ISA Dialog Manager

Automatisches Testen und Barrierefreiheit

A.06.03.b

Dieses Handbuch beschreibt Funktionen, die der ISA Dialog Manager bereitstellt um das automatische Testen der Benutzeroberfläche und die Entwicklung barrierefreier Anwendungen zu unterstützen.

ISA Informationssysteme GmbH

Meisenweg 33

70771 Leinfelden-Echterdingen

Deutschland

Darstellungskonventionen

DM wird in diesem Handbuch synonym zu "Dialog Manager" verwendet.

Die Bezeichnung UNIX schließt generell alle unterstützten UNIX-Derivate ein - außer in den explizit angegebenen Fällen.

Dort wo für geläufige englische Fachbegriffe keine gängigen deutschen Übersetzungen existieren, wird zur Vermeidung von Unklarheiten der englische Begriff verwendet.

< >	muss durch einen entsprechenden Wert ersetzt werden
color	Schlüsselwort ("keyword")
.bgc	Attribut
{ }	optional (0 oder einmal)
[ ]	optional (0 oder n-mal)
<A> | <B>	entweder <A> oder <B>

Beschreibungsmodus

Alle Schlüsselwörter sind fett und unterstrichen, z.B.

variable **integer** **function**

Indizierung von Attributen

Syntax für indizierte Attribute:

[I]

[I,J] bzw. [row,column]

Identifikatoren

Identifikatoren müssen mit einem Großbuchstaben oder einem "Unterstrich" ('_') beginnen. Die weiteren Zeichen können Groß-, Kleinbuchstaben, Zahlen oder Unterstriche sein.

Der Bindestrich ('-') ist für die Benennung von Identifikatoren als Zeichen ***nicht*** zugelassen!

Die maximale Länge eines Identifikators beträgt *31* Zeichen.

Beschreibung der zugelassenen Identifikatoren in Backus-Naur-Form

<Identifikator> ::= <erstes Zeichen>{<Zeichen>}

<erstes Zeichen> ::= _ | <Großbuchstabe>

<Zeichen> ::= _ | <Kleinbuchstabe> | <Großbuchstabe> | <Ziffer>

<Ziffer> ::= 1 | 2 | 3 | … 9 | 0

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

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

Inhalt

1 Einführung

Verfügbarkeit

Nur IDM FÜR WINDOWS, ab IDM-Version A.06.01.e

UI Automation (im Folgenden meist mit UIA abgekürzt) ist ein Framework von MICROSOFT um die Barrierefreiheit (Accessibility) und das automatisierte Testen von Benutzeroberflächen zu ermöglichen.

Der Fokus bei der Erweiterung des IDM liegt dabei auf der Bereitstellung der Infrastruktur für das automatisierte Testen.

Es macht dabei die Oberflächenelemente (UIA Elements) in einer Baumstruktur zugänglich. Wobei ein UIA Element Eigenschaften (Properties) aufweist, abhängig auch von seinem Typ (UIA Control Type). UIA Elemente können außerdem mehrere Steuerungsmuster (UIA Control Patterns) aufweisen, welche zusätzliche Eigenschaften oder Interaktionsmöglichkeiten (UIA Control Pattern Methods) bieten. Ereignisse zeigen dabei an, ob sich an einem UIA Element Änderungen ergeben haben.

Diese Dokumentation dient dazu, die Funktionalität des IDM bezüglich der UI Automation Unterstützung kenntlich zu machen und ist weitestgehend nur für Entwickler von UI Automation Clients interessant. Es stellt keine „Programmierdokumentation“ dar, sondern dokumentiert lediglich die Unterstützung durch den IDM. Für die Nutzung von UI Automation sei auf die Microsoft-Dokumentation verwiesen.

2 Zuordnung von IDM-Objektklassen zu UIA Control Types

Der IDM hat als ein Designprinzip die Verwendung der vom System bzw. Toolkit bereitgestellten Objektklassen (Controls, Widgets) um die Oberflächenobjekte zu implementieren und damit Bedienung und Aussehen möglichst konform zum System zu halten.

Eine Vielzahl von IDM-Objektklassen werden durch den „UI Automation Support for Standard Controls“ von MICROSOFT abgedeckt. Der IDM ergänzt wenn möglich und erforderlich noch die notwendigen Eigenschaften.

Im Hinblick auf die UI Automation Unterstützung im IDM lassen sich folgende Kategorien unterschieden:

D Besondere Objektklassen können nicht mit einer zusätzlichen UIA-Unterstützung durch den IDM ausgestattet werden, verfügen aber durch die Microsoft-Default-Implementierung über eine ausreichende Unterstützung. Dazu zählen z.B. ***menubox***, ***menuitem***, ***menusep***, ***messagebox*** und ***filereq***.

E Die meisten dieser Objektklassen erhalten ergänzende UIA-Unterstützung durch den IDM, insbesondere um deren Identifikation zu erleichtern.

X Vom IDM eigenimplementierte Objektklassen bzw. Objektklassen mit unzureichender Microsoft-Default-Implementierung erhalten eine eigene UIA-Unterstützung. Das sind hauptsächlich die Objektklassen ***image***, ***tablefield***, ***splitbox***, ***layoutbox***, ***progressbar***, ***poptext*** und ***toolbar***.

U Für besondere Objektklassen wie ***control***, ***canvas*** und USW-Objekte obliegt die UIA-Unterstützung dem Anwendungsprogrammierer bzw. Objekt-Implemetierer.

Die folgende Tabelle soll einen Anhaltspunkt für die Zuordnung der IDM-Objektklassen zu UIA Control Types liefern sowie der bereitgestellten Funktionalität durch die Pattern-Zugehörigkeit:

Kategorie	IDM-Objektklasse	UIA Control Type	Unterstützte UIA Control Patterns
U	***canvas***	Pane	Ist vom Canvas-Programmierer zu implementieren
E	***checkbox***	CheckBox	Toggle, Invoke
U	***control***	Pane	Ist vom OLE/Control-Objekt abhängig

Kategorie	IDM-Objekt-klasse	UIA Control Type	Unterstützte UIA Control Patterns
E	***edittext***	Edit	Systemabhängig – typischerweise Value, Text, Text2 Multiline-Texte mit Scrollbars haben meist noch das Scroll-Pattern
E	***edittext*** mit *.options[opt_rf] = true*	Document	Systemabhängig – typischerweise Value, Text, Text2
D	***filereq***	Window	Systemabhängig
X	***groupbox***	Pane Kind: Group	Invoke, Scroll (bei virtuellen Bereichen)
X	***image***	Button Kinder: Text, Image (wenn vor-handen)	Invoke, Toggle
X	***layoutbox***	Pane	Scroll (bei virtuellen Bereichen)
E	***listbox***	List Kinder: ListItem	Scroll, Selection, Invoke
D	***menubox***	Menu	ExpandCollapse
D	***menuitem***	MenuItem	Toggle, Invoke, ExpandCollapse, Selec-tionItem – abhängig vom *.style*-Attribut
D	***menusep***	Separator	–
D	***messagebox***	Window	Systemabhängig
E	***notebook***	Tab	Selection
E	***notepage***	TabItem/Pane	SelectionItem
X	***poptext***	ComboBox Kinder: Edit, List, Text (entsprechend dem *.style*)	ExpandCollapse, Value sowie Selection – abhängig vom *.style*-Attribut
X	***progressbar***	ProgressBar	RangeValue

Kategorie	IDM-Objekt-klasse	UIA Control Type	Unterstützte UIA Control Patterns
X	***pushbutton***	Button	Invoke
E	***radiobutton***	RadioButton	SelectionItem
X	***rectangle***	Button	Invoke
X	***scrollbar***	ScrollBar	RangeValue
X	***scrollbar*** (Slider)	Slider	RangeValue
X	***spinbox***	Spinner	RangeValue, Value, Selection – abhängig vom *.style*-Attribut
X	***splitbox***	Pane Kinder: Pane	Sichtbare Split-Bereiche sind Sub-Panes mit Transform-Pattern
X	***statictext*** (insensitiv)	Text	Text
X	***statictext*** (sen-sitiv)	Button	Invoke
E	***statusbar***	StatusBar	–
X	***tablefield***	Table Kinder: ListItem	Grid, Table, GridItem, TableItem, Invoke, Scroll, Selection
X	***toolbar***	ToolBar	Dock, Transform, Window (abhängig vom *.docking*-Attribut), Invoke
E	***treeview***	Tree	Scroll, Selection
E	***window***	Window	Window, Transform

Änderungen an der UI Automation Unterstützung können durch MICROSOFT jederzeit erfolgen, sodass bezüglich den Zuordnungen zu UIA-ControlTypes, Patterns, Properties und Events jederzeit Änderungen möglich sind.

3 Unterstützte UIA Properties und Control Pattern Methoden

Grundsätzlich sind die Properties, Methoden und Funktionalität durch die „UI Automation Specification“ vorgegeben. Dieses Kapitel hat also eher einen informativen Charakter um den Zusammenhang mit dem IDM herzustellen.

UIA Property/Method	IDM-Attribut	Bemerkungen
AcceleratorKey	*.accelerator*	Die Tastenkombination des ***Accelerators*** wird geliefert.
AccessKey	*.text*	Der **Mnemonic**-Buchstabe im Text wird zurückgeliefert, beispielsweise *"Alt+B"* für *.text "&Bearbeiten"*.
AutomationId	*.label*, *.acc_label*	*.acc_label* genießt, wenn gesetzt, Vorrang vor *.label*. Der *.label* wird unter Umständen ergänzt durch die Nummerierung „:[<I>]“ um eine eindeutige Identifikation auf Geschwisterebene zu erreichen.
BoundingRectangle	*.x*, *.y* *.width, .height* *.xauto*, *.yauto*	Position und Größe des Objekts.
ClassName	–	**Achtung** Systemnamen oder IDM-spezifische Namen die sich von Version zu Version ändern können und keine 1:1-Korrelation mit der IDM-Objektklasse besitzen.
ClickablePoint	–	Punkt innerhalb des BoundingRectangle zur Ausführung eines Mausklicks.
ControlType	*.class* (und anderen Attributen wie *.options*, *.style*, *.arrows*)	Die Umsetzung auf einen UIA Control Type hängt nicht nur von der Objektklasse ab, sondern unter Umständen von zusätzlichen Objektattributen.
FrameworkId	–	Wird von der UIA-Implementierung von MICROSOFT geliefert.

UIA Property/Method	IDM-Attribut	Bemerkungen
HasKeyboardFocus	*.focus*	Gibt wieder, ob das Objekt den Tastaturfokus besitzt.
HelpText	*.toolhelp* bzw. *.statushelp*	Falls kein Hilfetext in *.toolhelp* gesetzt wurde, wird *.statushelp* zurückgeliefert.
IsEnabled	*.sensitive* bzw. *.real_sensitive*	Liefert die Bedienbarkeit bzw. Sensitivität des UIA Elements.
IsKeyboardFocusable	–	Objekt ist sensitiv und kann den Tastaturfokus erhalten.
IsOffscreen		UIA Element ist sichtbar, kann aber durch darüberliegende Fenster oder Elemente verdeckt sein.
IsPassword	*.format*	Ein Format mit verdeckter Formatierung (Formatstring beginnt mit „S“) liefert für diese Property *true* zurück.
LocalizedControlType	–	Wird von der UIA-Implementierung von MICROSOFT geliefert.
LabeledBy	–	Ist dem IDM-Objekt ein insensitiver ***Statictext*** vorangestellt, wird das zugehörige UIA Element geliefert.
Name	*.text*, *.title*, *.acc_text* Bei Unterstrukturen (***Listbox***, ***Treeview***, Content) oft auch *.content*, *.content[I]*, *.content[R,C]*	Der Name des UIA Elements. Kann durch das Attribut *.acc_text* überschrieben werden.
NativeWindowHandle	*AT_WinHandle*	Entspricht meist dem HWND den **DM_GetToolkitData()** auf *AT_WinHandle* zurückliefert. Usually the same as the HWND that **DM_GetToolkitData()** returns for *AT_WinHandle*.

UIA Property/Method	IDM-Attribut	Bemerkungen
Orientation	*.direction*, *.docking*	Für IDM-Objektklassen wie ***splitbox***, ***scrollbar***, ***toolbar*** und ***tablefield*** wird hier die korrespondierende Ausrichtung geliefert.
OrientationType_ Horizontal	*.direction = 2*	
OrientationType_ Vertical	*.direction = 1*	
OrientationType_ Horizontal	*.docking = dock_<up|-down|window>*	
OrientationType_ Vertical	*.docking = dock_<left|-right>*	
ProcessId	–	Wird von der UIA-Implementierung von MICROSOFT geliefert.
ProviderDescription	–	Wird von der UIA-Implementierung von MICROSOFT geliefert.
RuntimeId	–	Wird meist von der UIA-Implementierung von MICROSOFT geliefert.

3.1 Value Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
ValueIsReadOnly	*.editable* *.editable[R,C]*	Eingabefeld oder Tabellenzelle ist editierbar. Eine ***Listbox*** liefert hier immer *true*.
Value	*.content* *.content[l]* *.content[R,C]*	Liefert den (formatierten) Inhalt eines Eingabefelds, ***Poptext***-, ***Treeview***- oder ***Listbox***-Elements oder einer Tabellenzelle.
SetValue()	*.content* *.content[l]* *.content[R,C]*	Verändert den Wert des Eingabefelds oder der Tabellenzelle.

3.2 RangeValue Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
ValueIsReadOnly	–	Range-Wert ist über SetValue() veränderbar.

UIA Property/Method	IDM-Attribut	Bemerkungen
Value	*.curvalue*	Range-Wert einer ***Spinbox*** oder ***Scrollbar***.
Minimum	*.minvalue*	Untere Range-Grenze.
Maximum	*.maxvalue*	Obere Range-Grenze.
SetValue()	–	Verändert den Range-Wert.

3.3 Scroll Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
HorizontallyScrollable	–	Horizontales Scrollen ist möglich.
HorizonalScrollPercent	–	Errechneter Wert aus *.vwidth* und realer Breite in %.
HorizontalViewSize	–	Errechneter Wert aus *.vwidth* und realer Breite in %.
VerticallyScrollable	–	Vertikales Scrollen ist möglich.
VericalScrollPercent	–	Errechneter Wert aus *.vheight* und realer Höhe in %.
VericalViewSize	–	Errechneter Wert aus *.vheight* und realer Höhe in %.
Scroll()	–	Scrollt relativ.
SetScrollPercent()	–	Scrollt zu einer bestimmten %-Position.

3.4 ScrollItem Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
ScrollIntoView()	–	Scrollt das Element in den sichtbaren Bereich.

3.5 Transform Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
CanMove	*.moveable*	***Fenster*** bzw. ***Toolbar*** kann verschoben werden.
CanResize	*.sizeable*	***Fenster*** bzw. ***Toolbar*** kann verkleinert und vergrößert werden. Oder der ***Splibox***-Bereich kann verkleinert und vergrößert werden.
CanRotate	–	Nicht vom IDM abgedeckt.
Move()	–	Erlaubt die Verschiebung eines ***Fensters*** oder einer ***Toolbar***.
Resize()	–	Erlaubt die Verkleinerung und Vergrößerung eines ***Fensters***, ***Toolbar*** oder ***Splitbox***-Bereichs.
Rotate()	–	–

3.6 Grid Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
ColumnCount	*.colcount*	Liefert die Anzahl der sichtbaren Spalten einer Tabelle.
RowCount	*.rowcount*	Liefert die Anzahl der sichtbaren Zeilen einer Tabelle.
GetItem()	–	Liefert das UIA Element einer Zelle.

3.7 GridItem Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
GridItemColumn	–	Spaltenindex (ab *0* und ohne unsichtbare Spalten).
GridItemColumnSpan	–	Ist immer *1* da im IDM keine Zellen zusammengefasst werden können.

UIA Property/Method	IDM-Attribut	Bemerkungen
GridItemRowCount	–	Zeilenindex (ab *0* und ohne unsichtbare Zeilen).
GridItemRowSpan	–	Ist immer *1* da im IDM keine Zellen zusammengefasst werden können.

3.8 Selection Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
SelectionCanSelectMultiple	*.multisel* *.selstyle* *.selection[]*	***Listboxen*** und Tabellen können eine Mehrfachselektion erlauben.
SelectionIsRequired	–	Ist eine Selektion erforderlich?
GetSelection()	–	Liefert die selektierten UIA Elemente.

3.9 SelectionItem Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
SelectionItemIsSelected	.active .activeitem	Liefert *true* für ein aktives Element.
AddToSelection()	–	Fügt ein Element zu einer Mehrfachselektion hinzu.
RemoveFromSelect()	–	Entfernt ein selektiertes Element aus einer Mehrfachselektion.
Select()	–	Selektiert dieses Element.

3.10 Toggle Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
ToggleState	*.active* *.state*	Aktivierung bzw. Zustand einer ***Checkbox***, ***Image***-Objekts oder ***Menuitem*** im *checkbox*-Stil.
Toggle()	–	Schaltet die Zustände zwischen *checked*, *unchecked* und *indeterminate* um.

3.11 ExpandCollapse Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
ExpandCollapseState	*.open* (***Treeview***)	–
Collapse()	–	Schließen eines Unterbaums.
Expand()	–	Öffnen eines Unterbaums.

3.12 Window Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
CanMinimize	*.iconifyable*	***Fenster*** kann minimiert werden.
CanMaximize	*.maximizable*	***Fenster*** kann maximiert werden.
IsTopmost	*.top_most*	***Fenster*** ist immer vor allen anderen Fenstern.
WindowIsModal	*.dialogbox*	***Fenster*** mit *.dialogbox* = *true* sind modale Fenster.
WindowInteractionState	–	–
WindowVisualState	–	Zustand des ***Fensters*** (minimiert, maximiert usw.).
Close()	–	Schließt das ***Fenster***.
SetWindowVisualState()	–	Erlaubt es, das ***Fenster*** zu minimieren oder zu maximieren.
WaitForInputIdle()	–	–

3.13 Dock Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
DockDockPosition	*.docking*	Liefert die Docking-Position einer ***Toolbar***.
SetDockPosition()	–	Kann zum Umdocken einer ***Toolbar*** verwendet werden.

3.14 Table Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
TableRowOrColumnMajor	*.direction*	Zeigt die Ausrichtung der Tabelle an.
GetColumnHeaders()	–	Liefert die UIA Elemente der sichtbaren Spalten-Header-Zellen.
GetRowHeaders()	–	Liefert die UIA Elemente der sichtbaren Zeilen-Header-Zellen.

3.15 TableItem Pattern

UIA Property/Method	IDM-Attribut	Bemerkungen
GetColumnHeaderItems()	–	Liefert die zugehörigen UIA Elemente im Header-Bereich *[row,1] … [row,.colheader]* der Tabelle.
GetRowHeaderItems()	–	Liefert die zugehörigen UIA Elemente im Header-Bereich *[1,col] … [.rowheader,col]* der Tabelle.

4 UIA Objektidentifikation

Die von UI Automation definierten Schlüssel-Properties für die Identifikation von UIA Elementen sind AutomationId, Name, ControlType und RuntimeId.

Um eine möglichst konsistente und sprachunabhängige Identifikation zu ermöglichen, wird deshalb bei den meisten IDM-Objekten der IDM Objekt-Label als AutomationId vergeben. Bei Mehrdeutigkeiten erfolgen bei geerbten Bezeichnern noch die IDM-typische Indizierung „:[<Nr>]“ nach dem Label für *Nr* >= *2*. Über das *.acc_label*-Attribut kann die AutomationId überschrieben werden. Dann hat allerdings der Anwendungsprogrammierer für die Eindeutigkeit zu sorgen.

Oft wird auch die Name-Property herangezogen, welche typischerweise den sichtbaren statischen Text-String (also nicht dem dynamischen Inhalt) widerspiegelt. Auch dies ist vom Anwendungsprogrammierer überschreibbar durch das *.acc_text*-Attribut.

In einem UI Automation Element Tree sind durch diese Properties damit Objekte meist eindeutig identifizierbar.

Die RuntimeId basiert im allgemeinen auf dem Window-Handle und kann sich somit auch bei Attributänderungen ändern. Sie eignet sich somit weniger für eine stabile Objektidentifikation.

5 Besonderheiten

5.1 Edittext

Für die Änderung des Textes ist die Methode SetValue() des Value Pattern zu benutzen. Dies führt zu einem *charinput*-Ereignis im IDM. Das *modified*-Ereignis um eine Änderung zu signalisieren wird allerdings erst versendet, wenn der ***Edittext*** den Fokus verliert. Konsistenter mit einer normalen Bedienung ist die „Simulation“ von Tastatureingaben. Generell sollte der Fokus über die SetFocus()-Methode von UIA auf das Eingabefeld gelenkt werden.

5.2 Poptext

Neben der üblichen Methodik, die Selektion über die Select()-Methode des SelectionItem Patterns zu verändern, kann ebenso die Methode SetValue() des Value Patterns benutzt werden. Handelt es sich dabei um einen editierbaren ***Poptext***, so wird, falls der Wert in der Poptext-Liste aufgeführt ist, eine Selektion vorgenommen und die entsprechenden *select*- und *activate*-Ereignisse erzeugt. Andernfalls wird der Inhalt des Editierbereichs umgesetzt und die Ereignisse *charinput* und *modified* ausgelöst.

5.3 Formatierte Eingabe

Wird bei einem Eingabefeld ein **Format** verwendet, so liefern Text und Value immer den Darstellungstext inklusive Formatierungszeichen. Ein Zugriff auf den Inhalt von Passwortfeldern bei einem verdeckten Format ist somit nicht möglich. Eine vollständige Änderung sollte ebenso über die SetValue()-Methode erfolgen und darf keine Formatierungszeichen beinhalten. Ansonsten sollten Testwerkzeuge die „Simulation“ der Tastatureingaben durchführen.

5.4 Client-Bereich

IDM-Objektklassen können aus mehreren UIA Elementen bestehen. Gruppierungsobjekte haben typischerweise ein übergeordnetes Element und einen Client-Bereich, welcher den Suffix „#client“ als AutomationId bekommt.

5.5 Virtuelle Bereiche

Gruppierungsobjekte erhalten einen virtuellem Bereich und optionale Scrollbars durch Setzen der Attribute *.vwidth* und *.vheight*. Das Client UIA Element erhält dann zusätzlich das Control Pattern Scroll und ScrollItem an den Kind-Objekten um so die Änderung des sichtbaren Bereichs und das Scrolling eines Kindes in den sichtbaren Bereich zu ermöglichen.

Durch die UIA-Methoden Scroll() bzw. ScrollIntoView() ist die Kontrolle über den angezeigten Ausschnitt möglich. Eine Scrolling-Kontrolle über die innen liegenden ScrollBar UIA Elemente ist nicht möglich.

5.6 Splitbox

Die einzelnen sichtbaren Bereiche sind als UIA-Kind-Elemente vom Typ Pane mit Namenssuffix „#client[1]“ … „#client[n]“ erreichbar und können durch die Resize() Control Pattern Methode in ihrer Größe geändert werden.

5.7 Menüs

Menüs in der Menüleiste wie auch Popup-Menüs sind nicht vom IDM überdefiniert sondern durch die Standard-Implementierung von Windows abgedeckt. Typischerweise sind die UIA Elemente der Menüs, Untermenüs und Menüeinträge erst beim Öffnen zugänglich. Kontextmenüs sind dabei unterhalb des Desktop UIA Elements eingeordnet. Die Menüs von Menüleisten liegen nach dem Öffnen direkt unter dem UIA Window Element.

5.8 Messagebox und Filereq

Beim **querybox()**-Aufruf werden diese durch Standard-Controls aufgebaut und sind typischerweise als UIA Control Type Dialog oder Window mit dem gesetzten Fenstertitel zu finden.

5.9 Image-Objekt

Um eine Erkennung des dargestellten Bildes zu erlauben, wird das ***Image***-Objekt in UI Automation durch das UI Element Button mit den Kind-Elementen Image und Text repräsentiert. Das Image-Element erhält, wenn vorhanden, den Alternativtext der verwendeten ***Tile***-Ressource in der UIA Name Property.

5.10 Toolbar-Objekt

Eine eingedockte ***Toolbar*** besitzt kein Transform Pattern, kann also nicht in Größe und Position variiert werden.

5.11 Treeview-Objekt

Eine erneute Selektion des gleichen Elements, über die Select()-Methode des SelectionItem Pattern, erzeugt kein erneutes *select*-Ereignis.

5.12 Benutzerinteraktion vs. Automatisierung

Die Funktionalität zur Automatisierung einer Benutzeroberfläche, die durch UI Automation möglich ist, deckt nicht alle Möglichkeiten ab, die einem Benutzer durch die Bedienung mit Maus und Tastatur gegeben sind.

» Tastatureingaben wie auch der Aufruf von ***Acceleratoren*** oder **Mnemonics** sind nicht direkt vorgesehen. Dies bedeutet aber auch, dass Aktionen bzw. Ereignisregeln, die z.B. auf `Enter` (`Return`), `Esc`, Cursortasten, usw. oder einer Funktionstaste basieren, nicht direkt durch UIA-Methoden angestoßen werden können. Es empfiehlt sich unter Umständen also die Simulation von speziellen Tasten um z.B. das Auslösen von *deselect_enter*-Ereignisregeln zu ermöglichen.

» Die Interaktionsmuster der Maus (Bewegen, Drücken und Loslassen einer Maustaste) sind ebenso kein Bestandteil der UIA-Methoden. Eine Interaktion in der Form „Poptext anklicken – mit gedrückter Maustaste nach unten fahren – über dem 3. Eintrag loslassen“ ist z.B. schlecht oder gar nicht simulierbar. Ein Mausklick entspricht meist der UIA-Methode Invoke() oder Select(), unter Umständen aber auch einem Expand() oder Collapse() und vieles mehr. Die Benutzerinteraktion ist also vom Control Type abhängig.

Grundsätzlich ist zu sagen, dass sich Benutzerinteraktion und Automatisierung bei den möglichen Aktionen, der Atomarität wie auch den Zustandsabhängigkeiten unterscheiden.

6 Hinweise zu HP UFT 12.52

Das Testwerkzeug HP UNIFIED FUNCTIONAL TESTING (UFT) bietet ab Version 12.52 eine Unterstützung für das Testen von UI Automation. Dabei weist UFT aber keine vollständige Unterstützung für alle UIA Control Types auf, sondern beschränkt sich auf die wichtigsten Control Types und Control Patterns. Dieses Kapitel ist also lediglich informativer Art und sollte keinen Blick in die UFT-Dokumentation ersetzen.

Die Unterstützung von UIA (UI Automation) durch UFT lässt sich wie folgt beschreiben:

- Das „UI Automation“ Add-In muss aktiv sein.
- Zur Identifikation über den Objekt-Spion muss der „UI Automation Modus“ aktiv sein.
- Die Objektidentifikation erfolgt über die UIA Properties Name und AutomationId.
- Es werden folgende UIA Control Types erkannt und in ein entsprechendes UFT-Objektmodell überführt:

UIA Control Type	UFT-Objektmodell
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

» Folgende UIA Control Patterns werden durch UFT über spezielle Methoden an den UIA-Objekten unterstützt:

UIA Control Pattern	UFT-Methoden
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

UIA Control Pattern	UFT-Methoden
Window	.Maximize .Minimize .Restore .Close

Dies bedeutet, dass nicht alle IDM-Objektklassen „unterscheidbar“ als Testobjekte mit eigener UIA-Klasse von UFT erkannt werden. Dazu gehören die IDM-Objektklassen:

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

Allerdings sollten diese von UFT als allgemeines UIAObject erkannt werden und durch die unterstützten Pattern-Methoden hinreichend test- und manipulierbar sein.

Es sei angemerkt, dass Microsoft mit UI Automation zwar die Bereiche „Accessiblity“ und „Test Automation“ abdeckt, allerdings keine vollständige und korrekte Erfassung von Benutzerinteraktion ermöglicht. Insofern ist ein „Recording“ der Benutzerinteraktion durch UFT im UI Automation Modus oft unvollständig und unter Umständen auch fehlerbehaftet.

Dies zeigt sich z.B. beim „Recording“ einer Menüinteraktion auf einem Untermenü. UFT generiert das folgende, nicht funktionsfähige Skript, welche keinen „Click“ auf das Menü enthält:

```
UIAWindow("Hauptfenster").UIAMenu("Anwendungsmenü").UIAObject("Datei").Expand
UIAWindow("Hauptfenster").UIAMenu("Datei").UIAObject("Untermenü").Expand
```

Ein funktionierendes Skript würde hingegen wie folgt aussehen:

```
UIAWindow("Hauptfenster").UIAMenu("Anwendungsmenü").UIAObject("Datei").Click
UIAWindow("Hauptfenster").UIAMenu("Datei").UIAObject("Untermenü").Click
UIAWindow("Hauptfenster").UIAMenu("Datei").Select "Untermenü;Menü A"
```

Beim „Recording“ im Windows-Modus erhält man etwa folgendes funktionstüchtiges Skript:

```
Window("Hauptfenster").WinMenu("Datei").Select "Datei;Untermenü;Menü A"
```

7 Attribut .acc_label

Mit diesem Attribut kann der Automation-Identifikator überschieben werden, der für ein Oberflächenobjekt über die MICROSOFT UIA-Schnittstelle vom IDM erfragt wird. Bei einem leeren String wird der Automation-Identifikator normalerweise vom Windows Control bzw. durch die UIA-Unterstützung im IDM sinnvoll vorgegeben.

Definition

Datentyp
string, *object* [***text***]

Zugriff
get, set

changed-Ereignis
ja

C
Bezeichner: AT_acc_label
Datentyp: DT_string, DT_text

COBOL
Bezeichner: AT-acc-label
Datentyp: DT-string, DT-text

U

Vererbung
ja

Standardwert
""

Klassifizierung
Standardattribut

n

terstützung des Attributs durch Objekte

Objekt	Unterstützung des Attributs
filereq, ***messagebox***	Attribut hat **keine** Auswirkung
menubox, ***menuitem***, ***menusep***	
canvas, ***spinbox***, ***statusbar***, ***tablefield***	Attribut wird unterstützt
groupbox, ***notebook***, ***notepage***, ***splitbox***	
image, ***layoutbox***, ***window***	
rectangle, ***scrollbar***	
checkbox, ***pushbutton***, ***radiobutton***	
edittext, ***poptext***, ***statictext***	
control, ***listbox***, ***treeview***	
andere Objektklassen	Attribut wird **nicht** unterstützt

Anmerkung

Dieses Attribut ist nur für die automatisierte Fremdsteuerung mit aktiver MICROSOFT UIA-Unterstützung relevant. Unter QT und MOTIF hat dieses Attribut keine Funktion.

Bei einer Überschreibung sollten die Regeln beachtet werden, welche in der MICROSOFT UI Automation Dokumentation für die AutomationId vorgegeben sind.

8 Attribut .acc_text

Mit diesem Attribut kann der Automation-Name überschieben werden, der für ein Oberflächenobjekt über die MICROSOFT UIA-Schnittstelle vom IDM erfragt wird. Bei einem Wert von *null* wird der Name normalerweise vom Windows Control bzw. durch die UIA-Unterstützung im IDM sinnvoll vorgegeben.

Definition

Datentyp
object [***text***], *string*

Zugriff
get, set

changed-Ereignis
ja

C
Bezeichner: AT_acc_text
Datentyp: DT_text, DT_string

COBOL
Bezeichner: AT-acc-text
Datentyp: DT-text, DT-string

Vererbung
ja

Standardwert
null

Klassifizierung
Standardattribut

U

n

terstützung des Attributs durch Objekte

Objekt	Unterstützung des Attributs
filereq, ***messagebox***	Attribut hat **keine** Auswirkung
menubox, ***menuitem***, ***menusep***	
canvas, ***spinbox***, ***statusbar***, ***tablefield***	Attribut wird unterstützt
groupbox, ***notebook***, ***notepage***, ***splitbox***	
image, ***layoutbox***, ***window***	
rectangle, ***scrollbar***	
checkbox, ***pushbutton***, ***radiobutton***	
edittext, ***poptext***, ***statictext***	
control, ***listbox***, ***treeview***	
andere Objektklassen	Attribut wird **nicht** unterstützt

Anmerkung

Dieses Attribut ist nur für die automatisierte Fremdsteuerung mit aktiver MICROSOFT UIA-Unterstützung relevant. Unter QT und MOTIF hat dieses Attribut keine Funktion.

9 Erweiterung der Tile-Ressource

Für den Zugang zu IDM-Oberflächenobjekten über die UI-Automation Schnittstelle unter MICROSOFT WINDOWS wurde die ***Tile***-Ressource ergänzt. ***Image***-Objekte, welche eine ***Tile***-Ressource als Bild (Muster) nutzen, erhalten einen Alternativtext um die Automatisierung und Zugänglichkeit („Accessibility“) zu erleichtern.

Dieser Alternativtext in kann dabei wie folgt angegeben werden:

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

{ export | reexport } tile  <Bezeichner>  <Dateiname>
  { scale } { text(<Alternativtext>) };

<Alternativtext> := ( <String> | <Text-Objektpfad> )
```

Beispiele

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

10 Ergänzung des Tracing

Um eine Trace-Ausgabe der UIA-Schnittstelle des IDM FÜR WINDOWS zu aktivieren, muss das UI Automation Tracing explizit (zum Beispiel in der `on dialog start` Regel) eingeschaltet werden:

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

11 Erweiterung zur C-Schnittstelle

Die Funktionen **DM_Control()** und **DM_ControlEx()** wurden um eine zusätzliche Aktion ergänzt. Standardmäßig ist die Unterstützung von UI Automation aktiv. Mit dieser Aktion kann die spezielle Unterstützung des IDM für seine besonderen Objekte abgeschaltet werden. Weiterhin aktiv bleibt aber die UI Automation Unterstützung von Microsoft für die Standard Controls.

Die Umschaltung muss dabei vor dem Aufruf von **DM_Initialize()** und nach dem Bootstrapping erfolgen. Ein erfolgreiche Umsetzung wir mit der Rückgabe von *DM_TRUE* angezeigt.

Aktion	Argument
DMF_UIAutomationMode	*0* deaktiviert oder *1* aktiviert den UI Automation Support des IDM

Index