ISA Dialog Manager

Automatic Testing and Accessibility

A.06.03.b

This manual explains features that the ISA Dialog Manager provides to support automatic user interface testing and the development of accessible applications.

ISA Informationssysteme GmbH

Meisenweg 33

70771 Leinfelden-Echterdingen

Germany

Notation Conventions

DM will be used as a synonym for Dialog Manager.

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

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

Description Mode

All keywords are bold and underlined, e.g.

variable **integer** **function**

Indexing of Attributes

Syntax for indexed attributes:

[I]

[I,J] meaning [row, column]

Identifiers

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

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

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

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

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

<digit> ::= 1 | 2 | 3 | … 9 | 0

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

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

Table of Contents

1 Introduction

Availability

IDM FOR WINDOWS only, since IDM version A.06.01.e

UI Automation (hereafter usually abbreviated as UIA) is a MICROSOFT framework to facilitate accessibility and automated testing of user interfaces.

The focus of the IDM enhancement is to provide the infrastructure for automated testing.

It makes the user interface elements (UIA Elements) accessible in a tree structure. A UIA Element has properties, also depending on its type (UIA Control Type). UIA Elements may also have several UIA Control Patterns that provide additional properties or interaction modes (UIA Control Pattern Methods). Events indicate whether changes have occurred on a UIA Element.

This documentation serves to illustrate the functionality of the IDM regarding the UI Automation support and is mostly only interesting for developers of UI Automation Clients. It does not provide “programming documentation”, but merely documents the support provided by the IDM. For the use of UI Automation, please refer to the Microsoft documentation.

2 Mapping of IDM Object Classes to UIA Control Types

The IDM has as a design principle the use of the object classes (controls, widgets) provided by the system or toolkit to implement the interface objects and thus to keep operation and appearance as compliant as possible with the system.

Many of the IDM object classes are covered by the “UI Automation Support for Standard Controls” from MICROSOFT. If possible and required, the IDM adds the necessary properties.

With regard to UI Automation support in the IDM, the following categories can be distinguished:

D Particular object classes cannot be equipped with additional UIA support through the IDM, but have sufficient support through the Microsoft default implementation. These include e.g. ***menubox***, ***menuitem***, ***menusep***, ***messagebox*** and ***filereq***.

E Most of these object classes get complementary UIA support through the IDM, especially to facilitate their identification.

X Object classes implemented by the IDM and object classes with inadequate Microsoft default implementation get individual UIA support. These are mainly the object classes ***image***, ***table-field***, ***splitbox***, ***layoutbox***, ***progressbar***, ***poptext*** and ***toolbar***.

U For special object classes such as ***control***, ***canvas*** and USW objects, UIA support lies in the hands of the application programmer or object implementer.

The following table shall serve as a reference point for the mapping of IDM object classes to UIA Control Types as well as the functionality provided by the Pattern affiliation:

Category	IDM Object Class	UIA Control Type	Supported UIA Control Patterns
U	***canvas***	Pane	To be implemented by the canvas programmer
E	***checkbox***	CheckBox	Toggle, Invoke
U	***control***	Pane	Depends on the OLE/control object
E	***edittext***	Edit	System-dependent – typically Value, Text, Text2 Multi-line texts with scrollbars usually also have the Scroll Pattern

Category	IDM Object Class	UIA Control Type	Supported UIA Control Patterns
E	***edittext*** with *.options[opt_rf] = true*	Document	System-dependent – typically Value, Text, Text2
D	***filereq***	Window	System-dependent
X	***groupbox***	Pane Child: Group	Invoke, Scroll (with virtual areas)
X	***image***	Button Children: Text, Image (if present)	Invoke, Toggle
X	***layoutbox***	Pane	Scroll (with virtual areas)
E	***listbox***	List Children: ListItem	Scroll, Selection, Invoke
D	***menubox***	Menu	ExpandCollapse
D	***menuitem***	MenuItem	Toggle, Invoke, ExpandCollapse, SelectionItem – depending on *.style* attribute
D	***menusep***	Separator	–
D	***messagebox***	Window	System-dependent
E	***notebook***	Tab	Selection
E	***notepage***	TabItem/Pane	SelectionItem
X	***poptext***	ComboBox Children: Edit, List, Text (according to *.style*)	ExpandCollapse, Value sowie Selection – depending on *.style* attribute
X	***progressbar***	ProgressBar	RangeValue
X	***pushbutton***	Button	Invoke
E	***radiobutton***	RadioButton	SelectionItem
X	***rectangle***	Button	Invoke

Category	IDM Object Class	UIA Control Type	Supported UIA Control Patterns
X	***scrollbar***	ScrollBar	RangeValue
X	***scrollbar*** (slider)	Slider	RangeValue
X	***spinbox***	Spinner	RangeValue, Value, Selection – depending on *.style* attribute
X	***splitbox***	Pane Children: Pane	Visible split areas are sub-Panes with Transform Pattern
X	***statictext*** (insensitive)	Text	Text
X	***statictext*** (sensitive)	Button	Invoke
E	***statusbar***	StatusBar	–
X	***tablefield***	Table Children: ListItem	Grid, Table, GridItem, TableItem, Invoke, Scroll, Selection
X	***toolbar***	ToolBar	Dock, Transform, Window (depending on *.docking* attribute), Invoke
E	***treeview***	Tree	Scroll, Selection
E	***window***	Window	Window, Transform

UI Automation support may be changed by MICROSOFT anytime, so that changes concerning the mapping of UIA ControlTypes, Patterns, Properties and Events are always possible.

3 Supported UIA Properties and Control Pattern Methods

Basically, the Properties, Methods and functionality are defined by the “UI Automation Specification”. Therefore, this chapter has rather informative character to show the relation to the IDM.

UIA Property/Method	IDM Attribute	Remarks
AcceleratorKey	*.accelerator*	The keyboard shortcut of the accelerator is returned.
AccessKey	*.text*	The **Mnemonic** character in the text is returned, for example *"Alt+E"* for *.text "&Edit"*.
AutomationId	*.label*, *.acc_label*	*.acc_label*, if set, takes precedence over *.label*. The *.label* may be supplemented by the numbering “:[<I>]” to achieve unique identification on sibling level.
BoundingRectangle	*.x*, *.y* *.width*, *.height* *.xauto*, *.yauto*	Position and size of the object.
ClassName	–	**Attention** System names or IDM-specific names that may differ from version to version and have no 1:1 correlation with the IDM object class.
ClickablePoint	–	Point inside the BoundingRectangle for executing a mouse click.
ControlType	*.class* (and other attributes like *.options*, *.style*, *.arrows*)	The mapping to a UIA Control Type does not solely depend on the object class, but possibly also on further object attributes.
FrameworkId	–	Provided through the UIA implementation by MICROSOFT.
HasKeyboardFocus	*.focus*	Returns whether the object has the keyboard focus.

UIA Property/Method	IDM Attribute	Remarks
HelpText	*.toolhelp* resp. *.statushelp*	If no help text is set in *.toolhelp*, *.statushelp* will be returned.
IsEnabled	*.sensitive* resp. *.real_sensitive*	Returns the operability respectively sensitivity of the UIA Element.
IsKeyboardFocusable	–	Object is sensitive and can receive keyboard focus.
IsOffscreen		UIA element is visible, but may be covered by overlying windows or elements
IsPassword	*.format*	For a format with hidden formatting (format string starts with "S") this Property returns *true*.
LocalizedControlType	–	Provided through the UIA implementation by MICROSOFT.
LabeledBy	–	If the IDM object is preceded by an insensitive ***statictext***, the appropriate UIA Element is returned.
Name	*.text*, *.title*, *.acc_text* With substructures (***listbox***, ***treeview***, content) often also *.content*, *.content[I]*, *.content[R,C]*	Name of the UIA Element. May be overwritten by the *.acc_text* attribute.
NativeWindowHandle	*AT_WinHandle*	Usually the same as the HWND that **DM_GetToolkitData()** returns for *AT_WinHandle*.
Orientation	*.direction*, *.docking*	For IDM object classes like ***splitbox***, ***scrollbar***, ***toolbar*** and ***tablefield*** the respective direction is returned here.
OrientationType_Horizontal	*.direction = 2*	
OrientationType_Vertical	*.direction = 1*	
OrientationType_Horizontal	*.docking = dock_<up|down|window>*	
OrientationType_Vertical	*.docking = dock_<left|right>*	

UIA Property/Method	IDM Attribute	Remarks
ProcessId	–	Provided through the UIA implementation by MICROSOFT.
ProviderDescription	–	Provided through the UIA implementation by MICROSOFT.
RuntimeId	–	Usually provided through the UIA implementation by MICROSOFT.

3.1 Value Pattern

UIA Property/Method	IDM Attribute	Remarks
ValueIsReadOnly	*.editable* *.editable[R,C]*	Input field or table cell is editable. A ***listbox*** always returns *true*.
Value	*.content* *.content[I]* *.content[R,C]*	Returns the (formatted) content of an input field, a ***poptext***, ***treeview*** or ***listbox*** item, or a table cell.
SetValue()	*.content* *.content[I]* *.content[R,C]*	Modifies the value of the input field or table cell.

3.2 RangeValue Pattern

UIA Property/Method	IDM Attribute	Remarks
ValueIsReadOnly	–	Range value may be changed via SetValue().
Value	*.curvalue*	Range value of a ***spinbox*** or ***scrollbar***.
Minimum	*.minvalue*	Lower Range limit.
Maximum	*.maxvalue*	Upper Range limit.
SetValue()	–	Modifies the Range value.

3.3 Scroll Pattern

UIA Property/Method	IDM Attribute	Remarks
HorizontallyScrollable	–	Horizontal scrolling is possible.

UIA Property/Method	IDM Attribute	Remarks
HorizonalScrollPercent	–	Computed value from *.vwidth* and real width in %.
HorizontalViewSize	–	Computed value from *.vwidth* and real width in %.
VerticallyScrollable	–	Vertical scrolling is possible.
VericalScrollPercent	–	Computed value from *.vheight* and real height in %.
VericalViewSize	–	Computed value from *.vheight* and real height in %.
Scroll()	–	Scrolls relatively.
SetScrollPercent()	–	Scrolls to a specific % position.

3.4 ScrollItem Pattern

UIA Property/Method	IDM Attribute	Remarks
ScrollIntoView()	–	Scrolls the element into the visible area.

3.5 Transform Pattern

UIA Property/Method	IDM Attribute	Remarks
CanMove	*.moveable*	***Window*** or ***toolbar*** can be moved.
CanResize	*.sizeable*	***Window*** or ***toolbar*** can be decreased and increased in size. Or the ***splibox*** area can be decreased and increased in size.
CanRotate	–	Not covered by the IDM.
Move()	–	Allows moving a ***window*** or ***toolbar***.
Resize()	–	Allows resizing a ***window***, ***toolbar*** or ***splibox*** area.
Rotate()	–	–

3.6 Grid Pattern

UIA Property/Method	IDM Attribute	Remarks
ColumnCount	*.colcount*	Returns the number of visible columns in a table.
RowCount	*.rowcount*	Returns the number of visible rows in a table.
GetItem()	–	Returns the UIA Element of a cell.

3.7 GridItem Pattern

UIA Property/Method	IDM Attribute	Remarks
GridItemColumn	–	Column index (from *0* and without invisible columns).
GridItemColumnSpan	–	Is always *1* because cells cannot be merged in the IDM.
GridItemRowCount	–	Row index (from *0* and without invisible rows).
GridItemRowSpan	–	Is always *1* because cells cannot be merged in the IDM.

3.8 Selection Pattern

UIA Property/Method	IDM Attribute	Remarks
SelectionCanSelectMultiple	*.multisel* *.selstyle* *.selection[]*	***Listboxes*** and tables may support multiple selection.
SelectionIsRequired	–	Is a selection required?
GetSelection()	–	Returns the selected UIA Elements.

3.9 SelectionItem Pattern

UIA Property/Method	IDM Attribute	Remarks
SelectionItemIsSelected	.active .activeitem	Returns *true* for an active item.

UIA Property/Method	IDM Attribute	Remarks
AddToSelection()	–	Adds an item to a multiple selection.
RemoveFromSelect()	–	Removes a selected item from a multiple selection.
Select()	–	Selects this item.

3.10 Toggle Pattern

UIA Property/Method	IDM Attribute	Remarks
ToggleState	*.active* *.state*	Activation or state of a checkbox, image object or menuitem in checkbox style.
Toggle()	–	Toggles the states between *checked*, *unchecked* and *indeterminate*.

3.11 ExpandCollapse Pattern

UIA Property/Method	IDM Attribute	Remarks
ExpandCollapseState	*.open* (***Treeview***)	–
Collapse()	–	Close a subtree.
Expand()	–	Open a subtree.

3.12 Window Pattern

UIA Property/Method	IDM Attribute	Remarks
CanMinimize	*.iconifyable*	***Window*** can be minimized.
CanMaximize	*.maximizable*	***Window*** can be maximized.
IsTopmost	*.top_most*	***Window*** is always in front of all other windows.
WindowIsModal	*.dialogbox*	***Windows*** with *.dialogbox* = *true* are modal windows.
WindowInteractionState	–	–

UIA Property/Method	IDM Attribute	Remarks
WindowVisualState	–	***Window*** state (minimized, maximized, etc.).
Close()	–	Close the ***window***.
SetWindowVisualState()	–	Enables to minimize or maximize the ***window***.
WaitForInputIdle()	–	–

3.13 Dock Pattern

UIA Property/Method	IDM Attribute	Remarks
DockDockPosition	*.docking*	Returns the docking position of a ***toolbar***.
SetDockPosition()	–	Allows to change the docking of a ***toolbar***.

3.14 Table Pattern

UIA Property/Method	IDM Attribute	Remarks
TableRowOrColumnMajor	*.direction*	Indicates the orientation of the table.
GetColumnHeaders()	–	Returns the UIA Elements of the visible column header cells.
GetRowHeaders()	–	Returns the UIA Elements of the visible row header cells.

3.15 TableItem Pattern

UIA Property/Method	IDM Attribute	Remarks
GetColumnHeaderItems()	–	Returns the corresponding UIA Elements in the header range *[row,1] … [row,.colheader]* of the table.
GetRowHeaderItems()	–	Returns the corresponding UIA Elements in the header range *[1,col] … [.rowheader,col]* of the table.

4 UIA Object Identification

The key Properties defined by UI Automation for the identification of UIA Elements are AutomationId, Name, ControlType and RuntimeId.

In order to enable the most consistent and language-independent identification possible, the IDM object label is therefore assigned as AutomationId for most IDM objects. In case of ambiguities, the IDM-typical indexing ":[<No>]" after the label for *No >= 2* is used for inherited identifiers. The AutomationId may be overwritten via the *.acc_label* attribute. In this case, however, the application programmer must ensure uniqueness.

Often the Name Property is also used, which typically represents the visible static text string (not the dynamic content). This may also be overwritten by the application programmer with the *.acc_text* attribute.

Through these Properties, objects in a UI Automation Element Tree usually are uniquely recognizable.

The RuntimeId is generally based on the Window Handle and may therefore change when attributes are modified. This makes it less suitable for reliable object identification.

5 Particularities

5.1 Edittext

To change the text, the SetValue() method of the Value Pattern has to be used. This triggers a *char-input* event in the IDM. However, the *modified* event to signal a change is not sent until the ***edittext*** loses focus. More consistent with normal operation is the “simulation” of keystrokes. In general, the focus should be directed to the input field via the SetFocus() method of UIA.

5.2 Poptext

In addition to the usual method of changing the selection with the Select() method of the SelectionItem Pattern, the SetValue() method of the Value Pattern may also be used. For an editable ***poptext***, if the value is listed in the poptext list, selection is carried out and the corresponding *select* and *activate* events are generated. Otherwise, the content of the input field is changed and the events *charinput* and *modified* are triggered.

5.3 Formatted Input

If a **format** is applied to an input field, Text and Value always return the displayed text including formatting characters. Therefore access to the contents of password fields with a hidden format is not possible. A complete modification should also happen through the SetValue() method and must not contain any formatting characters. Otherwise, test tools should execute the “simulation” of keyboard inputs.

5.4 Client Area

IDM object classes may consist of multiple UIA Elements. Grouping objects typically have a parent element and a client area which gets the suffix “#client” as AutomationId.

5.5 Virtual Areas

Grouping objects obtain a virtual area and optional scrollbars by setting the attributes *.vwidth* and *.vheight*. The Client UIA Element then additionally gets the Control Patterns Scroll and ScrollItem on the child objects to enable changing the visible area and scrolling a child into the visible area.

The UIA Methods Scroll() and ScrollIntoView() allow to control the displayed pane. It is not possible to control scrolling through the inside ScrollBar UIA Elements.

5.6 Splitbox

The individual visible areas are accessible as UIA Child Elements of type Pane with the name suffix “#client[1]” … “#client[n]” and can be changed in size using the Resize() Control Pattern Method.

5.7 Menus

Menus in the menu bar as well as pop-up menus are not redefined by the IDM but covered by the Windows standard implementation. Typically, the UIA Elements of the menus, submenus and menu items are only accessible when they are opened. Context menus are located under the Desktop UIA Element. The menus of menu bars after opening can be found directly under the UIA Window Element.

5.8 Messagebox and Filereq

With the **querybox()** call these are constructed from the standard controls and can typically be found as UIA Control Type Dialog or Window with the set window title.

5.9 Image Object

To enable recognition of the displayed image, the ***image*** object is represented in UI Automation through the UI Element Button with the child elements Image and Text. The Image element gets, if present, the alternative text of the used ***tile*** resource as UIA Name Property.

5.10 Toolbar Object

A docked ***toolbar*** does not possess the Transform Pattern, hence it cannot be altered in size and position.

5.11 Treeview Object

Reselecting the same element with the Select() method of the SelectionItem pattern does not generate a new *select* event.

5.12 User Interaction vs. Automation

The functionality available in UI Automation for automating a user interface does not cover all the possibilities offered to users by mouse and keyboard operation.

» Keyboard inputs as well as the activation of ***accelerators*** and **mnemonics** are not directly provided. However, this also means that actions or event rules e.g. based on `Enter` (`Return`), `Esc`, cursor keys, etc. or a function key cannot be invoked directly by UIA Methods. Therefore the simulation of special keys might be recommended, e.g. to enable the triggering of *deselect_enter*

event rules.

» The interaction patterns of the mouse (moving, pressing and releasing a mouse button) are neither part of the UIA methods. An interaction of the form “Click on poptext – move down with the mouse button pressed – release over the 3rd item” can e.g. hardly or not at all be simulated. A mouse click usually corresponds to the UIA Methods Invoke() or Select(), but sometimes also to Expand() or Collapse() or many more. Thus user interaction depends on the Control Type.

In general, user interaction and automation differ in possible actions, atomicity, and state dependencies.

6 Notes About HP UFT 12.52

The test tool HP UNIFIED FUNCTIONAL TESTING (UFT) provides support for UI Automation testing as of version 12.52. However, UFT does not fully support all UIA Control Types, but is limited to the most important Control Types and Control Patterns. Therefore, this chapter is only informative and should not replace reading the UFT documentation.

The UIA (UI Automation) support of UFT can be outlined as follows:

- The “UI Automation” add-in must be activated.
- For identification with the Object Spy, the “UI Automation Mode” must be enabled.
- Object identification is based on the UIA Properties Name and AutomationId.
- The following UIA Control Types are recognized and converted into an appropriate UFT Object Model:

UIA Control Type	UFT Object Model
Button	UIAButton
Calendar	UIACalendar
CheckBox	UIACheckBox
ComboBox	UIAComboBox
Edit	UIAEdit
HyperLink	UIAHyperLink
List	UIAList
RadioButton	UIARadioButton
Slider	UIASlider
Tab	UIATab
DataGrid	UIATable
SplitButton	UIASplitButton
Window	UIAWindow
Table	UIATable
... andere ...	UIAObject

- The following UIA Control Patterns are supported by UFT via special methods on the UIA Objects:

UIA Control Pattern	UFT Methods
ExpandCollapse	.Expand .Collapse
Grid	.GetItem
Invoke	.Click
RangeValue	.Decrement .Increment .SetValue
Scroll	.Scroll .ScrollDown .ScrollUp .ScrollLeft .ScrollRight .SetScrollPercent
ScrollItem	.ScrollIntoView
Selection	.Select .AddToSelection .RemoveFromSelection .GetSelection
SelectionItem	.Select .AddToSelection .RemoveFromSelection
Table	.GetRowHeaders .GetColumnHeaders
TableItem	.GetRowHeaderItems .GetColumnHeaderItems
Transform	.Move .Resize .Rotate
Toggle	.Set
Value	.SetValue
Window	.Maximize .Minimize .Restore .Close

This means that not all IDM object classes are “distinguishably” recognized as test objects with own UIA Class by UFT. This includes the IDM object classes:

- ***toolbar***
- ***statusbar***
- ***scrollbar***
- ***splitbox***
- ***menubox***, ***menuitem***, ***menusep***
- ***spinbox***
- ***progressbar***
- ***image***
- ***rectangle***
- ***control***
- ***layoutbox***, ***groupbox***, ***splitbox***
- ***edittext*** (RTF)

However, these should be recognized by UFT as generic UIAObjects and capable of being sufficiently tested and manipulated by the supported Pattern Methods.

It should be noted that with UI Automation Microsof covers the domains of “accessibility” and “test automation”, but does not provide full and correct capturing of user interaction. In this respect, the “recording” of user interaction by UFT in UI Automation Mode is often incomplete and may also be faulty.

This can be seen, for instance, when “recording” a menu interaction on a submenu. UFT generates the following malfunctioning script, which does not contain a “Click” on the menu:

```
UIAWindow("Main window").UIAMenu("Application menu").UIAObject("File").Expand
UIAWindow("Main window").UIAMenu("File").UIAObject("Submenu").Expand
```

A working script however would look like this:

```
UIAWindow("Main window").UIAMenu("Application menu").UIAObject("File").Click
UIAWindow("Main window").UIAMenu("File").UIAObject("Submenu").Click
UIAWindow("Main window").UIAMenu("File").Select "Submenu;Menu A"
```

“Recording” in Windows Mode produces the following working script:

```
Window("Main window").WinMenu("File").Select "File;Submenu;Menu A"
```

7 Attribute .acc_label

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

Definition

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

Access
get, set

changed event
yes

C
Identifier: AT_acc_label
Data type: DT_string, DT_text

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

Inheritance
yes

Default value
""

Classification
standard attribute

pport of attribute by objects

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

Remark

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

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

8 Attribute .acc_text

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

Definition

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

Access
get, set

changed event
yes

C
Identifier: AT_acc_text
Data type: DT_text, DT_string

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

Inheritance
yes

Default value
null

Classification
standard attribute

pport of attribute by objects

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

Remark

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

9 Enhancement to the Tile Resource

For the access of IDM user interface objects via the UI-Automation Interface on MICROSOFT WINDOWS, the ***tile*** resource has been enhanced. ***Image*** objects that use a ***tile*** resource as picture get an alternative text to facilitate automation and accessibility.

This alternative text in can be defined like this:

```
{ export | reexport } tile  <Identifier>  <x>, <y>,
      <Tilestring1>
  [ , <Tilestring> ]
{ scale } { text(<alternative text>) };

{ export | reexport } tile  <Identifier>  <file name>
  { scale } { text(<alternative text>) };

<alternative text> := ( <string> | <text object path> )
```

Examples

```
text TxBreak "Break"
{
  2: "Pause";
}
tile TiCoffeeBreak "coffeemug.gif" text("Break");
tile TiCoffeeBreak "coffeemug.gif" text(TxBreak);
```

10 Addition to Tracing

To activate trace output for the UIA Interface of the IDM FOR WINDOWS, UI Automation tracing must be turned on explicitly (for example, in the on `dialog start` rule):

```
setup.tracing["AU"] ::= true;  /* "AU" = Automation */
```

11 Enhancement to the C Interface

The functions **DM_Control()** and **DM_ControlEx()** have been enhanced with an additional action. UI Automation support is active by default. With this action the specific support of the IDM for its specific objects can be disabled. However, the UI Automation support provided by Microsoft for the standard controls remains active.

The switching must happen before calling **DM_Initialize()** and after bootstrapping. Successful switching will be indicated by the return of *DM_TRUE*.

Action	Argument
DMF_UIAutomationMode	*0* disables or *1* enables the UI Automation support of the IDM

Index

U

V

W