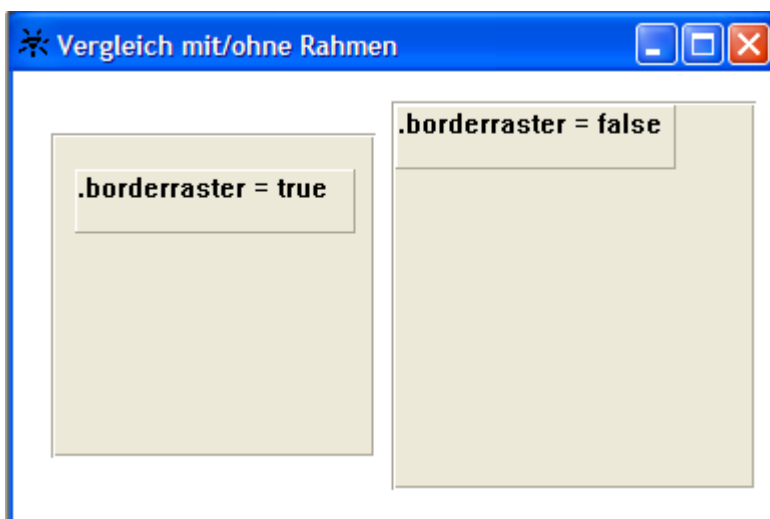
The default value of the attribute is *true*.

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

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

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

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



The difference is that the borders still appear, when *.borderraster = false*. The attribute can only make an appropriate adjustment if the object possesses a border and if raster coordinates are used. The

position (*.posraster = true*), the size (*.sizeraster = true*) and where applicable also the position of the child object can be influenced only when their positions are specified in raster units.

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

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

Object	Support of .borderraster
window	supported

30.6.1.1 The Meaning of *true*

The calculation of raster coordinates takes place in the usual way (analog to earlier versions of the Dialog Manager).

For objects with borders this means:

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

30.6.1.2 The Meaning of *false*

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

Note

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

30.6.2 References and Dependencies

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

Currently, the classification is as follows:

Objects Without Borders

canvas when *.borderwidth = 0*

checkbox

control	
edittext	when <i>.multiline = false</i>
groupbox	when <i>.borderwidth = 0</i>
image	
layoutbox	when <i>.borderwidth = 0</i>
notepage	when <i>.borderwidth = 0</i>
poptext	when <i>.style <> listbox</i>
progressbar	
pushbutton	
radiobutton	
rectangle	when <i>.borderwidth = 0</i>
scrollbar	
spinbox	
splitbox	when <i>.borderwidth = 0</i>
statictext	
toolbar	when <i>.docking <> dock_window</i> AND <i>.borderwidth = 0</i>

Objects With Borders

canvas	when <i>.borderwidth > 0</i>
edittext	when <i>.multiline = true</i>
groupbox	when <i>.borderwidth > 0</i>
layoutbox	when <i>.borderwidth > 0</i>
listbox	
notebook	
notepage	when <i>.borderwidth > 0</i>
poptext	when <i>.style = listbox</i>
rectangle	when <i>.borderwidth > 0</i>
splitbox	when <i>.borderwidth > 0</i>

statusbar when *.borderwidth > 0*
 toolbar when *.docking = dock_window* OR *.borderwidth > 0*
 tablefield
 treeview
 window

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

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

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

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

Single-lined Objects

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

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

30.7 New Additions to the Image Object (only Windows)

The attributes listed below have been added to the image object. These attributes allow for redockable menu bars or menu bars, which are “interchangeable” with other bars, to be designed. The image objects that have these attributes can be arranged accordingly in the toolbar object.

30.7.1 Attribute alignment

Identifier:

.alignment

Classification: object-specific attribute

Definition

Argument type: integer

C definition: AT_alignment

C data type: DT_integer

COBOL definition: AT-alignment

COBOL data type: DT-integer

Access: set, get

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

By using these attributes it is possible to define the text format.

-1 = right-aligned

0 = centered

1 = left-aligned

Determines how the text will be horizontally justified in the area provided by the **image** object (default value = 0).

An exact explanation of the effect the attribute *.alignment* can have depending on the attributes *.spacing* and *.tilestyle* can be found in chapter “Attribute tilestyle”.

30.7.2 New Attribute spacing

Identifier:

.spacing

Classification: object-specific attribute

Definition

Argument type:	integer
C definition:	AT_spacing
C data type:	DT_integer
COBOL definition:	AT-spacing
COBOL data type:	DT-integer
Access:	set/get
Value Range:	-32767 ... 32767
Default Value:	0

This describes the space between the image and the text. This space is also taken into consideration when either image or text are missing (additional shifting).

An exact explanation of the effect the attribute *.spacing* can have depending on the attributes *.alignment* and *.tilestyle* can be found in chapter “Attribute tilestyle”.

30.7.3 Attribute tilestyle

Identifier:

.tilestyle

Classification:	object-specific attribute
-----------------	---------------------------

Definition

Argument type:	enum
C definition:	AT_tilestyle
C data type:	DT_enum
COBOL definition:	AT-tilestyle
COBOL data type:	DT-enum
Access:	set,get

This attribute describes the position of the image (*.picture*) within the **image** object. If there is no image, this will be handled like images whose width and height are 0.

Possible values are:

<code>tilestyle_icon</code>	As was in the past, the image is placed at a certain distance above the text (<i>.spacing</i>). The object is horizontally centered, the text is horizontally connected to the image according to <i>.alignment</i> (normal: centered). Vertically the object and the text are seen as one unit that is centrally aligned. If the image is scalable then the text is placed at the bottom border and the remaining space is taken up by the image itself.
<code>tilestyle_left</code>	Here the image is placed at the far left side of the image object. The display area for the text begins at a defined position (<i>.spacing</i>) to the right and continues to the end of the right border. By using <i>.alignment</i> the text is connected to the display area. The image as well as the text is vertically centered. If the image is scalable, then the text will be situated on the far right side and <i>.alignment</i> becomes meaningless. The image then takes up the remaining free space.
<code>tilestyle_right</code>	The image is placed at the right border of the <i>image</i> object. The display area for the text begins at a set distance <i>.spacing</i> to the left and continues to the far left border. The text is connected to the display area according to <i>.alignment</i> . The image as well as the text is vertically and centrally aligned. If the image is scalable then the text is found on the left side and <i>.alignment</i> becomes meaningless. The image then takes up the remaining free space.
<code>tilestyle_top</code>	The image is placed at the upper border of the <i>image</i> object. The display area for the text begins at a defined position <i>.spacing</i> just below the image. The image is horizontally centered and the text is connected (horizontal) to the image object according to <i>.alignment</i> . If the image is scalable then the text is found at the lower border and the image then takes up the remaining available space.
<code>tilestyle_bottom</code>	The image is located at the bottom border of the <i>image</i> object. The display area for the text begins at a defined position <i>.spacing</i> just above the image. The image is horizontally centered and the text is connected (horizontal) to the image object according to <i>.alignment</i> . If the image is scalable then the text is found at the upper border and the image then takes up the remaining available space.
<code>tilestyle_background</code>	The image is displayed as a background. The text is situated on top of the image. The image is in a central, horizontal position and the text is connected (horizontal) to the image object via <i>.alignment</i> . The image and the text are displayed in a vertical/centered position. If the image is scalable then the image will take up the entire space provided.

30.8 Keyboard Navigation Adjustments for Motif

The keyboard navigation in the Dialog Manager for Motif has been changed so that no changes in the navigation order arise when objects are made visible subsequently or when they are forced to be newly created in the window system due to changes made to the attributes.

The order in which objects can be navigated in a Dialog Manager application via the keyboard (**Tab** key) is determined according to the order sequence in the child vector (normally according to the

order of set up). Grouping objects such as groupbox cannot hold a keyboard focus but are seen more as a place holder since navigable child objects could be contained inside them.

A further special feature found in the Dialog Manager to comply with user interface style guides is the grouping of checkboxes and radiobuttons that are in the same father object. These are seen as a separate “super object”, name it group if you will, within the `Tab` navigation. Here the navigation takes place via the control keys (Cursor `Up`, `Down`, `Left`, `Right`) and not via the `Tab` key.

Objects can be taken out of the keyboard navigation via the `.navigable` attribute; they remain however focusable via the mouse.

The list of navigation possibilities via the keyboard for Motif-WSI can be set up as follows. Please note that not every object can offer all possibilities because some objects have pre-set keys for their own purposes – for example the use of the cursor key through the *edittext* objects.

Key	Function
<code>Tab</code>	Keyboard focus switches to the next navigable object or group according to the order of the children. When the focus is on the final element in the object hierarchy, then the focus will go to the next navigable object in the next sister hierarchy or to the very beginning
<code>Shift + Tab</code>	Reverse direction of <code>Tab</code>
Cursor <code>Down</code> , <code>Right</code>	Jumps to the next object within a group or to the beginning if the focus is already on the last object. If the object itself does not require this key for its operation, then changing the focus takes place analog to the <code>Tab</code> key. This is not true for checkboxes or radiobuttons
Cursor <code>Up</code> , <code>Left</code>	Reverse direction of Cursor <code>Down</code> , <code>Right</code>
Home (<code>beginLine</code>)	Jumps to the first object within a group

The MS Windows system has a number of special navigation keys such as `End` that allows the user to jump to the end of a group or the key combinations `Ctrl + Tab` and `Ctrl + Shift + Tab` that allow the user to jump to the next or the previous tabs. These are also very effective in notebooks.

The Dialog Manager tries its best not to influence the normal windows keyboard navigation process in order to maintain the system-specific navigation mechanisms.

The following is a small example of keyboard navigation:

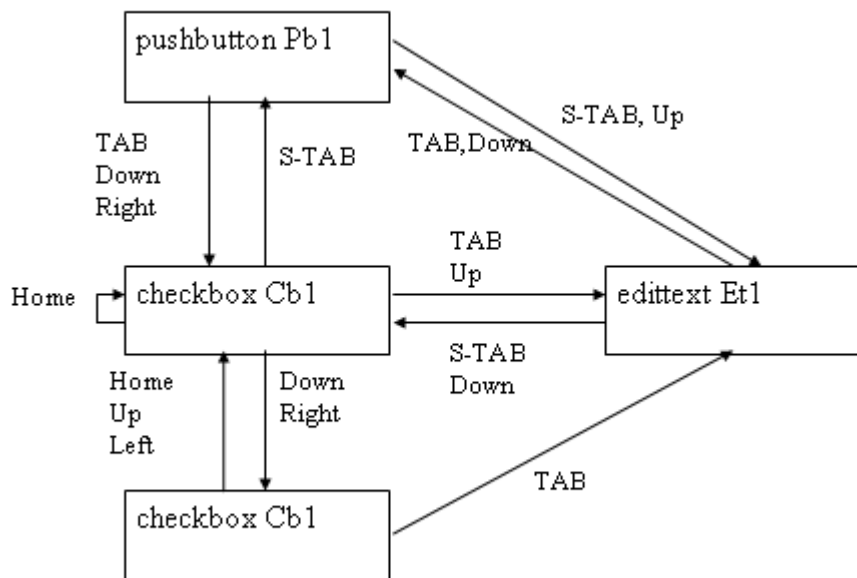
```
dialog D
window Wi { .title "Wi"; .width 400; .height 400;
  pushbutton Pb1 { .text "Pb1"; }
  groupbox Gb { .ytop 30; .xauto 0; .yauto 0;
    checkbox Cb1 { .text "Cb1"; }
    edittext Et1 { .content "Et1"; .xleft 100; .width 80; }
    checkbox Cb2 { .text "Cb2"; .ytop 30; }
```

```

}
on close {
    exit();
}
}

```

By using the keyboard one is able to navigate as follows:



30.9 Support for on open on Menuboxes under Motif

The *open* event for menuboxes is now also supported by the Dialog Manager for Motif platforms beginning with the version Motif 2.1. On Motif 1.2 systems no *open* events are created; this should be kept in mind during the implementation of dialogs on various platforms.

This event is created for pop-up menus (object menus) for pull down menus (menus in one menu bar) and also for sub-menus and is called before the menu appears. Only changes to the sub-menus, i.e. transformation to sensitive, switching to visible, adding or removing menuitems or sub-menus are allowed to be made.

The following specific attributes of the *thisevent* objects are set as follows:

Attribute	Description
.value	Object is activated via the menu. This value is only set for the top popup menu.

Attribute	Deascription	
.x / .y	In popup menus these contain the coordinates where the mouse button had been pressed (relative to the object in Pixel). Supported by the objects: canvas, listbox, image, layoutbox, poptext, groupbox, notepage, splitbox, tablefield, treeview	
.index	For popup menus of the objects listed below this attribute holds a reference to the element/sub-area over which the mouse cursor had been located when the menu was opening. Possible types are integer I and index [F,S]. Setting can only take place when an evaluation is possible.	
	poptext	I when clicking within a list in a listbox style, cursor position [F,S] when the mouse is positioned in the text field in an edittext/listbox style.
	listbox treeview	Entry I, above which the mouse cursor was located.
	tablefield	Entry [F,S], above which the mouse cursor was located.
	edittext	Cursor position [F,S], above which the mouse cursor was located, whereby F and S have the same value.
	splitbox	Split area I above which the mouse cursor was located. Implemented for the bar area.

Due to an error in the Motif library, object-specific values for a **treeview** object are only supported in the OpenMotif platform beginning with the version 2.2.3.

Please note that menuboxes for **rectangle** objects are not supported in the Motif platforms.

The normal Motif menubox knows two operating modes for a menu:

1. The menu button of the mouse is held down. The menu opens. While pressing the mouse key, move to an entry item and release the button to select the item. The menu closes. If the mouse key is released outside of the menu, no menu entry is selected.
2. Menu button is pressed and then released immediately. The menu opens and remains in this state. A menu entry is selected either with the right or left mouse button. Thereupon the menu closes. By clicking elsewhere the menu is closed without selecting a menuitem.

The differentiation between the two modes from Motif is made through the inclusion of the MultiClick time. As a rule, when dealing with *open* rules whose execution exceeds the allowed time limit, the Dialog Manager opens the menu analog to mode 2.

30.10 Windows

- » New: The new attribute *.borderraster* for objects with borders is now available. For more information refer to chapter “Borderless Geometry (only for MS Windows)”.
- » Changed: A change to the processor symbols has been made to allow for query definitions that function (unaltered) under other Windows versions. The symbol WSIWIN, which is defined by the Windows version number, has been added.
Now the following applies:
WSIWIN32 correlates to 0x400
WSIWINXP correlates to 0x501
(for Windows Vista: 0x600)
- » Gnats 10269: Spontaneous crashing of the Dialog Manager (dialog was ended without notice) has been eliminated.
- » Gnats 10268: Sometimes a dirt-like effect on the border of a tablefield object appeared. It was as if the connected edittext object was displayed in the border region. This has been eliminated.
- » Gnats 10267: An Assfail that occurred in connection with an editable table (tablefield object with an assigned edittext object) has been eliminated.
- » Gnats 10150: A crash within a modularized environment, that arose while accessing previously destroyed objects of another module, has been eliminated. This was discovered in the IDM Editor in connection with the reloading of a module file.
- » Gnats 10234: The focus borders in an image object without text are now again shown in the W2k version.
- » Gnats 10131: The sensitive status of **scrollbar** objects (especially on **tablefield** objects) is correctly displayed again.
Problems were observed on **tablefield** objects being filled in an invisible state and having “Visual Styles” activated, when the arrows in the scrollbar were activated while the tablefield was constructed but still invisible. These problems can occur either when the attribute *.mapped* on the affected object (or a father object) is set to *false* or when the object in question is located in an inactive notepage.
- » Gnats 10122: A rectangle object with raster coordinates within a window no longer flickers when changing the size or position of the window.
- » Gnats 10121: The cursor no longer becomes invisible when the list of a poptext object is opened via the keyboard (Workaround for a Microsoft Bug).

Note

If the list of a poptext is open, the cursor is not changed by the poptext object. In particular, the cursor is no longer hidden when the edittext is edited. This is caused by the Microsoft Windows object (Windows XP SP2) and thus can change again in future Windows versions.

- » Gnats 10120: On a splitbox object without borders (*.borderwidth* = 0), the cursor is now only changed within the “splitbar” and not one pixel next to it. As a result, the user no longer experiences non-triggered shifting problems.

- » Gnats 10118: A **toolhelp** object is now again displayed with borders even when on the **setup** object the option is set to `.options[opt_balloon_toolhelp] false`.
- » Gnats 10114: With active “Visual Styles” the lower or rather right border of the **tablefield** object, with non-displayed scrollbars, is now again displayed in the correct color.
- » Gnats 10112: A system crash no longer occurs when an application, that uses the OLE options, is dynamically linked and a **control** object gets the focus.
- » Gnats 10109: An opened **toolhelp** on a **poptext** object no longer flickers when the list of the **pop-text** object is opened.
- » Gnats 10108: The value of `.real_size[l]` of a splitbox object is now again returned for the correct area.
- » Gnats 10106: Changing the attribute `.edittext` on a visible **tablefield** object no longer leads to a system crash.
- » Gnats 10077: When a mouse click incidentally hits a **toolhelp** object that is being opened or is already open, it no longer disappears.
- » Gnats 10076: The closing and re-opening of an already opened context menu by the user no longer leads to the context menu only being partially displayed. In this case, no “*ASSERTION failed*” messages are recorded in the tracefile anymore.
- » Gnats 10071: A **toolhelp** object that is opened and then immediately closed reappears when the object in question is touched with the cursor.
- » Gnats 10060: The Dialog Manger for the Windows XP platform now again displays **toolhelps** in the operating system Windows 2000 (and older versions).
- » Gnats 9411: The **statusbar** object is now able to hold **image** and **progressbar** objects. These are only used for display purposes and are therefore implicitly insensitive. It is also now possible to create **image** and **progressbar** objects as children of **statusbar** objects. Please note that the children of statusbar objects are used for display purposes only; they have no additional function and for this reason the attribute `.sensitive` must be set to `false`. Furthermore, child objects are not allowed to be freely positioned. Only the width (`.width`) and size raster (`.sizeraster`) independent of the connection (`.xauto`) is used for calculating the width. The height is automatically set via the **statusbar** object and is calculated through the font (`.font`) that is set on the **statusbar** object. This cannot be set differently.

Note

The **statictext** object, as a child of a **statusbar** object, now takes into account its own font (`.font`). In order to sustain the original behavior, the attribute `.font` must be set to `null`. In this case the font of the father object or rather the **statusbar** object is used.

- » Gnats 8605 and 9952: Additions to **image** objects, see chapter “New Additions to the Image Object (only Windows)” for further information.
- » Eliminated: The errors within the manifest file for the XP platform have been eliminated. Now the example applications can be started again.

30.11 Motif

- » New: The **spinbox** is now available under Motif 2.x.
The spinbox object is now available, analog to the existing Windows implementation, for the Motif 2.x platforms.

Special Features

In contrast to the `XmSimpleSpinBox` widget, the use of all keyboard functions is not possible. Only the cursor keys `Up` and `Down` lead to a spinning that guarantees that an **edittext** child can be edited. The IDM sets the size of the arrows to half of the spinbox height (maximum of 16 pixels) so that an automatic scaling takes place and the appearance remains the same for the lower spinboxes.

- » New: Support for transparent icons on **window** objects has been added.

Note

The dynamic conversion of a window icon to *null* is supported by the Dialog Manager, but unfortunately ignored by Motif and the respective Windows manager. As a result the last set icon remains visible.

- » New: The *open* event on the **menubox** object is now supported.
For further information please refer to chapter “Support for on open on Menuboxes under Motif”.
- » Gnats 10004: The support for **notepage** objects within the graphical editor has been improved.
- » Gnats 10014: Several problems that arose with the **poptext** object in connection with the conversion of the *.content* attribute and formats have been corrected.
- » Gnats 3590, 9116, 9769 and 9822: Through adjustments of the keyboard navigation these problems have been solved.
For further information please refer to chapter “Keyboard Navigation Adjustments for Motif”.

30.12 Core

- » New: The addition of a validity range to the Dialog Manager, see chapter “Validity Range for Better Type Checking”.
- » New: It is now possible to install a handler function that is called when errors, which are recognized by the rule interpreter, occur.
For further information refer to chapter “New Interface Function `DM_ErrorHandler`”.
- » New: More possibilities for path referencing in the static part of Dialog Manager files.
For further information please refer to chapter “Addition of a Value Resolution in the Path Referencing”.
- » Gnats 10275: Errors that are detected in the rule analyzer are displayed as error boxes. If no line number is given, then the faulty rule is recorded. This allows for the error to be detected in the file.
- » Gnats 10261: A problem arose during the binary writing of modules in connection with global variables. This has been taken care of as best as possible. Now, no value replacement occurs in paths that are resolved in a normal way (not complete) and which have a variable at the end of their path.

- » Gnats 10250: A problem that arose in connection with the new validity test has been eliminated. This problem sometimes leads to a disruption of the module loading process.
- » Gnats 10182: The interface file is now correctly written for functions whose parameters use default values.
- » Gnats 10152: Flushing by CR in a memory buffer no longer creates an endless recursion.
- » Gnats 10105: Correction of memory allocation through `pass` and `fail` to fix a system crash.
- » Gnats 9456: An Assfail no longer occurs when the execution of an `on open` rule on a `menubox` is triggered while changing the `.text` attribute on a ***statictext*** object.
- » Gnats 9132: The change of the `.content` attribute on a `pop`text object (combobox) no longer triggers the sending of an *.activeitem changed* event.

30.13 Editor

- » Gnats 10151: A spelling error in the editor has been corrected.
- » Gnats 10236: While moving an object that happens to be a child object of an invisible object to another father object in the IDM Editor, the object now remains visible.
- » Gnats 10153: Named rules in an editable sub-module are now again written out when saved.

30.14 COBOL

- » Gnats 10164: A problem that arose during the transformation of COBOL character strings into C character strings has been eliminated. This problem was noticed in that the interface function `DMcob_LSetVectorValueBuff` considerably slowed down in speed. This was especially true when dealing with larger data fields.

30.15 Java-Platform

- » Gnats 9903: A statically, by means of a character string defined format is now regarded.
- » Gnats 9488: Now clicking the mouse in a ***listbox*** object creates only one *select* event.
- » Gnats 9479: The built-in function **`applyformat()`** and the interface function **`DM_ApplyFormat`** are now supported in the Java platforms. However, please note that numerical formats are not supported at this time.

The Java interface has been enhanced with the addition of the following methods for formatting strings:

Class `DM_Display`

```
public String applyFormat(DM_String format, DM_String str, DM_Options
options)
    throws DM_Exception
```

```

public String applyFormat(DM_String format, DM_String str)
    throws DM_Exception

public String applyFormat(String format, String str, DM_Options options)
    throws DM_Exception

public String applyFormat(String format, String str)
    throws DM_Exception

```

Class DM_Object

```

public String applyFormat(DM_String str, DM_Options options)
    throws DM_Exception

public String applyFormat(DM_String str)
    throws DM_Exception

public String applyFormat(String str, DM_Options options)
    throws DM_Exception

public String applyFormat(String str)
    throws DM_Exception

public String applyFormat(DM_String format, DM_String str, DM_Options
options)
    throws DM_Exception

public String applyFormat(DM_String format, DM_String str)
    throws DM_Exception

public String applyFormat(String format, String str, DM_Options options)
    throws DM_Exception

public String applyFormat(String format, String str)
    throws DM_Exception

```

Parameters

format

The format string.

str

The string that is to be formatted.

options

Additional options. If these are not specified, then DM_Options.DEFAULT is used.

Return Value

The result of the format application.

Note 1

When no format parameter is given, then the object must be a format resource or a format function.

Note 2

Java WSI still does not support numerical formats.

- » Gnats 8812: The **setup** object now also returns WSI values, i.e. for *setup.screen_width*.

30.16 Debugger

- » Gnats 10229: The Assfail that occurred when the debugger came to a halt in code lines containing “\$” has been eliminated.

31 Version A.05.01.h

31.1 Java

- » Gnats 10346: Dialog events are now generated when a dialogbox (**window** with `.dialogbox = true`) is open.
- » Gnats 10345: Visible objects can now again be made invisible dynamically.
- » Gnats 10343: The examples for the Java interface are now part of the installation.

31.2 New Functions for Error Analysis

The new functions for error analysis (see “Version A.05.01.g3”) are also available in version A.05.01.h.

32 Version A.05.01.g4

32.1 Core

- » Gnats 9456: If while changing the *.text* attribute of a ***statictext*** object an on open rule of a menubox is executed, no more assfails will occur.

33 Version A.05.01.g3

33.1 New Functions for Error Analysis

33.1.1 Overview

The IDM runtime libraries have been enhanced in order to simplify the error analysis of IDM applications. The purpose of adding the new functions is to further assist the developer in the analyzation and classification of errors in special situations such as: interpreter error messages, application crashes¹ or fatal error messages. The following table contains a rough classification of possible errors in relation to typical points of error and the likely impact of such errors. The effect that an error remains undetected, leads to a changed/faulty functionality or may be transferred onto another component, is not explicitly mentioned in the table.

Point of Error	Description	Impact
Rule code	Error in rule code of an IDM application, e.g. invalid access to an attribute or object, type errors or uninitialized values.	*** Eval Error in logfile/tracefile [E: ...] or [W: ...] messages, YE-error codes in the tracefile
Application function	Several error types, e.g. access violations, damaged memory management that in some cases may show through their impacts on the rule processing or the IDM library.	eventual crash
IDM library	Faulty conditions within the IDM are normally secured through an assertion. However, an error can have an effect on the rule processing or it can appear as an application error.	FATAL ERROR in logfile/tracefile [E: ...] or [W: ..] message in tracefile
External functions	External, IDM-independent functions, e.g. in a different thread or different library, can contain errors which can then be brought into other areas via actions such as overwriting memory or faulty synchronization.	e.g. crash

¹A crash in this sense is an unhandled or a non-interceptable error situation, i.e. invalid memory access, division through 0.

In the following further information pertaining to the new additions for easier error analysis and their purpose can be found.

Comments

- » In principle the new additions are not meant to replace or substitute a debugger or a tool such as DR. WATSON under MS Windows. They are to be seen as add-ons in order to deliver meaningful information that cannot be delivered by the mentioned tools.
- » If previous serious errors have already occurred, then the correctness of the given information cannot be completely guaranteed.

33.1.1.1 Safety Tracing

In order to use the tracing function within the IDM for situations, in which application errors occur only after a longer period of time, the tracing safety mode can be used.

Normally, if the tracing function is running alongside application use this can lead to a decrease in performance and to a very large, cumbersome trace file. In the safety mode the tracing takes place in a ring buffer with a limited number of lines and line lengths that are kept in the main memory. The buffer is then written in the tracefile only after an application has been ended or after an error event.

33.1.1.2 Dumpstate

Without tracing, the user receives no clues when crashes, “*FATAL ERROR*” or “*Error in Eval*” messages occur in applications that use binary data and the IDM runtime library. The dumpstate reveals status information of the IDM specifics which include the callstack, the errorstack, events and hints pertaining to the number of IDM objects and the use of memory through the IDM.

The primary goal is to output as much information as possible when an error occurs in order to be able to properly assess the error and at the same time simplify the causal research. Only information that is known to the IDM can be gathered; no information pertaining to the application or foreign functions. The writing of the dumpstate can be programmatically achieved via the Rule Language or the DM interface function.

Important Note

Security-related information, e.g. passwords that are entered into a login window, should be deleted by the application immediately after they have been used so that they cannot be revealed through a dumpstate and as a result be made accessible to others.

33.1.1.3 Exception Catcher

In order to be able to write out the safety tracing as well as the dumpstate after a crash, a global exception catcher is registered for the application. The exception catcher passes on the exception in order to allow for its further usual processing through the operating system (core dump under Unix/Linux or the writing of a dump file under MS Windows via DR. WATSON).

33.1.2 Dumpstate Output

The dumpstate is the status information of IDM-relevant information in order to simplify the error analysis within an IDM application. The content of the dumpstate is divided into different sections that are variable and that are adapted to the error situation. In addition, the dumpstate is influenced by the errors that have previously occurred. For example an unsuccessful memory allocation leads to information concerning the memory usage by the IDM in the next dumpstate output. If no IDM objects or identifiers can be created, then the utilization of IDM objects and identifiers is dumped.

The dumpstate information is always enclosed between `*** DUMP STATE BEGIN ***` and `*** DUMP STATE END ***` and can have the following sections, which are described in detail in the paragraphs below:

- » **PROCESS:** Process and thread number, date/time.
- » **ERRORS:** Complete content of set error codes.
- » **CALLSTACK:** Contains rules, DM interface functions and application functions directly called by the IDM.
- » **THISEVENTS and EVENT QUEUE:** Actual processed thisevent objects and their values as well as events that are still in the queue.
- » **USAGE:** The number of created objects, modules and identifiers and the size of the memory that is used by the rule interpreter and for string transfer.
- » **MEMORY:** Memory usage as far as it can be detected by the IDM.
- » **SLOTS:** Hints about IDM objects that have not been correctly released.
- » **VISIBLE OBJECTS:** A list of the visible objects and their respective values.

In order to keep the output to a minimum, this is usually displayed in a shortened form. Normally, a shortening of the IDM strings (in "...") to a maximum of 40 characters always occurs. Their entire length is attached in []. Byte size information is given in kilo, mega or gigabytes (k/m/g).

The dumpstate is normally automatically written to the logfile or the tracefile when one of the following situations occurs. Hereby, the situation also influences the dumpstate content.

Error Type	Dumped Sections
The rule interpreter delivers an <i>"EVAL ERROR"</i>	Errorstack, Callstack, Events
The IDM library recognizes an error of its own and delivers a <i>"FATAL ERROR"</i>	All
An catchable exception (e.g. access violation, stack overflow, division by zero) comes about and the IDM exception catcher is active	All – in addition, the exception code (Windows) or the signal number (UNIX) is initially displayed
Normal shutdown via <code>DM_ShutDown()</code> but information about errors are at hand	According to the hints

The writing of the dumpstate can explicitly take place via the built-in function **dumpstate()** or the interface function **DM_DumpState()**. In addition, it is also possible to influence the dumpstate output independent of the type of error or to define the type of error in which a dumpstate output takes place via the options **-IDMdumpstate** and **-IDMdumpstateseverity**.

Due to the amount of complex information that is gathered by the dumpstate, crashes within the dumpstate functionality can happen and cannot be intercepted.

The following contains a detailed description of the various sections that can be found in the dumpstate:

33.1.2.1 Process

This area contains the process ID, the number of the thread in which the dumpstate is called as well as the thread number of the IDM main thread. If the dumpstate is not called from the IDM main thread, then the following message is displayed:

```
ATTENTION: not in IDM main thread!
```

When this happens extreme care should be taken. This indicates either an application error, an error in an external function or the improper use of the IDM. Please note that the callstack only contains the call list of the IDM main thread!

```
PROCESS:
pid=2984, thread=4080, IDM-thread=4080, date=2009-11-10, time=16:20:38
```

33.1.2.2 Errors

Lists all currently set error codes. See also interface function **DM_QueryError** in manual “C Interface - Functions”.

```
ERRORS:
#0: IDM-E-UnkAttr: Attribute not available for this type of object
[object/thing:639]
```

33.1.2.3 Callstack

The callstack (call list) is the most important tool in order to recognize if the reason behind a system crash is due to an error that occurred in an application function, in a rule/method, in the IDM library or in an external function.

The callstack contains all calls of rules, methods, built-in functions, DM interface functions as well as the application functions called by the IDM. Hereby, the parameter values are displayed. However, in order to minimize the effect on performance it is not possible to carry out a correct string coding in every situation.

In addition, the this-object, the file name in which the rule is found, for ASCII dialogs the start line of the rule as well as a %-output, which delivers the approximate position in the intermediate code of the rule, is also included. For the first rule on the stack the local variables (locvars list) as well as the content of the value stack (valstack), which is necessary for the expression evaluation, is also listed.

```
STACK:
#0: rule D.AfterError, this=dialog D, file=dumping.dlg:68+39%
    locvars=[dialog D, "initvar2", nothing]
    valstack=[100]
#1: rule on dialog start, this=dialog D, file=dumping.dlg:78+35%
#2: DM_StartDialog(dialog D, null)
```

Abridgements

Only the first 20 and the last 20 entries are listed.

Important

If the message *"ATTENTION: not in IDM main thread!"* appears at the beginning of the dumpstate output, extreme care should be taken. The callstack only displays the callstack of the IDM main thread!

33.1.2.4 Events

In this section the actual processed events (THISEVENTS) and the events that are currently waiting in line (EVENT QUEUE) to be processed are listed.

The source indicates the event trigger: **dialog**, **setval**, **destroy** or **external**. This reveals whether this has to do with an event that was triggered by a user or the system, if it was triggered by a change in value of an attribute, if it has to do with the destruction of a module or if it has to do with an external event.

The object, which is the recipient of the event, is also shown. The event queue, the data value and the arguments as well as the specific attributes of THISEVENTS (e.g. .x, .y) are also displayed.

```
THISEVENTS:
#0: source=dialog, object=dialog D, event=start
EVENT QUEUE#1:
#0: source=dialog, object=window D.Wi, event=activate
#1: source=external, object=pushbutton D.Wi.Pb, data=1, args=["EvData"]
```

Abridgements

A maximum of 10 events are listed

33.1.2.5 Usage

This section is made up of three tables:

- » Information concerning the number of IDM objects, their distribution on defaults, models and instances, their total number and their approximate memory need in the IDM core (not the displayed user interface elements). This table is sorted in an ascending order in the columns **alloc** and **count**.
- » Information pertaining to dialogs and modules. The number of all dialogs and modules is displayed. Furthermore, the sum of the object slots contained in the dialogs and modules, the memory that is needed for the slot administration, the complete number of identifiers and the memory needed for this is also shown. In the **<maximum>** line the dialogs and modules with the largest values are listed in the individual columns. This makes the dialogs and modules, upon which no further objects can be created or whose space for identifiers is becoming scarce, easier to identify.
- » The third table is of an informative nature and contains the number and the memory size of the rule frames that are used by the rule interpreter. It also contains information pertaining to the number and memory size of buffers that are necessary for the storage of interim results.

class	defaults	models	instances	count	alloc

thisevclass	0	0	1	1	84
clipboard	0	0	1	1	100
setupclass	0	0	1	1	140
accelerator	1	1	2	4	336
module	0	0	1	1	628
pushbutton	1	0	1	2	752
dialog	0	0	1	1	783
text	1	1	7	9	828
edittext	1	0	1	2	980
window	1	1	1	3	2k
rule	2	3	5	10	19k
<total>	7	6	22	35	25k

module	count	slots	alloc	labels	labelsize	name

--						
dialog	1	27	3k	29	3k	
module	1	8	3k	1	3k	
<maximum>		27	3k	29	3k	dialog D

frames	f-alloc	scratch	s-alloc	values	v-alloc

2	22k	4	4k	69	17k

Abridgements

A maximum of 126 classes are listed. No object class-related allocation sizes are determined (column contains only zeros).

33.1.2.6 Memory

This section contains the size of the allocated heap storage and gives clues about the amount of memory used for the entire application. This functionality is only available under Windows or with an activated IDM memory debugging. The sum of the allocated memory blocks is determined via **HeapWalk()** and for the C-runtime via `_heapwalk`. The heap that is used directly by the IDM is marked with “*” and the default heap of the process is marked with “P”.

MEMORY:

	heap	alloc	other
-	-	-	-
* crt		386k	56k
0x02CC0000		4k	61k
0x02BA0000		5k	167k
0x029C0000		386k	613k
0x02A10000		9k	56k
0x006F0000		7k	999k
0x00340000		12k	52k
0x00630000		11k	53k
P 0x007B0000		137k	797k
<total>		568k	3m

Abridgements

Only the first 20 heaps are displayed.

Note

Please be aware that the heaps also contain IDM-foreign memory allocations that are, for example, used by the application function, system routines or external functions (DLL, In-Process-OLE-Control).

33.1.2.7 Slots

The output of slots serves especially for the recognition of errors in the IDM library, which are related to the referencing, the release and the destruction of objects. A list of object slots, which could not be completely eliminated or that displayed suspiciously high referencing and locking counters, is created.

SLOTS:

slot	class	refs	locks	drop	hull	object
-	-	-	-	-	-	-
[0x0020028]	window	1	1	1	0	window D.WINDOW:[1]

Abridgements

A maximum of 20 slots is listed. All locked slots are displayed in full.

If the start option, built-in or interface function with the parameter *dump_locked* (see chapter “New Built-in Function dumpstate”) is used for the dumpstate output, then additionally all attributes of the locked objects are dumped. Attribute values of resources, rules and functions cannot be dumped.

33.1.2.8 Visible Objects

In this section all of the visible objects are written out including the values of their pre-defined attributes and visible child objects. This section should deliver copies of most of the relevant objects and attributes when display problems occur in order to be able to reproduce errors more quickly.

```
VISIBLE OBJECTS:
window Wi {
  .repos_id "";
  .userdata nothing;
  .visible true;
  .sensitive true;
  .navigable true;
  .fgc null;
  .bgc null;
  .font null;
  :
}
```

In the *dump_full* and *dump_uservisible* mode (see chapter “New Built-in Function dumpstate”) all visible objects that are directly attached under a dialog or a module are written out. All of the pre-defined and user-defined attributes including all of the visible and invisible child objects and child records are written out for these objects.

In principle no resources, rules, methods or functions can be dumped.

Abridgements

Only the top visible objects are dumped, without value/child objects.

33.1.2.9 Example

Constellation

An IDM application started an individual thread via C that crashed after 10 seconds because it was trying to access an invalid storage address. In the thread approximately 5 x 10MB memory is allocated. During this process the IDM is constantly creating windows and records whereby the windows are eventually destroyed, but not the records.

In this case the end of the logfile looks somewhat like the following:

```
FATAL ERROR: Exiting due to exception 0xC0000005 (ACCESS VIOLATION)
*** DUMP STATE BEGIN ***
```

ATTENTION: not in IDM main thread!

PROCESS:

pid=3024, thread=5176, IDM-thread=4684, date=2009-11-16, time=16:58:52

STACK:

#0: create(window, dialog D, true)
#1: rule D.Test ("c-thread-access-violation"), this=dialog D,
file=bad.dlg:78+40%
valstack=["c-thread-access-violation", window, true]
#2: rule on extevent 1 , this=dialog D, file=bad.dlg:112+71%
#3: DM_EventLoop(null)
THISEVENTS:
#0: source=external, object=dialog D, data=1

USAGE:

	class	defaults	models	instances	count	alloc
-----	-----	-----	-----	-----	-----	-----
	record	1	0	24704	24705	0
	window	1	0	1	2	0
	clipboard	0	0	1	1	0
	dialog	0	1	1	2	0
	module	0	0	1	1	0
	setupclass	0	0	1	1	0
	thisevclass	1	0	1	2	0
	accelerator	1	0	3	4	0
	function	0	0	6	6	0
	rule	1	1	5	7	0
	text	1	6	16	23	0
	<total>	6	8	24740	24754	0

module	count	slots	alloc	labels	labelsize	name
-----	-----	-----	-----	-----	-----	-----
--						
dialog	1	24745	99k	28	3k	
module	1	9	3k	1	3k	
<maximum>		24745	99k	28	3k	dialog D

frames	f-alloc	scratch	s-alloc
-----	-----	-----	-----
2	13k	5	5k

VISIBLE OBJECTS:

#0: window D.WiMain

MEMORY:

	heap	alloc	other
-	-	-	-
* crt	11m	3m	
0x030D0000	4k	61k	
0x030E0000	5k	167k	
0x00300000	11k	54k	
0x02EF0000	11m	4m	
0x027A0000	6k	58k	
0x00030000	13k	52k	
P 0x00A50000	48m	791k	
<total>	59m	5m	

*** DUMP STATE END ***

The following can be read from the dumpstate output:

- » The message “*ATTENTION: not in IDM main Thread!*” implies that the crash (**ACCESS VIOLATION**) does not take place in the IDM. For this reason the callstack is, in this situation, unsuspecting. When the crash occurred in another thread the IDM was calling the built-in function **create**.
- » The number of **record** instances in the USAGE section is suspiciously high. This can be seen in the **record** line of the **class** table as well as in the **<maximum>** line of the **module** sub-table. In addition, the enormous storage need is reflected in the **crt** line of the MEMORY table.
- » The MEMORY table shows that an enormous amount of memory (over 50 MB) has been accumulated in the default heap of the process (labeled as “P”).

33.1.3 Noteworthy for Windows/Threads

The IDM manages its own callstack for rule calls, methods, builtin functions, DM interface functions and DM application functions (see also chapter “Callstack”). For the DM interface functions (except for **DM_SendEvent**, **DM_QueueExtEvent**, which **do not** appear on the callstack) a check takes place to see if they are called from the same thread in which the IDM was initialized. If this is not the case, then an error message is sent.

For the most part (with only a few exceptions) the IDM and its interfaces are not designed for use in multi-threading situations. The callstack only serves to manage IDM-specific functions and not for arbitrary application functions.

33.1.4 Safety Tracing

Safety tracing is a special tracefile mode. In order to keep the tracefile running during long application sessions without experiencing a slower running system and the use of too many resources, safety tracing uses a limited ring buffer that is held in the memory. For this use the option **-IDMstracefile <filepath>** instead of **-IDMtracefile <filepath>** or when using **-IDMtracefile <filepath>** switch on safety mode with the option **-IDMstrace**.

With respect to the ring buffer, limited means that the number of lines and the number of characters per line (without indents) is restricted. In addition, the length of the string values (typically displayed in "...") is shortened and the string length is attached in brackets []. The limitations can be influenced via the option **-IDMstraceopts**. For the hierarchical indentation in the tracefile two blank spaces are used. This **cannot** be influenced by the option **-IDMindent**. If the issued lines are discontinuous this is marked by a line and a colon. If the maximum length of the line is exceeded, this is marked with a tilde (~).

The content of the ring buffer is written to the tracefile only after the application has been ended. It is necessary to have an active exception catcher in order to guarantee that files are saved in case of system crashes (see chapter "Exception Catcher").

Switching off the tracing or the output of certain tracecodes is **not possible** in the safety tracing.

The following limitations apply within the safety tracing

	Minimum	Maximum	Standard
Bytes per line	20	65.536	200
Lines	20	100.000	1.000
String length			40
Indent depth		50 (100 white spaces)	

33.1.5 Exception Catcher

During bootstrap the IDM installs a global exception catcher for catchable exceptions (for structured exceptions under MS Windows, signal handler under UNIX).

When an exception appears, the IDM reports a FATAL ERROR together with the exception number. If necessary and if set accordingly, the dumpstate and safety tracefiles are written.

When setting up own exception handlers please ensure that these pass on an exception.

33.1.6 New Start Options

33.1.6.1 -IDMdumppstate <enum>

This option influences the output of IDM status information (dumpstate). Via the <enum> parameter the output of certain information is forced (see chapter "New Built-in Function dumpstate").

33.1.6.2 -IDMdumppstateseverity <string>

The output of a dumpstate normally occurs when an EVAL ERROR or a FATAL ERROR arises. With this option one is able to force a dumpstate output for other types of errors and messages.

Parameter

First Character	Dumpstate-Output...
E	in case of normal error messages [E: ...]
F	in case of fatal error messages [F: ...]
I	in case of informative messages [I: ...], warnings and error messages
O	only in case of EVAL ERROR
W	by warnings [W: ...] and error messages

Default setting: *F*

33.1.6.3 -IDMcallstack <boolean>

The administration of the callstack, which saves the call list of rules, methods, built-ins, application and interface functions and which is in turn written out by the dumpstate, can be prevented with this option.

Default setting: *true*

33.1.6.4 -IDMcatchexceptions <boolean>

With this option one is able to prevent an exception catcher from being set up.

Default setting: *true*

33.1.6.5 -IDMstrace

The tracefile is used in the safety mode. In addition, the tracing must be activated via the **-IDMtracefile <filepath>** option.

33.1.6.6 -IDMstracefile <filepath>

This option is the short form for the combination of the options **-IDMtracefile <filepath>** for activating the tracing and **-IDMstrace** for setting the safety mode.

33.1.6.7 -IDMstraceopts <string>

This option activates the safety tracing and at the same time defines the setting for this.

The string parameter influences the setting as follows:

Template	Setting
c<integer>	Bytes per line
h	Hierarchical output: Retention of as many hierarchical levels as possible
l<integer>	Number of lines
r	Rotating output (standard): The oldest line is replaced by the newest
s<integer>	Length limit for strings

Through concatenation it is possible to carry out numerous settings at the same time. The order is arbitrary.

Instead of this option the environment variable `IDM_STRACEOPTS` can be used.

Example

A safety tracing with 300 lines and 80 bytes per line is activated via the option **-IDMstraceopts l300c80**.

33.1.7 New Built-in Function `dumpstate`

IDM status information (`dumpstate`, see also chapters “Dumpstate Output” and “Dumpstate”) is written out to the logfile or tracefile with this function.

Syntax

```
void dumpstate(
    enum State input
)
```

Parameter

`enum State input`

This parameter influences which sections of the status information is written out.

The following values are possible

Value (enum)	Meaning
<code>dump_all</code>	All sections are written out in an abbreviated form. This corresponds to the output in case of a FATAL ERROR.
<code>dump_error</code>	The sections ERRORS, CALLSTACK and EVENTS are written out in an abbreviated form. This is the normal output in the case of EVAL ERRORS.

Value (enum)	Meaning
<i>dump_events</i>	The section THISEVENT/EVENT QUEUE is written out in full.
<i>dump_full</i>	All sections are written out in full.
<i>dump_locked</i>	The section SLOTS is written out in full. In addition, for locked objects their attribute values are written out.
<i>dump_memory</i>	The section MEMORY is written out in full.
<i>dump_none</i>	No action (nothing is written out)
<i>dump_process</i>	The section PROCESS is written out in full.
<i>dump_short</i>	All sections (excluding SLOTS) are written out in an abbreviated form.
<i>dump_slots</i>	The section SLOTS is written out in full.
<i>dump_stack</i>	The section CALLSTACK is written out in full.
<i>dump_usage</i>	The section USAGE is written out in full.
<i>dump_user-visible</i>	The section VISIBLE OBJECTS is written out in full for all visible top-level objects including their children, the pre-defined and user-defined attributes.
<i>dump_visible</i>	The section VISIBLE OBJECTS is completely written out.

Example

```

dialog D
window Wi
{
    .title "dumpstate()-example";
    edittext Et
    {
        .content "66";
        .xauto 0;
        on deselect_enter
        {
            variable string Content := this.content;
            if fail(Pg.curvalue := atoi(Content)) then
                print "Conversion error!";
        }
    }
}

```

```

        dumpstate(dump_error);
    endif
}
}
pushbutton Pb
{
    .ytop 33;
    .text "dump objects";
    on select
    {
        dumpstate(dump_visible);
    }
}
progressbar Pg
{
    .yauto -1;
    .xauto 0;
}
on close { exit(); }
}

```

33.1.8 New Interface Function DM_DumpState

IDM status information (dumpstate, see also chapters “Dumpstate Output” and “Dumpstate”) is written out to the logfile or tracefile with this function.

Syntax

```

void DML_default DM_EXPORT DM_DumpState
(
    DM_Enum state,
    DM_Options options
)

```

Parameters

-> DM_Enum state

This parameter influences which sections of the status information are written out. Possible values are listed in chapter “New Built-in Function dumpstate”. A combination of multiple sections is not possible.

-> DM_Options options

This parameter is reserved for future versions. Until further notice, please only enter 0.

34 Version A.05.01.g2

34.1 Network

- » Gnats 10264: Due to deterioration in performance on the application side (compared to the Dialog Manager Version A.04.02) the routines for string conversions have been revised and improved.

35 Version A.05.01.g

35.1 COBOL

- » Gnats 10164: A problem that arose during the transformation of COBOL character strings into C character strings has been eliminated. This problem was noticed in that the interface function `DMcob_LSetVectorValueBuff` considerably slowed down in speed. This was especially true when dealing with larger data fields.

35.2 Editor

- » Gnats 10153: Named rules in an editable sub-module are now again written out when saved.
- » Gnats 10152: An editor crash that occurred in connection with the processing of named rules (appearance, change) has been eliminated.

36 Version A.05.01.f

36.1 Important Clarification

- » Clarification concerning the object monitors and text changes within the release notes for version A.05.01.d.

New Text

Additional changes when using "YiRegisterUserEventMonitor":

The administration of the monitors "YI_OBJ_MONITOR" or rather "YI_OBJFRAME_MONITOR" has been converted internally.

This leads to window messages, that are sent while creating an object (i.e. WM_CREATE, ...) being no longer made available to the monitor "YI_OBJFRAME_MONITOR".

In order for a monitor function to be able to recognize that an additional IDM object has been created, then the message "IDM_HWND_SUBCLASSED" is delivered after the "sub-classing". The message parameter "IParam" contains the Windows Handle (HWND) of the IDM object (identical to the **GetToolkitData** via *AT_WinHandle* that can be called up via *Handle*). Thus, in combined IDM objects now one can see if a subordinate window object has been created. Note that this parameter can also be 0.

36.2 Windows NT/Windows XP

- » New: The **setup** object now supports access to Windows clipboard (interim storage area). For further information refer to the manual "C Interface - Functions", chapter "DM_GetToolkitData" or chapter "DM_SetToolkitData".
- » Gnats 10131: The sensitive state of a **scrollbar** object (especially on a **tablefield** object) is now again displayed correctly.
These problems occurred on the **tablefield** object while filling in an invisible state and in active "Visual Styles" when the cursor on the scrollbar was activated and the object was in a created, but invisible state. This condition can arise either when the attribute *.mapped* is set to false on the object in question (or on a father object) or when the object in question is situated in an inactive notepage object.
- » Gnats 10121: The cursor is no longer invisible when the list of a **poptext** object is opened by using the keyboard (Workaround the Microsoft Bug).

Note

When the list of a **poptext** object is opened, the cursor is not changed via the **poptext** object. In particular, the cursor is no longer hidden when the editbox is worked on. This is because this is a Microsoft Windows object (Windows XP SP2) and thus can change again in future Windows versions.

- » Gnats 10122: A rectangle object with raster coordinates within a window no longer flickers when the size or position of the window is changed.

- » Gnats 10120: On a splitbox object without borders (*.borderwidth = 0*), the cursor is only changed within the “splitbar” and not one pixel next to it. As a result of this the user no longer experiences (non-triggered) shifting problems.
- » Gnats 10118: A **toolhelp** object is now again displayed with borders even when on the **setup** object the option is set to *.options[opt_balloon_toolhelp] false*.
- » Gnats 10114: Within active “Visual Styles” the lower or rather right border on the **tablefield** object, by non-displayed scrollbars, is now again displayed in the correct color.
- » Gnats 10109: An opened **toolhelp** on a **poptext** object no longer flickers when the list of the **poptext** object is opened.
- » Gnats 10108: The value of *.real_size[l]* on a splitbox object is now now again returned to the correct area.
- » Gnats 10106: Changing the attribute *.edittext* on a visible **tablefield** object no longer leads to a system crash.
- » Gnats 10084: A **poptext** containing text of more than 256 characters no longer leads to a program crash.
- » Gnats 10077: When a mouse click incidentally hits a **toolhelp** object that is being opened or is already open, it no longer disappears.
- » Gnats 10076: The closing and re-opening of an already opened context menu by the user no longer leads to the context menu only being partially displayed. In addition, no further “*ASSERTION failed*” messages are recorded in the tracefile in this case.
- » Gnats 10075: The display problem in the tablefield object when scrolling too quickly with the mouse wheel has been eliminated.
- » Gnats 10073: When the Dialog Manager libraries were dynamically connected, a crash of the system was possible. Earlier, the release of an OLE error message after an OLE method was called up or after accessing an OLE characteristic was not possible. This problem has been corrected.
- » Gnats 10071: A **toolhelp** object that is opened and then immediately closed reappears when the object in question is touched with the cursor.
- » Gnats 10070: A **splitbox** object with a set size raster (*.sizeraster true*) is now considered by an interactive shifting of the “splitbar”. Thus, a “jumping away” of the “splitbar” no longer happens when the mouse is released.
- » Gnats 10060: The Dialog Manger for the Windows XP platform displays **toolhelps** now again in the operating system Windows 2000 (and older).
- » Gnats 10041: MDI child windows (when *.sizeable = false*) are no longer automatically reduced when the window size of the father is reduced.
Further notes: Maximal window sizes are found in chapter “Window” in the “Object Reference”.
- » Gnats 10028: When a dialogbox (window object where *.dialogbox = true*), a messagebox or a file-requester object is opened, then the window that is opened last is declared as the “owner”. In the case that the application possesses additional windows, then these are seen as able to be activated despite the opened dialogbox or the opened messagebox or filerequester objects.

Note

Under Microsoft Windows it cannot be avoided that a window displayed in the task bar is able to be activated. If at this point in time a window object with the attribute *.dialogbox = true* (dialogbox) or a messagebox or a filerequester object is open, then at first an erroneous object is activated. Shortly afterward the correct object, i.e. the dialogbox, the messagebox or the filerequester is activated.

The following limitations are true:

In certain cases the Dialog Manager must heuristically search for the correct object. In order to avoid the wrong object being opened, please avoid the following:

- » Several messageboxes or filerequesters being open at the same time. Top level objects that were not created by the Dialog Manager can also cause problems.
- » When the objects are closed the order of sequence is not taken into account (the last one opened is the first to be closed).
- » After a modal object is opened the attributes *.sensitive*, *.mapped*, *.iconic* or *.visible* on the top level window application are changed.
- » Gnats 10027: Emerging from the Rule Language the *.content* attribute on the poptext object was not able to be changed when at the same time an object monitor had called up the attribute *.activeitem*. The cause of this was an internal puffer that was written over with older content when the object monitor had called up the attribute *.activeitem*. Now static puffers are no longer used which means that the chance of the monitor function causing this error has been reduced.
- » Gnats 10025: When the attribute *.topitem* possesses the value *1* and the entry item, that is situated outside of the actual visible area (long list), is activated, then an incorrect entry is now longer displayed at the top when the listbox object contains *.selstyle <> single*.
- » New (Gnats 9950): Can now take place under the object. In this case the attribute *.options[opt_center_toolhelp] = true* (default value is false) needs to be set on the desired objects. The toolhelp display does not appear at the cursor but rather centered directly under the object as long as there is enough room provided.

36.3 Motif

- » Gnats 10038: The determination of the charset encoding of a **font** resource is now carried out case insensitive.

36.4 Core

- » Information for the automatic positioning of toolbars:
There are small differences to the earlier versions regarding the size and position of toolbars when they are automatically positioned.
As is described in the manual, no splitters (separators between the toolbars) are considered for the last toolbar in a line or a column during the positioning process. Other small differences in the

size of the unused remaining window-toolbar area can also arise when the necessary *sizeraster* of all the toolbars are taken into consideration

- » Gnats 10123: In toolbars that use a size raster an endless loop, that occurred when attribute *.auto-size true* was set, no longer happens.
- » Gnats 10105: Correction of memory allocation through *pass* and *fail* to fix a system crash.
- » Gnats 10069: Certain heredity constellations that occurred when discharging or stopping a module sometimes lead to a crash. This is no longer the case.
- » Gnats 10045: When the design is given in the **select** method of the **doccursor** object in XPath syntax, this no longer causes a crash.
- » Gnats 9132: The conversion of the *.content* on a poptext object (combobox) no longer causes the sending of an *.activeitem changed* event.

36.5 Editor

- » Gnats 10068: When loading, reloading or closing a file the editor interface is completely disabled up to the complete actualization of the editor interface.
- » Gnats 10040: Under MS Windows multiple line texts (i.e. rules from the rule area of the editor) are completely transferred to the clipboard and not partially cut off.
- » Eliminated: A newly loaded module or newly loaded dialogs are no longer marked as changed.
- » Eliminated: A cursor click on the spinbox is now observed under Microsoft Windows.
- » Eliminated: The pasting action within a rule area after having carried out a cut or copy action triggers the activation of the post/discard pushbutton. The pasted text is marked as changed.
- » Eliminated: A cut or paste action now functions via the menu and the toolbar.

37 Version A.05.01.e2

37.1 Windows NT / Windows XP

- » Gnats 10075: The display problem that arose in the tablefield object when scrolling too quickly with the mouse wheel has been eliminated.
- » Gnats 10073: When the Dialog Manager libraries were dynamically connected, a crash of the system was possible. Earlier, the release of an OLE error message after an OLE method was called up or after accessing an OLE characteristic was not possible. This problem has been corrected.

38 Version A.05.01.e

38.1 Important Modifications

- » In Microsoft Windows the setting of the focus while opening a context menu was changed.
- » The checkbox object has a new attribute *.navigation*. With this attribute a better tabulator navigation could be performed.
- » In Microsoft Windows the functions **DM_Execute** or **execute()** support now the values *DM_EXE_maxwindow* or *maxwindow* in the *windowtype* parameter.
- » The Editor was expended the possibility to define own tools and a file modification check (message, if a opened file was changed from another program).
- » The Dialog Manager for the Windows XP platform no longer supports Window 95. This step was necessary so that Dialog Manager for Windows XP (32 bit) could be started in WOW64 under MS Windows XP 64bit.
- » Added support for OpenMotif 2.3.0 to be able to use UTF-8 locales in Linux (z.B. RHEL).
- » The restriction of older IDM versions not to use menus in dialogboxes in Motif exists in the version 5 no longer. The display is like Microsoft Windows now.

38.2 New Features of the IDM Editor

Now there is the possibility to define own tools in the IDM Editor. Furthermore a message if opened files were changed from another program is possible.

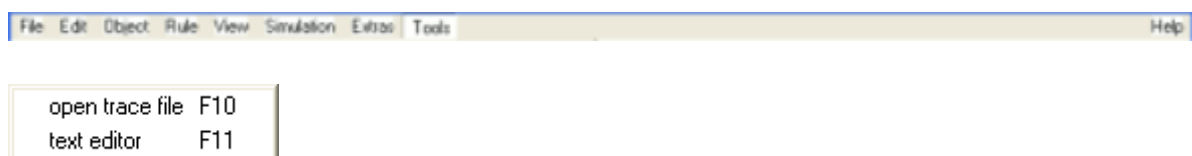
Remark

This is not available in Dialog Manager for VMS.

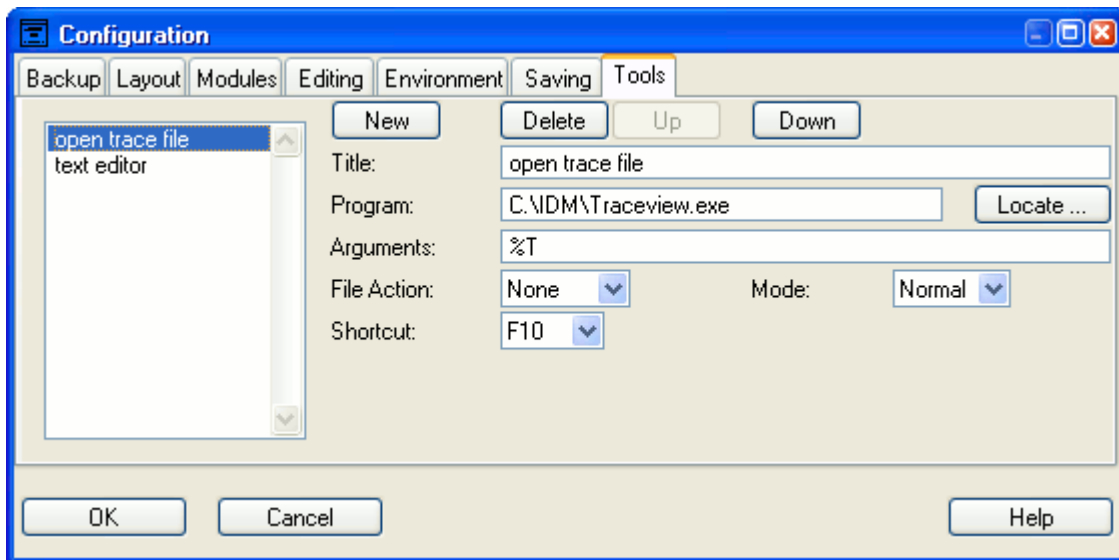
38.2.1 Tools

The Editor now allows the definition of own tools, for example for opening a external text editor or to work with the opened file in another program.

As soon as external tools are defined there would be a menu tools in the main window. There you can select your tools.



With the buttons “New” and “Delete” tools could be created or destroyed. With the buttons “Up” and “Down” the tools can be sorted in that way they should appear in the tools menu of the main window.



A tool always needs

- » *“Title”*: the title displayed in the tools menu
- » *“Program”*: the path to the program
- » *“Arguments”*: the arguments used by the program

The following placeholders are valid for tool arguments. They relate to the actual edited file.

- %P Pathname (drive and directory, without file name) flush with a “/” or “\”
- %B Base file name without ending (without path)
- %F File name with ending (without path)
- %N Identifier of the dialog/module
- %D Identifier of the dialog/module shortened to 8 characters
- %T Name of the trace file how it is used in the simulation parameters

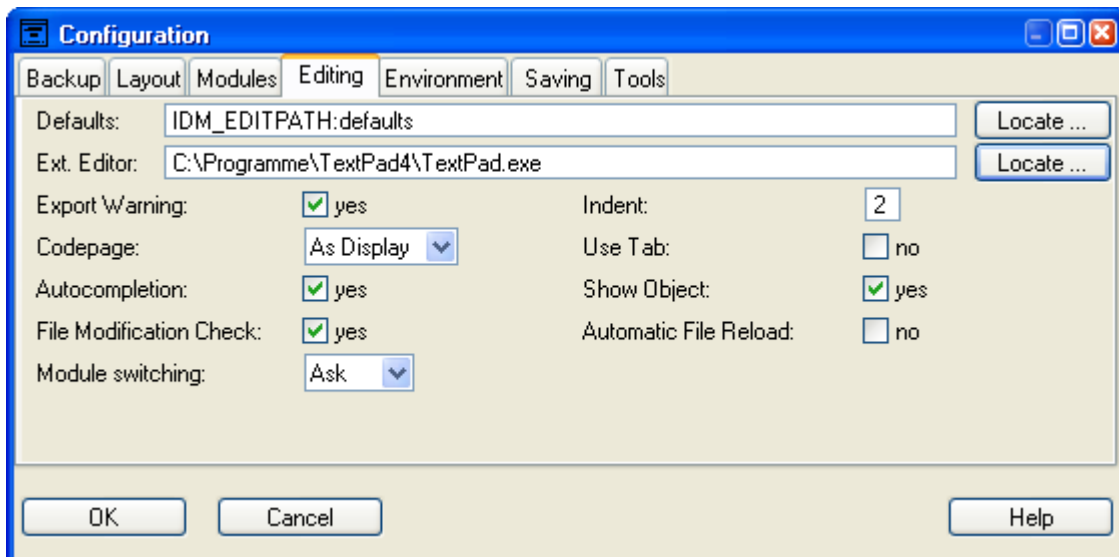
With the point *“File Action”* there is a file action before the start of the tool configurable. It is possible to save the actual edited file or all modified files in the IDM Editor here if necessary.

Via the *“Mode”* selection the sometimes necessary differentiation between Windows and Shell programs in MS Windows could be done.

In the Editor for tools the function keys **F9** ... **F12** are valid as *“Shortcut”* for tools. So the tool can be selected by key.

38.2.2 File Modification Check

The settings for the file modification check would be done in the configuration window of the IDM Editor, Editing tab.



“File Modification Check” switches the periodical changes check of open files on and off.

“Automatic File Reload” switches the request if the file should be reloaded on and off (if there are changes in the IDM Editor there would be always a dialogbox request).

38.3 New Attribute .navigation

This attribute is only available at the checkbox object and regulates the modality of the key navigation (Tab and cursor keys) and the association to a Tab group.

38.3.1 Attribute .navigation

Identifier:

.navigation

Classification: object-specific attribute

Definition

Argument type: enum

C definition: AT_navigation

C data type: DT_enum

COBOL definition: AT-navigation

COBOL data type: DT-enum

Access: set, get

This attribute defines the type of key navigation at the checkbox object. This navigation differences objects navigated by using the **Tab** key and object groups (e.g. several radiobuttons or checkboxes) were the navigation is done with the cursor keys.

Valid Values

<i>navi_default</i>	Present behavior (default)
<i>navi_grouped</i>	Composes a group with other objects also set to <i>navi_grouped</i> . Within this group the key navigation is done with cursor keys
<i>navi_tab</i>	The object stands alone and could be accessed with the Tab key.

It should be made sure that all objects of a group, that means all object inside a grouping object (e.g. **groupbox**, **window**), have the same value.

38.4 Windows NT / Windows XP

- » Gnats 10036: For the display time of the *.toolhelp* attribute the Dialog Manager furthermore uses the standard settings of Microsoft Windows. Some other applications, including Microsoft Office, are displaying the tooltip for a longer time but there is no advice in the Style guides what displaying times making sense and there is no system setting for it.
If there is a longer displaying time wished so this could be through a C function. See Gnats 9945.
- » Gnats 10033: A mnemonic is now deactivated if a text of an object is changed to a string without mnemonics.
- » Gnats 10023: At activating an accelerator the object on which the accelerator is defined would be focused.
The mistake was noticed through a lost focus change to a default button and therefore the not shipped edittext object *deselect* or *modified* event.
- » Gnats 10015: If a dialogbox (window with *.dialogbox = true*) without a owner window was closed no other window of the application was activated. Now another window of the application was randomly chosen. The problem could always occur if dialogboxes weren't nested incorrectly or they were opened while no other window of the application was visible.
To work well dialogboxes need another window of the application as owner. For this reason always another visible and sensitive application window should exist at the time opening a dialogbox. This window not should be closed before closing the dialogbox. Also the dialogboxes should be nested correctly, that means the dialogbox opened at last must be closed first. If this rules are not noticed it could be happen htat application windows are not activated automatically or dialogboxes are displayed behind a application window.

Recommendation

If windows always be opened through user actions and the closing of the window is initiated through a user action on this window the above shown rules are not targeted.

- » Gnats 10013: A window fastened up and down (*.yauto = 0*) or left and right (*.xauto = 0*) with coordinates given in raster values (*.posraster = true*), were the raster value is set to 1 or is undefined, the coordinates wouldn't changed any more during visualization. Also such a window keeps his size while its moved.
- » Gnats 10011: The size of the working area of an visible and maximized window object is adjusted again if a statusbar object is switched to visible or invisible. If the window is not maximized the size of the working area is leaved unchanged.
- » Gnats 10010: Deleting the marked area in a formatted edittext object with cursor movement leaving the cursor position unchanged rules no longer to a beep. This matches the behavior of previous Dialog Manager versions.
- » Gnats 10003: An edittext object with the option *.options[rtf] = true* and RTF string in the *.content* attribute now shows the whole text again and not only the first character.
- » Gnats 10002: In Microsoft Windows the functions **DM_Execute** or **execute()** support now the values *DM_EXE_maxwindow* or *maxwindow* in the *windowtype* parameter. With this the specified application could be started maximized.

Attention

The started application have not to note this parameter.

If the *exetype* parameter has the value *DM_EXE_shell* or *exeshell* the command interpreter would be started maximized and not the application started through the command interpreter.

- » Gnats 1001: The focus is now set to the before focused object of a toolbar object again if in the meantime another window was activated and then the primary window with the toolbar object was activated again.
- » Gnats 9995: In Microsoft Windows with "Visual Styles" child objects of a toolbar object were positioned in a wrong way if during uncovering the parent window the toolbar object was resized automatically. This behavior was only observed at right and down tethered child objects.
- » Gnats 9993: In a layoutbox or a splitbox object child object set to *.mapped = false* are no longer lead to empty areas.
- » Gnats 9992: A maximized window is displayed completely (with its border) after the user has locked and unlocked the computer.
On taskbar setting changes in Microsoft Window windows could be moved in a way that they are displayed completely. The same behavior could be observed if a the user has locked and unlocked the computer. This is a Microsoft Windows general behavior and could also observed with other applications (MS Internet Explorer, Comand Prompt (DOS box), MS Outlook, Notepad ...). Even a maximized window with limited size settings is treated in that way and so it would be moved.
- » Gnats 9991: The drawing mistake at the lower border of a notepage object which could be observed setting the object to visibility is corrected.
- » Gnats 9990: Partial smaller fonts as in the version 4 of the IDM were chosen. This was ascribed to rounding problems (only some fonts in some sizes were effected) and is corrected.

- » Gnats 9988: A scrollbar object without arrows (*.arrows = false*) now creates scroll events again if the slider is moved.
- » Gnats 9887: No error message is written anymore in the log file setting the focus to a visible but insensitive tablefield object (*.focus := [I,J]* or *.focus[I,J] := true*).
- » Gnats 9986: The selection in a tablefield object with set *.selstyle = extended*) the selection could be expanded with **Shift** + “cursor keys” now.
- » Gnats 9981 and 9820: Doing a Drag&Drop operation or opening a context menu at a treeview object loosen the mouse button in an area without items or clicking in this area an index of 1 instead of **no index** was delivered misleadingly in the *paste* or *open* event.
In Windows XP please notice that an treeview item needs nearly the whole width even only a short text is displayed on the left hand side.
- » Gnats 9979: a scrollbar object without arrows (*.arrows = false*) created insensitive and then changed to sensitive can be operated again.
- » Gnats 9970: The slowed down filling of a tablefield object if compared with IDM version 4 is corrected. Especially the tablefield object is no longer redrawn if the content of a cell beyond the visible area is changed.
- » Gnats 9967: If a radiobutton object was set to *.active = true* in its insensitive state (creating the object as well) and afterwards was changed to sensitive (*.sensitive = true*) then at key navigation (**Tab** key) the first radiobutton object was activated.
The same problematic occurs on activating a notepage object with the mouse too.

Remark

In Microsoft Windows the focused radiobutton object is automatically activated. Therefore avoid setting the active radiobutton object insensitive.

Attention

Should the active radiobutton object insensitive the focus is set to the next radiobutton object and thereby this object is activated.

- » Gnats 9965: The toolhelp is now also displayed by moving the mouse over a inactive window.
- » Gnats 9963: Till now the focus border in a cell of a tablefield object was drawn over the text. This is no longer desired. Now the focus border is drawn at the cells border. To leave enough space for the focus border the attributes *.xmargin[l]* and *.ymargin[l]* could be set to 2 or greater.
- » Gnats 9960: In a statictext object with centred text (*.alignment = 0*) the right side of the focus border is now again displayed in the right way. Furthermore the focus border now is drawn around the text and not mistakenly over the text.
- » Gnats 9958: If the focus object was changed directly after closing a dialogbox there could occur a runtime error. Thru this the focus object change are not done and the object which has had the focus before opening the dialogbox gets the focus again (or kept it). The problem occurs especially by using **querybox()** and MDI windows and it is corrected.

- » Gnats 9953: An edittext dedicated to a tablefield object using *.editpos = true* are now treated the right way from a layoutbox object or a splitbox object. The edittext does not longer occur as own object or as empty split area.
- » Gnats 9945: The function **DM_GetToolkitData** of the setup object was extended about the attribute *.toolhelp*.
- » Gnats 9934: The Microsoft **test container for ActiveX Controls** crashes including a OLE control with *.mode = mode_server*. This is an error in the Microsoft test container.
- » Gnats 9925: The *.alignment* attribute could be changed at one lined edittext objects again. At RTF edittext (*.options[opt_rtf] = true*) the text alignment reverts to the RTF formatting and not to the attribute *.alignment*.
- » Gnats 9921: New attribute *.navigation* for the checkbox object improved (see chapter “New Attribute *.navigation*”).
- » Gnats 9920: The object at which a context menu is defined would now get the focus if the context menu is opened. If this object is a grouping object (e.g. window, groupbox) the focus is set to the first child object which can get the focus. Indeed the focus will only be changed if neither a direct or indirect child object nor the object the context menu was defined has had the focus before.
- » Gnats 9900: Setting a notebook object visible without explicit set active notepage child now again the content of the notepage object with active tab is displayed and not the content of the initial visible notepage child.

Take Care of This

In a notebook object always exactly one notepage child is active (*.active = true*). As long as the notebook object is invisible (*.real_visible = false*) there is either the first notepage child or the last active notepage child.

Once the notebook object is visible only the also visible notepage child had to set active. If on the other side a invisible notepage child of a visible notebook object is set active (*Np.active := true*) this setting is lost. Under special circumstances it could be that an error (*.active* had not to change to false) for a visible notepage child is written in the trace file.

Before setting a notebook object visible first the active notepage child had to set visible (*Np.visible := true*) and it only one notepage child had to be active. If there is no active notepage child at the time setting the notebook object visible the behavior is undefined and could be different between different versions and patches; any visible notepage child could be activated.

- » Gnats 9886: In a multiline RTF edittext (*.options[opt_rtf] = true*) *.startsel* and *.endsel* are displayed in the right way now and not sometimes displayed shifted.
- » Gnats 9882: Through a correction of an error the calculation of the preferred size and position has changed. Though this had caused problems with existence dialogs the compatibility to the Dialog Manager version 4 is established again.
- » Gnats 9869: With active “Visual Styles” and very much nested Grouping objects there was a bad performance (delay) in redrawing of inner objects. If these inner objects had a background color then additionally a flickering could be observed. Both problems are corrected.

- » Gnats 9849: No pure shell commands could be processed via **DM_Execute** or **execute()** because the command was always embedded in double quotation marks, if the parameter has had spaces but no double quotation marks. Additionally the value of the *ExeType* parameter had to be *DM_EXE_normal / exenormal* or *DM_EXE_shell / exeshell*.
Furthermore now commands build from *command* and *args* parameters are automatically embedded in quotation marks if the *command* parameter has begun with double quotation marks (given or automatically added) and the *exetype* parameter has had the value *DM_EXE_shell / exeshell*. If this behavior isn't favored the *command* parameter could be left empty (*null* or *""*) and the command which should be executed (in this case its forced) given thru the *arguments* parameter.
- » Gnats 9839: The value of the spinbox objects *.activeitem* attribute is now up to date if it is requested from his child's on *charinput* event rule.
- » Gnats 9833: The background color settings at checkbox or radiobutton in Windows XP are now considered.
- » Gnats 9808: At a picture displayed in a poptext object the lower offset was greater than the upper offset. This is corrected. Indeed the rectangle defined through *.picwidth* and *.picheight* further on is fixed at the left upper corner to ensure compatibility to the other Dialog Manager objects.
- » Gnats 9793: Calling the Rule Language **execute()** function no longer a error message is written in the trace file if the *ExeType* parameter is used correctly.
- » Gnats 9780: The setting of a cursor at the toolbar object effects the grid area again.
- » Gnats 9767: In grouping objects some problems with the background pattern (*.tile* attribute) were corrected:
 - » A lucent background pattern has looked odd. This problem exists no longer because the background of grouping objects now lucent by definition
 - » A centered (*.tilestyle = tile_centered*) or stretched (*.tilestyle = tile_strechted*) background pattern was only displayed once in the splitbox object. Now it is displayed in each split area once.
- » Gnats 9725: An error probably based at the existence of a splitbox object in a self defined dialogbox could, since Version A.04.04.g, no longer reproduced.
- » Gnats 9686: A window opened while a *topmost* set dialogbox was opened was also set to *topmost*. This was already corrected in A.04.04.k.
- » Gnats 9674: If a splitbox object was set the second time visible there could be a "ASSERTION FAILED" message in the log file. This behavior was never noticed since A.04.04.g
- » Gnats 9657: At an RTF edittext (edittext with *.options[opt_rtf] = true*) the method **:setformat()** is setting the right typeface again.
- » Gnats 9594: The not from toolbar object covered area of the toolbar area of a window was in Windows XP drawn darker then the toolbar objects. The menu line or the window background. The problem was using the system color to draw the background of 3D objects. In Windows 2000 this color is the same as the color for windows backgrounds which was used for the toolbar area. In Windows XP the system color for the windows background is defined darker than in Windows

2000 so the problem occurs. Now the system color for menu lines are used as toolbar area background color.

- » Gnats 9445: The *.fgc* settings at the treeview object are observed again since the correction to Gnats 9673.
- » Gnats 9428: There are no longer flickering of a mini window using RTF methods at invisible RTF edittext objects (edittext with *.options[opt_rtf] = true*).
- » Gnats 9424: The text in the edit field of a poptext object (*.style <> poptext*) are no longer placed further right than in a discrete edittext object.
- » Gnats 9394: The IDMMuser.h now contains key words for the used calling convention of the IDM interface functions. Through this no **unresolved externals** occur compiling with an other calling convention using a command line option.
- » Gnats 9366: The implemented **beep()** function now allows the setting of tone pitch and tone duration.
In Microsoft Windows the implemented **beep()** function are now supports three parameters: **beep (volume, duration, frequency)**; although the parameter *volume* is ignored. The parameter *frequency* has a duration from 37 ... 32767.
In Microsoft Windows 95, 98 and ME the output is only a standard tone.
- » Gnats 9217: Since a basic change in the tile resource management, done in A.04.03.c, a crash during displaying a animated GIF picture in a listbox object does no longer occur.
- » Gnats 8773: Setting *.style = fr_directory* and *.changedir = true* at an filerequester object the choice was restricted to the directory which was caring the chosen directory. Through this after a change in the subdirectory there could be not gone back to the origin directory.

Change

The filerequester *.directory* attribute only restricts the choice in modus *.style = fr_directory* only if the attribute *.changedir* has the value *false*. Otherwise (*.changedir = true*) the choice is unrestricted and the directory set in the *.directory* attribute is the default selection.

- » Gnats 8584: The filerequester is opened modally again even if the attribute *.style* has the value *fr_directory*.
- » Changed: The Dialog Manager for Windows XP platforms no longer supports Windows 95. This was necessary for the reason Dialog Manager for Windows XP (32 bit) could start in WOW64 in Windows XP 64bit.
- » Corrected: If the working area of a window object could be to small or negative due to defining the distances to the borders in a wrong way and the window was fixed up and down (*.yauto = 0*) or left and right (*.xauto = 0*) there could be a crash. This is corrected.
- » Corrected: A memory leak occurring during cutting a text string from a formatted edittext object was corrected.
- » Corrected: changes in the tablefield object now only lead to redrawing the necessary cells and no longer to redrawing the whole content.

- » Corrected: The calculation of *.real_height* and *.real_width* of the statusbar object with *.borderwidth* < 0 are now again alright.
- » Corrected: The rectangle object coordinates are actualized during *.borderwidth* changes again.
- » Corrected: There could be a flickering at unnecessary actualizing of the objects layoutbox, notebook, spinbox, splitbox and window. Now the actualizing only occurs if a direct child object was changed.

38.5 Core

- » Gnats 10044: When using return, pass or throw previously allocated dynamic memory for expression evaluation is now correctly released.
- » Gnats 10024: Exporting of binary data which draw modules via 'export import' constructs is performed correctly again and no longer causes any 'Name of import must be unique' error.
- » Gnats 10018: With !! comments which are placed before attributes, the Dialog Manager does not distinguish between attribute declaration and default value allocation, as these are in one line anyhow when they are output.
In contrast to predecessor versions, !! comments are now merged for the same attribute, so that e.g. all comments of multiple placements as well as separate declaration and default value placements are preserved with their last value.
Caution should be taken with comments on shadow attributes. These comments can only be obtained for the attribute declaration, as the actual value is usually stored on another object.
- » Gnats 10012: The binary export of models and instances of the document and transformer objects is now performed in the correct sequence. The models of the mapping objects, which occur in the same module as the instances inferred from them,, can therefore be used once again.
- » Gnats 9984: A dialogbox, which had been opened with the built-in function querybox() could not be closed with the built-in function closequery(), if closequery() was invoked within an on extevent or on <Attribute> changed rule. The dialogbox was only closed after a dialog event. This behavior has been remedied.
- » Gnats 9975: Child objects with *.posraster = true* were positioned displaced in the layoutbox object. This lead to the overlapping of child objects if the size of the child was smaller than the position grid. In the described case, the child objects did not have a height (*.height = 0*) and the preferred height was smaller than the grid height. As the layoutbox object determines the position of the child object in pixels, the *.posraster* of the child objects is set to false in a layoutbox.
A grid layout within the layoutbox object is not supported.
- » Gnats 9921: The checkbox object is extended with the *.navigation* attribute (see chapter "New Attribute *.navigation*").
- » Gnats 9702: The subcontrol object can now also use the hierarchic attributes (*.window*, *.control*, *.notepage*, *.layoutbox*, *.toolbar*, *groupbox*).
- » Gnats 8641: When attempting to visualize an object (here: Edittext) directly beneath a module (i.e. without window!) no assfail occurs any longer.

- » Corrected: The virtual size of a layout object with activated size grid is calculated properly again.
- » Corrected: **DM_CreateObject** now accepts *NULL* as father object when creating a dialog object.

38.6 Editor

- » Gnats 10018: The problem concerning lost comments for user-defined attributes is now remedied. Please also see chapter “Core”, [Gnats 10018](#) (detail description).
- » Gnats 10017: An endless loop and the resultant inoperability of the editor when selecting rules or methods in the object browser has been remedied.
- » Gnats 10009: A double listing of menu child objects in a menubox in the 'Arrange children' dialog has been remedied.
- » Gnats 10008: A duplication of objects no longer forces the entry of a descriptor, but only offers this possibility.
- » Gnats 10007: If file tracking is deactivated, files changes following the appearance of the 'reload' question dialog do not result in any new questions.
- » Gnats 10006: When processing a window with *.dialogbox = true* the attribute is no longer switched to *false*.
- » Gnats 10004: The crash that occurs when allocating a tile resource to a notepage model has been remedied.
- » Gnats 10000: After closing an error message to a rule that has just been created and completed with 'assign', focus (and therefore also the mouse cursor) is once again in the rules area (editing field) of the properties toolbar.
- » Gnats 9999: Under Windows the editor crashed, when an object was dragged with the mouse and the mouse hit the arrows of a notebook object that has many, partly hidden pages. This has been remedied.
- » Gnats 9996: The pushbutton for the external editor ('ext. Editor') in the rules area of the properties toolbar is now correctly set to sensitive (if the configuration of an external editor has been entered). The same applies to the update of the menu “Rule” -> → “Buffer”.
- » Gnats 9994: The 'assign'/'cancel' pushbuttons in the rules area of the properties toolbar are now once again correctly updated when changing the rule buffer. Now rule changes can once again be conducted to two rules simultaneously.
- » Gnats 9983: The editor was expanded with the option to define own tools and track files (a message is returned if an opened file has been changed externally), see also chapter “New Features of the IDM Editor” (not for the Dialog Manager under VMS).
- » Gnats 9859: Editor support for notebook/notepage objects under Microsoft Windows has been improved. Now a notepage object can be selected with **Ctrl** + “Click” on the tab and the properties toolbar can be opened by pressing **Ctrl** + “Double Click” on the tab.

- » Gnats 9720: A window maximized within the design area under Microsoft Windows is once again drawn with the title bar and can therefore be minimized again. Plus the MDI window is supported once again.
- » Gnats 9220: Support for the splitbox object under Motif has been improved. By redirecting the Resize/Moving button from direct children of the splitbox to the splitbox itself, it is now possible to move or change the size of the splitbox object.
- » Gnats 8624: The size of the file history can now be adjusted via the configuration window on the Saving page to between 4 and 99 files.
- » Corrected: The pressing and releasing of the resize button on one window under Motif no longer leads to the window object being moved.
- » Corrected: In MS Windows, a further object could be moved to a spinbox object, even if the spinbox object already had a child object, which led to a full-window portrayal of the newly moved object in the spinbox object.
- » Changed: The editor no longer forces the minimization of child windows, which leads to a quicker start of the simulation for large, modularized dialogs.

38.7 Network

- » Gnats 9761: If multiple network connections are configured on a computer, under some circumstance an incorrect IP address was returned for an '.exec' connection. To prevent this, first a connection to the 'rexecd' service is opened and then the local IP address of this connection is determined.
- » Gnats 9141: When establishing a network connection with .exec, the password was not allowed to have a colon.
In order to use a colon before the syntactically required colon (in names or passwords) it has to be doubled. In the command part of the .exec attribute (after the syntactically required colon), colons are now transferred directly.

38.8 COBOL interface

- » Gnats 9977: Parameters defined under COBOL as *PIC 9(4)* were incorrectly passed when calling a COBOL function from the Rule Language. The problem, however, only occurred in architectures with the byteorder 4321 and has been remedied.
- » Gnats 9804: After initializing a DM value structure from the COBOL side, the IDM COBOL interface functions could no longer evaluate strings as the *DM-value-string-size* was set to 0 and all strings were therefore regarded as empty strings. Actually, initialization should never be performed from COBOL (unless the respective DM functions call this). For comparability reasons (e.g. for A.04.02.h) the standard length 80 is now used automatically if 0 has been stated as the string length.
The *DM-value-string-size* element of the DM value structure is an internal field and may under no circumstances be changed directly.

38.9 Unix

- » New: RHEL 4 (Kernel 2.6.9, 32bit, gcc 3.4.5) with OpenMotif 2.3.0 is available as a new platform. However OpenMotif 2.3.0 should be installed independently at a later time as it is not part of the standard installation of the distribution.
- » New: RHEL 4 (Kernel 2.6.9, 32bit, gcc 3.4.5) with OpenMotif 2.2.3 is available as a new platform. However OpenMotif 2.2.3 should be installed independently at a later time as it is not part of the standard installation of the distribution.

39 Version A.05.01.d

39.1 Important modifications

- » Changes in Object-Monitor and Frame-Monitor under MS Windows.
- » The network interface (DDM option) has been revised completely (see chapter “Change of Network Interface (DDM)”).
- » Due to an expansion of the rule interpreter binary modules are no longer downward compatible.
- » Subsumption of differences between versions 5 and versions 4 of the Dialog Manager under MS Windows (clarification):
 - » A statictext object with no special height setting (*.height= 0*) is preferred set to the height of one text line even it carries no text.
 - » The notebook object now gets as its preferred dimensions the preferred dimensions of the groupbox object plus the height of a tab.
 - » If no “Visual Styles” are activated, the preferred dimension of the statictext and the progressbar object from A.05.01.d is consistent to the version 4 of Dialog Manager.
- » Note for the listbox object:

The listbox object under Microsoft Windows does not support texts which need more than 65535 pixels to display. These texts are displayed but the horizontal scrolling and the setting of *.firstchar* are only working until a special value (65535 divided by medium character width).
- » Note for the treeview object:

The treeview object under Microsoft Windows allow texts of any length but only the first 260 letters are drawn.

Remark: This behavior may be changed with a newer version of **comctl32.dll** (> 6.0) and so the Dialog Manager do not create any message if the text is to long.

39.2 Change of Network Interface (DDM)

The previous behavior of the distributed Dialog Manager (DDM) of responding to network errors with the direct termination of the client or the server has been abolished with Version A.05.01.d.

The application object as a link and abstraction concept for the definition of application parts (e.g. that are available locally, via C/COBOL, via a dynamic library or network) for the Rule Language has the ability to detect errors and to respond accordingly.

The following brief set of rules describes the expected behavior of the IDM in this respect

- » There is an error if either a technical network error occurs (errors that are reported by network system functions, e.g. hardware errors, loss of connection through 'canceled' server process), the IDM versions are incompatible on client and server side, or the IDM network protocol has been

violated. The latter must be reported to IDM Support.

If the application is a dynamically connected library, it may just as likely be a file that hasn't been found.

A non-existent connection (e.g. the idmndx was not linked) for the application type should also be considered an error.

Termination of the program e.g. by a KILL signal or by terminating a process via Task Manager cannot be recognized as an application error. To the other side, this is merely reported as a network error.

The `.active` attribute on the application object reflects, as before, the activation status.

- » As soon as an error is detected, the respective application is deactivated. In addition, an error code is placed on the error stack.
- » For a successfully activated application (initial, as well as when setting the `.active` attribute), a start event is triggered. For an application that is being deactivated (e.g. with `exit()`, `DM_StopDialog`, as well as changing the `.active` attribute), a finish event is triggered.
start/finish events occur at initialization or at the end of an application or after a dialog start/finish, but otherwise asynchronously!
- » Further information about the error that has occurred is found in the attributes `.errorcode` as well as `.systemerror`.
- » If an error occurs while an application function is being processed, the application is deactivated and either the simulation rule is called or a fail is generated.

If the DDM server recognizes an error, once the current server application function has been processed, the network service loop is left and `AppFinish()` called, however with 0-IDs.

39.2.1 Consequences for the existent dialogs

There are some effects that one has to take into consideration as an IDM application developer.

The IDM application (client) or the server part does **not** end itself any longer, or at least no longer straight away!

The start/finish event is not triggered for a local application, but rather for network applications and for local applications with dynlib transport.

From the Rule Language side, an error with a previously existing active application can be recognized via a finish event that indicates that the application has been deactivated, as well as through a sudden failure of an application function or when calling a simulation rule.

On the server side, an `AppFinish()` is called once an error is recognized in order to give the application programmer the opportunity to "clean up". Errors while processing an application function should be treated correctly as the server function is still being processed.

39.2.2 Application example

The following client/server example illustrates the use of simulation rules as well as the response to start/finish events.

Client

```
!! sample.dlg
dialog D

default editttext {
    .borderwidth 0;
}
default statictext {
    .sensitive false;
}
default window {
    on close {
        exit();
    }
}

application Appl
{
    .active false;

    integer SimCount := 0; // counter for calls of simulation rules
    integer FailCount := 0; // counter for fails that occurred

    function string FuncSimple(integer I)
    {
        !! Simulation rule in case of error
        this.SimCount := this.SimCount+1;
        return "SIM-"+this.SimCount;
    }

    !! Server function, that calls a rule on the client side.
    !! Now without simulation rule, error is caught by fail().
    function string FuncComplex(object Rule, integer I);

    on start
    {
        StStatus.text := "CONNECTED";

        PbSimple.sensitive := this.active;
        PbComplex.sensitive := this.active;
        CbConnect.active := this.active;
    }

    on finish
    {
        if this.errorcode <> error_none then
```

```

        StStatus.text := "FAIL: "+this.errorcode+ " - "+this.systemerror;
    else
        StStatus.text := "";
    endif

    PbSimple.sensitive := this.active;
    PbComplex.sensitive := this.active;
    CbConnect.active := this.active;
}
}

rule string Convert(integer I)
{
    return "CONV-"+I;
}

window Wi
{
    .title "DDM NetErrors";
    .width 400;
    .height 200;

    integer SimCount := 0;

    checkbox CbConnect
    {
        .width 80;
        .active false;
        .text "Connect";
        on activate
        {
            Appl.connect := EtConnect.content;
            Appl.active := true;
            EtConnect.sensitive := false;
        }
        on deactivate
        {
            EtConnect.sensitive := true;
            Appl.active := false;
        }
    }
    edittext EtConnect
    {
        .xauto 0;
        .xleft 80;
        .content "localhost:4711";
    }
}

```

```

}
statictext StStatus
{
    .ytop 30;
    .xauto 0;
}
pushbutton PbSimple
{
    .ytop 60;
    .text "Call Simple-Func";
    .sensitive false;

    on select
    {
        variable integer I;

        Appl.SimCount := 0;
        Appl.FailCount := 0;

        if fail(I := atoi(EtCount.content)) then
I := 0;
        endif
        while(I>0) do
            if fail(StCount.text := FuncSimple(I)) then
!! Fail only occurs when function is not tethered by server.
!! Normally the simulation rule is called in case of an error.
Appl.FailCount := Appl.FailCount+1;
StCount.text := "FAILED-"+Appl.FailCount;
endif
            updatescreen();
            I := I-1;
        endwhile
    }
}
edittext EtCount
{
    .ytop 60;
    .xleft 200;
    .content "2000";
}
pushbutton PbComplex
{
    .ytop 90;
    .text "Call Complex-Func";
    .sensitive false;

```

```

on select
{
    variable integer I, FailCount:=0;
    this.window.SimCount := 0;

    Appl.SimCount := 0;
    Appl.FailCount := 0;
    if fail(I := atoi(EtCount.content)) then
        I := 0;
    endif
    !! Trigger loop with an extevent in order
    !! to enable the processing of finish.
    sendevent(this,1,I);
}
on extevent 1(integer I)
{
    !! Call of the network function which in turn calls the convert rule.
    if (I>0) then
        if fail(StCount.text := FuncComplex(Convert,I)) then
            !! Network errors are caught by fail.
            Appl.FailCount := Appl.FailCount+1;
            StCount.text := "FAILED-"+Appl.FailCount;
        else
            I := I-1;
            sendevent(this,1,I);
        endif
        updatescreen();
    endif
}
statictext StCount
{
    .ytop 90;
    .xleft 200;
    .xauto 0;
}
}

```

Server

```

/* sample.c
*/
#include <stdio.h>
#include <IDMuser.h>
#include <samplefm.h>

static trace_errors(char *txt)

```



```

{
DM_ErrorCode errbuf[100];
uint nerrors, i;

nerrors = DM_QueryError(errbuf, sizeof(errbuf), 0);
for (i=0; i<nerrors; i++)
    DM_TraceMessage("%s - error #%d", DMF_LogFile|DMF_Printf, txt, errbuf[i]);
}

DM_String DML_default DM_ENTRY FuncSimple __1((DM_Integer, I))
{
    static char buf[100];

    sprintf(buf, "FUNC-%d", I);

    return buf;
}

DM_String DML_default DM_ENTRY FuncComplex __2((DM_ID, ID),(DM_Integer, I))
{
    DM_Value args[1], retval;
    DM_String str = NULL;

    args[0].type = DT_integer;
    args[0].value.integer = I;

    if (DM_CallRule(ID, ID, 1, args, &retval, 0)
        && retval.type == DT_string)
    {
        str = retval.value.string;
    }
    else
        trace_errors("callrule");
    return str;
}

int DML_c AppInit __4(
(DM_ID, appl),
(DM_ID, dialog),
(int, argc),
(char **, argv))
{
    DM_TraceMessage("AppInit called", DMF_LogFile);

    BindFunctions_Appl (appl, dialog, 0);
}

```

```

        return 0;
    }

    int DML_c AppFinish __2(
        (DM_ID, appl),
        (DM_ID, dialog))
    {
        DM_ErrorCode errbuf[100];
        uint nerrors, i;

        DM_TraceMessage("AppFinish (%d,%d) called", DMF_LogFile|DMF_Printf, appl,
            dialog);
        trace_errors("appfinish");

        return 0;
    }

```

39.2.3 Further Information

IDM generally does not conduct any automatic/periodic checks of network connections. If this is desired by the application, this may be realized e.g. via the timer object.

39.2.4 Changes to the Application Object

The following describes the changes to the application object.

39.2.4.1 New Attributes

The attributes *.errorcode* and *.systemerror* have been newly introduced.

39.2.4.1.1 .errorcode Attribute

Identifier

.errorcode

Classification: Object-specific attribute

Definition

Argument type:	enum
C definition:	AT_errorcode
C data type:	DT_enum

COBOL definition: AT-errorcode

COBOL data type: DT-enum

Access: get

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

Possible error types are:

Enum Value	Meaning
error_none	Activation or deactivation of an application took place without any errors.
error_network	Error occurred when calling the network system routines. This therefore enables a clear distinction between DDM protocol errors.
error_protocol	A non-reparable error has occurred in the DDM protocol. We advise that customers contact IDM Support in such cases. This error is not necessarily signaled by both sides, but only by the side that notices it first. As the network connection is closed in the case of a protocol error, the other side general receives report of a network error.
error_version	DDM protocol version between client and server is incompatible.
error_file	File error. e.g. that occurs when loading a DYNLIB library.
error_unavail	Application functionality is not available – i.e.: NDX- or DYNLIB extension has not been integrated.

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

39.2.4.1.2 .systemerror Attribute

Identifier

.systemerror

Classification: object-specific attribute

Definition

Argument type: string

C definition: AT_systemerror

C data type: DT_string

COBOL definition: AT-systemerror

COBOL data type: DT-string

Access: get

Can only be read.

When an error occurs while using the application functionality (also see `.errorcode`), this attribute is set with the error string that the system provides and can be queried in the event of an error. It is only supported for the error types `error_network` and `error_file`, in all other cases an empty string is returned.

It should be noted that the error string cannot be influenced by IDM in terms of its format and language. Neither can IDM ensure that the system provides an adequate error string for each error condition.

39.2.4.2 start/finish Event

The start event is triggered in the application object once the application has been activated successfully (without errors).

The finish event is triggered if the application was active before and has been deactivated, which can occur as a result of an error, if activation fails initially, when ending the application or when changing the `.active` attribute.

If an error occurs during the event loop, the start/finish events of the application are processed in line with the event sequence. This also means that the current `.active` status at that time does not necessarily belong to the event.

The activation of an application object, which initially had set `.active` to true, takes place when starting the dialog via `DM_StartDialog` or `start()` from the Rule Language. Accordingly, the start/finish rule is executed before processing the dialog start rule and has, as before, ensured that local applications can connect to their functions.

If one ends a dialog, e.g. via `exit()` with an application that is still active, the finish rule is only executed after the finish rule of the dialog.

39.2.4.3 Erratic Behavior in Application Functions

If an error occurs during the processing of an application function, the application is deactivated and an existing simulation rule is called. If there is no simulation rule, a fail is returned in the rule interpreter.

39.2.5 Network Application Side

If an error is detected while calling a DM function on the network side (a network error or protocol error), the function will be returned with an error status.

The network stub is exited with an error when ending the server function. The AppFinish() function is called if this happens as a result of a network or protocol error as long as an Applnit() was called before. However, no application ID or module ID is provided. In this case too, a respective error is on the error stack.

The system error text can be queried via DM_ControlEx with the DMF_GetSystemError action. A (DM_String *) is to be passed as data. If TRUE is returned, an error text exists and the string is stored in the application code page in the data variables.

39.3 Windows NT / Windows XP

- » New: To provide the editor support for composed objects the filter functions are revised. Especially the frame monitor passes through the filter functions.
Remark using **YiRegisterUserEventMonitor** in Microsoft Windows:
The monitors YI_OBJ_MONITOR and YI_OBJFRAME_MONITOR respectively are called for additional Microsoft controls. If a ISA Dialog Manager is built of more than one Microsoft Control the YI_OBJ_MONITOR could be called for each of them.
- » Additional changes during usage of **YiRegisterUserEventMonitor**:
The management of the monitors YI_OBJ_MONITOR and YI_OBJFRAME_MONITOR are changed internally.
Thereby the Windows messages sent at object creation (e.g. WM_CREATE, ...) no longer provided for the YI_OBJFRAME_MONITOR. To recognize in a monitor function that another IDM object is created there is delivered the message IDM_HWD_SUBCLASSED after "sub classing". The parameter IParam of the message carries the Windows Handle (HWND) from the IDM object (identical to the Handle *AT_WinHandle* asked with **GetToolkitData**). So in composed IDM objects you can recognize if a inferior object is created. Be careful, this parameter can be 0.
- » Changed: The IDM option **edextra** is adjusted to the new Editor support.
- » Gnats 9962: At the interactive movement of a toolbar object the calculation of *.offset* by an vertical toolbar object (*dock_left* or *dock_right*) has not considered the toolbar objects docked on top. Through this the moved toolbar object was placed about the size of the top toolbar area to far below when loosing the mouse key. This is solved.
- » Gnats 9961: Till now a toolbar object only could be moved by handling with the toolbar plate or with its title bar or docked in And out by double-click.
- » Gnats 9951: Sometimes a toolbar object were initial not visible. This is solved.
- » Gnats 9948: A window with a statusbar or a multiline menu line and bounded on top and on bottom (*.yauto = 0*) the lower border would be drawn in the right way now again.
- » Gnats 9947: At maximizing a MDI parent window the MDI child windows with *.xauto = 0* or *.yauto = 0* changes their dimensions again.
- » Gnats 9931: The adjustment of Gnats 9637 had to be removed because there were to many problems in customer dialogs.
Therefore this malpractice exists:

An editable pop text object creates a modified event at the lost of the keyboard focus not only if the content was changed but also if another item of the list was selected.

- » Gnats 9927: If a window with *.dialogbox = true* was opened the associated application window flickered and was marked at the taskbar. This is solved because the Dialogbox would be displayed in the front at any rate.
- » Gnats 9898: The edittext object has not react on WM_UNDO. Though there was no charinput event delivered and the content was not actualized by selecting “Undo” from the context menu.

Remark

An undo is not possible if the edittext object has an format because the format routines do not support this.

- » Gnats 9894: If the title (*.title*) of a notepage object was changed while it was visible the mnemonic table was not actualized. In this case mnemonics of notepage models which are instantiated in a visible state were not working.
- » Gnats 9885: The menubox object open event now delivers the object who has called the context menu in *thisevent.index* again.
- » Gnats 9883: In the select event of a pop text object with *.style = edittext* the correct index is delivered now and not the value of the *.activeitem* before the select event.
- » Gnats 9878: A toolbar object which only contains insensitive child objects couldn’t activated which a mouse click in the titlebar if it was docked out as a independent window.
- » Gnats 9873: The pop text object partly lost the displayed text if it was changed in its dimensions in the Editor. The problem only takes place in the IDM for XP under Microsoft Windows XP and was solved providing the new Editor support.
- » Gnats 9864: If additional menubox objects are changed to visibility in the menu line of a visible window the minimal and maximal dimensions of this window are actualized immediately. Though the window could be higher now.
the minimal size of a window had to set in a way the width and the height of the work space couldn’t be less than 0. Especially the minimal height hat to set in a way that all menus of the menu line could be displayed if the window is displayed with the minimal width. Because of this it could be that a window is displayed higher than desired or a window enlarges itself if additional menus were set to visible. To avoid such effects it is enough to choose the minimal width (*.minwidth*) in a manner that the menu line won’t get multiline anymore (all menus fit in line).
- » Gnats 9836: There could have been crashes by processing the open event, because the event was processed synchronous, that means in the execution of any other rule. This behavior was changed for the context menu. The open event there would be processed as delayed as any other event too.

Remark

Using the open event of a menubox object the parent object of this menubox object should not destroyed in any circumstances. These will lead to unintentional effects. Special dangerous are the open events of menubox objects if they are child objects of another menubox object or a window object.

- » Gnats 9832: Calculating the preferred dimensions of an object (*.width = 0* or *.height = 0*) line breaks are now considered.
But only the given line breaks are considered in the calculation. If an object needs an line break because it is too small these additional line breaks were not considered. Furthermore the object's property as a single-line object will remain unaffected. That means, a normally single-line object and therefore at *.height = 0* it is centred in a grid row will continue to centre in a grid row.
- » Gnats 9782: A undocked toolbar object (*.docking = dock_window*) without a titlebar (*.titlebar = false*) couldn't be moved any more. Now the toolbar object could be moved by pressing the left mouse button in the working area. The working area, as a precondition, therefore should not be covered totally from the toolbar's child objects.
- » Gnats 9781 and Gnats 9946: Sometimes a maximized window initially would not be displayed maximized. The problem always appears if the "normal" window dimensions have values which had to be corrected (e.g. minimal window dimensions of Windows are under-run).

Remark

If incorrect dimension values are set for a window then the dimensions are corrected automatically and a resize event is sent. An incorrect value is a value rejected from Microsoft Windows. A violation from *.minwidth*, *.maxwidth*, ... is an dialog script error. Therefore the behavior is undefined.

- » Gnats 9747: The allocation of the keys `Cursor Up` and `Cursor Down` to `Cursor Left` and `Cursor Right` were twisted at a single line edittext object. This concludes in stepping one position backwards if the cursor has achieved the border and `Cursor Up` or `Cursor Down` was selected. Generally despite of this mistake the cursor movement was all right.
- » Gnats 9678: Optional scrollbars are not disappear any more if the virtual dimension in pixel becomes greater than 32767.
- » Gnats 9546 and Gnats 9586: Drawing errors affecting the statictext object if the attribute *.focus_on_click* has had the value *false* were already corrected in version A.05.01.a.
- » Gnats 9541: At the window object were only small icons set. This results in displaying the small icon stretched by task changes with `Alt + Tab`. So the appearance may be bad sometimes. Now both, small and big icons are set.
- » Gnats 9538: If the mouse was moved during a double click at a toolbar object a movement operation could happen. The behavior is enhanced in the way that to start a movement the same heuristic as at Drag&Drop operations is used. That means the mouse must move at least about the value presented in the key **DragMinDist** or the left mouse button had to be pressed longer than the **DragDelay** key value.
Both keys can be found in the Windows registry under "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\IniFileMapping\win.ini\Windows" (earlier this was the Windows area in the **win.ini** file).

Remark

To an user it is very difficult (nearly impossible) to realize whether he had done a double click or a movement operation with an additional single click. At both actions a undocked toolbar object

changes in the docked state. Because of this change the user could sometimes mean that both, the movement operation and a double click was performed.

- » Gnats 9526: The setting of *.topitem* at the treeview object to another value does not work if the redrawing of the Windows control is prohibited. Therefore the setting of *.topitem* does not work if simultaneously an attribute is changed which causes a deletion or the filling of the treeview object (at the moment these are the attributes *.content*, *.contentvec*, *.itemcount*, *.level*, *.open* and *.picture*).
- » Gnats 9270: A context menu is displayed according to object at the following place:
 - » listbox, treeview and tablefield object
At the bottom of the item which has had the focus at the time the key event was initiated. If no item was focused then the context menu is displayed below the upper border of the object.
 - » edittext and editable poptext object
Beneath the cursor on the right hand side (*.endsel*) except for the situation where the cursor is at the end of the text. Then the context menu is displayed at the bottom of the object.
If the cursor is displayed at the beginning of the line in a multiline edittext object the context menu is displayed at the end of the line before.
 - » grouping object
Beneath the upper object border.
 - » other objects
At the bottom.
- » Gnats 9077: It is alright that the *.activeitem* of a poptext object couldn't be set to 0.
- » Corrected: At the change between docked and undocked state of a toolbar object with double click the original position was lost.
- » Corrected: The spinbox object couldn't be handled with the keyboard. Now the spinbox object also responds to keyboard actions unless the child object is not insensitive. If the child object is insensitive the spinbox object can only be handled with its arrows because the child object does not notice any key events.
- » Corrected: It was possible that in a notebook object in which no notepage object was displayed before by displaying a new notepage object there was a border left.
This was noticed in the IDM Editor.
- » Corrected: Between statusbar and design area of a window there is no space left now.
- » Corrected: A MDI child window is restored correctly after maximizing.
- » Corrected: Opening a context menu the focus now is set additional to the object if this object is a grouping one.

39.4 Motif

- » Gnats 9935: The object toolbar no longer reconfigures itself when the size of a child widget changes.

- » Gnats 9910: The minimum size restriction of objects is also 1 pixel when resizing them (e.g. objects beneath a window) as it normally is during setup.
- » Gnats 9904: The psize (self-determining size, if .width=0/.height=0) of an image object, which is not connected left aligned, is now determined correctly again.
- » Gnats 9893: The activate/deactivate events for a notepage in Motif-WSI are now analogical to the Windows-WSI.
 - » "activate" is triggered
 - » initially when visualizing a notebook for the active notepage
 - » when activating via mouse or key command
 - » when activating by setting the .active attribute to true on another notepage
 - » when hiding or destroying an active notepage for the currently activated notepage
 - » "deactivate" is triggered
 - » when hiding a notebook for the currently active notepage
 - » when activating the active notepage by mouse/key command
 - » by setting the .active attribute to true on another notepage

Please note

When displaying (clobbering) a notebook/notepage that is visible anyway, triggered by some special attribute changes, activate/deactivate events are also sent. The deactivate event is not triggered if the notebook/ notepage or the superordinate object is destroyed via destroy().

- » Gnats 9879: The sensitive activation of objects with pop-up menus no longer leads to an insensitive menu.
- » Gnats 9821: The edittext object under Motif with activated scrollbars is no longer turned into a multiline edittext by mistake.
- » Gnats 9816: The system no longer crashes when applying an empty string to a poptext object with format.
- » Gnats 9301: Workaround for the Motif bug in the treeview object implemented. The dbselect event did not take place when the entry had already been active. It took 3 clicks to call the event. Plus now unnecessary select/activate events are suppressed when pressing any random key.
- » Gnats 9188: Double ampersand (&&) no longer underlines the first & character and setup of an & mnemonic.
- » Gnats 9077: A getvalue (explicit and implicit via allocation operator) on .activeitem of an editable poptext now returns consistent values analogical to Windows-WSI and therefore also 0 if the content of .content does not correspond to the content of .text[l].

Warning (clarification of existing functionality)

The .activeitem attribute on an editable poptext object ('.style = listbox' or '.style = edittext') returns a "dynamic" value for a poptext object rendered visible if .content has been changed. The value returned is typical for the window system. Generally, the window system will search the list for the

first line that corresponds to the `.content` value. Then the line number of this line is returned. A 0 value means that no matching line has been found.

Reproducible values can only be returned if the texts are unique and no text is the start text of a previous text. Example of an unfavorable sequence:

```
.text[1] = "smaller";  
.text[2] = "small" // is start text of .text[1]
```

If this is not observed, the delivered value for `.activeitem` depends on the previous history.

As a `poptext` object normally always represents a text from the list, the attribute `.activeitem` can only be set to a value between 1 and `.itemcount` (inclusive in each case). Particularly setting to 0 is not allowed. If as an exception, a `poptext` object should show a text that is not contained in the list (`.text[l]`), the `.content` attribute has to be set dynamically.

- » Corrected: Toolbars that are located inside a window now ignore when the following attributes are set: `.xleft/.xright/.ytop/.ybottom/.xauto/.yauto`.

39.5 Core

- » Gnats 9942: Decoding errors reading dialogs that are using converting tables (e.g. CP1252) are solved.
- » Gnats 9928: The selection at a `poptext` object with format are working again.
- » Gnats 9923: Recursive extevent events (possible through `querybox` calls for example) the last call doesn't overwrite the parameters of the previous calls.
- » Gnats 9922: At the toolbar object the heritage of the attributes `.height/.width` are working now again.
- » Gnats 9919: A `getvalue` access on `.useerdata[0]` of the `poptext` object are now supplying a fail if `.activeitem` is also 0.
- » Gnats 9918 and Gnats 9902: There are now record structures with more than 64k are possible in C or COBOL.
- » Gnats 9909: A statical set `.curvalue` of a `spinbox` are now working also in modularized dialogs.
- » Gnats 9899: The kilo symbol in numerical formats are now passed at inserts/changes of the display site (modify task of the format).
- » Gnats 9818: At writing the tablefield attributes `.xmargin[l]` or `.ymargin[l]` with **-writedialog** (or saving in the Editor) they are not twisted anymore.
- » Gnats 9890: For the trace codes "[SV]" (setvalue) and "[VV]" (setinherit) the scope of the object is also logged in the tracefile.
- » Gnats 9889: A `getvalue` on `poptext .activeitem` supplies now the right inherited value.
- » Gnats 9884: NULL strings are now correctly realized by the `DM_SetVectorValue` function on user-defined arrays. There are no subsequent error deallocating memory in a subsequent value setting.
- » Gnats 9881: Error message concerning a wrong runstate in DM calls now only written as WE messages in the tracefile and not written in a logfile.

- » Gnats 9876: EvalError arguments correctly given to “[FE]” trace messages.
- » Gnats 9860: Remarks to objects with virtual dimensions:
 The clipping of child objects from objects (e.g. groupbox, window, notepage or toolbar) with virtual dimensions are different between MS Windows and Motif.
 Motif is clipping the child objects like the virtual size, MS Windows like the size of the parent object.
 This has visual effects if the virtual size is smaller than the size of the parent object. This must be noticed designing a dialog.
- » Gnats 9845: There is no error anymore using the **:find()** method at empty user defined or associative arrays. Invalid index values trigger a fail again.
- » Gnats 9831: The relaying of fails in a over defined built-in method are now possible.
 Advice: Due to this enlargement of the rule interpreter the binary modules are not downwards compatible.
 The following new keywords were added to the Rule Language and are especially determined to the use in **:get()** (in **:set()** they are needless because the error handling there is results from the return value):

pass <expression>	Evaluate an expression like a return statement to deliver the result to the calling side. If an error occurs the error state are also delivered to the calling site. A normal return statement would be interrupted in that case and processing the next statement.
-------------------	---

So this is as a simplification equivalent to

```
variable anyvalue V;
if fail(V:=<expression>) then
    throw;
else
    return V;
endif
```

throw	With this keyword an error is given to the calling site and the current rule is finished.
-------	---

Example

```
dialog D
model edittext MEt {
    :get() {
        case Attribute
            in .text:
                if this.content="" then
                    throw;
                else
                    return this.content;
                endif
            endcase
    }
```

```

        pass this:super();
    }
}
on dialog start {
    variable string S := "???";
    print fail(MEt.text);           // => true
    MEt.content := "Hello";
    print fail(S := MEt.text);     // => false
    print S;                       // => Hello
    print fail(MEt:get(.docking)); // => true
    exit();
}

```

- » Gnats 9766: There is no Assfail anymore writing a dialog using **–writedialog** or using the Editor (saving or simulating the dialog), even if the dialog carries models with hierarchical inherited **document** or **transformer** objects.
- » Gnats 9793: An Assfail at inserting numbers with zeros at the end in a formatted edittext, having decimal places, is corrected.
- » Gnats 9132: The texture of the poptext *.content* attribute was improved .
 1. **:get()** at subscripted *.content[l]* are now creating an error like **:set()** does.
 2. Setting *.content* having style poptext no longer sets *.activeitem* to 0 if the value of *.content* wasn't found in *.text[l]*. Also the setting of *.content* creates a change event on *.activeitem* if this is changed during the setting.
Additionally in the Motif WSI the setting of *.content* is no longer wrongly done and therefore the display not inconsistent anymore.

39.6 Editor

- » Gnats 9056: Rules of inherited hierarchical child objects are displayed in the Editor again.
- » Gnats 8963: Rules created at a not explicit created default are written out correctly now.
- » Changed: The balloon help (toolhelp with speech balloons) is switched off.
- » Changed: if the creation of objects of a special class goes wrong in the Editor then the create dialog is closed.
- » Corrected: The grid masking is correct again. The grid of the parent object is no longer used.
- » Corrected: A layoutbox without child objects could be moved or changed in size in the Editor again.

39.6.1 Changes in the Editor on Microsoft Windows

- » New: Now it is possible to move a IDM object with the left mouse button in the IDM Editor.

Exceptions

toolbar: For moving the standard model of the toolbar object is used.

splitbox: The left mouse button is further on used to move the splitbars.

- » New: In the Editor the size of the toolbar object now can be changed interactively. A docked toolbar object only can be resized at those sites where no additional position change is necessary. With other words a docked up toolbar object only can be resized below and left.
- » New: A notepad object now can be interactively moved from one notebook object to another.
- » Changed: A double click on a toolbar object edited in the IDM Editor now opens the object/attribute window and not docking of the toolbar.
- » New: A double click on the spinbox arrows in the IDM Editor now opens the object/attribute window.
- » New: It is possible to set the size of an object to 0 in the IDM Editor. Therefore the object must be sized smaller and smaller until it is escalated.

Remark

A object with size 0 is not positioned into the grid. So if using a grid the interactive size changes would result in jumps.

- » New: The poptext object now can be resized in the IDM Editor even if its style has not value poptext.
- » Changed: You can no longer change the size of **notebook**, **spinbox** and **splitbox** direct child objects interactively.
- » New: A started size change in the IDM Editor could be aborted with the **Esc** key now.
- » Gnats 9809: Resizing the width of an object interactively in the IDM Editor it could happen that the value 0 of another value was also changed. This is fixed.
- » Gnats 9763: A size change of an object in the IDM Editor does not change its position anymore.

Remark

A one lined object with position set in raster coordinates (*.posraster = true*) and which is not bounded up and down (*.yauto <> 0*) would change its position further on if the attribute *.height* changes between the values =0 and >0. This is due to the specification of raster coordinates and height 0 which prescribe a centered positioning.

- » Gnats 9729: A poptext object in the IDM Editor are again changeable in size via the new Editor support. The problem has only occurred in the IDM for XP under Windows XP.
- » Gnats 9622: It is possible again to move or change a poptext object in size in the IDM Editor for Windows even if it has set the *.style* attribute to *listbox* or *edittext*.
- » Gnats 9554: A undocked toolbar object (*.docking = dock_window*) are displayed now in the Editor again.
- » Corrected: A spinbox object could be moved at its arrows in the IDM Editor now.
- » Corrected: A edited window wrongly was getting a white background. This is fixed.

39.7 Network

- » New: The network protocol starting from ISA Dialog Manager A.05.01.d is incompatible with the protocol of the predecessor versions. Therefore, **no** connection can be established from the display side of the ISA Dialog Manager A.05.01.d to an application side of an ISA Dialog Manager with a smaller version number.
The application side of the ISA Dialog Manager, however, does recognize older display sides and can support them. However, the new functions will not work and errors that have already been remedied will reoccur.
Also see chapter “Change of Network Interface (DDM)”.
- » Gnats 9944: An error treatment has been introduced for network errors. The network interface of the IDM (DDM) has been changed for this purpose, see chapter “Change of Network Interface (DDM)”.
- » Gnats 8745: No assfail/protocol error when using DM_Set/Get-Content or DM_Set/GetVectorValue or DM_*MultiValue functions for count values >65535. These are forwarded now without errors in the DDM protocol and an error status is returned if the values are exceeded.

Warning

The DDM protocol has changed so that there is no compatibility between A0501c and A0501d in DDM.

Also see chapter “Change of Network Interface (DDM)”.

39.8 COBOL interface

- » Gnats 9943: The string length specification for functions as well as user-defined attributes has been extended so that now strings >64k can also be used with COBOL. The valid value range is [0 ... 2147483647].
Now string lengths >65535 are also taken into consideration by the functions DMcob_LGetValueLBuff and DMcob_LSetValueLBuff.
With this it is possible to exchange string parameters as well as string attributes larger than 64k between COBOL applications and the IDM.
Please note that the DM-Common-Data structure has remained the same for comparability reasons and therefore the DM-value-string and DM_va-value-string as well as the respective -putlen, -getlen and -size definitions remain the same as they are made for strings with max. 80 characters.
- » Gnats 9877: The interface function DMcob_GetArgString now always fills the output buffer DM-Buffer with the fill code specified in DM-StdArgs and no longer with the low value (0).
- » Gnats 9858: The function DMcob_GetRecordMember no longer changes the size in the DM-value-string-size on architectures for which the byteorder has been changed.
- » Gnats 9857: DMcob_GetVectorValue and DMcob_LGetVectorValueBuff return the wrong data type for the attribute AT-userdata if the user data was stored in a string. The error only occurred in architectures in which the byteorder is changed.

- » Gnats 9843: When calling COBOL functions, the bytes were in part not exchanged correctly for all data types. As a result, under Linux the value from the function parameters with the *boolean* type was always *false*. This applied to the data types *boolean*, *datatype*, *enum*, *event* and *scope* on architectures in which the bytes have to be exchanged. This did not affect COBOL under Microsoft Windows.
- » Gnats 9799: The names of internal constants were changed to prevent ambiguities and the associated query problems in **IDMcobsw.cob**.

39.9 Unix

- » New: Sparc Solaris10 is now also supported.

40 Version A.05.01.c3

40.1 Windows NT / Windows XP

- » Gnats 9931: The adjustment of Gnats 9637 had to be removed because there were too many problems in customer dialogs.
Therefore this malpractice exists:
An editable **poptext** object creates a *modified* event at the loss of the keyboard focus not only if the content was changed but also if another item of the list was selected.
- » Gnats 9927: If a **window** with *.dialogbox = true* was opened the associated application window flickered and was marked at the taskbar. This is solved because the dialogbox would be displayed in the front at any rate.
- » Gnats 9883: In the *select* event of a **poptext** object with *.style = edittext* the correct index is delivered now and not the value of the *.activeitem* before the *select* event.

41 Version A.05.01.c

41.1 Important Modifications

- » Until now, in ASCII Dialogs misleadingly access to not exported, inherited child objects was allowed. In this case, after converting the dialog files into binaries, this could cause functionally problems (in extreme situations the dialog wouldn't start).

Example

Definition of an exported model with child objects in Modul1:

```
export menubox MMb {  
  child menuitem Mi1 {  
    .text "New";  
  }  
}
```

Supply by Modul2:

```
export import Modul1 "modul1.if" {}
```

Incorrect usage in the Dialog (or in a modul)

```
import Modul2 "modul2.if" {}  
window WnMain {  
  child MMb Mb {  
    .Mi1.text "&Newer"; !! not allowed, because it is not exported  
  }  
}
```

This behavior is fixed with Gnats 9844. Now rightly such an static access to not exported, inherited child objects also in the developer version (ASCII) is detected an error message is issued.

- » The poptext object (*.style <> listbox*) attends under Microsoft Windows the height setting. This could cause undesired effects if till now a arbitrarily height was set. In such a case the height of the object must set to 0 to get the old behavior back.
- » The appearance of the toolhelp can be set at the setup object with the *.options* attribute.

```
boolean setup.options[opt_balloon_toolhelp]
```

true (default value): Behavior like yet, the toolhelp appears as a speech balloon

false: The toolhelp appears as a rectangle.

41.2 Multiscreen Support Under Motif

The IDM for Motif now has programming support for multiple screens. This support is also available for the IDM 4 version starting from A.04.04.j.

Meaning of Multiscreen Support Under X

This applies to an X server configuration with several screens (several screens either via a graphics card with multiple frame buffers or several graphics cards in the same display host). The screens may differ in terms of resolution as well as color options. They do, however, share input devices such as a mouse and keyboard. Their arrangement to one another can usually be defined in the X-Server configuration (e.g. Screen#1 is to the right of Screen#0).

Also before an IDM application could be shown in a specific screen, e.g. via the X option `–display "<host>:<display>.<screen>"`. It is **new** that **windows can be opened in various screens** within an IDM application.

Support by IDM

IDM supports the application programmer to the extent that he can query the available screens and their properties (this is performed via the **setup** object, see also the attributes `.screencount`, `.screen`, etc.).

The IDM also enables the allocation as to which windows are to be shown on which screens. This is done via the `.display` attribute of the window. It has to be set to a **display** resource which has to be defined by the user. The IDM then takes care of the necessary resource administration for colors, cursor and images.

A dynamic switch of a window to another screen can be performed either via the `.display` attribute or the display resource. For a sample dialog, also see the documentation on the **display** resource.

Dialog windows as well as messageboxes or filerequesters are shown in the same screen as the father specified during the querybox call. If no father has been specified, visualization takes place in the default screen.

Comments

- » Unfortunately, the layout of the screens among one another cannot be determined via X/Motif and is therefore not accessible from IDM.
- » The storage requirements of the IDM are optimized for the standard case of the single screen application.
- » Multiscreen and non-default visuals: Via the environment variable `IDM_VISUAL_ID`, IDM allows for the specification of a visual ID to use another visual than the one defined (e.g. gray-scale screen on a TrueColor display). In a multiscreen configuration, this only affects the default screen, i.e. that the windows in other screens are shown in the default visual.
- » Support for the new attributes and resources for the IDM Editor is now available.

41.2.1 New attributes

The following shall explain the new or extended attributes.

41.2.1.1 screencount

Identifier:

.screencount

Classification: object-specific attribute

Definition

Argument type: integer

C definition: AT_screencount

C data type: DT_integer

COBOL definition: AT-screencount

COBOL data type: DT-integer

Access: get

No changed event triggered.

Can only be read, i.e. it is only allowed to define the attribute but not to change it.

This attribute is available for the setup object and returns the number of available screens. The value returned is in all cases ≥ 1 or 1 if there is no multiscreen.

Comments

- » This attribute is only supported by Motif-WSI and supplies the number of screens that are configured for the used display.
- » An example dialog can be found at the display resource.

41.2.1.2 screen or screen[I]

Identifier:

.screen or **.screen[I]**

Classification: object-specific attribute

Definition

Argument type: integer

C definition: AT_screen

C data type: DT_integer

COBOL definition: AT-screen

COBOL data type: DT-integer

Access: (set)/get

Setup Object

No changed event is sent.

The attribute can only be read.

Accessing the scalar attribute provides the screen number of the default screen. The screen numbers of all existing screens can be queried via the indexed attribute, where the index must be between 1 ...-setup.screencount.

Comments

Only the IDM for Motif has multiscreen support. The provided screen numbers correspond to the numbers provided by xdpinfo and are normally 0 ... (screencount-1). The numbers that can be queried via .screen[I] are not necessarily in ascending order.

An example dialog can be found at the display resource.

Display Resource

Only the .screen attribute is available, i.e. without index.

No changed event is generated.

The default value of the attribute is -1.

The attribute can be set and read.

This attribute is used for the dynamic requesting and setting of screen numbers in a display resource. The value range is between -32767 and +32767, whereby only the numbers ≥ 0 are valid screen numbers (see also the comments). Invalid screen numbers are treated like the default screen.

Changing the attribute will result in all windows that use the affected display resource moving to the specified screen.

Comments

Only the IDM for Motif has multiscreen support. An example dialog can be found at the display resource. Valid screen numbers can either be determined by the program xdpinfo or they can be requested via the .screen[I] attribute on the setup object.

41.2.1.3 real_screen

Identifier:

.real_screen

Classification: object-specific attribute

Definition

Argument type:	integer
C definition:	AT_real_screen
C data type:	DT_integer
COBOL definition:	AT-real-screen
COBOL data type:	DT-integer
Access:	get

Can only be read.

The attribute is available on the display resource.

This attribute returns the actual and valid screen number which is always ≥ 0 , even when an invalid one has been defined on the display resource.

Comments

Only the IDM for Motif has multiscreen support.

41.2.1.4 display

Identifier:

.display

Classification:	object-specific attribute
-----------------	---------------------------

Definition

Argument type:	object
C definition:	AT_display
C data type:	DT_display
COBOL definition:	AT-display
COBOL data type:	DT-display
Access:	set/get

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

This attribute defines the screen in which the window is to be displayed.

When entering zero or a display resource with an "invalid" screen number or for WSI without multiscreen support, the window will be displayed in the default screen.

A messagebox or a filerequester that is opened via querybox with a window as father is displayed in the screen of the father window, otherwise in the default screen.

Comments

Only the IDM for Motif has multiscreen support. An example dialog can be found at the display resource.

41.2.1.5 Attribute extensions for the setup object

Identifier:

.screen_width[I]

.screen_height[I]

.pointer_width[I]

.pointer_height[I]

.xdpi[I]

.ydpi[I]

.colorcount[I]

.color_type[I]

These attributes can only be read!

The above-listed attributes of the setup object are now no longer scalar but can also be additionally requested via the index in area 1 ... setup.screencount. You then receive the respective value for the specified screen index. The scalar attribute provides the value for the default screen.

Comments

Only the IDM for Motif has multiscreen support. It should be noted that the screen index is not the screen number. Indexed attributes in IDM normally have the start index 1 and for the above attributes go to the end index .screencount.

An example dialog can be found with the display resource.

41.2.2 Display Resource

Keyword:

display

Definition

```
{export} display <Identifier> { screen( <Number> ) } ;
```

Attributes

Attribute	RSD	PSD	Property	Brief description
screen	integer	integer	S,G/-/-	Screen number
real_screen	integer	integer	-,G/-/-	Real screen number

The display resource is used for the subsequent allocation of a window to a certain screen. The actually selected screen can be queried via the *.real_screen* attribute. The use of an invalid screen number is automatically transferred to the default screen (usually, but not necessarily, 0). Valid numbers are ≥ 0 . This means that e.g. when stating -1, the default screen is used.

Particularities

Multiscreen support is only in place with IDM for Motif. Valid screen numbers can be determined via **xdpyinfo** and the **setup** object.

The *.screen* attribute can also be changed dynamically via Rule Language.

Example

The following small dialog shows how one can place two windows on different screens and how e.g. a single-screen configuration can be handled.

```
dialog MultiScreen

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

default window WINDOW {
    .width 200;
    .height 100;
    on close {
        exit();
    }
}

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

    statictext StDefScreen {
    }
}

// secondary window on screen#1
window WiSec {
```

```

.display DpySec;
.title "Secondary Window";
listbox LbScreens {
    .xauto 0;
    .yauto 0;
    // move secondary window to another screen dynamically
    on select {
        this.window.display.screen := this.userdata[thisevent.index];
    }
}
}

on dialog start {
    variable integer I;

    StDefScreen.text := "Default Screen: "+setup.screen;

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

    // list available screens
    for I:=1 to setup.screencount do
        LbScreens.content[I] := "Screen#" + setup.screen[I] +
            " " + setup.screen_width[I] + "x" + setup.screen_height[I];
        LbScreens.userdata[I] := setup.screen[I];
    endfor
    LbScreens.activeitem := LbScreens:find(.userdata, DpySec.real_screen);
}

```

41.3 Extension Repository Identifier

So-called Repository Identifiers are used to link an IDM application to an external application in an IDM object basis or to simplify this process (in short **ReposId**). This is an additional string per object, **user-defined** attribute and text variant, which e.g. is used to note a reference text to an object in a database.

The ReposId is a local piece of information which is not inherited but is also written into the binary file. The removal of all ReposId information from a dialog/module is performed with the option **-noreposid** when using the idm program to export a dialog via **-writedialog**.

A ReposId is defined directly behind the definition of an object, resource, text variant or attribute via the keyword **reposid()**. The exact syntactic position is defined as follows:

- » Dialog, module, application, variable as well as for all resources (accelerator, etc.): behind the descriptor
- » Import: behind the file name
- » Text variants: entirely at the end of the variant
- » Function and rule: behind the argument definition
- » Attributes: behind the descriptor or an optional convformat() description.

For all other objects, the definition of the ReposId has to be conducted as an attribute placement of *.repos_id*. This placement type is also allowed for all other object classes and should be used with preference.

The following example demonstrates the definition of the ReposId:

```
dialog D reposid("Dialog.123")
color CoRed reposid("Farbe-ROT")"RED";
function integer Func(string S, boolean B) reposid("FUNKTION.69");
text Tx "Hi" reposid("Text.Hi.317") {
    1: "Hallo" reposid("Text.318");
    2: "Hello" reposid("Text.319");
}
window Wi {
    .repos_id "Window.456";
    checkbox Cb {
        .repos_id "Cb.301";
    }
    integer Wert reposid("Wi.Wert.761") := 10356;
}
on dialog start reposid("Regel.999")
{
    print this.repos_id;
    print Tx.repos_id[2];
    print Wi.repos_id[.Wert];
    exit();
}
```

Dynamic read and write access to the ReposId is performed simply via the *.repos_id* attribute.

The indexed variants of the *.repos_id[]* attribute are decoupled from the existence of text variants and user-defined attributes. However, exporting in ASCII is always conducted in accordance with the language syntax rules, which requires the saving only of existing attributes and text variants. Unlike the process of generating a binary file, where ReposId information of non-declared attributes or text variants are stored.

Moving through all indexed ReposId information must take place analogically to a user-defined associative field via *.itemcount* and *:index()*.

ReposId saving is not optimized for speed but rather for memory capacity. Meaning, if not in use, there is not increased storage capacity when compared to the predecessor version.

41.3.1 New attributes .repos_id and .repos_id[I]

Brief Description

Attribute	Data type	Value range	Properties	Brief description
.repos_id and .repos_id[]	string	("")	SG	Repository information of user-defined attributes or text variants of an object

Identifier and Attribute Properties

	Rule language	C	COBOL	Properties
Identifier	.repos_id	AT_repos_id	AT-repos-id	SG
Data type	string	DT_string	DT-string	

Value range: String

Default value: ""

Classification: Object-specific attribute

Value Meaning and Description

This attribute is available for any object and resource. Any text string can be saved in it. This is a local attribute that is not inherited and for which no changed event is triggered for changes. The attribute is saved in the binary file.

For object classes that allow for the definition of user-defined attributes, an additional indexing can be performed via attributes in order to save a ReposId text per attribute definition.

The text resource allows for indexing via an integer value in order to assign an own ReposId for each variant.

For the indexed variants of the .repos_id attribute, the number of indexed values (<IndexQuantity>) can be queried via access to <Object>.itemcount[.repos_id]. The determination of the respective index is performed via method call <Object>.index(.repos_id, <Index-No>), whereby <Index-No> may vary between 1...<IndexQuantity>.

41.4 New options for grouping objects under Windows

Due to a problematic with "Visual Styles" and bugfixes in size calculation (and therefor different behavior) the attribute .options at all grouping objects (groupbox, layoutbox, notebook, notepage, spinbox, splitbox, statusbar, toolbar and window) has the following new values:

» `.options[opt_wntsizebug_compat]`

Default value: *true*

The value *false* stands for setting the position and size of all direct child objects in the right way.

The value *true* stands for setting the position and size of all direct child objects in a bug compatible way. In Windows versions there was a very old malpractice doing the calculation of position and size of the objects edittext, groupbox, listbox poptext and treeview. Cause to this malpractice these objects are displayed one pixel smaller at each side than desired.

This option is preallocated with the value *true*. Therefore no changes at existing dialogs is necessary.

If a dialog must be designed new (e.g. in the course of porting to Windows XP), then the bug compatible way can be removed step by step.

» `.options[opt_w2kprefsize_compat]`

Default value: *false*

The value *true* stands for calculating the preferred size of all direct child objects in a way as if the application is running under Windows 2000. The value *false* stands for calculating in the Windows XP version of the IDM the preferred size of all direct child objects according to the "Visual Styles".

This option is preallocated with the value *false*. Therefore objects with no dimensions settings appear under active "Visual Styles" also in optimized dimensions.

If the appearance of a dialog is broken than you can turn on this option at the default objects and, as the next step, you can adjust your dialog step by step.

41.5 Windows NT/Windows XP

» Changed: The option `.options[opt_no_clipping]` was only available for the out of date 3D support. Cause this is no longer necessary and its also deleted, this option would not longer evaluated. Therefore the option `.options[opt_no_clipping]` rightly is considered as outdated. That means, by reading a dialog or module it is ignored and by writing (IDM Editor) it is discarded.

» New: The Online Documentation of the Windows version can now used with SUN Java Plug-in version 1.5. The using of the Online Documentation with the Microsoft Virtual Machine (no longer issued) is no longer possible. In addition a pure HTML version is issued, also all single documents are available as (linked from Online or HTML Documentation) PDF-files.

» Gnats 9848: The position of a notepage object title was not actualised if the title was changed in the visible state of the object. The problem appears only if the title was left-aligned.

» Gnats 9837, 9835 and 9803: Sometimes there where drawing mistakes at the treeview object. In the main that occurs with the implicit tooltip of the treeview object if a context menu was closed or in the treeview where scrolling actions.

Also the rectangle between the scrollbars of an grouping object won't drawn another time (affected all objects in a splitbox with vertical and horizontal scrollbars).

Remark

This is an Error in Windows XP. The Error was reported to Microsoft and from there confirmed.

» Gnats 9834 and 9852: A statusbar object is now again display in a MDI Parentwindow.

- » Gnats 9826: Sometimes there was a false modified event at an edittext object with a format. The Problem was only present if the format code page was different from the internal code page. Noticed was this only at a tablefield object with an assigned edittext object.
- » Gnats 9825: The runtime libraries of the IDM display the value of the toolhelp attribute in a multiline way if there are line breaks.
- » Gnats 9824: The vertical scrollbar of the tablefield object was 1 pixel to high (wide). This is especially recognized if "Visual Styles" are active and one of both scrollbars are not displayed.
- » Gnats 9823: An ASSFAIL (object/table2.c line 372) could be occurred if the mouse was moved over an area where a field was drawn before. This could take place if rows or columns of a table where erased. This is fixed.
- » Gnats 9817: The multiline edittext object processed the WM_MOUSEWHEEL messages and do not forward them to his parent object. This is state of the art.
- » Gnats 9814 and 9815: By doing the changes for "Visual Styles" support a lot of mistakes in doing size calculation was fixed. These malpractice exists until support for 3D objects in Windows was supported (since Windows 95 and Windows NT respectively). Especially Dialogs which uses pixel exact design are very affected by these bug fixes. Therefore the XP version of the IDM is now made "bug compatible". Also two new options for grouping objects are added (see chapter "New options for grouping objects under Windows").
- » Gnats 9812: The text in a tablefield object cell was not drawn till the object border if "Visual Styles" (Windows XP style) were active. Instead there was one pixel space between border and text. This was taken back.
- » Gnats 9811: If a checkbox object is positioned in a way that it must be cut (*.ytop = -3*) this was good visible as recently as "Visual Styles" in the IDM for Windows XP are activated. This is because the checkbox object without activated "Visual Styles" is drawn with a small box with a 2 Pixel wide 3D border. From this 3D border the top pixel (dark grey) is lost, what's not remembered practically because a line (black) is already there. Are there "Visual Styles" active, then the small box would be drawn with a 1 pixel border and at the same size as if there were no "Visual Styles" active. Now the top pixel is lost and so the upper border is not there what is recognized immediately. For this case there are no changes.
- » Gnats 9810: The Windows objects checkbox and radiobutton are using the false background if "Visual Styles" are active (Windows XP style). The IDM for Windows XP tries to go round that. That this is successful could not be guaranteed because this is addicted from many "Visual Styles" properties. To play it safe especially at the notepage object should be background color settings. Though you'll see a additional border at the border of the notepage object.
- » Gnats 9803: At a treeview object there would be drawing mistakes sporadically. This mostly take place in connection with the implicit tooltip of the treeview object if there were scrolling actions or by closing a context menu.
Also the rectangle between the scrollbars of an grouping object won't drawn another time (affected all objects in a splitbox with vertical and horizontal scrollbars).

Remark

This is an error in Windows XP. The error was reported to Microsoft and from there confirmed.

- » Gnats 9800: A control object with settings `.mode = mode_client`, `.visible = true` and `.connect = true` are created at `DM_StartDialog` like any other object, too. On the other hand a control object with settings `.mode = mode_server`, `.visible = true` and `.connect = true` is recognized after the on start rule of a dialog or module, in which it is defined.
- » Gnats 9790: There could be a “Debug Assertion” if a MFC ActiveX Control compiled as debug variant was used. This is fixed.
- » Gnats 9789: At this version the HTML Helpfiles were also shipped with the Windows versions.
- » Gnats 9788 and 9783: A crash from `idmole` at completion (access violation) of a displayed OLE control is fixed. The problem was recognized with `"RICHTEXT.RichTextCtrl"`.
- » Gnats 9779: The toolhelp (attribute `.toolhelp`) was displayed as a speech bubble by the IDM. For some applications this was to big. The type of displaying could now configured via the setup object (see chapter “Important Modifications”).
- » Gnats 9778: The wheel mouse scrolling were to slow if a groupbox object has a lot of child objects. The problematic is that each child object must be positioned new. This repositioning is interrupted if new scroll messages arrives. So no long scrolling after the last user action can be avoided. To go round this problem insert additional groupbox objects:

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

    !! workaround for scrolling problem
    groupbox {
        .xauto 0;
        .yauto 0;
        .borderwidth 0;

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

- » Gnats 9776: It was not possible to get a already visible MDI child window with settings of `.visible := true` or `.active := true` in the foreground.
- » Gnats 9773: The ISA Dialog Manager recognizes on Windows platforms from version A.05.01.c his previous Versions (also from A.05.01.c) and offers a substitution at installation time. Previous Versions of A.05.01.c are only recognized as unknown ISA Dialog Manager version. This versions should not be substituted; they should be uninstalled before (if it is desired).
- » Gnats 9765: The toolbar object is again drawn with the right background images.

- » Gnats 9759: The notebook object draws the area beside the notepage tabs always in the "3D-object" background color. To achieve a seamless integration of the notebook object in its parent object the background color of the parent object should be set to "CLR_BUTTONFACE".
- » Gnats 9754: The position of a slider from a grouping object was not actualized if an attribute which affected the position (e.g. *.xorigin*) was changed. This mistake was not noticed if the size of the object had to be calculated, too. Furthermore it was possible to set the values of the attributes *.xorigin* and *.yorigin* beyond the "dynamical" value range. Now the values would be changed to the next possible value.
- » Gnats 9752: If the value of the *.icon* attribute from a window object was set to *false*, the window would be placed in the foreground although a dialogbox was opened. This is fixed.
- » Gnats 9748: Setting the *.edittable* attribute from an invisible poptext object to *false* is now recognized.
- » Gnats 9745: The background color of a poptext list would be changed immediately while the list is opened.
- » Gnats 9743: In the following circumstances the list of a poptext object with *.style = listbox* would not be drawn in the right way:
 - » initial creation of the poptext object with a height that is smaller than the height of the poptext editfield
 - » additional enlargement of the height in a case, that the list could not display all elements and a scrollbar is necessary.

If the poptext object was one time high enough (list does not need a scrollbar) the mistake didn't take place further more.

This problem is fixed.

- » Fixed: the attribute *.top_most* is now recognized if setting dialogboxes visible/invisible.
- » Gnats 9738: If a window with *.dialogbox = true* is opened the last active window from the application is registered as the "Owner". If the application has other windows too, these windows could be activated via Windows Taskbar. While it could not be avoided that all windows in the Windows Taskbar could be activated, the Dialog Manager shortly after the "wrong" window was activated activates the dialogbox another time.
 Something near that applies to invisible windows which could also be activated. In that case, shortly after that the last active window of the application would be activated. Is there no such window, the insensitive window keeps activated.
- » Gnats 9717: The second access on an indicated OLE property from a control or subcontrol object now works also when the property supplies an OLE object. The problem is near the problem in Gnats 8678, but at the subcontrol object from the first access you don't have to set any label, as in this case it is possible that there could be detected a wrong object.
- » Gnats 9703: The data type *DT_instance* was not accepted by control and subcontrol objects on OLE method calls or OLE properties access. The problem only affected the C Interface because there you get the data type *DT_instance* instead of *DT_object*.

- » Gnats 9692: On setting the attribute *.connect* of a control object to false maybe no finish event would created or the application crashes. To get this Error, first a OLE action must be proceed which created a subcontrol object.
- » Gnats 9685: Dialog Manager Drag&Drop would now also supported by the poptext object. As in Drag&Drop normally, only the active item of the list could be selected (dragged). Pay attention, dragging a item out of the list would change the content of the editfield like it was a selection. The thisevent object *.index* attribute could have the following data types for paste or cut event:
 - » *void*
The position in the poptext object could not identified exactly or the editfield is hit at position 0.
 - » *integer*
The list was hit at the position specified.
 - » *index*
The editfield was hit at the area specified
- » Gnats 9675: if the attributes *.activeitem* and *.content* at the poptext object where set back-to-back the *.content* setting was lost even though it was the last setting. The error only take place if both settings were combined performed (e.g. from C without any GetValue or updatescreen between the settings).
- » Gnats 9637: A editable poptext object could create a modified event without any direct change of the *.content* attribute. This always takes place if first a different item of the list was selected and then the keyboard focus was lost.
- » Gnats 9626: Errors in the tracefile were logged although the OLE call was al right. This causes to a superclass of the control object which has already crated an error (without deleting it). This mistakes conspicuous working with C-functions; by working with the Rule Language these errors only at the end of the program are logged, if at all.
- » Gnats 9611: At visualizing a poptext object the *.height* attribute was overwritten with 0. If at the same time *.posraster = true* was set, a position fault has taken place.
- » Gnats 9598: If the dimensions of a child object of a layoutbox object were changed dynamically the dimensions of this child object were not actualized. This is fixed.
- » Gnats 9576 and Gnats 9751: An already visible window could not brought to foreground by setting *.visible = true*.

Remark

If the application is not active a window would not brought to foreground. Instead it would give a short flicker and, at least under Windows XP, in the taskbar it would be presented with a different color.

- » Gnats 9571: By closing the open list of a poptext object sometimes there may be an inconsistency of the *.activeitem* attribute. This took place if the list was closed by opening another window (or messagebox object) or changing the content of the poptext object.

Attention

If the poptext object carries double texts than this inconsistency could not solved clearly.

- » Gnats 9549: In a method of a control object with *.mode = mode_server* a network connection couldn't be established. The problematic was being in an OLE access in the Windows network queue. Therefore no other network queue could be started, the blocking network call was canceled. Now this case would be noticed and as a workaround a filtering of the messages is processed. For this filter we couldn't guarantee so network calls in OLE calls should be avoided. These critical situations are written in the tracefile as warnings.
- » Gnats 9523: A control object with *.mode = mode_client* which was not a child of a window object (must no direct one) provides no OLE interfaces for "InPlace Activation" to an OLE control. In this situation it could be that maybe the OLE control could not be accessed. In such a case the control object had to transform to a child of a window object, then "InPlace Activation" would be possible.
- » Gnats 9505: If the dimensions of a visible child object of a layoutbox object is changed later, the dimensions of these child were displayed correctly but the positions of the child objects of the layoutbox object were not actualized. This is fixed.
- » Gnats 9202: It would be possible that the width of an object was miscalculated if the object text contains Unicode characters. This behavior is conducted to the used font. A statictext object in a statusbar is sometimes displaying alternate characters instead of Unicode ones by using some font resources. This behavior is conducted to the statusbar object of Windows and can not be discovered from the IDM. In such a case a false width is calculated. The problematic takes not place if "Visual Styles" are activated (only with IDM for XP possible).
- » Gnats 9191: Drag&Drop at an edittext object with a language displayed right to left is now possible.
- » Gnats 9047: If a tablefield object was higher than the display the lower border of the object was not drawn. This mistake was located in a Windows function which is not used any longer. In Windows XP this problem was not detected.
- » Gnats 8716: If the attribute *.dialogbox* at a window object (Wn1) was changed to *true* while another window object (Wn2) with *.dialogbox = true* were open the first window object (Wn1) keeps locked. This is fixed.
- » Gnats 8712: The open event for a submenu of a context menu is now considered again.
- » Gnats 8680: An implicitly created subcontrol object without any label couldn't be connected any more if *.connect* was set to *false*. The suggestion to drop such a subcontrol object immediately would not be implemented because this suggestion will lead to not meaningful error reports.
- » Gnats 8678: The second (explicit) *getvalue* on a OLE property of a control or subcontrol object would now work if the property is resolving an OLE object. Instead of resolving the subcontrol object created at the first *getvalue*, another subcontrol was created without any label. If the *getvalue* was done indirectly using *Control.Prop* (instead of *Control:get(.Prop)*) the access works anyway.
- » Gnats 8556: the miscalculation of an Enhanced Meta File (EMF) picture dimensions is solved. The picture wouldn't be unnecessarily compressed anymore.

- » Changed: The position of the objects listbox, poptext and treeview in the XP version under Windows XP are now positioned like the edittext object. Up to now each border of these objects are one pixel smaller than the edittext object.
- » Solved: the multiline edittext object (*.multiline = true*) process WM-MOUSEWHEEL messages even though it has no vertical scrollbar.
- » Solved: If a splitbox or tablefield object was only partial visible the movingline was nevertheless drawn in full length. Now the movingline is limited by the parent object. The moving area will remain unaffected that means the line could be moved out of the parent object area.
- » Solved: The poptext object (combobox) now adjust its dimensions if a to small or to big height was set.
- » Solved: The spinbox object could be scrolled with the mouse wheel again.

41.5.1 Wheel mouse and wheel mouse support with Microsoft Windows

The kind of scrolling a object using a scrollbar normally is like this: The object having the scrollbar is scrolled. To scroll these object, the mouse cursor has to be over the scrollbar.

By scrolling with the keyboard even its clear that the object, which has the focus, would be scrolled.

These clear things are not as clear if you are scrolling using a mouse wheel. Even the object under the mouse cursor could be scrolled but the object having the focus could be also. The "Windows User Experience Guidelines" from Microsoft don't say anything about which object should be scrolled with the wheel mouse. And so in reality there are actually two different models. Which model is used is not addicted to the application, it is addict to the used mouse driver (later more).

Asking computer users, the model the user is familiar to would be declared logical.

Why the mouse driver defined the behavior?

In the online helpfiles from Microsoft you can read the following things to the special mouse wheel message WM_MOUSEWHEEL:

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

That means Microsoft will determine the model the focus object should be scrolled and not the object under the mouse cursor. The IDM support this by handling the message as requested.

But the first wheel mice are there before the WM_MOUSEWHEEL message was. The drivers of these wheel mice utilizing the fact that an object is scrolling if it is getting the WM_VSCROLL or WM_HSCROLL message. That means, a WM_VSCROLL or WM_HSCROLL message and not a special WM_MOUSEWHEEL message is sent. At newer drivers it could be set up how they should work.

Conclusion

The behavior of the IDM is correct and the WM_MOUSEWHEEL message is processed rightly. If the mouse driver only creates WM_VSCROLL or WM_HSCROLL messages instead of a WM_MOUSEWHEEL message it is a driver or driver settings problem.

Unfortunately no official Style Guide is known by us describing which object should be scrolled using a mouse wheel.

41.5.2 Installation of the IDM 5 versions, modification

The ISA Dialog Manager since version A.05.01.a recognizes on Windows platforms his previous versions (also since A.05.01.c) and is offering an exchange. The versions before A.05.01.c are only recognized as unknown ISA Dialog Manager Versions. These Versions should not be exchanged, they should be uninstalled before (if desired).

Warning

In no case a installed IDM 5 version must not be moved, deleted or renamed from or in his installation directory. If one of theses actions is needed you had to uninstall the concerned version in the Windows system control panel and install it another time in the new destination.

Installation Details

Insert the ISA Dialog Manager CD in your Computer and start the specified setup program

IDM_<version number>_<platform>_<language>.exe

whereas

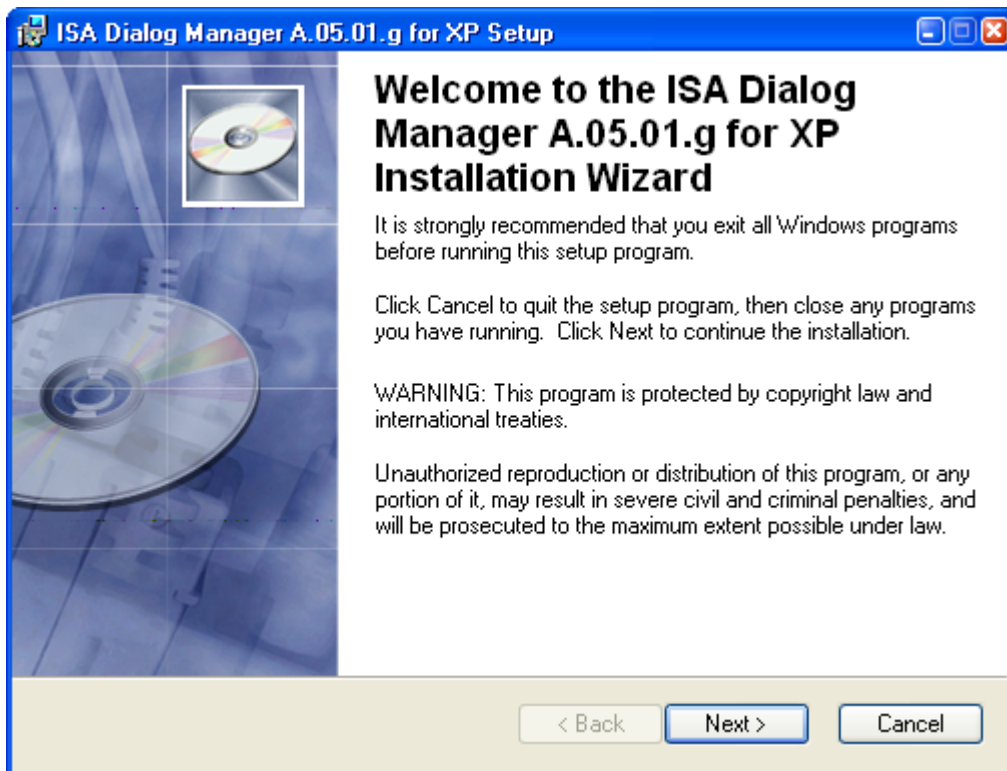
version number = IDM version

platform = Windows 2000 (W2K), Windows XP (XP), Java for Windows (Java)

language = german (de), english (en).

Below the needed steps are shown as an example with the Windows XP platform, english, Version A.05.01.g.

After starting the setup program you'll get the below start dialog



Please select “Next” and in the following dialog insert

- » the Username
- » the company
- » the product number (license key) on the back of the CD Inlay

and select, if you have administration privileges, if Dialog Manager should be installed for the current or for all users.

ISA Dialog Manager A.05.01.g for XP Setup

User Information
Enter the following information to personalize your installation.

Full Name:

Organization:

Product ID: - - - - -

The settings for this application can be installed for the current user or for all users that share this computer. You must have administrator rights to install the settings for all users. Install this application for:

☒ Anyone who uses this computer
☐ Only for me (Rupert Beitel)


Wise Installation Wizard (R)


< Back Next > Cancel

After that you get a dialog with the installation options “*Complete*” (recommended) or “*Customized*”.

ISA Dialog Manager A.05.01.g for XP Setup

Select Installation Type
Select the desired installation type.

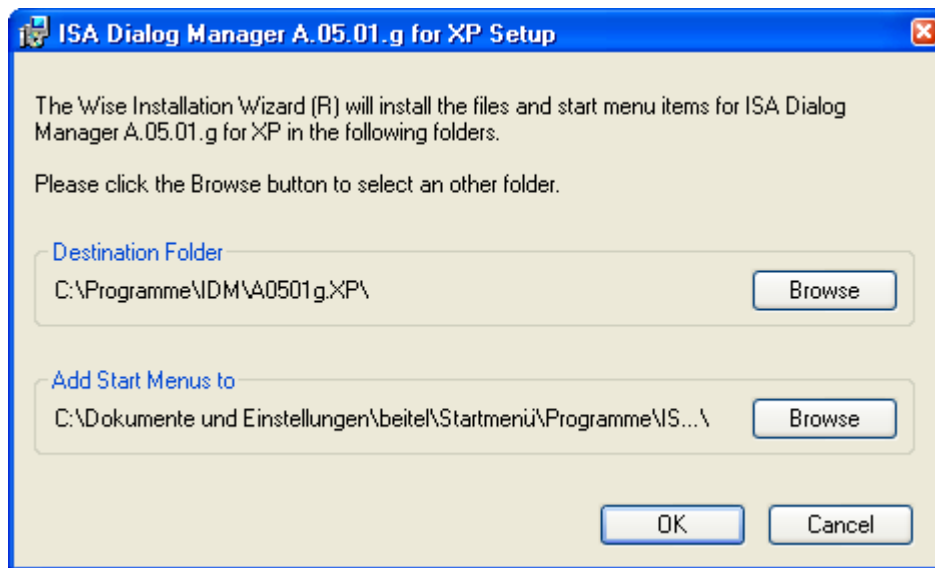
☒ **Complete**
 All application features will be installed. This option is recommended for the best performance.

☐ **Custom**
 Use this option to choose which application features you want installed. Recommended for advanced users.

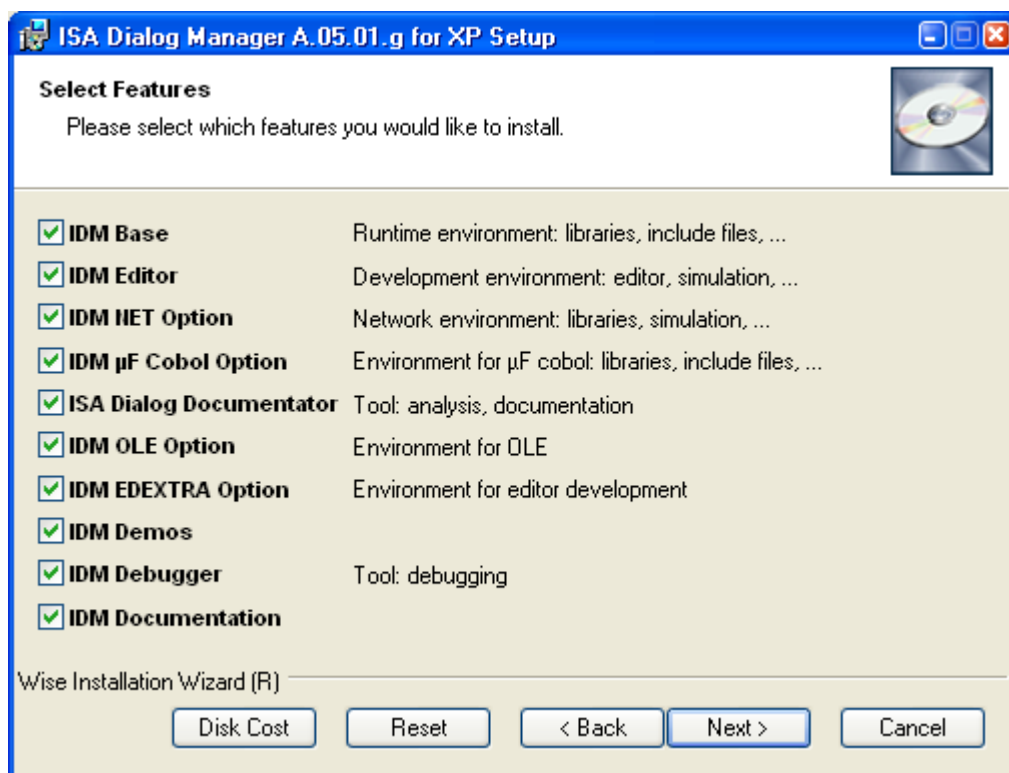
Wise Installation Wizard (R)

Options < Back Next > Cancel

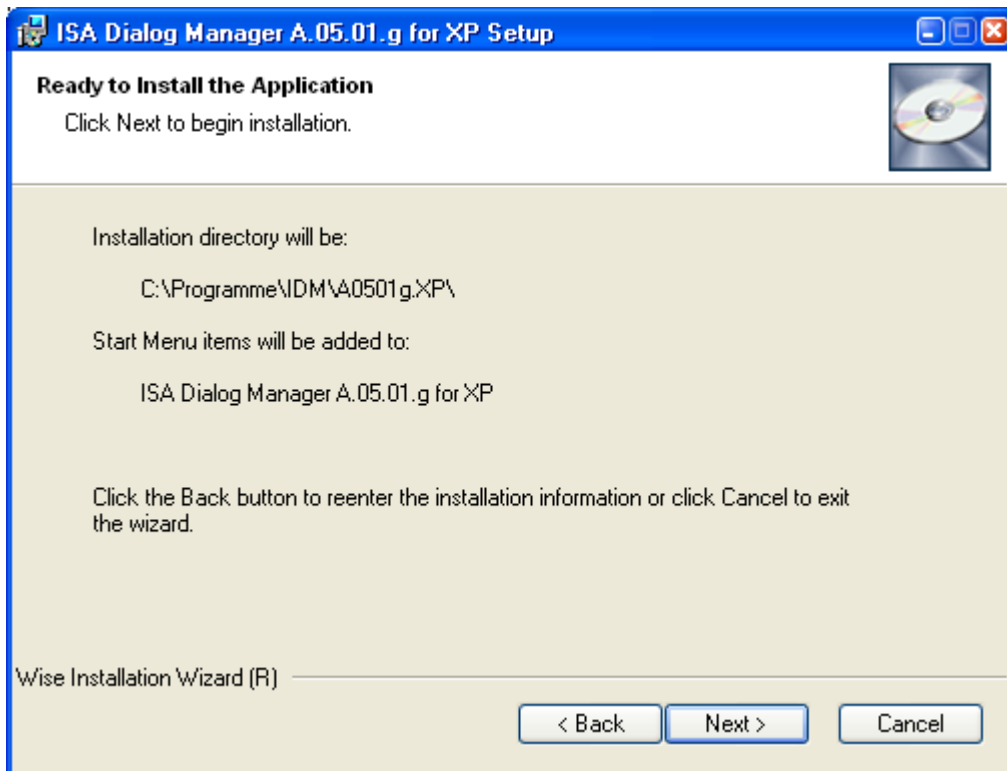
By selecting “Options” in the following dialog the installation directory (destination folder) and the start menu item can be changed



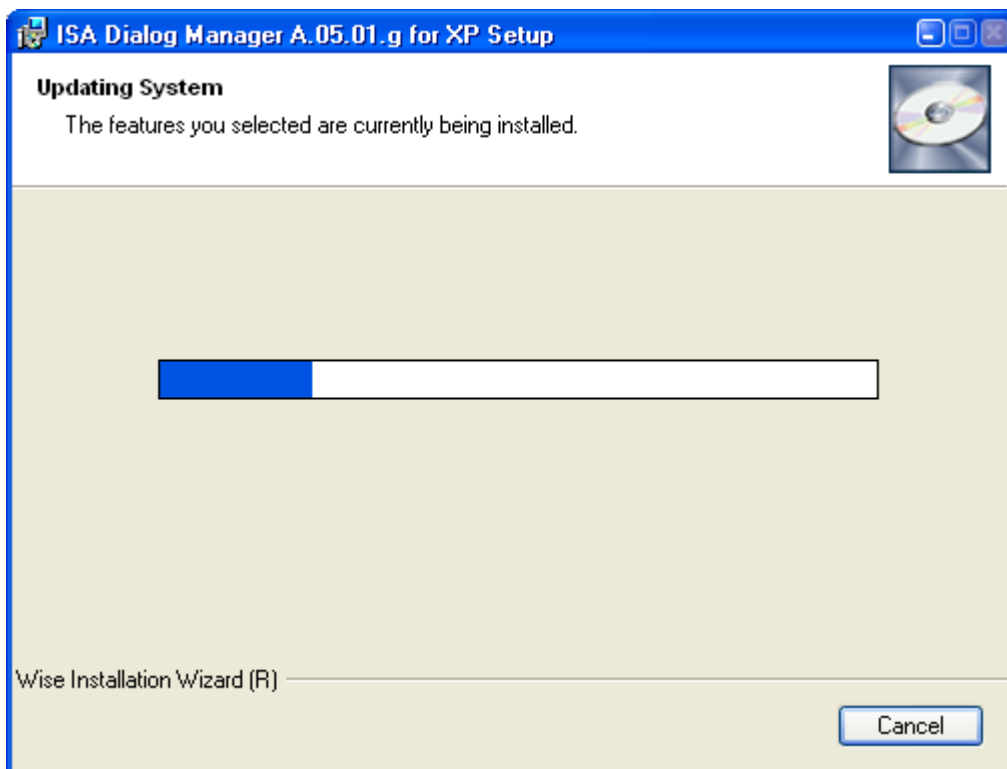
By selecting the customized installation (“Custom”) in the previous dialog window you can choose the options you want to be installed (as far as available in the license).



The needed space and the available space on the volumes could be shown by selecting “Disk Cost”.
At any rate you’ll get an dialog showing the installation properties by selecting “Next”.



Now the installation would be activated and in another dialog the progress is shown.



At last the following dialog is shown.



With selecting *"Finish"* the installation is completed.

41.5.2.1 Further options on installed previous versions

After the *"User Information"* dialog you entered your serial number the *"Select Installation Type"* became visible. This dialog could carry the following additional options:

- » A checkbox *"Exchange ISA Dialog Manager..."*
Here you could choose whether the previous version should be replaced or not. If a previous version with replacement support was found, the checkbox is selected. Further information if a replacement is supported or not is also displayed.
- » A checkbox *"Install in the same directory as..."*
Here you can choose whether the new version has to be installed in the same directory as the previous one or not.
This checkbox is not selected as default.
- » Both checkboxes wouldn't displayed if no previous version was found.

41.6 Motif

- » New: ISO10646 font encoding is now recognized (for Linux Unicode support).
- » Changed: In a font list the first font is used as default font and not, as before, the first single byte font.

- » Gnats 9863: The dynamic Create of an image object whose model has a picture with *.visible := false*, is now again correctly hidden.
- » Gnats 9798: No more "Bad Window" X errors occur when using IDM without Window Manager.
- » Gnats 9771: Minimizing or maximizing a window with a notebook and the contained layoutbox no longer results in the application "freezing".
- » Gnats 9749: Multiscreen support for the Motif window system (see chapter "Multiscreen Support Under Motif" for details).
- » Corrected: A *.bgc* color setting on the groupbox or layoutbox object now also takes effect when a *.tile* (background image) is set.

41.7 Java Interface

- » Gnats 9773: A parallel installation of multiple IDM versions was not possible with the 5 versions under Microsoft Windows. Predecessor versions from A.05.01.c are now recognized and a replacement is offered but not forced (also see chapter "Installation of the IDM 5 versions, modification").

41.8 Core

- » New: A additional tracing code [EF] (evaluation failure) was established, issued at an error of evaluation while processing rule code. This readout is additional to the existing readouts and it is issued even the expression is inside of an *fail()*-bracket. Therefore it should be much easier to recognize errors in *fail()*-brackets.
- » Gnats 9844: The access to not exported, inherited child objects now rightly creates an error message.
- » Gnats 9819: The turnaround in a spinbox object was frozen if it contained text items (*.style = string*) which were only different in upper and lower case.
- » Gnats 9807: The value of the *.convformat* attribute of user defined attributes now were also written in the binary file.
- » Gnats 9805: A return value *false* of an overwritten *:set()* method are now passed correctly.
- » Gnats 9794: The error message "can't identify root-object ... of path" which occurred at the binary reading of a module which references an object via an *export-import* construction from a not actual loaded betwixt module is solved.
- » Gnats 9772: *.options[opt_new_align]* of the tablefield object is now written binary correctly.
- » Gnats 8871: Objects with *.posraster = true* and *.height = 0* and normally displayed in one line are not displayed without overlapping in a layoutbox object. The problem is that such objects have a rasterized height and are centered in the grid. This could not be realized from the layoutbox object, by what overlapping depends.
- » Gnats 8483: Sorting child objects of a splitbox objects in another way no Assfail will occur.

41.9 Editor

- » Changed: In Linux now ISO10646 fonts in UTF-8 local are used. The IDM Editor therefore is setting up font variant 3 and uses lucidia as well as lucidatypewriter fonts. The delivered default file is also extended with this fonts.
- » Gnats 9842: In the Editor there is now the possibility to apply a method on a model with the same name as a method in the instance.
The not possible creating/renaming of attributes (attribute/userdata section) with a label also used as method name in the instance would be adverted with an error message.
- » Gnats 9795: The crash at closing a toolbar in the graphical Editor under VMS is corrected.
- » Gnats 9622: Interactive moving or dimension changes of a poptext object with `.style = edittext` or `.style = listbox` is now possible again.
- » Gnats 8506 and Gnats 8508: Several types of objects couldn't be handled in the source code window of the old Editor (Editor II). This is invalid because the source code window is inexistent in the new Editor (since A.05.01.a).
- » Gnats 8490: Objects displayed in the "Code Area" of the Browser toolbar are not tagged automatically. This is solved.
- » Gnats 8479: Sometimes a dialog in Microsoft Windows under some circumstances would be marked as changed directly after loading. In the new Editor this does not appear.

41.10 Network

- » Changed: Under Windows it may occur that under certain circumstances network calls are executed while you are in Windows message processing. This problematic situation is revealed with a warning in the trace file. The earlier messages of an "unsupported network call" are no longer used.

41.11 COBOL interface

- » New: Two enums were added in IDMcobls.cob and IDMcobws.cob.

Rule language	COBOL
<i>nodetype_processing_instruction</i>	NODETYPE-processing-instr pic 9(4) binary
<i>nodetype_document_fragment</i>	NODETYPE-document-frag pic 9(4) binary

- » Gnats 9865: Trampoline C file for COBOL functions with records containing child records is correctly exported.

42 Version A.05.01.b5

42.1 Motif

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

43 Version A.05.01.b2

43.1 Windows NT / Windows XP

- » Installation procedure changed so that the Window Installer no longer checked for existing old or new versions or forbids multiple installations.

When the ISA Dialog Manager (version 5) is installed, a systematic survey to see, if another level 5 version already exists is no longer carried out. Now it is possible to install multiple level 5 versions at the same time.

Attention

The previous version needs to be explicitly uninstalled, if it is no longer needed. Each installation is viewed as a parallel installation and **does not** delete the previous version. The installation index should **never** be moved or renamed!

43.2 Motif

- » Gnats 9715: Using **Ctrl** + **Tab** in special conditions under KDE, the faulty notepage activation no longer occurs.
- » Gnats 9770: Pop-up menus on insensitive objects (in static cases) are now correctly set (disabled) under Motif2.1

44 Version A.05.01.b

44.1 Windows

- » New: Microsoft Development Environment .NET 2003 projects have been added to the examples of the IDM version for Windows XP. In the course of this change the Microsoft Developer Studio 6.0 projects were also removed from this version.
The IDM-Version for Windows NT still contains the Microsoft Developer Studio 6.0 projects.

- » New: The text of an image object can now contain more than one line.

Attention

When height = 0 the number of text lines for the height adjustment of an image object on the tile resource is not taken into consideration.

- » 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 presentation of the order; the indexes (*.text[l]*) remain the same. The attribute *.options[opt_sort]* determines if the content should be sorted or not. Generally, the attribute is set to *false*. If sorting is desired, the attribute must be set to *true*.
In this way the standard behavior of Microsoft Windows is better supported with respect to a pop-text object, which prerequisites an sorted content.
- » New: The treeview object now supports the setting of text color (*.fgc*). As a result, the appearance of existing dialogs can change.
- » New: Internal tiles now display the actual background color with the characters "." and " ". If this is not desired, then the desired color can be changed through the attribute *.imagebgc* on the object image.
- » Gnats 9760: During the destruction of a grouping object (groupbox, notepage, etc.), the focus was not forwarded correctly. This error has been corrected.
- » New: Gnats 9728: The statictext object (Windows XP version) now also supports "Visual Styles". As a result, the background of a static object in the XP version is the same as the background of a notepage father object.
- » New: Gnats 9677: The attribute *.toolhelp* now supports multiline texts.
- » The method *:getformat()* of the edittext object (*.options[opt_rtf] = true*) returned *nothing* under the non-unicode capable Microsoft systems. The error existed in *idmuni.dll* and has been corrected.
- » Sometimes it was not possible to use a parameter, that contained a space, for example by the definition for an OLE server.
- » The size of a non-optional scrollbar slider was not correctly reset after the size of a window had been interactively changed. One could see, for example, that after the object affected was made visible or invisible the size of the slider had also changed.

- » Through an interactive changing of the window size, the maximum size of the window was not taken into consideration.
- » The size of maximized MDI children windows was not correctly adjusted when the size of the father window was changed or when toolbar objects were docked or undocked.
- » Gnats 8669: The examples have been adapted to run on pidm.exe. They are now executable with a dedicated P-license (porting license).
- » Gnats 9760: The focus is now set correctly after the destruction of a grouping object (groupbox, notepage, etc.). In some cases this can be very time consuming in its use.
- » Gnats 9741: The method :setformat() on the edittext object (.options[opt_rtf] = true) did not function correctly. This problem was caused by Gnats 9672. Please take note that the setting of a format on the insertion mark (startsel = endsel) can only take place during runtime (.visible = true), and is only temporary. This is a characteristic of a Microsoft Windows object, which unfortunately cannot be avoided. Please also take note of the comment for repairing Gnats 9672, which is noted further on in these release notes.
- » Gnats 9733: An Assfail in object/slot 2284 was possible when a poptext object with .style <> pop-text had 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 was not processed by IDM. In this reported case, this happens for the **Tab** key and the modifier keys **Alt** and **Ctrl**.
- » Gnats 9730: When a maximized window was made visible, it possessed the maximized condition (this could be seen on the buttons in the title line), but it obtained the normal size.
- » Gnats 9723: The checkbox, radiobutton and statictext objects now use the same background color as the father object when no other background color has been set. Up to now the system background color for 3D objects was used in this case.
- » Gnats 9723: Docked toolbar objects were covered from the work area in MDI father windows.
- » 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 indexes.
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");
```
- » 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 letter remained unmarked. This problem only happened when the cursor was moved by more than one letter at a time (movement with **Shift + Home** or **Ctrl + Shift + Cursor Left**). This problem did not occur when a format (was empty) or no format was set.

- » Gnats 9705: When the trace file could not be opened, the application was ended. This led to the problem that the simulation was not able to be started from the editor when the surrounding variable `IDM_TRACEFILE` was set to a fixed filename. Now, the application continues to run without tracing after showing an error message. In addition, there is now an additional space holder "%A" for entering the name of the log or the trace file. For additional information please refer to the chapter entitled Core.
- » Gnats 9698: The setting of a cursor for the entire application (`setup.overridecursor`) was not reported to the statusbar and toolbar objects. Due to this, an Assfail in object/slot 2283 occurred in some cases.
- » Gnats 9696: Under Windows 95 additional menus can't be created.
- » Gnats 9680: The Microsoft Windows poptext object does not discern between capital and small case letters. Due to this it was possible, that an incorrect entry was displayed when the poptext object contains identical texts which only differs in upper/lower-case characters.

Note for Windows 95

This problem can not be avoided in Windows 95.

- » Gnats 9673: The treeview object draws a set background color (`.bgc`) only where no text existed. In order to correct this error `comctl32.dll` version 4.71 or higher (beginning with Internet Explorer 4.0) is needed.
- » Gnats 9672: When the method `:setformat` is used on an edittext object that has `.options[opt_rtf] = true`, and when the attributes `.startsel` and `.endsel` or the parameters `Start` and `End` have the same value, then the edittext object throws out the format setting when a change is made to `.endsel`. This also happened when one of the methods, which had `.options[opt_rtf] = true` for an edittext object, was called up with the parameters `Start` and `End`. This can now be avoided. The above-mentioned standard behavior of the Windows object remains the same.

Note

- » Method `:setformat()`
A format setting made to the insertion marker (`.startsel = .endsel` or `Start = End`) is only temporary. The applied format remains valid up to the point where `.endsel` is changed. Here, it makes no difference if `.endsel` is changed by the user or through the dialog script. This is the reason why a format setting where `Start = End` has no effect when `Start <> .startsel` and `End <> .endsel`. In the last mentioned case, the method `:setformat()` returns a faulty value. In addition, it is important to know that a format setting on the insertion marker is not assumed in the RTF-text and as a result remains ineffective when it is carried out on an object that is invisible at that moment.
- » Attribute `.content` and Method `:gettext()`
The RTF-text is changed by the object itself (not formatted). It exists only logically. In other words, for example, a letter is cursive and bold – but the order of events of a change in format (first cursive and then bold or vice versa) can change.
- » The use of the parameters `Start` and `End`

If one of the following methods :findtext(), :getformat(), :gettext(), :replacetext() and :setformat() is called up with the Start and End parameters, take note, that the marked area remains as it was, but that the insertion marker located to the right of the marked area can be slightly shifted out of place.

- » Gnats 9671: When a invalid tile resource, was applied to an attribute, then the attribute was handled incorrectly - as if no tile resource was applied to it. Now the attribute is regarded as being set.

Example

The tile resource TInSens refers to the GIF file "NotExisting.gif", which does not exist. This resource is allocated to the attribute .picture[tile_insensitive] of an image object. When this image object is set to insensitive, then the image object will be filled with the background color; previously .picture[tile_default] was indicated.

- » Gnats 9652: Filerequester dialogs (filereq object) no longer functioned under Microsoft Windows 98. The problem was, that an incorrect size was set in idmuni.dll.
- » Gnats 9639: When a poptext object with .style = edittext contained a number of entries that began with the same characters, then, after opening the list, the first entry and not the entry that corresponded to the active item was marked. Unfortunately, since this is the standard behavior of the poptext objects under Microsoft Windows which cannot be changed. However, an event (select or activate depending upon .options[opt_old_select]) is created when the .activeitem attribute is changed when it is opened. The content of the .content attribute remains the same in this case.

Note for Windows 95

This problem cannot be recognized under Windows 95. As a result no event is fired.

44.2 Unix

- » The examples in IDM.EXAMPLES now use the pidm, as a result these are also compilable with a P-license (porting license).

Attention

Beginning with A.04.01.a, the P-license is only delivered with the pidm and no longer with idm. Now, pidm is a part of every IDM license.

44.3 Motif

- » New: Under Motif1.2, as with Motif2.1, a consistent way for making pop-up menus visible has been established. By activating the menu buttons, the visible pop-up menu (.visible set to true) of the topmost sensitive object is displayed. This is analog to Windows with only one exception: under MS Windows insensitive pop-up menus behave like "invisible" ones.
- » Gnats 9764: The predefined value of width and height of a rectangle object is now again 0/0.
- » Gnats 9746: The &-character (mnemonic-character) was not handled correctly by the listbox, treeview, poptext and filereq objects.

- » Gnats 9740: Maximized windows are now completely visible including all of its decorations.
- » Gnats 9737: The initial size of windows with menu bars is now again correct.
- » Gnats 9715: Falsly activation of the Notepage below the mouse pointer corrected when pressing ALT-TAB to switching between windows.
- » Gnats 9713: When an element of a treeview is activated via the setting of `.activeitem` now the father element is correctly opened and its activation is guaranteed.
- » Gnats 9708: Making a dialogbox visible under certain Window Managers was sometimes delayed. In addition, corrections needed to be made, which also had an effect on the process of the window size/position with respect to the interaction with the Window Manager.

Peculiarities of the window object under Motif

The position values (`.xleft`, `.ytop`, etc.) refer to the outer coordinates of the window. Window Manager or desktops with panels normally carry out a correction of the position/size themselves in order for the panels to remain visible. This is reflected again in the position - this can be seen in relation to the whole screen. Very few and rare Window Managers, for example TWM, do not allow IDM to determine the position of the window including its decorations. This leads to windows moving when they repeatedly switched from a visible to invisible status.

- » Gnats 9620: Attributes `.xmargin` and `.ymargin` has been added to the ***edittext*** object. These attributes (type *integer*) determine the space between the outside edges and the text in pixels. Changes to the margin values have an effect on the spacing to the upper and left side. When `.height = 0` or `.width = 0` and `.yauto <> 0` or `.xauto <> 0`, then this can effect the other two sides.
Please take note, that the default values of these attributes with other WSI's (i.e. Windows) are different. The value range is -127 ... +127. The default value for Motif is 5, for Windows 1. Windows-WSI supports `.xmargin` since A.03.07.c, but not `.ymargin`.
- » Gnats 9665: Workaround for Solaris/Motif errors when working together with a different X-display. Pop-up menu now opens itself with Solaris-IDM on a non-Solaris X-display.
- » Gnats 9662: Warnings by the creation or selection of notepages in the editor have been removed. Crashes, which occurred when notepages where made invisible under Suse, no longer occur.

44.4 Core

- » New: An additional space holder %A has been introduced for using the application name in the log or trace filename.
%A is replaced by the application name. Under Microsoft Windows the ending ".exe" is removed from the name.
- » New: DM_LoadDialog now interprets the character "~" (i.e. "~:" in place of "MOD:"). Now a dialog, which is stored in the same folder as the application, can be found without a problem.
- » Gnats 9735: When the format codepage was not UTF8, then no umlaut was displayed in an edit-text object with format. Instead two characters were displayed. This error was caused by the correction of Gnats 9695 and has been corrected.

- » Gnats 9724: The use of path variables when writing, for example, interface files now functions correctly in version A.04.04. This means if the file is not in the path, then the first path is placed at the beginning of the variable.
- » Gnats 9711: Binary writing no longer breaks down in the path resolving over the master import.
- » Gnats 9709: When reading dialogs with `repositid()`, loading isn't aborted but the `repositid` information is rather read over.
- » Gnats 9697: The start order of modules, which changed after Gnat 9577 was corrected, has also been taken care of.
- » Gnats 9695: `applyformat()` with format string now also works correctly when `AppFormatCode-page!=CP_internal`.
- » Gnats 9658: `Setvalue` on `.content` on a `poptext` is now also possible in a `:init-Rule`.
- » Gnats 8697: A static assignment to a path-ID (forward reference to an object) on a shadow attribute resolves the path-ID now correct.

44.5 Editor

- » Gnats 9721: Split areas remember their size even when they are closed for a short period of time.
- » Gnats 9719: When selecting objects in the design area the active element within the browser is moved into the visible area.
- » Gnats 9716: No more external windows are shown when a window is created in the editor and no default window exists.
- » Gnats 9714: The interactive selection of a notepage within the editor makes these active.
- » Gnats 9589: Under Microsoft Windows, objects which possessed raster coordinates, were partially shifted when their size was reduced or enlarged. This problem no longer exists.

44.6 Network

- » Gnats 9712: The DDM for Java-WSI is now also able to process asynchronous Ext-Events from the server side.

44.7 COBOL Interface

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

45 Version A.05.01.a

45.1 New Improvements to the Graphical Editor (Editor III)

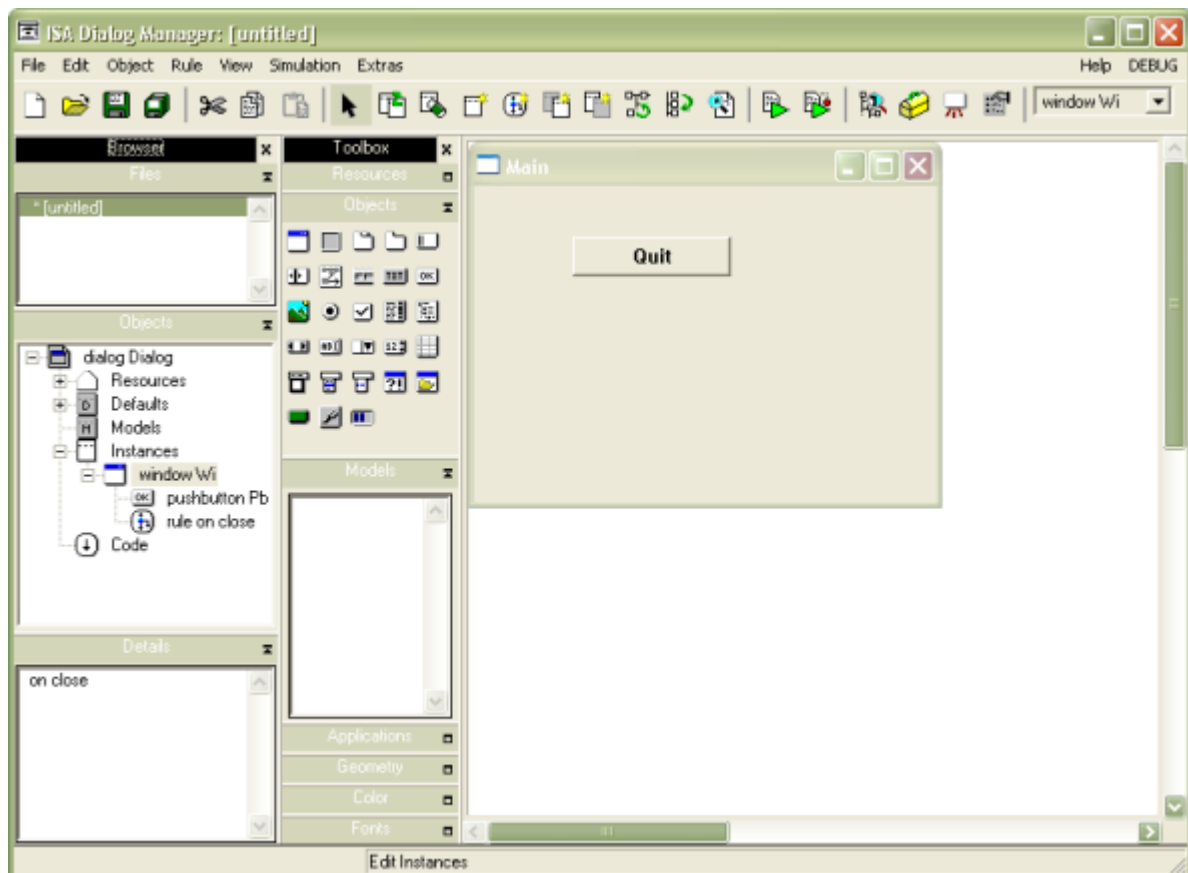
This documentation, in the most part, explains the changes that have been made to the editor documentation. Therefore, each and every detail is not explicitly covered.

45.1.1 Overview

The “Editor III” now allows an even easier and clearer interactive creation of dialogs as this was the case with the “Dialog Manager Editor II”. The goal was to establish a more object-oriented handling while at the same time retain a clear and simple, but nevertheless well-known user structure. For this purpose, the previous secondary windows now have been integrated into the main window and the layout can now partially be configured by the user himself.

The recognition, the access and the creation of objects has been made easier through the increased use of symbols even for the “non-visual” object classes (i.e. one click on the rule-set button in the function bar for creating an object-oriented event rule or method).

The reason for bringing the secondary window into a character area was to make the predefined and user-defined attributes and rules, which are necessary for the development, available at a glance.



Along with the menu, the function bar and a statusbar, the main window consists of four function areas, which can be switched by the user from visible to invisible. These include:

» *“Browser”*

This area deals with the various aspects of selection:

Firstly, with the file that is currently being worked on.

Secondly, it deals with the object and when necessary with other details such as the rules or variants that belong to it.

The central component is the object browser, which reflects the edited dialog/module and allows for operations to be carried out on objects. It also shows the selected object, that is to be edited.

» *“Toolbox”*

Here one can find all the object classes and models needed for creating objects, plus all the additional tools needed for simplifying positioning tasks and for changing color or text properties.

» *“Designer”*

Here the selected object, which is to be visually edited, is displayed.

» *“Properties”*

This allows for the displaying and processing of attributes or rules of the selected object.

45.1.2 In General

The operations, that take place during the selection of an object in the browser area or in the design area depend upon the method (i.e.: selection, deleting, setting of an object, etc.) being used. This is shown either in the function bar or in the toolbox area.

The width of the function areas can be altered by the user. The area can be made invisible via the close button in the title, the function bar or the view menu.





The sub-structure of the function area can be vertically altered by the user. In addition, a sub-area can be closed, for example the *“Resources”* area in the picture above.

The function areas, as is with the toolbars, can be shifted to the left or the right and can also be docked or undocked (docking via shifting is not allowed under Motif). It is now possible to easily dock and undock by clicking on the caption of the function area (i.e. *“toolbar”*).

The arrangement, width and height of the area and the sub-structure is also stored when the configuration is saved (this is true when the saving feature of *“Windowpos”* in the configuration dialog is turned on).

In the object browser, small symbols are shown as visual representation of the object classes, similar to the toolbox. The basic design of the symbols for certain classes follows the pattern described below:

-  For all objects, that are resources
-  For rules, methods and functions

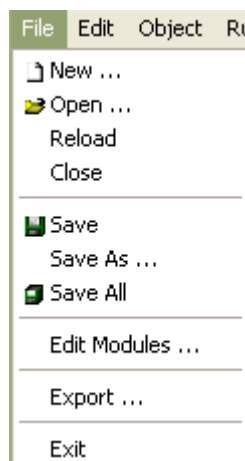
The color of the editor depends on the desktop settings. This is true for MS Windows and now also for Motif.

45.1.3 Menus

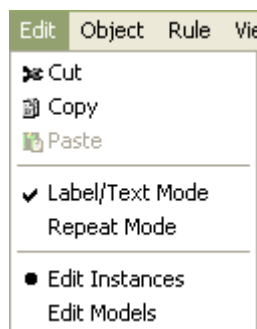


The file menu is used for creating, loading or for saving of modules and dialogs. Also the export dialog for writing interface files etc. is also accessible via this menu.

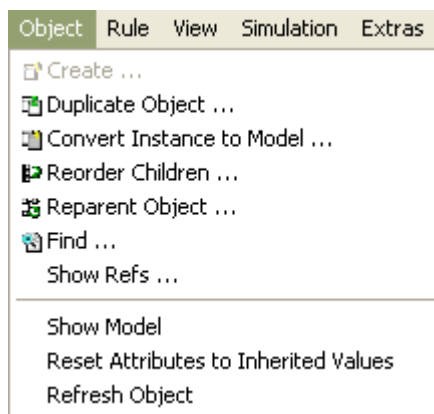
For modularized dialogs it is not necessary to explicitly reload the module for editing purposes. The currently loaded module can be used for editing purposes. Any changes have an immediate effect.



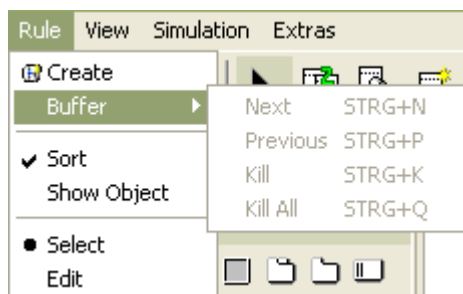
The edit menu allows for the typical cut, copy and paste operations. If you are in the rule window, then the operations to be carried out will affect the rule text - otherwise it affects the selected object. When the Label/Text mode is activated while creating a new object, you will be questioned for the identifier. If the repeat mode is active, then the action will not be reset after the operation has been carried out – therefore, this is suitable for the multiple execution of the same operation on various objects.



The object menu offers operations, which can only be used on objects.

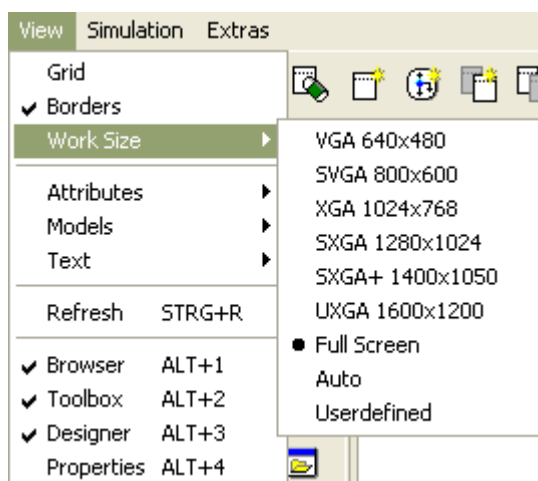


The rule menu contains the functions, that are necessary for the management of the rule buffers as well as display settings.

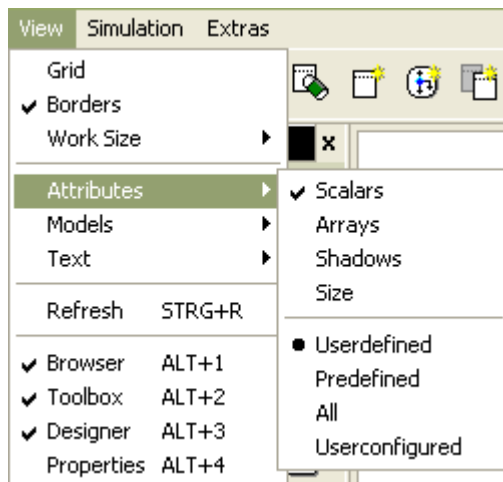


The view menu influences the way things are displayed in the function areas. On the one hand, the areas analog to the function bar can be switched from visible to invisible and vice versa.

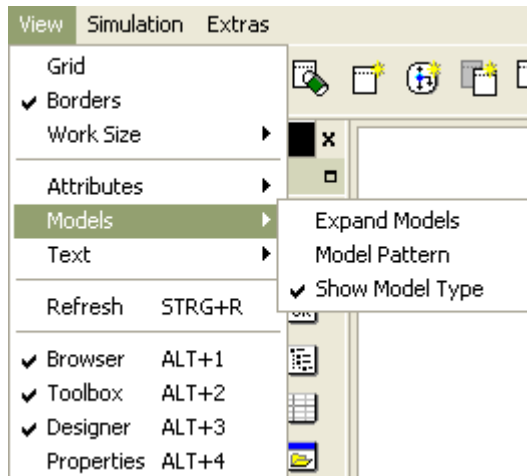
The first three entries affect the presentation in the design area. By clicking on the “Grid” option it is possible to make the grid, in which the selected object and its children are positioned, visible. The size of the design area can be changed to various sizes by clicking on the “Work Size” menu option. The limitation of standard screen sizes can be shown via the “Borders” menu option.



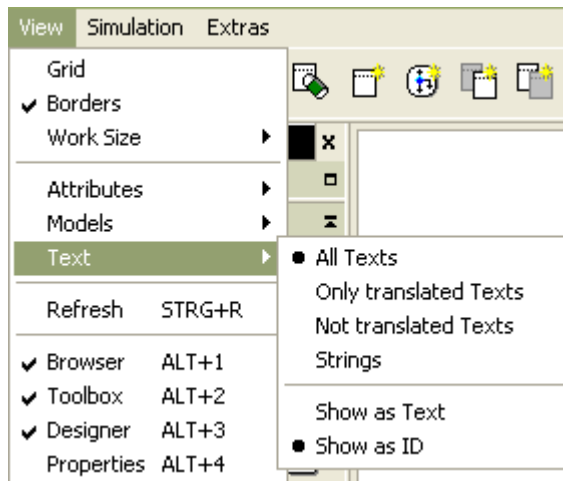
The view of the attribute area in the properties page can be changed via the “Attributes” menu option.



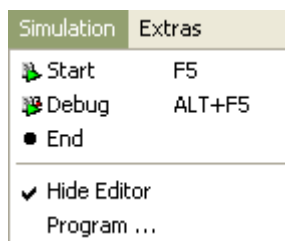
The display of the models within the toolbox allows the filtering with an pattern alongside the masking of object types and children.



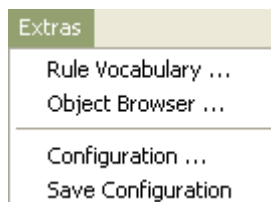
The “Text” sub-menu option controls, analog to the previous resource secondary window, the display of text resources and influences the detail showing of text resources in the browser area.



The “*Simulation*” menu option is used for starting and configuring the simulation mode and also offers the user the possibility of directly calling up the debugger.



The resources of the rule vocabulary, the object browser and the configuration menu of the editor, which were previously “hidden” within the object and rule secondary windows, can now be found under the menu option “*Extras*”.



45.1.4 Function Bar



The function bar is a “shortcut” for actions that can also be carried out via the menus. The bar is comprised of six areas, that are separated by a vertical line:

- » Actions dealing with files
- » Use the clipboard
- » Operations (mode) on an object

- » Dialog simulation
- » Controlling the visibility of the function areas
- » History of the most recently selected object

These are the actions in detail:

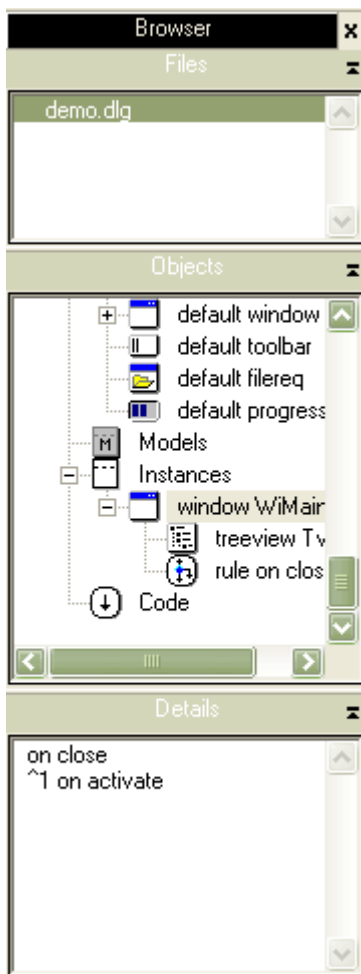
1. Create new file
2. Open a file
3. Save a file
4. Save all altered files
5. Cut (Object / Text)
6. Copy (Object / Text)
7. Paste (Object / Text)
8. Mode: Selection
9. Mode: Duplicate (not via clipboard, but rather directly. The positioning of the duplicated data is located just slightly to the side of the original object).
10. Mode: Delete object
11. Create an object
12. Create a rule for the selected object
13. Mode: To instantiate an object from a model
14. Mode: Make a model from an instance
15. Mode: Reparent the selected object to a new father
16. Mode: Reorganize the selected objects in the father's child list
17. Search an object by his name
18. Switch the visibility of the browser area
19. Switch the visibility of the toolbox area
20. Switch the visibility of the design area
21. Switch the visibility of the properties area
22. Simulate a dialog
23. Debug a dialog
24. Object history: displays the last 10 selected objects independent of the file

In contrast to the menu operations, which can be applied to the selected objects, the mode operations serve as a way for applying an operation to the next object, that is selected by the user.

45.1.5 Browser

The browser area is divided into three areas: the data, object and detail area, and serves for the selection of the edited file, the objects, that were just edited and the structural representation of the edited

file.



The functions are listed separately below:

» *“File Browser”*

Via the file browser the user decides in which dialog or module he is editing at that moment.

» *“Object Browser”*

The object browser displays the object structure of the module/dialog that was just edited and offers analog to the object menu a context menu with corresponding operations. In addition, the selected object is chosen/displayed with it. Operations that are chosen via the function bar or via menus are able to be applied to an object through selection within the object browser analog to the design area. By double clicking the object, the properties area is opened and allows further editing of the object.

» *“Detail Area”*

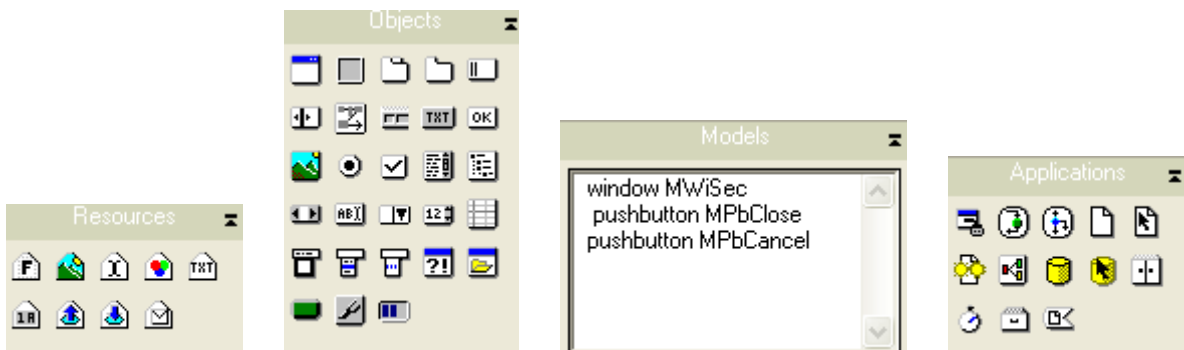
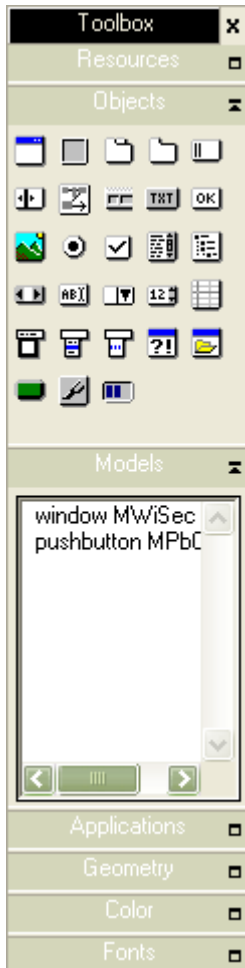
This area is for displaying further details of an object.

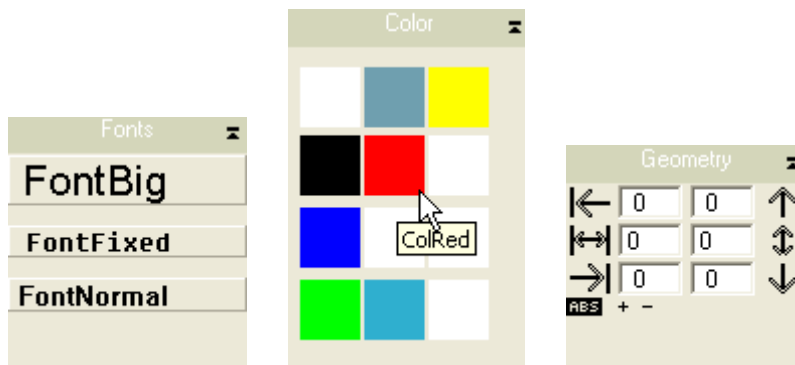
When an object is selected, then all the rules (including the inherited rules) are displayed which allows an uncomplicated access.

Resources possessing variants are displayed in a list and are made selectable for further editing purposes.

45.1.6 Toolbox

The toolbox contains all possible object classes, defaults and models, that can be used when creating an object (instance). These diverse classes are subdivided into the categories resources, (default) objects, models and applications and serve, in comparison to the “old” editor, for quick and easy access to “non-visual” object classes (variable, record, timer, etc.).





45.1.7 Design Area

In contrast to earlier editor versions, the windows of the edited dialogs are no longer displayed as independent windows. The design area in the main window serves for visual editing purposes. In this area defaults, models and instances of object classes, which have a visual representation, can be shown.

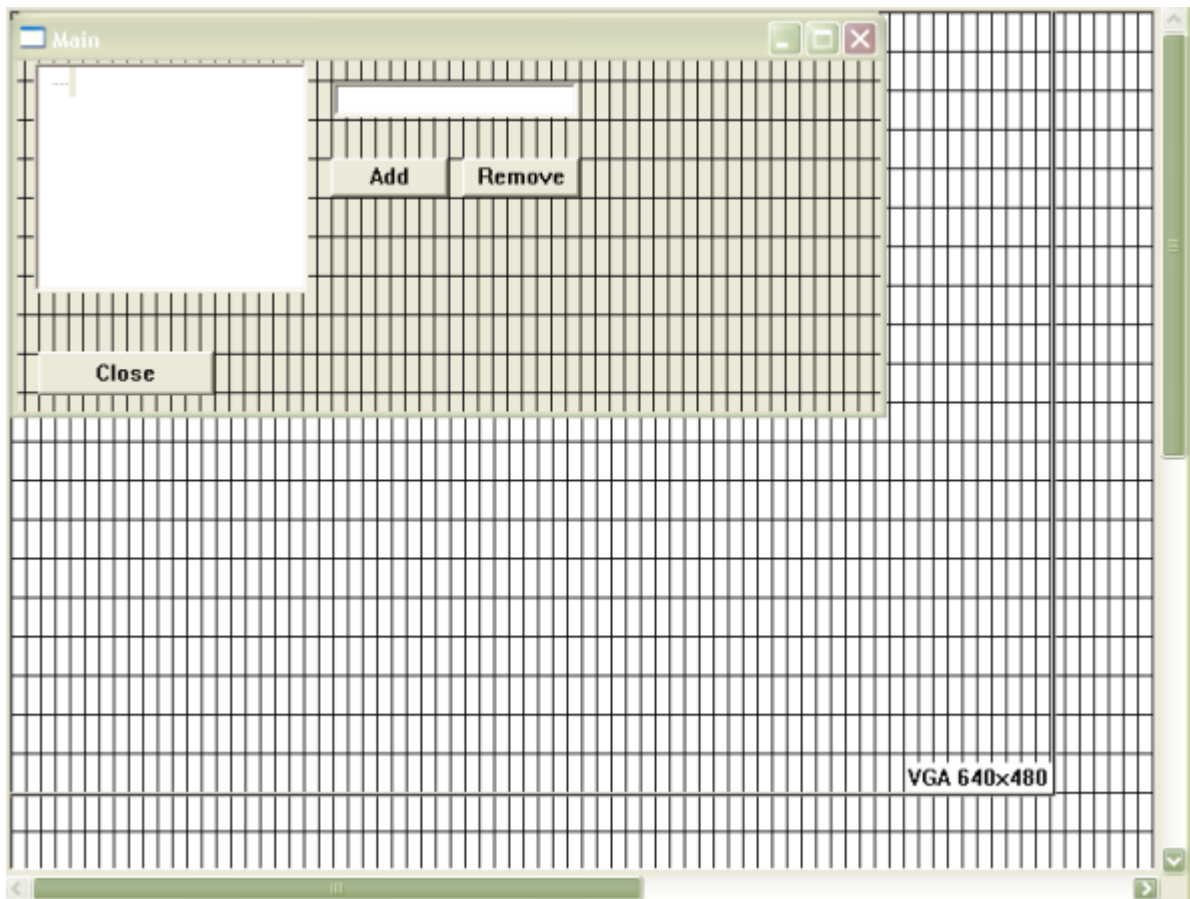
When selecting an object, with the exception of default objects, not just the object itself but rather the entire model or window including all of its children is displayed. Child objects that are switched to `.visible=false` can be made visible, as in the previous version, via the *"Edit"* field within the properties area.

Otherwise, it is still possible to move (position or place in the object hierarchy) or adjust the size of selected objects within the design area.

Via the *"View"* menu option it is possible to adjust three characteristics in the design area:

- » Display of a positioning grid
- » Limit lines for the resolution VGA, SVGA, XGA, SXGA, SXGA+, UXGA
- » Virtual size of the design area - corresponding either to the standard resolution, to the screen resolution, to user-defined values or adjusted to the real size of the design area.

Objects that are made visible by being called up via `querybox()` can not be edited in the design window. However, these can be made visible via the test button (specific page within the properties area).



Specialties of Motif

The decoration is not shown for an window (title, frame, etc..).

Other objects, that can not be displayed include the messagebox, menus, and filerequesters. The menubar of a window, however, are shown.

Specialties of Windows

Under MS Windows the following objects are not visible in the design area:

Menus (including those underneath a window), statusbar, messagebox and filerequester.

45.1.8 Properties

The properties area is a combination of the previous secondary window objects/attributes, resources, application, attributes and rules. The sub-area object still allows the alteration of attributes and properties corresponding to the selected object in a prepared/processed and categorized manner.

The attribute sub-area serves for the editing of user-defined attributes of selected objects, but can also be applied to pre-defined attributes by switching over to the "View" menu.

The rule area behaves as if it is "detached" from selected objects. The selection of an object in the browser or design area has no affect on the rule area. Through buffer management it is allowed to edit

up to 10 rules. The selection of a rule in the detail area or in the object browser brings the rule into view.

When a new rule is created and confirmed, then the object, that was just selected determines the object to which the rule/method belongs to. Afterward, the object to which the rule belongs is stipulated.

Properties

Object

☒ Identifier:
☐ Text
 ☐ Export
 ☒ Edit

Standard

Specific

Text

Geometry

Layout

Scrollbars

Userdata

Comment

☐ Titlebar:
 ☒ yes
 ☐ Userplaced:
 ☐ no

☐ Closeable:
 ☒ yes
 ☐ Iconifyable:
 ☒ yes

☐ Sizeable:
 ☒ yes
 ☐ Iconic:
 ☐ no

☐ Moveable:
 ☒ yes
 ☐ Dialogbox:
 ☐ no

☐ Clipping off:
 ☐ no
 ☐ Topmost:
 ☐ no

☐ Autoclose:
 ☒ yes
 ☐ Preedit:

☐ Icon:

☐ Ignorecursor:
 ☐ no
 ☐ Preedit:

Apply

Revert

Attributes

Identifier	Datatype	Value	Comment
.xauto	integer	1	
.xleft	integer	0	
.xorigin	integer	0	
.xraster	integer	1	
.xright	integer	0	

Rules

WiMain: on close

☐ Export

```

on close
{
  exit();
}

```

Apply

Revert

45.1.9 Other Changes

The source text window is no longer available.

The inheritance view in the object browser is also no longer supported.

45.1.10 Writing a Dialog into a File

The indentation depth of the source code, that is shown in or rather written out of the editor can now be changed via the edit page within the configuration window. In addition, the substitution of 8x-indentation with a TAB character can be turned off.

Please take note of the following:

The default indentation value is 2 spaces for the source code; in the editor without the TAB substitution, in the normal IDM application with TAB substitution. Up to now, this was a combination of 2 and 4 spaces – now this has been standardized.

The indentation of source texts or dialog files, that has written out of IDM applications, as well as the substitution of 8x-indentations via TABs, can now be adjusted. The adjustment of the TAB substitution of every indented output, as well as from tracing, is taken into consideration.

This can take place either via the environment variable `IDM_INDENT` or the option `-IDMindent` when starting the program with normal IDM applications, whereby the latter of the two has priority.

The format is as follows:

```
"<indent>[:<tabsize>[:<traceindent>]]"
```

Source code indentations `<indent>` allow for full numbers `>=0`; by optional sizes `<tabsize>` only the values 0 and 8 are allowed. Tracing indentation `<traceindent>` is also possible and can be controlled, as usual, via the setup object.

In order to achieve an indentation of three character spaces without the tab substitution, then `3:0` must be entered. If `8:8` is entered, then the indentation will occur each time through a TAB. The same thing applies for the editor: '3 characters, no TAB substitution' or '8 characters, TAB substitution'.

45.1.11 Motif

When using system names for color resources it is now possible to address the default colors of Motif. The following system names are available for this purpose:

Symbolic Color Names

- » `ACTIVE_BACKGROUND`
- » `ACTIVE_BOTTOM_SHADOW`
- » `ACTIVE_FOREGROUND`
- » `ACTIVE_SELECT`
- » `ACTIVE_TOP_SHADOW`

- » BACKGROUND
- » BOTTOM_SHADOW
- » FOREGROUND
- » SELECT
- » TEXT_BACKGROUND
- » TEXT_BOTTOM_SHADOW
- » TEXT_FOREGROUND
- » TEXT_SELECT
- » TEXT_TOP_SHADOW
- » TOP_SHADOW

Please take note, that not every Desktop and Window Manager influences the presentation of the colors. With Motif-conform desktops such as CDE, the desktop colors appear to have the same effect as by the Motif applications. With respect to other systems, where this is not supported (Motif 1.2, HP-UX), the user receives calculated default colors derived from the background color.

45.2 Windows

- » New: An additional platform exists: Windows XP.
For this platform MS Visual Studio.net is recommended. For further details please refer to chapter "Changes Resulting from the Introduction of the Windows XP Platform"
- » New: The examples, that were delivered with the IDM contain a project file, that can be run under MS Visual Studio.net (*.dsw) instead of the make files.
- » New: The toolbar object is now able to be resized in a docked condition. As a result, the event 'resize' can now be sent in a docked condition. For further details please refer to chapter "Toolbar" in the "Object Reference".
- » Within a notepage object that has an border (.borderwidth > 0), the child objects, which are aligned either to the right (.xauto = 0; or .xauto = -1) or underneath (.yauto = 0; or .yauto = -1) were positioned either too far to the right or too low – this resulted in the right or the lower border being somewhat cut off. This error has been corrected. However, now the affected child objects are displayed more to the left, smaller or higher than before.
- » It was not possible to read the tracefile when the Dialog Manager was running. This error has been corrected.
- » The two functions DM_QueueExtEvent and DM_SendEvent now use a critical section in order to be sure, that they have been carried out completely before one of the two functions is called up a second time. If one of the two functions is called up in a situation in which a critical section is not permitted, then the use of the critical section can be prevented through the option "DMF_NoCriticalSection"

Attention

A thread, that carries out one of the two functions is not allowed to be stopped.

- » Support for the old Windows interface (NT3.51) and for WIN32s has been removed. Due to this, the option `.options[opt_use_3d]` is no longer of use (this is now ignored and left out when writing a dialog).

45.2.1 Changes Resulting from the Introduction of the Windows XP Platform

The Dialog Manager for WINDOWS XP is reversely compatible to the Dialog Manager for Windows. This also means, that it supports all Microsoft Window systems, that are also supported by the Dialog Manager for Windows. *WINNT 5.1* identifies the Windows XP Platform in the ISA Dialog Manager.

When using the ISA Dialog Manager for Windows XP under older Window versions, the following points should be taken into consideration:

- » The ISA Dialog Manager must be installed on each non-XP computer (no pure copying!).
- » The libraries of the 'redist' indexes must be connected in order to deliver an application.

The Dialog Manager for Windows can be run under Windows XP, but does not support this. This means that the Dialog Manager for Windows don't work together with the new Common Controls DLL (`comctl32.dll`) version 6 or higher.

45.2.1.1 Dialog Manager Objects and "Visual Styles"

Dialog Manager objects can generally be categorized as follows:

- » Dialog Manager objects, that also exist as Microsoft standard objects, and which are displayed via these Microsoft standard objects.
- » Dialog Manager objects, that also exist as Microsoft standard objects, but are not displayed via the Microsoft standard objects
- » Dialog Manager objects, that have no counterpart under Microsoft Windows and for this reason are independently implemented.

By the first two categories the "Visual Styles" are used, which are intended for the object type in question. Unfortunately, no such "Visual Styles" exist for the third category. In order to achieve an XP-look, objects from the third category were brought together from a number of different "Visual Styles".

A number of changes occur when "Visual Styles" are used:

- » The presentation of the object is determined by "Visual Styles". This is especially true with respect to the borders.
- » The feedback with respect to user actions and the presentation of object conditions is also determined by "Visual Styles". This means, that "Visual Styles" control how a selected object is marked, or how the object, that currently has the mouse focus on it, is marked.

- » “Visual Styles” can contain transparent areas. Therefore a repaint forces also a repaint of the father object. This can lead to the actualization of an object lasting longer than it would without “Visual Styles”. For this reason, it is even more important to be sure, that no further changes are made to objects when they are in a visible state.

Special features resulted from this for the following objects:

- » ***Edittext***

When the attribute `.options[opt_rtf]` is set, then the RichEdit control from Windows is used – normally this uses “Visual Styles” only for the scrollbars. Otherwise, the object possesses a normal border just as under Windows 2000. One can compare this with the Microsoft application Wordpad, which uses a RichEdit control.

The way the size is calculated changed slightly. See also chapter “Calculating the Raster Size”.

- » ***Groupbox***

In the standard XP style the border possesses “transparent” areas. Therefore, in order to guarantee, that the border is displayed correctly it is important, that the father object of the groupbox object has the same background color as the groupbox object.

In the standard XP style the border is now wider as it previously was. As a result, groupbox objects, that possess a border are now larger because, according to the definition of the Dialog Manager, the border is drawn on the outside of a groupbox.

- » ***Image***

The setting of the attribute `.fgc` and `.bgc` can lead to a unattractive appearance due to the fact that the colors are not coordinated with the “Visual Styles”.

- » ***Progressbar***

The attribute `.continous` is no longer observed due to the use of “Visual Styles”.

- » ***Tablefield***

The setting of colors can lead to a unattractive appearance because the colors are not coordinated with those of the “Visual Styles”.

45.2.1.2 Calculating the Raster Size

The calculation of the raster size is still carried out through the attribute `.reffont` and therefore the layout does not change.

Note

The height of a single-line edittext object no longer corresponds to the grid height. As a result, it makes a difference, if the attribute `.height` has the value of 1 or 0. In the actual Windows XP Style a single-line edittext object whose attribute `.height = 0` will be lower than one whose attribute `.height = 1` (in other “Visual Styles” this can be different).

45.2.1.3 Object Scrollbar and Scrollbars of Other Objects

Under Microsoft Windows XP it is not customary, that the scrollbar arrows are gray when the slider is either located at the top or bottom of the slider area. For this reason the Dialog Manager for Windows

XP also does not make the arrows grey.

45.3 Motif

- » New: Now the toolbar object is also available under Motif. For further details please refer to chapter “Toolbar” in the “Object Reference”.
- » New: Now the layoutbox can be used under Windows and Motif.
- » New: The progressbar is now available under Motif.

45.4 Core

- » New: The new operators `andthen` and `orelse` have been added. In contrast to the operators `and` and `or`, boolean expressions for these operators are only evaluated when necessary. Expressions, that contain `andthen` and `orelse` evaluate the expression only to a certain extent until the rest of the expression can no longer influence the result. Afterward, the evaluation is stopped and the result is returned. This form of evaluation is also used in C/C++ and JAVA and is labeled “lazy evaluation” (non-strict evaluation).

Example

```
variable boolean X := false;
variable boolean Y := true;
```

```
if X andthen Y then
```

This expression is only evaluated up to variable “X”, because then it is recognized that this expression cannot be *true*.

This comes in handy, as can be seen in the following example, which tests if an object has already been created or not:

```
variable object O := null; !! not instantiated yet
```

```
if O <> null andthen O:DoSomething() then
```

=> no error message, because `andthen` discontinues the analysis after the evaluation of `O <> null`.

- » Gnats 9682: Checking for the available object memory has been corrected. In certain cases this error sometimes caused an Assfail when loading a dialog.
- » Gnats 9661: An Assfail occurred when the `:delete-method` was called up on a `tablefield` after the previous focus became invalid while trying to delete. This has been corrected.
- » Gnats 9608: The depth of the indentation as well as the replacement through tabs can now be set. For further details please refer to chapter “Writing a Dialog into a File”.
- » Gnats 7775: The attribute `.focusitem` on the `listbox` object is obsolete.

45.5 Editor

- » New: The separate windows within the graphical editor have been replaced by a workplace with toolbars. The resulting improvements and changes are explained in the chapter entitled “New Improvements to the Graphical Editor (Editor III)”.
- » Gnats 9588: No jump onto an exported object by selecting a hierarchical inherited child.
- » Gnats 9587: When an object is selected this is then scrolled into the visible part of the design area when possible. This feature can be turned off via the “*Edit/Show Object*” option within the configuration window.
- » Gnats 9580: The setting of color and font types via the toolbox or in the properties area now works again.
- » Gnats 8756: Now it is possible to enter escaped characters for filereq-pattern, as well as for texts by other objects and string values of variables and attributes.
- » Gnats 8477: The German translation of “Size” has been adjusted.

Index

A

- AT-borderraster [132](#)
- AT-constant [130](#)
- AT-display [213](#)
- AT-indexscope [123](#)
- AT-preeditssel [45](#)
- AT-real-screen [213](#)
- AT-repos-id [218](#)
- AT-scope [122](#)
- AT-screen [212](#)
- AT-screencount [211](#)
- AT-typescope [123-124](#)
- AT_borderraster [132](#)
- AT_constant [130](#)
- AT_display [213](#)
- AT_indexscope [123](#)
- AT_preeditssel [45](#)
- AT_real_screen [213](#)
- AT_scope [122](#)
- AT_screen [211](#)
- AT_screencount [211](#)
- AT_typescope [123-124](#)
- attributes
 - display [215](#)

C

- character encoding [28](#)
- code page [28](#)
- code page constant [28](#)

- color_type[] [214](#)
- colorcount[] [214](#)
- CP_appl [29](#)
- CP_display [29](#)
- CP_format [29](#)
- CP_input [29](#)
- CP_output [29](#)
- CP_system [29](#)

D

- display [213-214](#)
- display resource [214](#)
- DM_StrCreate [28](#)
- DT-display [213](#)
- DT-preeditssel [45](#)
- DT_display [213](#)
- Dumpstate [154](#)
 - Callstack [155](#)
 - Errors [155](#)
 - EVENT QUEUE [156](#)
 - Events [156](#)
 - STACK [155](#)
 - THISEVENTS [156](#)
 - Usage [156](#)

E

- entry behavior [45](#)
- explicit activation [45](#)
- export
 - inheritance [31](#)

I

identifier [3](#)
implicit activation [45](#)

P

pointer_height[l] [214](#)
pointer_width[l] [214](#)
preeditset [45](#)
presel_all [46](#)
presel_begin [46](#)
presel_default [45](#)
presel_end [46](#)

R

real_screen [212](#)
reexport [31](#)
 inheritance [31](#)
repos_id [217-218](#)
repos_id[] [217-218](#)
reposid
 keyword [216](#)
ReposId [216-217](#)
 attribute [217](#)
Repository Identifier [216](#)
resource
 display [214](#)

S

Safety-Tracing [161](#)
screen [211](#)
screen[l] [211](#)

screen_height[l] [214](#)
screen_width[l]" [214](#)
screencount [211](#)

T

tablefield
 entry behavior [45](#)
 explicit activation [45](#)
 implicit activation [45](#)

X

xdpi[l]" [214](#)

Y

ydpi[l] [214](#)