

## 1 A.06.03.a

### 1.1 HighDPI Support

The resolution of modern screens is constantly growing. This means that the requirements for the design and quality of the operating elements and interfaces are also changing. Problems are often that applications appear much too small and text and graphics are displayed blurred or pixelated. The resolution, DPI and scaling set in the system, as well as fonts and the size and quality of images, influence the appearance of an application. To ensure that existing applications are also displayed in detail and sharpness on high-resolution screens, they must be prepared for HighDPI.

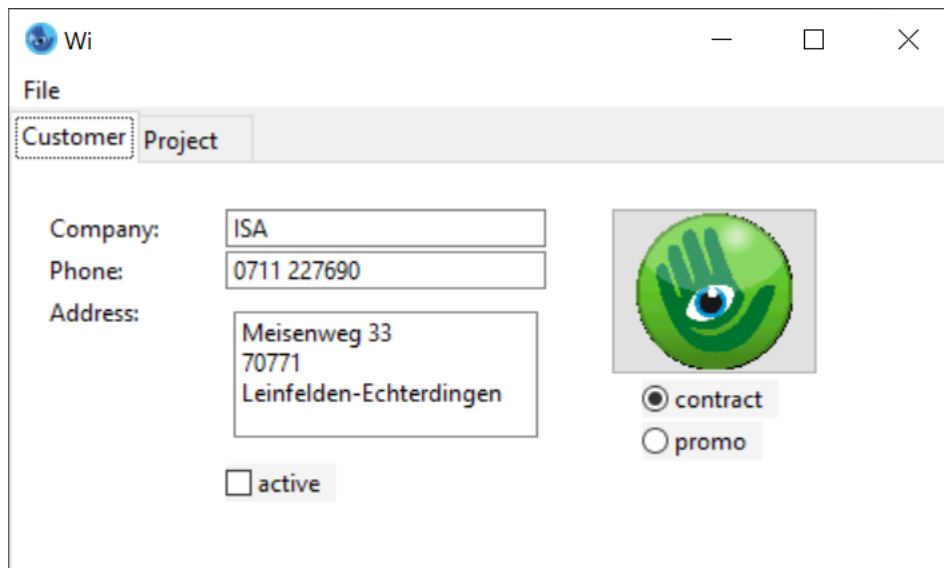


Figure 19-1: without HighDPI support

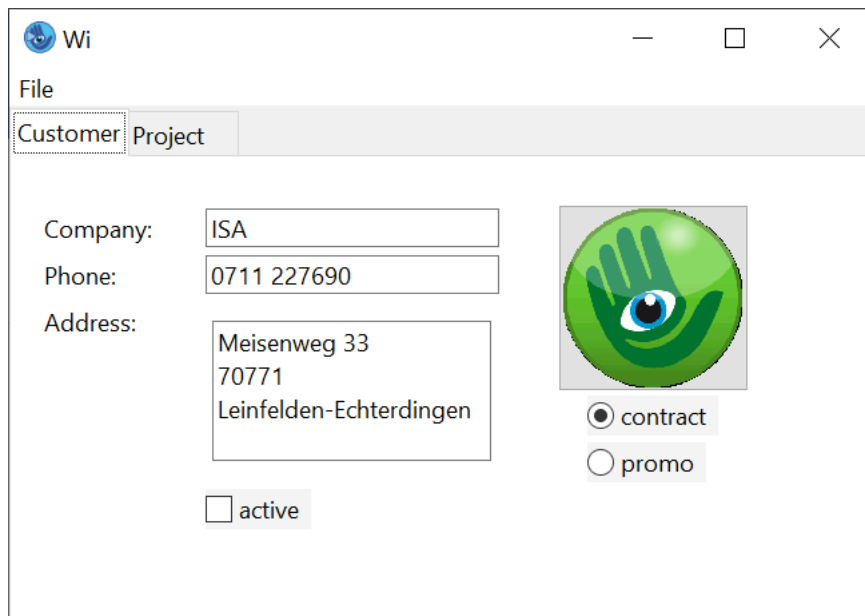


Figure 19-2: with HighDPI support

The IDM for MOTIF, QT and MICROSOFT WINDOWS as of version A.06.03.a (only the IDM FOR WINDOWS 11 on MICROSOFT WINDOWS) now provides the appropriate tools to support high-resolution screens and future-proof applications.

In addition, working with multiple screens has been improved. IDM applications now respect the system's DPI settings and scaling factors. As a result, previous problems such as pixelated, muddy, or too-small applications or reduced font sizes at high resolutions and small screen sizes (the effects can vary by system) no longer occur. The IDM now automatically adjusts geometry, layout, font sizes, graphics and cursors to the appropriate resolution and DPI determined by the system.

The scaling factor is based on the system settings of the respective desktop environment.

However, the user is free to make other settings or intervene creatively through various new attributes, options and mechanisms.

#### 1.1.1 Layout resources

Until now, the available predefined resources in IDM were always WSI- or system-dependent and always required complex adaptations when transferred to other systems. IDM A.06.03.a now offers WSI-independent and HighDPI-compliant predefined layout resources for the first time. This includes [font](#), [color](#) and [cursor](#) resources, which are

available on all systems in similar form and meaning. This offers the enormous added value that dialogs which have been completely converted to the new UI resources can now be transferred to other systems easily and without effort. These resources can be identified by the naming scheme ***UI\*\_FONT***, ***UI\*\_COLOR*** or ***UI\*\_CURSOR***.

***Example:***

```
module Resc

font FontNormal "UI_FONT";
font FontBig "UI_FONT", 16;
font FontFixed "UI_FIXED_FONT";

cursor CurStop "UI_STOP_CURSOR";

!! Background color for group objects
color ColGrp "UI_BG_COLOR";
!! Background color for input objects
color ColInp "UI_INPUTBG_COLOR";
```

The UI\_resource names can also be queried - like all other predefined resource names- at the ***setup*** object via the attributes *.colorname[integer]*, *.fontname[integer]*, *.cursorname[integer]*.

**Cursor**

The uniform Cursor resources have the naming scheme ***UI\*\_CURSOR*** and are defined as follows:

UI_CURSOR	Default cursor usually arrow
UI_IBEAM_CURSOR	Edittext insertion marker
UI_WAIT_CURSOR	Busy indicator, "hourglass"
UI_CROSS_CURSOR	Cross
UI_UP_CURSOR	Up arrow
UI_SIZEDIAGF_CURSOR	Sizing arrow from upper left to lower right
UI_SIZEDIAGB_CURSOR	Sizing arrow from lower left to upper right
UI_SIZEVER_CURSOR	Sizing arrow from top to bottom
UI_SIZEHOR_CURSOR	Sizing arrow from left to right
UI_MOVE_CURSOR	Moving arrow in all directions
UI_STOP_CURSOR	Prohibition sign (crossed out circle)
UI_HAND_CURSOR	Hand symbol (index finger used for pointing)
UI_HELP_CURSOR	Arrow with question mark (for context help mode)

**Fonts**

The unified font resources have the naming scheme ***UI\*\_FONT*** and are defined as follows:

UI_FONT	Font used by default for UI elements (Default or Dialog Font).
UI_FIXED_FONT	Font with a fixed character width.
UI_MENU_FONT	Font used for menus.
UI_TITLE_FONT	Font used in the window title bar.
UI_SMALLTITLE_FONT	Font used in a small/low window title bar.
UI_STATUSBAR_FONT	Font used in the status bar.
UI_NULL_FONT	Behavior as if no font setting was made (null). The WSI uses the default system font provided for the UI element or a system fallback font.

**Color**

The color resources have the naming scheme ***UI\*\_COLOR*** and are defined as follows:

UI_BG_COLOR	Color used by default for general backgrounds of (mostly group) objects. Used for object classes with attribute .bgc/.bgc[], e.g. window, groupbox, notepage, statictext, pushbutton...
UI_FG_COLOR	Color used by default for general foreground/text color of (mostly group) objects. Used for object classes with attribute .fgc/.fgc [], e.g. window, groupbox, notepage, statictext, pushbutton...
UI_INPUTBG_COLOR	Color used by default for backgrounds of input/selection objects. Use e.g. for edittext, lists, tables, poptext, treeview...(e.g. object classes with attributes .bgc/.textbgc)
UI_INPUTFG_COLOR	Color used by default for foregrounds of input/selection objects, such as edittext, lists, tables, poptext, treeview (e.g. object classes with attributes .fgc/.textfgc).

UI_BUTTONBG_COLOR	Color used by default in the background of button-like objects, e.g. pushbutton or image (e.g. object classes with attribute .bgc/.imagebgc).
UI_BUTTONFG_COLOR	Color used by default when foregrounding button-type objects, e.g. pushbutton or image (e.g. object classes with attribute .fgc/.imagefgc).
UI_BORDER_COLOR	Color used by default as border (e.g. at .bordercolor).
UI_ACTIVEBG_COLOR	Color that is used by default as background color of the active element, e.g. for lists, pop text, tables, possibly progressbar, markers etc. Mostly inversion of the normal foreground and background colors of the widget (e.g. object classes with attribute .bgc).
UI_ACTIVEFG_COLOR	Color that is used by default as foreground/text color of the active element, e.g. for lists, poptext, tables, possibly progressbar, markers etc. Mostly inversion of the normal foreground and background colors of the object (e.g. object classes with attribute .fgc).
UI_TITLEBG_COLOR	Color used by default as background color for title bars or borders of windows/dialogs, if supported (e.g. at attribute .titlebgc).
UI_TITLEFG_COLOR	Color to be used by default as foreground/ text color for title bars or borders of windows/dialogs, if supported (e.g. at attribute .titlefgc).
UI_MENUBG_COLOR	Color to be used by default as background color for menus, if supported (e.g. at attribute .titlebgc).
UI_MENUFG_COLOR	Color to be used by default as foreground/text color for menus, if supported (e.g. at attribute .titlefgc).
UI_HEADERBG_COLOR	Color used by default as background color in table headers - if supported.
UI_HEADERFG_COLOR	Color used by default as foreground/text color for table headers – if supported. Use with tablefield (e.g. at attribute .rowheadfgc, .cloheadfgc).
UI_NULL_COLOR	Behavior as if no color (null) was set. WSI draws according to its default palette.

It may happen that not all UI\*COLORS differ from each other. This is due to the fact that most colours are "calculated" from a few basic colours from the desktops.

#### Notes Motif:

In addition to the **UI\*\_COLOR** colors, the system names for color resources have been completed. Newly added are:

```

INACTIVE_BACKGROUND
INACTIVE_FOREGROUND
INACTIVE_TOP_SHADOW
INACTIVE_BOTTOM_SHADOW
INACTIVE_SELECT
SECONDARY_BACKGROUND
SECONDARY_FOREGROUND
SECONDARY_TOP_SHADOW
SECONDARY_BOTTOM_SHADOW
SECONDARY_SELECT

```

#### Notes Windows:

The **UI\*\_COLOR** colors access the Windows system colors. It should be noted that with the introduction of Visual Styles, the individual objects no longer use the system colors, but the colors defined by the theme. Depending on the selected theme there can be strong discrepancies. Therefore, if possible, no colors (or better the **UI\_NULL\_COLOR**) should be set under Windows.

Furthermore, it should be noted that with Windows 10 the color variety has been reduced. For example, the colors **UI\_HEADERBG\_COLOR**, **UI\_HEADERFG\_COLOR**, **UI\_MENUBG\_COLOR**, **UI\_MENUFG\_COLOR**, **UI\_TITLEBG\_COLOR** and **UI\_TITLEFG\_COLOR** are no longer supported. This means that these colors are still available and usable for the user, but they are currently not used by any object by default.

#### Notes Qt:

The **UI\*\_COLOR** colors access the colors of the standard palette offered by Qt. The colors are based on the ColorGroup Normal or Active. The colors of the standard palette are determined by the currently set system style.

In addition to the **UI\*\_COLOR** colors, the predefined colors have been supplemented by further ColorRoles offered by the Qt Toolkit. New additions (in all forms of the ColorGroups) are:

```

BRIGHTTEXT
ALTERNATEBASE
TOOLTIPBASE
TOOLTIPTEXT
LINK
LINKVISITED
PLACEHOLDERTEXT

```

### 1.1.2 Enhancement to the tile resource

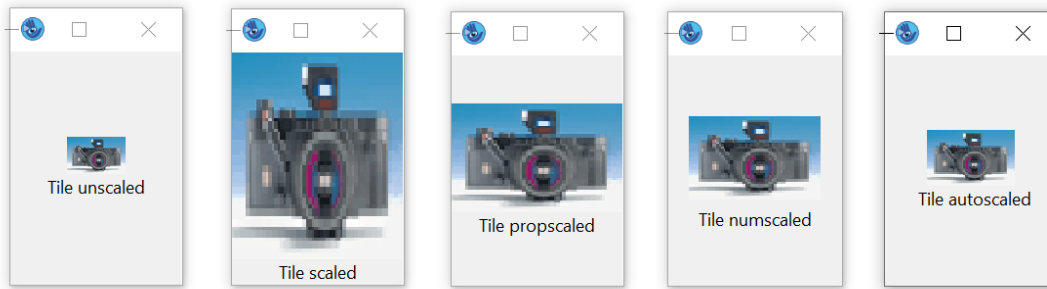
The existing scaling options at the Tile resource have been extended. It is now possible to specify a *.propscale* that determines how a tile should be scaled. Thus, tiles can now be scaled according to the following characteristics:

```

automatic
according to a specific factor

```

proportional  
free in all directions  
based on the DPI value and no scaling



**Figure 19-3:** the same picture with different values of the attribute `.scalestyle` at a system scaling of 150%

The following values are available for selection in the rule language:

Definition on Tile	Dynamic setting	Meaning
noscale	scalestyle_none	The pattern or image is not scaled. <code>.tiledpi</code> has no impact.
scale	scalestyle_any	The height and width of the pattern or image are fully enlarged to fit the available area.
propscale	scalestyle_prop	Height and width of the pattern or image are enlarged to the available area, while height and width proportions of the pattern or image are maintained in any case. I.e. free spaces can be created above and below or left and right.
numscale	scalestyle_num	The scaling of the pattern or image is done by a numerical scaling divider. The scaling is done in quarter steps, i.e. 1.25-fold, 1.5-fold, 1.75-fold, 2-fold, 2.25-fold, 2.5-fold, and so on. A downscaling is done down to a maximum of 0.25-fold.
dpi	scalestyle_dpi	The pattern or image is always scaled according to the set screen scaling.
Not provided	scalestyle_auto	The pattern or image is scaled according to the set screen scaling. A scaling compatible to the previous version takes place. <b>Default value</b>

The **`scalestyle`** can be specified either directly with the **tile definition** or as a **dynamic attribute**.

**Tile definition:**

```
tileSpec ::= <tileBitmap> | "<file path>" | "<graphics resource>" { scale | noscale | numscale | propscale | dpiscale};
```

**Dynamic attribute:**

	Identifier	Data Type
Rule Language	<b><code>.scalestyle</code></b>	enum
C	AT_scalestyle	DT_enum
COBOL	AT-scalestyle	DT-enum
Classification	object-specific attribute	
Objects	<b><code>tile</code></b>	
Access	get, set	Default value scalestyle_auto
changed event	no	Inheritance —

**Example:**

```
dialog D
tile Ti_Rect_Pattern 5,5,
    "####",
    "# #",
    "# #",
    "# #",
    "####" noscale; // corresponds to scalestyle_none
tile Ti_BG „bg.gif“ scale; // corresponds to scalestyle_any
tile Ti_Logo „logo.gif“; // default: scalestyle_auto
...
on dialog start {
    if setup.scale > 100 then
        Ti_Logo.scalestyle := scalestyle_prop; // dynamic
    endif
}
```

See also chapter “tile” in manual “Resource Reference”.

1.1.3 Enhancement to the font resource

The font resource has a new property `.propscale`. This controls whether the horizontal and vertical raster should be set proportionally to the maximum value, which is determined by calculating the value of xraster and yraster of the font raster.

The **property** can be specified either directly whith the **font definition** or as a **dynamic attribute**.

The divisor is used to refine the grid.

Font definition:

```
fontspec ::=
...
...
{ x:= * <xscale> % + <xoffset> | / <xdivider>} { y:= * <yscale> %
+ <yoffset> | / <ydivider>}
{ propscale }
{ r:= "<RefString >" } ;
```

Dynamic attribute:

	Identifier	Data Type
Rule Language	<u>.propscale</u>	boolean
C	AT_propscale	DT_boolean
COBOL	AT-propscale	DT-boolean
Classification	object-specific attribute	
Objects	<i>font</i>	
Access	get, set	Default value false
changed event	no	Inheritance –

See also chapter “font ” in manual “Resource Reference”.

1.1.4 Enhancement to the setup object

	Identifier	Data Type
Rule Language	<u>.tiledpi</u>	integer
C	AT_tiledpi	DT_integer
COBOL	AT-tiledpi	DT-integer
Classification	object-specific attribute	
Objects	<i>setup</i>	
Access	get, set	Default value 96
changed event	no	Inheritance –

This attribute determines for which resolution the images/patterns were designed. The size of an image/pattern is converted to the currently valid DPI value based on this value.

See also chapter “tile ” in manual “Object Reference”.

1.1.5 Enhancement to the image object

For better alignment and extended layout options, the following attributes on the Image object can now be used to control margins or spacing:

	Identifier	Data Type
Rule Language	<u>.xmargin</u>	integer
C	AT_xmargin	DT_integer
COBOL	AT-ymargin	DT-integer
Classification	object-specific attribute	
Objects	<i>image</i>	
Access	get, set	Default value 0
changed event	no	Inheritance ja

This attribute determines the horizontal distance between the border and the display area.

	Identifier	Data Type
Rule Language	<b><u>.ymargin</u></b>	integer
C	AT_ymargin	DT_integer
COBOL	AT-ymargin	DT-integer
Classification	object-specific attribute	
Objects	<b><i>image</i></b>	
Access	get, set	Default value 0
changed event	no	Inheritance ja

This attribute determines the vertical distance between the border and the display area.

See also chapter “[tile](#)” in manual “[Object Reference](#)”.

### 1.1.6 Support of HiDPI image variants

To achieve an optimal design of the interface regarding images and application icons when using the higher resolution, it is recommended to use images adapted to the scaling factor.

Until now, there were only limited options for the Windows and Qt window systems. Up to now, Windows has offered the ICON/BITMAP mechanism and Qt the Icon Resource mechanism for support in this regard. Both mechanisms allow multiple resolution levels to be maintained, but are limited in terms of image type or usage. Such a mechanism does not exist for MOTIF.

The IDM A.06.03.A therefore provides an option for loading multiple resolution levels that can be applied equally to Windows, Qt and MOTIF and is thus also system-independent.

The tile management of the IDM was only able to load one image file (e.g.: tile Ti “smiley.gif”) and did not keep several resolution factors of these (except for the exceptions mentioned above). If there is a scaling of the application, the new loading mechanism for tile files now tries to load the image data for the corresponding scaling factor first. For this purpose, the extension **@*nx*** is added to the file name and in front of the suffix, similar to the Qt mechanism. Here ***n*** stands for the scaling factor between 2 and 9. The number 2 corresponds to a scaling of 150%-249%, 3 of 250%-349%, and so on.

Reasonably the size changes of the variant images should correspond to the scaling factor, whereby a check by the IDM does not take place. So, for example, original image “smiley.gif” (32x32 pixels), smiley@2x.gif (64x64 pixels), smiley@3x.gif (96x96 pixels), smiley@4x.gif (128x128 pixels).

#### Example:

```
tile Ti „smiley.gif“; // at a scaling of 200% the IDM
                     // automatically loads the image
                     // smiley@2x.gif, if present (Windows, Motif, Qt)

                     // at a scaling of 300% the IDM
                     // automatically loads the image
                     // smiley@3x.gif, if present (Windows, Motif, Qt)

tile Ti2 „smiley.ico“ // only Windows! ICON mechanism of Windows
                     // takes the best image for the resolution level
                     // from the .ico file

tile Ti3 „:/smiley“  // only Qt! The resource mechanism of Qt takes
                     // the best image for the resolution level
                     // from the resource file with the given alias
                     //
```

### 1.1.7 Start options

#### **-IDMscale** <integer>

The startup option **-IDMscale** <integer> can be used to enable or disable scaling by the IDM. Under Qt and MOTIF, the value specifies the scaling in %.

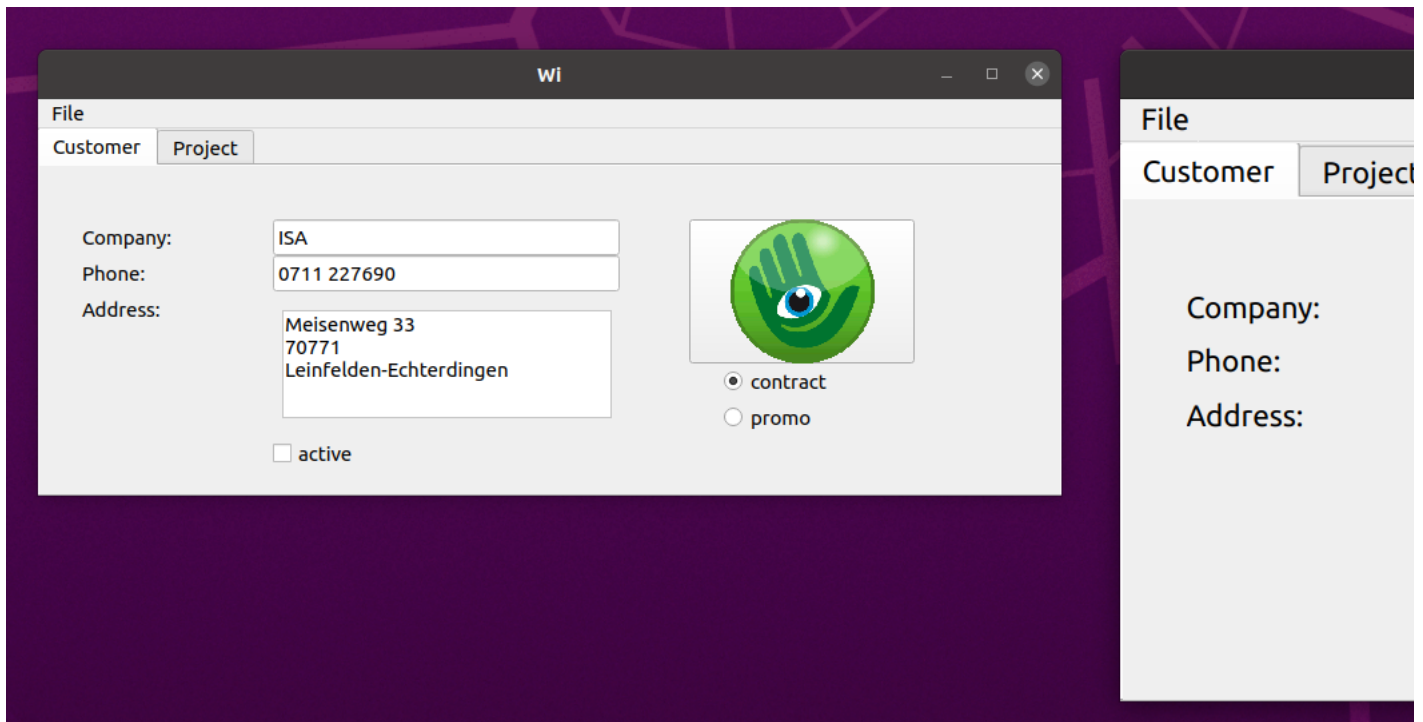


Figure 19-4: left started normally (system scaling 100%), right started with 150% scaling

#### Note

This option should not be used under MICROSOFT WINDOWS. DPI awareness is a property of the application and should be specified in a manifest file. For testing purposes, DPI awareness can be turned on (value: 1) or off (value: 0).

#### Attention:

It is recommended to not use a scaling > 0 and < 100%, as this may impact the display and operation of objects.

#### **-IDMtildepi <integer>**

Graphics are automatically scaled according to the set magnification factor. It is assumed here that the images have been designed for a DPI value of 96. If the images of the application have been designed for a higher resolution, this can be set via the start option **-IDMtildepi <integer>**. The size of an image is then converted to the currently valid DPI value based on this value and scaled accordingly.

Likewise, this setting can be made directly on the **setup** object with the **.tildepi** attribute (see [A.06.03.a](#)).

### 1.1.8 Installation notes

To prepare an application for different resolutions, it may be necessary to keep the images for the different scaling levels. If you want to use this feature and not use the ICON mechanism of WINDOWS or the resource mechanism of QT, you should take care to deliver the image files according to the @ naming (see chapter "Support of HiDPI image variants") with your application or include them in the installation packages if necessary.

### 1.1.9 Geometry and coordinates

Coordinates and dimensions are transformed by the IDM to match the scale factor of the system to ensure a representation consistent with the system settings.

The geometry units are set in **IDM pixel values** or *raster* as before.

The **IDM pixel values** are internally extrapolated and converted into **real pixel values** according to the scaling factor. This scales the application as a whole and maintains ratios and proportions of objects to and among each other.

This conversion is done in both directions. When querying the size and position of objects as well as during event processing, the **real pixel values** from the IDM are also cleaned up and returned in **IDM pixel values**. The scaling factor is then calculated out again. This applies to all pixel-oriented geometry attributes.

*Raster values* are either linked to the (*HighDPI-capable*) character set used or are also set in **IDM pixel values** and then scaled up. The *raster units* can also be determined to **IDM pixel values** based on the underlying font.

### 1.1.10 High resolution support under Motif

The Motif toolkit itself does not contain any special technology to support high resolutions, but if the X display used has the Xrender extension implemented and there is support for XFT fonts with the corresponding fonts, IDM applications for Motif can be run on HighDPI screens respecting the scaling factor of the desktop.

What can be done to check whether support is available on the X-Display/desktop used? For this purpose, the server information of the X-Display can be queried via the **-IDMserver** option:

```
$ idm -IDMserver
ServerVendor The X.Org Foundation
VendorRelease 12013000
Motif at compiletime @(#)Motif Version 2.3.8
Motif at runtime 2003
XRenderVersion 11 (active)
XFT Support yes (active)
Scaling 200% (active)
Screen dpi 96 (tile dpi: 96, scaled dpi: 192)
```

The version of the Xrender extension and its usability (active) is listed as well as the presence of XFT and its usability. The "Scaling" line shows the scaling factor set by the user in the desktop/ display settings, or the scaling factor overwritten by the user.

The dpi information provided by X is mostly the default value of 96dpi and thus does not necessarily reflect the real DPI value available on the monitor. The optionally output "tile dpi" and "scaled dpi" are used to control the DPI values used for the images and controls.

Dynamic scaling change is not provided for Motif applications. Likewise, a conversion of the scaling cannot be detected in the desktop.

## XFT / Font Handling / Motif Font Support

For a GUI application, the basis for different desktop scaling is provided by the fonts used. In contrast to the fonts that could previously be used under Motif/X, which were mostly specified via an XLFD (X Logical Font Descriptor), XFT fonts are also scaled according to the defined scaling of the desktop.

Until now, the fonts that could be used for IDM FOR MOTIF had not automatically adapted to different desktop scaling.

*XFT support is available in the IDM only on Linux platforms. Since most desktops on Linux platforms do not set the default font for Motif widgets properly, the IDM uses **Liberation Sans** as the default font and **Liberation Mono** as the fixed-width font if no font is set in the dialog. This causes texts in IDM dialogs to scale even without a font set. By explicitly disabling scaling (via the `-IDMscale=0` startup option), the old default fonts can be preserved.*

To set an XFT font, only the font name and any additional modifiers are required, as with previous font definitions in the IDM. If the font is available as an XFT font on the system, it will be loaded. The list of XFT fonts available on the system can be viewed in the IDM EDITOR.

Under CDE-compliant platforms such as AIX and HP-UX, the IDM uses `"-dt-interface_system-medium-r-normal-"` as the default font and `"-dt-interface_user-medium-r-normal-"` as the fixed-width font if no font has been set. Depending on the scaling factor, different sizes of the font are used. Between `1%...75% xs` (extra small) is used, between `75%-150% m` (medium) is used, between `150%-250% l` (large) is used and from `250% xxl` (extra extra large) is used. This allows scalability for such applications even if no XFT fonts are available.

### 1.1.11 High resolution supported by IDM for Windows 11

There is now one **IDM for Windows 11** and one for **Windows 10**. The IDM FOR WINDOWS 11 supports DPI awareness and High DPI.

The IDM FOR WINDOWS 11 supports high resolutions of the "PerMonitor V2" model. Since the support of high resolutions is a feature of the application, it must be marked accordingly in the application manifest. Corresponding settings can be found in the IDM examples. The IDM then recognizes this independently and converts the coordinates accordingly to the configured scaling factor. In other words, the pixel coordinates of the IDM are virtual pixels that are converted according to the defined scaling factor.

The manifest files of the IDM examples mark the built applications as "High DPI-Aware". To build an application that is not "DPI-Aware", the entire `<asmv3:application>` block must be removed from the manifest file.

#### Notes IDM for Windows 11

Note for all functions that process Microsoft Windows messages: If the application supports high resolutions, then all coordinates received via Microsoft Windows messages are the real high-resolution coordinates. The IDM, on the other hand, uses coordinates adjusted for the scaling factor. It must therefore be taken into account that in the case of support for high resolutions, the IDM coordinates are generally no longer identical to the Microsoft Windows coordinates. This affects, for example, input handler, canvas, monitor and subclassing functions, as well as the USW implementations.

#### Notes IDM for Windows 10 and 11:

The support of high resolutions requires a different processing of Windows messages. As a result, this means that it is no longer possible to adjust windows to underlying constraints, such as grid coordinates, while moving or zooming. Only after releasing the mouse button the window is adjusted to the next grid coordinate.

Internally defined cursors are now enlarged or reduced to the size required by the system in order to automatically adjust them to the defined scaling factor. This ensures that the cursors are displayed correctly if the application supports high resolutions. Previously, the cursors were either filled with transparent areas or simply cut off if the size did not fit.

Images (**tile** resource) are automatically enlarged according to the defined scaling factor. It is assumed that the images are designed for a DPI value of 96. If the images of the application are designed for a higher resolution, then this can be set in the setup object with the attribute **.tiledpi**. Automatic adjustment applies only to the **tile** resource. Images specified directly via filename at the **.picture** attribute of the image object are not automatically adjusted.

The dialog or message font defined in the system parameters is now used as the default font, since this adapts to the resolution. This can lead to changes in the display of objects that do not have a font set. The mentioned font can be configured via Windows system settings.

The old system fonts are not DPI-Aware. Only the new **UI\*\_FONT** fonts adapt to the set resolution.

The old Windows fonts "ANSI\_FIXED\_FONT", "ANSI\_VAR\_FONT", "DEVICE\_DEFAULT\_FONT" and "SYSTEM\_FONT" cannot be adapted to different resolutions, so their use is discouraged. The use of fonts without size specification is not recommended, because in such a case a font-specific default size is chosen by Windows font mapper, which does not adapt to different resolutions.

### 1.1.12 High resolution support under Qt

To support high resolutions, the IDM enables Qt-side HighDpi scaling as well as the possible rendering of images beyond their actual requested size. I.e. the scaling is done based on the pixel density of the monitor.

Among Linux desktop environments, the support for HighDPI is unfortunately very different and advanced. Since the values for logical and actual pixel density as well as scaling factors are obtained from the Qt framework, less popular desktop environments may provide insufficient values here.

#### Notes for HighDPI preferences

Qt still offers some environment variables that can be used to control the display at high resolutions. When using such environment variables, however, you should be aware that this overrides the behavior of the IDM, which can lead to unwanted display effects.

#### Remarks for tile scaling

Objects are displayed by default with the values specified by the desktop style by the WSI/Toolkit. It can therefore happen in certain situations that the desired settings regarding the size and scaling of tiles are not implemented. This applies to the use of tiles for objects with tabs such as the notebook or notepages as well as menus with menuitems. Although the IDM sets the desired display options, the final display is subject to the display policy of the respective DrawingEngine of the set UI style. This means that the display may differ from the set options depending on the style or options may be ignored completely.

### 1.1.13 Enhancements/changes to the IDMED

Arbitrary selection of the font for the user interfaces or rule code is no longer possible. Instead, the font size can be selected between Small - Normal - Large - Extra Large. Of course, this has an influence on the window size.

Under Motif, XFT fonts are also listed in the fonts setting dialog. In addition, the list of selectable fonts can be reduced to UI fonts, X fonts, and XFT fonts for clarity.



The IDMed, TracefileAnalyzer as well as the debugger now use UI resources.

### 1.1.14 High-resolution support for USW programming

The **DM\_GetToolkitDataEx** function has been improved to support HighDPI. The **Toolkit Dat trutur DM\_ToolkitDataArgs** has been extended for this purpose and now provides detailed information about DPI, scale factor and image. These can be requested via the **AT\_DPI** and **AT\_XWidget** attributes.

#### Note Motif

The USW programmer must convert the coordinates between the IDM (this concerns attributes as well as events) and the Motif values/structures/functions (*X.../ Xt.../ Xm...*). The data required for a conversion can be requested via function **DM\_GetToolkitDataEx** using **AT\_DPI** and the **Toolkit Dat trutur DM\_ToolkitDataArgs** structure (*tile.scale.factor*). Depending on the image type, scaling may also be required.

#### Note for Windows 11

The IDM for Windows 11 supports high resolutions. If an application is released for high resolutions, then the USW objects it uses must also support high resolutions. This means that the USW objects from Windows will have real coordinates (scaled up by the scaling factor), while the IDM will work with virtual coordinates (not scaled up). Practically, not much will change, since the IDM performs the coordinate calculations for the USW object. But it should be taken into account that the **control** call for "UC\_C\_PrefSize" expects real coordinates, since these are normally calculated from toolkit values. If fixed numbers are used here as in the *ucarrow* example, then these must be corrected by the current DPI value (see *ucarrow* example).

If the application was started as DPI-Aware, the task "UC\_I\_clientarea" of the function **UC\_Inquire** now also expects the high-resolution values.

#### Extensions to the C Interface for DM\_GetToolkitDataEx and DM\_GetToolkitDataEx()

In order to support high definition, the C Interface functions "DM\_GetToolkitData()", "DM\_GetToolkitDataEx()" and "DM\_ToolkitDataArgs" (for toolkit data structure) have the following:

##### Attribute **AT\_DPI**

Renders the DPI value of the system or the displayed object. The function "DM\_GetToolkitDataEx()" must be selected with a pointer on a "DM\_ToolkitDataArgs" toolkitdata structure. Further DPI related information is rendered in this structure. It is extended with the data field "dpi" and the sub data structure "scale".

##### Attribute **"AT\_Tile" resp. "AT\_XTile"**

This attribute already exists for the "tile" ressource. Now the function "DM\_GetToolkitDataEx()" can be selected by a pointer on the toolkit data structure "DM\_ToolkitDataArgs" that has been extended with the sub data structure "tile". This structure contains, among other values, the DPI value for which the "tile" ressource has been developed.

##### Attribute **"AT\_IsNull"**

Renders a value unequal "0" if the specified "color" or "font" ressource has previously been defined to "UI\_NULL\_FONT" resp. "UI\_NULL\_COLOR".

See also chapter "Toolkit Dat trutur DM\_ToolkitDataArgs" in manual "Resource Reference".

See also chapter "DM\_GetToolkitDataEx" in manual "C Interface - Functions".

See also chapter "DM\_GetToolkitDataEx" in manual "C Interface - Functions".

## 1.2 Multi-monitor support under Windows

With Windows it is now possible to query the **setup** object for the number of monitors (*.screencount*) and the coordinates of the monitors (*.screen\_x[integer]*, *.screen\_y[integer]*, *.screen\_width[integer]* und *.screen\_height[integer]*). The requested values are dynamic, because one can change the scaling of a monitor or add or remove monitors at any time.

If several monitors are used, they are positioned in a virtual desktop. The coordinates of this virtual desktop can be queried with *.vscreen\_x*, *.vscreen\_y*, *.vscreen\_width*, *.vscreen\_height*. The coordinates of the workspace (without the taskbar) are rendered. The workspaces are included in the calculation of the virtual desktop.

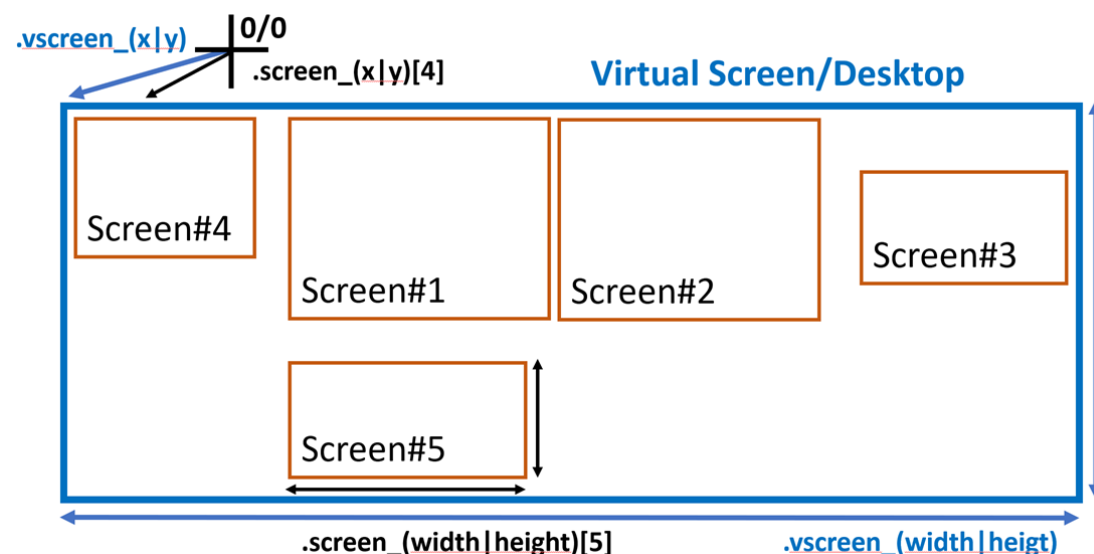


Figure 19-5: relation of *.screen\_\** and *.vscreen\_\** attributes

#### Attention:

If the magnification settings of the monitors differ, please note:

### IDM for Windows 11

The values do not appear consistent. Based on the recommended use of `.xauto = 1` and `.yauto = 1` for windows, the position is converted with the system scaling factor, while the size is converted with the monitor scaling factor. This allows a window to be set to a position and size that fills the entire monitor. The position and size of the virtual desktop are converted using the system scaling factor. The monitor to which a window that spans several monitors is assigned can depend on the monitor on which the window is currently located. For a defined behavior, coordinates should therefore only be changed in the invisible state.

### IDM for Windows 10

Microsoft Windows internally scales applications that are DPI-unaware. In this case, the different scaling factors lead to misrepresentations. To avoid this, either all windows should be opened on the primary monitor only, or all monitors should have the same scaling factor.

#### Remark for Windows 10 and 11 with several monitors:

In Windows, after adding/removing a monitor or changing the display settings, you may encounter the following problems:

No moving frame appears when moving a toolbar object. It can also happen that a rectangle the size of the **toolbar** frame is displayed with an incorrect background on another monitor.

Undocked **toolbar** objects do not adjust to changes in DPI value. Both when moving to another monitor or changing the magnification.

The problem is due to Windows, which on the one hand incorrectly converts the coordinates when drawing to the desktop and on the other hand does not send important messages (`WM_DPICHANGED`) to so-called tool windows and does not change the window DPI value.

After the screen saver was active or you logged in again (and after a reboot), the problem no longer occurs.

## 1.3 Compatibility

### 1.3.1 Motif

The IDM FOR MOTIF now displays an application enlarged when system scaling is active. This behavior can be turned off using the `-IDMscale=0` startup option.

### 1.3.2 Windows

If screen scaling is active, the grid of the IDM FOR WINDOWS 11 cannot be proportional to the scaling, as the font size is specified in logical points. The more pixels are available, the better the logical value corresponds to the real one.

### 1.3.3 Qt

IDM FOR QT now has HighDPI scaling enabled. This means that when system scaling is active, the application is now inevitably and automatically adjusted, which now affects not only fonts but also geometry.

## 1.4 Core

**IDM-13192:** A *finish* event of application objects affected the handling of finish events of dialog/module objects. This resulted in duplicate execution of *finish* rules on the dialog/module and could affect exiting or not exiting the event loop. This misbehavior is now fixed.

**IDM-13191:** Invalid memory accesses in the rule resolver no longer occur when resolving paths in value expressions. Also a possibly omitted attribute access should not happen anymore.

**IDM-13072:** The "sharing" of imported modules, which can be activated via environment variable **DM\_SHARED\_MODULES** to release imports with the same identifier from the presence of the same import in the super module, is restricted to ensure consistency regarding *start/stop/init/finish*, dialog membership and the visibility state of objects. Only modules in the same dialog can be shared!

#### Attention:

The sharing of modules via **DM\_SHARED\_MODULES** is not a standard feature of the IDM. It makes more sense to use "use" to import modules and thus safely prevent multiple loading of modules in complex dialogs.

**New:** Added the *hinttext* attribute to the *edittext* and *poptext* objects, which can be used to display a placeholder or hint text, if the object has no content set yet. The *poptext* object must have the *edittext* style set..

	Identifier	Data Type
Rule Language	<i>hinttext</i>	string
C	AT_hinttext	DT_string
COBOL	AT-hinttext	DT-string
Classification	object-specific attribute	
Objects	<i>edittext</i> , <i>poptext</i>	
Access	get, set	Default value —
changed event	yes	Inheritance ja

#### Attention:

If there is a space as content, the placeholder or hint text will not be displayed, this also is the case some formats.

#### Particularities:

This attribute is supported only by WINDOWS and QT.

#### Remark for Windows:

On Microsoft Windows the attribute is currently supported on the following objects:

**edittext** object if the attributes *.multiline* and *.options[opt\_rtf]* have the value *false*

**poptext** object if the attribute *.style* has the value *edittext* or *listbox*

The hint text is displayed when the object does not have the focus and the content is empty.

#### Remark for Qt:

If the *.multiline* attribute of the **edittext** object has the value *true* and layout attributes (e.g. *.alignment*, *.xmargin*, *.ymargin* etc.) are set, these are ignored when the hint or placeholder text is displayed. The hint or placeholder text is always attached to the top left.

## 1.5 Windows

**New; Attention:** From now on there is one IDM FOR WINDOWS 11 and one IDM FOR WINDOWS 10. The IDM FOR WINDOWS 11 supports **DPI Awareness** and **High DPI**.

**New;** MICROSOFT VISUAL C++ 2022 is now supported with IDM FOR WINDOWS 11.

**IDM-13194:** After a drag and drop action, the error message "WSI error processing clipboard.value" appeared. The prerequisite was that a *:setclip* method existed that changed the clipboard.

**IDM-13215:** Under Windows, the geometry attributes were not inherited if they were changed interactively on the model.

**IDM-12999:** An exception occurred in a **UIAutomation** test application when the **opt\_hscroll** option was set on a **poptext** object. This is now avoided by not releasing the **UIAutomation** object for the poptext list until the poptext **UIAutomation** object is no longer needed. Since then, the **UIAutomation** object for the poptext list was released when the poptext list was no longer needed.

**IDM-12967:** It could happen that the first tracing message of **UIAutomation** was empty. The problem occurred when the first object was made visible only after the **UIAutomation** tracing was switched on.

**IDM-13188:** When a window was maximized from the minimized state, then objects bound at the bottom were positioned incorrectly, they appeared too far down.

**IDM-11065:** When the application was running under Terminal Services, no warning beep sounded. Now the Windows **Beep** function is used under Terminal Services. Attention, the change only affects the sounds generated by the IDM.

**IDM-10294: Mnemonics** are now no longer triggered on objects on inactive **notepage** pages.

**New:** The attribute names *AT\_Font* and *AT\_XFONT* were introduced as an alternative to *AT\_wsidata* for the **ToolkitData** functions at **font** Ressource. The usage is identical to *AT\_wsidata*.

**New:** The attribute names *AT\_Color*, *AT\_Tile* and *AT\_Widget* introduced as an alternative to *AT\_XColor*, *AT\_XTile* and *AT\_Widget* for the **ToolkitData** functions. The usage is identical to the corresponding attribute with the X.

**IDM-12697:** Problems occurred during an installation to a network drive. Attention, the installation to a network drive is not intended. The attempt will usually lead to an abort after all files have already been unpacked. If no abort occurs the unlocking of the files takes place. This can only be done in the last phase of the Windows installation. This phase is executed by the Windows Installer under the "SYSTEM" account and usually has no access to network resources.

**Changed:** For the IDM Editor, more information about the available fonts is now displayed, so that it can be recognized which options are possible. In addition, the *type* parameter is now taken into account during selection.

**New:** The **opt\_show\_ticks** option is now supported by the scrollbar object.

**Fixed:** The setup object was not created/initialized in the WSI. As a result, an initially set *.overridecursor* was not adopted.

**New:** The **font** resource has been extended with new options for *black*, *demibold* and *light*. In addition, the *roman* font family is now supported.

**IDM-13057:** The moving frame of a **toolbar** object did not appear over the **toolbar** object when a scaling factor was set. This problem is fixed now.

#### Remark Windows 10 and 11 with multiple monitors:

In WINDOWS, after adding/removing a monitor or changing the display settings, you may encounter the following problems:

No moving frame appears when moving a toolbar object. It can also happen that a rectangle the size of the **toolbar** frame is displayed with an incorrect background on another monitor.

Undocked **toolbar** objects do not adjust to changes in DPI value. Both when moving to another monitor or changing the magnification.

The problem is due to Windows, which on the one hand incorrectly converts the coordinates when drawing to the desktop and on the other hand does not send important messages (*WM\_DPICHANGED*) to so-called tool windows and does not change the window DPI value.

After the screen saver was active or you logged in again (and after a reboot), the problem no longer occurs.

**New; IDM-11904, IDM-11753:** The IDM FOR WINDOWS now offers multi-monitor support. See [chapter 1.2 "Multi-monitor support under Windows"](#)

In order for a window to be maximized correctly in a multi-monitor configuration, the calculation of the maximum size must be left to Microsoft Windows. This can cause windows to be covered by the taskbar.

**Fixed; Attention:** Change of the calculation method for the raster width.

With IDM version A.06.03.a, the calculation of the raster width has been substantially changed. If no reference string is set, the raster width is now calculated from an internal reference string ("M") to avoid excessive width growth due to excessively wide letters within a font.

For compatibility reasons, however, the *opt\_fontraster\_compat* option, the **-IDMfontraster\_compat** startup option, or the *IDM\_FONTRASTER\_COMPAT* environment variable can temporarily reactivate the old raster width calculation (with the drawback that excessive width growth may occur again).

When using the *opt\_fontraster\_compat*, the size calculation is partly based on the system font, which is not High DPI capable, so High DPI capable applications created with IDM FOR WINDOWS 11 should not use this option.

**Note:** The **DM\_GetToolkitData/DM\_GetToolkitDataEx** attributes *AT\_Tile*, *AT\_XTile* and *AT\_wsidata* were extended by **Toolkit Data** **DM\_ToolkitDataArgs** and are assigned as follows:

*tile.gfxtype*

Set to the GFX Type that fits best.

*tile.datatype*

Is set to the value that previously had to be queried via *AT\_DataType*.

*tile.data*

Set to the data, the exact type depends on *tile.gfxtype* resp. *tile.datatype*:

HICON: if *DM\_GFX\_ICO* resp. *DMF\_TikDataIcon*

HBITMAP: if *DM\_GFX\_BMP* resp. 0  
HMETAFILE: if *DM\_GFX\_WMF* resp. *DMF\_TkDataIsWMF*  
HENHMETAFILE: if *DM\_GFX\_EMF* resp. *DMF\_TkDataIsEMF*

*tile.mask*

Set to the monochrome mask if one is present. This can only occur with *DM\_GFX\_BMP*.

*tile.palette*

Set to the color palette if one exists.

*tile.dpi*

The DPI value for which the images loaded by the IDM were designed.

*rectangle*

Set to the size that matches the requested DPI value. See remark.

*dpi*, *scale.dpi* and *scale.factor*

Are set to match the requested DPI value. See remark and notes for *AT\_DPI*.

#### Remark on the requested DPI value:

The requested DPI value is determined from the following information in the specified order:

If the *DM\_TKAM\_handle* bit is set in the *argmask*, then the DPI value of the Microsoft Windows control is determined whose Windows handle (HWND) is in the *handle* element.

If the *DM\_TKAM\_scaledpi* bit is set in the *argmask*, then the system DPI value is determined (corresponds to the scaling factor of the primary monitor).

Otherwise, the original image size, as the DPI value for which the images were designed.

**Attention:** This does not mean that the image data has the appropriate size, these may need to be scaled.

The return value remains as before, i.e. corresponds to *tile.data*.

Note: The elements of the [Toolkit Dat trutur DM\\_ToolkitDataArgs](#) are assigned as follows when querying the attribute *AT\_DPI* in [DM\\_GetToolkitData/DM\\_GetToolkitDataEx](#) attribute *AT\_DPI*:

*dpi*

Set to the default DPI value used by applications that are not DPI Aware. This value is defined by MICROSOFT WINDOWS as constant *USER\_DEFAULT\_SCREEN\_DPI* (value: 96). This value is used by the IDM for its pixel coordinates.

*scale.dpi*

Set to the current DPI value of the object, it corresponds to the return value of *AT\_GetDPI*. This value is dynamic for IDM FOR WINDOWS 11 if the application is DPI Aware. Otherwise, DPI Unaware or IDM FOR WINDOWS 10, the value is fixed at 96.

*scale.factor*

Integer percentage value calculated from *scale.dpi* and *dpi*.

The return value (converted to int) corresponds to *scale.dpi*. To get the DPI value for a Microsoft Windows control the *handle* element in the [Toolkit Dat trutur DM\\_ToolkitDataArgs](#) can be set to the Windows handle (HWND) of the control. In the *argmask* the bit *DM\_TKAM\_handle* must be set and as object 0 or the *setup* object must be specified in the [DM\\_GetToolkitDataEx](#) call.

New: With [DM\\_GetToolkitData](#) or [DM\\_GetToolkitDataEx](#) the current DPI value can be queried under MICROSOFT WINDOWS with *AT\_GetDPI*. Depending on the *objectID* parameter the following values are returned:

ID of a visible object:

The DPI value that applies to this object. Value depends on the monitor on which the object appears.

NULL ID or ID of the *setup* object:

The system DPI value or if the Microsoft Windows Window Handle (HWND) is passed in the *data* parameter the DPI value of this object.

otherwise:

Either a random old value (if object not visible) or 0 and an error message.

The return value must be converted to "int". Attention, all values are dynamic and can be changed by the user via the control panel. If the application is not DPI Aware (for example IDM FOR WINDOWS 10) then the default DPI value of 96 is always delivered.

Changed: Removed linker option to subsystem version so that the application is no longer set to XP compatibility.

Fixed: The *scrollbar* object had not updated the *.real\_width* and *.real\_height* attributes when no size was set and the *.arrows* attribute had the value false.

Changed: Additions to the fontraster compatibility setting for the Windows WSI. This is normally set to false/inactive. This compatibility setting can be set/activated in the following ways:

start option *-IDMfontraster\_compat*

option *setup.options[opt\_fontraster\_compat]*

For information on raster calculation, see the corresponding chapters in the Windows WSI.

New: [DM\\_GetToolkitData](#) or [DM\\_GetToolkitDataEx](#) supports the *AT\_Font* and *AT\_XFont* attributes on the *setup* object under Microsoft Windows. The font handle of the used default font is returned (the return value must be converted to *HFONT*). The font is scaled for the system DPI value. In the *data* parameter another DPI value can be specified as integer (conversion: (DM\_Pointer) (size\_t) dpi).

New: [DM\\_GetToolkitData](#) or [DM\\_GetToolkitDataEx](#) supports the attributes *AT\_Font* and *AT\_XFont* on the IDM objects under Microsoft Windows. The font handle of the used font is returned (return value must be converted to *HFONT*). The font is scaled for the current DPI value of the object.

New: [DM\\_GetToolkitData](#) or [DM\\_GetToolkitDataEx](#) supports the *AT\_IsNull* attribute at the font and color resource under Microsoft Windows. This can be used to query whether the resource was defined to *NULL* (*UI\_NULL\_FONT* or *UI\_NULL\_COLOR*). The resource was defined to *NULL* if the return value is not 0.

Changed: The [DM\\_GetToolkitDataEx](#) function returns a *NULL* value for font and color resources under MICROSOFT WINDOWS if the resource has been defined to *UI\_NULL\_FONT* or *UI\_NULL\_COLOR*, respectively. This affects the following attributes:

*AT\_wsidata*, *AT\_Font*, *AT\_XFont*:

The return value for *UI\_NULL\_FONT* becomes (*HFONT*) 0.

*AT\_Color*, *AT\_XColor*:

The return value for *UI\_NULL\_COLOR* becomes (*COLORREF*) -1L.

*AT\_Tile*, *AT\_XTile*:

The return value for `UI_NULL_COLOR` becomes (`HBRUSH`) `0`.

**New:** The `.scalestyle` attribute on the `tile` resource is now also respected by the `treeview`, `notebook/notepage`, `menuitem` and `window` objects.

**New:** The two attributes `.real_x` and `.real_y` corresponding to `.real_width` and `.real_height` introduced.

**Changed:** At the `tile` resource the predefined names `OLD_UPARROW`, `OLD_DNARROW`, `OLD_RGARROW`, `OLD_LFARROW`, `OLD_REDUCE`, `OLD_ZOOM` and `OLD_RESTORE` were removed, the OEM resource names without `OLD_` must be used.

**Fixed:** If the `.reffont`, or `.xraster` / `.yraster` attribute was changed on the `toolbar` object, the size of the toolbar object was not adjusted if `.sizeraster = true`.

**Fixed:** If on the `poptext` object the `.format` attribute was changed dynamically, then the list content was not adjusted.

**Fixed:** If the attribute `.imagefgc` was set on the `image` object, then this color was mistakenly also used as text color.

**Changed:** There was a visual difference in some objects between a set font that does not exist and a font set to `null`. This difference no longer occurs.

**New:** Added DPI awareness information to the trace file header. As well as more warnings and error messages during DPI initialization that indicate unsupported settings.

## 1.6 Motif

**Fixed:** The layout box did not immediately adjust the children when scaling down.

**Fixed:** The conversion of the positions of windows and toolbars (`dock_window`) is done as far as possible taking into account real window decorations in order to position the window/toolbar at the same place again when toggling `.visible` twice.

**Fixed:** Insensitive text at the `image` object is now drawn grayed out with shadows cast analogous/similar to other MOTIF widgets from and including MOTIF-2.3.

**New:** The workarea specified by newer desktop and window managers (e.g. by menus or dock areas) are taken into account by the IDM and thus reduce the usable screen size.

## 1.7 Qt

**IDM-13186:** When switching background tiles in the `groupbox`, it was not considered whether the `groupbox` had a virtual size.

**New:** The `opt_show_ticks` option is now supported by the scrollbar object.

**New:** Added system colors to color resource, that were previously not noticed.

**New:** `DM_GetToolkitData` now supports `AT_XTile` and `AT_XWidget`. `AT_XTile` always returns the type `DM_GFX_QPIXMAP` with `QPixmap`.

**Fixed:** If a tile was subsequently set dynamically at the `splitbox`, it could happen that it was not drawn.

**Fixed:** If a tile was subsequently set dynamically at the `layoutbox`, it could happen that it was not drawn.

**Fixed:** The `window` object did not respect a dynamic change of `.reffont` and as a result did not resize.

**Fixed:** The `:openpopup()` method had displayed the context menu under Qt only at the mouse pointer and not directly at the object.

**Fixed:** When changing the direction of the `scrollbar` in the visible, the size of the scrollbar was not adjusted.

## 1.8 DDM (Network)

**New:** The `ssh://` scheme now also supports the OPENSSH command. This scheme is used with the `.exec` attribute to determine how the server side is started. First it is checked if the `libssh` DLL is available. If it is not available it is checked if an `ssh` command is available. Additionally there is the scheme `sshlib://` to use only `libssh` and `sshcmd://` to use only OPENSSH. The disadvantage of the command is that no password can be used. OPENSSH must be configured to connect without a password (see the `ssh` man pages).

**New:** OPENSSL version 3 is now supported.

**Fixed:** Old OPENSSL DLLs could not be loaded under certain circumstances because function `SSL_CTX_set_options` is not available. The problem could occur if build and deployment system had different OPENSSL versions. Now optionally the old `SSL_CTX_ctrl` function is used.

**Fixed:** The network side behaved differently between UNIX and WINDOWS when an open port should be used by a second process. Now under Windows the socket option `SO_REUSEADDR` is no longer set, so that an error also occurs if the port is already in use.

**Fixed:** Verification of SSL certificates could fail if default search paths were not loaded.

**Fixed:** The application side stopped responding when SSL errors occurred. Now, if errors occur during the creation of the SSL context, the process is no longer terminated, but the connection continues to be established and thus the resulting errors are logged on both sides.

## 1.9 Debugger

**IDM-13002:** If a too large trace file was loaded in the IDM TRACEFILE ANALYZER section by section, it could happen that the application stopped responding. This is now prevented.

**New:** The IDM TRACEFILE ANALYZER now allows section-by-section loading of trace files that are too large.

**Fixed:** The IDM TRACEFILE ANALYZER under MICROSOFT WINDOWS did not fully utilize the display area in some cases. Editor

## 1.10 Editor

**Changed:** The preview of a tile first determines the real path from the edited module to better support the display of image names with “~”.

**Changed:** Editing options of the `.scale` attribute for `tile` resources changed to enum/selection for `.scalestyle`.

**New:** Propscale property for font resources now editable.

**New:** The IDM TRACEVIEW ANALYZER under MICROSOFT WINDOWS had partly not fully utilized the display area.

## 1.11 USW

IDM-13057 (Keine Änderung): Note regarding High DPI support for IDM on Windows 11: Upon starting the IDM as DPI Aware, the IDM for Windows 11 expects to receive the real, high resolution values from the task **UC\_I\_clientarea** of the function **UC\_Inquire**

See also chapter in manual "[Programming Techniques](#)", "ProgrammingTechniques"

Fixed: The USW examples for Windows had not displayed a reasonable image in **uctable** for quite some time. The examples now display icons in the correct size.

Fixed: Keys used by the IDM for navigation (e.g. `TAB`, `cursor 1eft`, etc.) were not forwarded to child objects of the USW widget under MICROSOFT Windows although the key was marked as required in the **UC\_I\_querykey** Inquire.

Fixed: The surrounding IDM object of USW widgets now has no frame. The USW widget should be responsible for its own frame.

## 1.12 IFD

Changed: The IFD option is no longer supported.

## 1.13 Demos

IDM-13189: The README file of the samples only stated that the project "usw\usw" could not be started. A reference to the project "usw\lucsample" was missing.

New: USW example **ucgrid** introduced to show *HiDPI* support for USW widgets.

Fixed: The **ucframe** widget of the USW examples had not taken into account the set font when calculating the size under Windows, thus headline was cut off.

## 1.14 Documentation

New: Added chapter "HighDPI Support" in the "Programming Techniques" manual.

Changed: Documentation for **DM\_GetToolkitData** and **DM\_GetToolkitDataEx** substantially revised. Documentation for **Toolkit Dat trutur DM\_ToolkitDataArgs** created in the manual "C Interface - Basics".

IDM-13198: Important notes that the **opt\_hscroll** option on **poptext** is problematic under MICROSOFT Windows, now included in the corresponding chapters of the attributes manual for **.options[enum]** as well as objects manual for **poptext**.

---

Support

Product

Company

© 1987 – 2023; ISA Informationssysteme GmbH, Leinfelden-Echterdingen, Germany