Guile-GNOME: Atk

version 2.15.93, updated 2 September 2007

Bill Haneman
Marc Mulcahy
Padraig O’Briain
Short Contents

1 Overview ..................................................... 1
2 AtkAction .................................................. 2
3 AtkComponent ............................................. 4
4 AtkDocument ............................................... 8
5 AtkEditableText .......................................... 10
6 AtkGObjectAccessible ................................. 12
7 AtkHyperlinkImpl ....................................... 13
8 AtkHyperlink ............................................. 14
9 AtkHypertext ............................................. 16
10 AtkImage .................................................. 17
11 AtkNoOpObjectFactory ................................. 19
12 AtkNoOpObject .......................................... 20
13 AtkObjectFactory ....................................... 21
14 AtkObject .................................................. 22
15 AtkRegistry .............................................. 27
16 AtkRelationSet ........................................... 29
17 AtkRelation ............................................... 31
18 AtkSelection .............................................. 33
19 AtkStateSet .............................................. 35
20 AtkState ................................................... 38
21 AtkStreamableContent ................................. 39
22 AtkTable ................................................... 41
23 AtkText .................................................... 47
24 AtkUtil ..................................................... 56
25 AtkValue ................................................... 58
Concept Index ............................................... 60
Function Index .............................................. 61
1 Overview

The ATK wrapper for Guile is a part of Guile-GNOME. Maybe write more here at some point.
2 AtkAction

The ATK interface provided by UI components which the user can activate/interact with,

2.1 Overview

<atk-action> should be implemented by instances of <atk-object> classes with which the user can interact directly, i.e. buttons, checkboxes, scrollbars, e.g. components which are not "passive" providers of UI information.

Exceptions: when the user interaction is already covered by another appropriate interface such as <atk-editable-text> (insert/delete test, etc.) or <atk-value> (set value) then these actions should not be exposed by <atk-action> as well.

Also note that the <atk-action> API is limited in that parameters may not be passed to the object being activated; thus the action must be self-contained and specifiable via only a single "verb". Concrete examples include "press", "release", "click" for buttons, "drag" (meaning initiate drag) and "drop" for drag sources and drop targets, etc.

Though most UI interactions on components should be invocable via keyboard as well as mouse, there will generally be a close mapping between "mouse actions" that are possible on a component and the AtkActions. Where mouse and keyboard actions are redundant in effect, <atk-action> should expose only one action rather than exposing redundant actions if possible. By convention we have been using "mouse centric" terminology for <atk-action> names.

2.2 Usage

**atk-action-do-action (self <atk-action*>) (i int) ⇒ (ret bool)**

Perform the specified action on the object.

*action* a <gobject> instance that implements AtkActionIface

*i* the action index corresponding to the action to be performed

*ret* ‘#t’ if success, ‘#f’ otherwise

**atk-action-get-n-actions (self <atk-action*>) ⇒ (ret int)**

Gets the number of accessible actions available on the object. If there are more than one, the first one is considered the "default" action of the object.

*action* a <gobject> instance that implements AtkActionIface

*ret* a the number of actions, or 0 if action does not implement this interface.

**atk-action-get-description (self <atk-action*>) (i int) ⇒ (ret mchars)**

Returns a description of the specified action of the object.

*action* a <gobject> instance that implements AtkActionIface

*i* the action index corresponding to the action to be performed

*ret* a description string, or ‘#f’ if action does not implement this interface.
atk-action-get-name (self<atk-action*>)(i int) ⇒ (ret mchars)  [Function]
Returns the name of the specified action of the object.

  action a <gobject> instance that implements AtkActionIface
  i      the action index corresponding to the action to be performed
  ret    a name string, or ‘#f’ if action does not implement this interface.

atk-action-get-localized-name (self<atk-action*>)(i int) ⇒ (ret mchars)
Returns the localized name of the specified action of the object.

  action a <gobject> instance that implements AtkActionIface
  i      the action index corresponding to the action to be performed
  ret    a name string, or ‘#f’ if action does not implement this interface.

atk-action-get-keybinding (self<atk-action*>)(i int) ⇒ (ret mchars)
Returns a keybinding associated with this action, if one exists.

  action a <gobject> instance that implements AtkActionIface
  i      the action index corresponding to the action to be performed
  ret    a string representing the keybinding, or ‘#f’ if there is no keybinding for
          this action.

atk-action-set-description (self<atk-action*>)(i int)(desc mchars) ⇒ (ret bool)
Sets a description of the specified action of the object.

  action a <gobject> instance that implements AtkActionIface
  i      the action index corresponding to the action to be performed
  desc   the description to be assigned to this action
  ret    a gboolean representing if the description was successfully set;
3 AtkComponent

The ATK interface provided by UI components which occupy a physical area on the screen.

3.1 Overview

<atk-component> should be implemented by most if not all UI elements with an actual on-screen presence, i.e. components which can be said to have a screen-coordinate bounding box. Virtually all widgets will need to have <atk-component> implementations provided for their corresponding <atk-object> class. In short, only UI elements which are *not* GUI elements will omit this ATK interface.

A possible exception might be textual information with a transparent background, in which case text glyph bounding box information is provided by <atk-text>.

3.2 Usage

- **atk-component-add-focus-handler**: (Function)
  ```c
  (atk-component-add-focus-handler (self <atk-component*>) (handler <atk-focus-handler>) ⇒ (ret unsigned-int))
  ``
  
  Add the specified handler to the set of functions to be called when this object receives focus events (in or out). If the handler is already added it is not added again.

  - component: The <atk-component> to attach the handler to.
  - handler: The <atk-focus-handler> to be attached to the component.
  - ret: A handler id which can be used in atk_component_remove_focus_handler or zero if the handler was already added.

- **atk-component-contains**: (Function)
  ```c
  (atk-component-contains (self <atk-component*>) (x int) (y int) ⇒ (ret bool))
  ``
  
  Checks whether the specified point is within the extent of the component.

  - component: The <atk-component>
  - x: X coordinate
  - y: Y coordinate
  - coord-type: Specifies whether the coordinates are relative to the screen or to the component's top level window
  - ret: '#t' or '#f' indicating whether the specified point is within the extent of the component or not.

- **atk-component-get-extents**: (Function)
  ```c
  (atk-component-get-extents (self <atk-component*>) (coord_type <atk-coord-type>) ⇒ (x int) (y int) (width int) (height int))
  ``
  
  Gets the rectangle which gives the extent of the component.

  - component: The <atk-component>
Chapter 3: AtkComponent

\begin{itemize}
\item \textit{x} \quad \text{address of \texttt{<gint>} to put x coordinate}
\item \textit{y} \quad \text{address of \texttt{<gint>} to put y coordinate}
\item \textit{width} \quad \text{address of \texttt{<gint>} to put width}
\item \textit{height} \quad \text{address of \texttt{<gint>} to put height}
\item \textit{coord-type} \quad \text{specifies whether the coordinates are relative to the screen or to the components top level window}
\end{itemize}

\begin{verbatim}
\textbf{atk-component-get-layer} (\texttt{self \texttt{<atk-component*>}}) \Rightarrow (\textit{ret} \quad \texttt{<atk-layer>})
\end{verbatim}

Gets the layer of the component.

\begin{itemize}
\item \textit{component} \quad \text{an \texttt{<atk-component>}}
\item \textit{ret} \quad \text{an \texttt{<atk-layer>} which is the layer of the component}
\end{itemize}

\begin{verbatim}
\textbf{atk-component-get-mdi-zorder} (\texttt{self \texttt{<atk-component*>}}) \Rightarrow (\textit{ret} \quad \texttt{int})
\end{verbatim}

Gets the zorder of the component. The value G\_MININT will be returned if the layer of the component is not ATK\_LAYER\_MDI or ATK\_LAYER\_WINDOW.

\begin{itemize}
\item \textit{component} \quad \text{an \texttt{<atk-component>}}
\item \textit{ret} \quad \text{a \texttt{gint} which is the zorder of the component, i.e. the depth at which the component is shown in relation to other components in the same container.}
\end{itemize}

\begin{verbatim}
\textbf{atk-component-get-position} (\texttt{self \texttt{<atk-component*>}}) (\textit{coord-type} \quad \texttt{<atk-coord-type>}) \Rightarrow (\textit{x int} \quad (y int))
\end{verbatim}

Gets the position of \textit{component} in the form of a point specifying \textit{component}'s top-left corner.

\begin{itemize}
\item \textit{component} \quad \text{an \texttt{<atk-component>}}
\item \textit{x} \quad \text{address of \texttt{<gint>} to put x coordinate position}
\item \textit{y} \quad \text{address of \texttt{<gint>} to put y coordinate position}
\item \textit{coord-type} \quad \text{specifies whether the coordinates are relative to the screen or to the components top level window}
\end{itemize}

\begin{verbatim}
\textbf{atk-component-get-size} (\texttt{self \texttt{<atk-component*>}}) \Rightarrow (\textit{width int}) \quad (\textit{height int})
\end{verbatim}

Gets the size of the \textit{component} in terms of width and height.

\begin{itemize}
\item \textit{component} \quad \text{an \texttt{<atk-component>}}
\end{itemize}
width    address of <gint> to put width of component
height   address of <gint> to put height of component

atk-component-grab-focus (self <atk-component*>) ⇒ (ret bool) [Function]
Grabs focus for this component.

component
an <atk-component>

ret     ‘#t’ if successful, ‘#f’ otherwise.

atk-component-remove-focus-handler (self <atk-component*>)
                                      (handler_id unsigned-int) [Function]
Remove the handler specified by handler-id from the list of functions to be executed
when this object receives focus events (in or out).

component
the <atk-component> to remove the focus handler from

handler-id the handler id of the focus handler to be removed from component

atk-component-set-extents (self <atk-component*>)(x int) (y int) (width int) (height int) (coord_type <atk-coord-type>) ⇒ (ret bool)
Sets the extents of component.

component
an <atk-component>

x       x coordinate
y       y coordinate
width   width to set for component
height  height to set for component
coord-type
        specifies whether the coordinates are relative to the screen or to the
        components top level window

ret     ‘#t’ or ‘#f’ whether the extents were set or not

atk-component-set-position (self <atk-component*>)(x int) (y int) (coord_type <atk-coord-type>) ⇒ (ret bool)
Sets the position of component.

component
an <atk-component>

x       x coordinate
y       y coordinate
coord-type
        specifies whether the coordinates are relative to the screen or to the
        components top level window

ret     ‘#t’ or ‘#f’ whether or not the position was set or not
atk-component-set-size (self <atk-component*>)(width int) (height int) ⇒ (ret bool)
Set the size of the component in terms of width and height.

component
  an <atk-component>
width
  width to set for component
height
  height to set for component
ret
  ‘#t’ or ‘#f’ whether the size was set or not

atk-component-get-alpha (self <atk-component*>) ⇒ (ret double)
Returns the alpha value (i.e. the opacity) for this component, on a scale from 0 (fully transparent) to 1.0 (fully opaque).

component
  an <atk-component>
ret
  An alpha value from 0 to 1.0, inclusive.

Since ATK 1.12
4 AtkDocument

The ATK interface which represents the toplevel container for document content.

4.1 Overview

The AtkDocument interface should be supported by any object whose content is a representation or view of a document. The AtkDocument interface should appear on the toplevel container for the document content; however AtkDocument instances may be nested (i.e. an AtkDocument may be a descendant of another AtkDocument) in those cases where one document contains "embedded content" which can reasonably be considered a document in its own right.

4.2 Usage

atk-document-get-document-type (self <atk-document*>) ⇒ (ret mchars) [Function]

Gets a string indicating the document type.

document a <gobject> instance that implements AtkDocumentIface
ret a string indicating the document type

atk-document-get-document (self <atk-document*>) ⇒ (ret <gpointer>) [Function]

Gets a 'gpointer' that points to an instance of the DOM. It is up to the caller to check atk_document_get_type to determine how to cast this pointer.

document a <gobject> instance that implements AtkDocumentIface
ret a 'gpointer' that points to an instance of the DOM.

atk-document-get-attribute-value (self <atk-document*>) (attribute_name mchars) ⇒ (ret mchars) [Function]

Returns:

document a <gobject> instance that implements AtkDocumentIface
attribute-name a character string representing the name of the attribute whose value is being queried.
ret a string value associated with the named attribute for this document, or NULL if a value for <attribute-name> has not been specified for this document.

Since ATK 1.12

atk-document-set-attribute-value (self <atk-document*>) (attribute_name mchars) (attribute_value mchars) ⇒ (ret bool) [Function]

Returns:

document a <gobject> instance that implements AtkDocumentIface
attribute-name
a character string representing the name of the attribute whose value is being set.

attribute-value
a string value to be associated with attribute-name.

ret
TRUE if value is successfully associated with attribute-name for this document, FALSE otherwise (e.g. if the document does not allow the attribute to be modified).

Since ATK 1.12

atk-document-get-attributes (self <atk-document*>) ⇒ (ret <atk-attribute-set*>)
Gets an AtkAttributeSet which describes document-wide attributes as name-value pairs.
Returns:

document  a <gobject> instance that implements AtkDocumentIface
ret  An AtkAttributeSet containing the explicitly set name-value-pair attributes associated with this document as a whole.

Since ATK 1.12

atk-document-get-locale (self <atk-document*>) ⇒ (ret mchars)
Gets a UTF-8 string indicating the POSIX-style LC_MESSAGES locale of the content of this document instance. Individual text substrings or images within this document may have a different locale, see atk_text_get_attributes and atk_image_get_image_locale.

document  a <gobject> instance that implements AtkDocumentIface
ret  a UTF-8 string indicating the POSIX-style LC_MESSAGES locale of the document content as a whole, or NULL if the document content does not specify a locale.
5 AtkEditableText

The ATK interface implemented by components containing user-editable text content.

5.1 Overview

<atk-editable-text> should be implemented by UI components which contain text which
the user can edit, via the <atk-object> corresponding to that component (see <atk-
object>).

<atk-editable-text> is a subclass of <atk-text>, and as such, an object which im-
plements <atk-editable-text> is by definition an <atk-text> implementor as well.

5.2 Usage

atk-editable-text-set-text-contents (self <atk-editable-text*>)(string mchars)
Set text contents of text.

  text          an <atk-editable-text>
  string        string to set for text contents of text

atk-editable-text-insert-text (self <atk-editable-text*>)(string mchars)(length int) ⇒ (position int)
Insert text at a given position.

  text          an <atk-editable-text>
  string        the text to insert
  length        the length of text to insert, in bytes
  position      The caller initializes this to the position at which to insert the text. After
                 the call it points at the position after the newly inserted text.

atk-editable-text-copy-text (self <atk-editable-text*>)(start_pos int)(end_pos int)
Copy text from start-pos up to, but not including end-pos to the clipboard.

  text          an <atk-editable-text>
  start-pos     start position
  end-pos       end position

atk-editable-text-cut-text (self <atk-editable-text*>)(start_pos int)(end_pos int)
Copy text from start-pos up to, but not including end-pos to the clipboard and then
delete from the widget.

  text          an <atk-editable-text>
  start-pos     start position
  end-pos       end position
atk-editable-text-delete-text (self <atk-editable-text*>)  
\( (\text{start-pos int}) \) \( (\text{end-pos int}) \)  
Delete text \text{start-pos} up to, but not including \text{end-pos}.

\text{text} \quad \text{an <atk-editable-text>}

\text{start-pos} \quad \text{start position}

\text{end-pos} \quad \text{end position}

atk-editable-text-paste-text (self <atk-editable-text*>)  
\( (\text{position int}) \)  
Paste text from clipboard to specified \text{position}.

\text{text} \quad \text{an <atk-editable-text>}

\text{position} \quad \text{position to paste}
6 AtkGObjectAccessible

This object class is derived from AtkObject and can be used as a basis implementing accessible objects.

6.1 Overview

This object class is derived from AtkObject. It can be used as a basis for implementing accessible objects for GObjects which are not derived from GtkWidget. One example of its use is in providing an accessible object for GnomeCanvasItem in the GAIL library.

6.2 Usage

atk-gobject-accessible-for-object (obj <gobject>) ⇒ (ret <atk-object>)

[Function]

Gets the accessible object for the specified obj.

obj a <gobject>

ret a <atk-object> which is the accessible object for the obj

atk-gobject-accessible-get-object (self <atk-gobject-accessible>) ⇒ (ret <gobject>)

get-object

[Method]

Gets the GObject for which obj is the accessible object.

obj a <atk-object>

ret a <gobject> which is the object for which obj is the accessible objectc
Chapter 7: AtkHyperlinkImpl

7 AtkHyperlinkImpl

An interface from which the AtkHyperlink associated with an AtkObject may be obtained.

7.1 Overview

AtkHyperlinkImpl allows AtkObjects to refer to their associated AtkHyperlink instance, if one exists. AtkHyperlinkImpl differs from AtkHyperlink in that AtkHyperlinkImpl is an interface, whereas AtkHyperlink is a object type. The AtkHyperlinkImpl interface allows a client to query an AtkObject for the availability of an associated AtkHyperlink instance, and obtain that instance. It is thus particularly useful in cases where embedded content or inline content within a text object is present, since the embedding text object implements AtkHypertext and the inline/embedded objects are exposed as children which implement AtkHyperlinkImpl, in addition to their being obtainable via AtkHypertext:getLink followed by AtkHyperlink:getObject.

7.2 Usage

atk-hyperlink-impl-get-hyperlink (self <atk-hyperlink-impl*>) [Function]

⇒ (ret <atk-hyperlink>)

Gets the hyperlink associated with this object.

obj a GObject instance that implements AtkHyperlinkIface

ret an AtkHyperlink object which points to this implementing AtkObject.

Since ATK 1.12
8 AtkHyperlink

An ATK object which encapsulates a link or set of links in a hypertext document.

8.1 Overview

An ATK object which encapsulates a link or set of links (for instance in the case of client-side image maps) in a hypertext document. It may implement the AtkAction interface. AtkHyperlink may also be used to refer to inline embedded content, since it allows specification of a start and end offset within the host AtkHypertext object.

8.2 Usage

atk-hyperlink-get-uri (self <atk-hyperlink>) (i int) ⇒ (ret mchars) [Function]
get-uri [Method]
Get a the URI associated with the anchor specified by i of link.
Multiple anchors are primarily used by client-side image maps.

link an <atk-hyperlink>
i a (zero-index) integer specifying the desired anchor
ret a string specifying the URI

atk-hyperlink-get-object (self <atk-hyperlink>) (i int) ⇒ (ret <atk-object>) [Function]
get-object [Method]
Returns the item associated with this hyperlink's nth anchor. For instance, the returned <atk-object> will implement <atk-text> if link is a text hyperlink, <atk-image> if link is an image hyperlink etc.
Multiple anchors are primarily used by client-side image maps.

link an <atk-hyperlink>
i a (zero-index) integer specifying the desired anchor
ret an <atk-object> associated with this hyperlink's i-th anchor

atk-hyperlink-get-end-index (self <atk-hyperlink>) ⇒ (ret int) [Function]
get-end-index [Method]
Gets the index with the hypertext document at which this link ends.

link an <atk-hyperlink>
ret the index with the hypertext document at which this link ends

atk-hyperlink-get-start-index (self <atk-hyperlink>) ⇒ (ret int) [Function]
get-start-index [Method]
Gets the index with the hypertext document at which this link begins.

link an <atk-hyperlink>
ret the index with the hypertext document at which this link begins
atk-hyperlink-is-valid (self <atk-hyperlink>) ⇒ (ret bool)  [Function]

is-valid  [Method]
Since the document that a link is associated with may have changed this method returns ‘#t’ if the link is still valid (with respect to the document it references) and ‘#f’ otherwise.

link  an <atk-hyperlink>
ret  whether or not this link is still valid

atk-hyperlink-is-inline (self <atk-hyperlink>) ⇒ (ret bool)  [Function]

is-inline  [Method]
Indicates whether the link currently displays some or all of its content inline. Ordinary HTML links will usually return ‘#f’, but an inline &lt;src&gt; HTML element will return ‘#t’. a *

link  an <atk-hyperlink>
ret  whether or not this link displays its content inline.

atk-hyperlink-get-n-anchors (self <atk-hyperlink>) ⇒ (ret int)  [Function]
get-n-anchors  [Method]
Gets the number of anchors associated with this hyperlink.

link  an <atk-hyperlink>
ret  the number of anchors associated with this hyperlink

atk-hyperlink-is-selected-link (self <atk-hyperlink>) ⇒ (ret bool)  [Function]
is-selected-link  [Method]
Determines whether this AtkHyperlink is selected

Returns:

link  an <atk-hyperlink>
ret  True is the AtkHyperlink is selected, False otherwise

Since ATK 1.4 @Deprecated: This method is deprecated since ATK version 1.8. Please use ATK_STATE_SELECTED to indicate when a hyperlink within a Hypertext container is selected.
9 AtkHypertext

The ATK interface which provides standard mechanism for manipulating hyperlinks.

9.1 Overview

An interface used for objects which implement linking between multiple resource or content locations, or multiple ‘markers’ within a single document. A Hypertext instance is associated with one or more Hyperlinks, which are associated with particular offsets within the Hypertext’s included content. While this interface is derived from Text, there is no requirement that Hypertext instances have textual content; they may implement Image as well, and Hyperlinks need not have non-zero text offsets.

9.2 Usage

\begin{description}
\item[atk-hypertext-get-link (self <atk-hypertext*> (link_index int) ⇒ (ret <atk-hyperlink>)]
\hspace{1em} Gets the link in this hypertext document at index link-index.
\item[hypertext] is an <atk-hypertext>
\item[link-index] is an integer specifying the desired link
\item[ret] is the link in this hypertext document at index link-index.
\end{description}

\begin{description}
\item[atk-hypertext-get-n-links (self <atk-hypertext*> ⇒ (ret int)]
\hspace{1em} Gets the number of links within this hypertext document.
\item[hypertext] is an <atk-hypertext>
\item[ret] is the number of links within this hypertext document.
\end{description}

\begin{description}
\item[atk-hypertext-get-link-index (self <atk-hypertext*> (char_index int) ⇒ (ret int)]
\hspace{1em} Gets the index into the array of hyperlinks that is associated with the character specified by char-index.
\item[hypertext] is an <atk-hypertext>
\item[char-index] is a character index
\item[ret] is an index into the array of hyperlinks in hypertext, or -1 if there is no hyperlink associated with this character.
\end{description}
10 AtkImage

The ATK Interface implemented by components which expose image or pixmap content on-screen.

10.1 Overview

<atk-image> should be implemented by <atk-object> subtypes on behalf of components which display image/pixmap information onscreen, and which provide information (other than just widget borders, etc.) via that image content. For instance, icons, buttons with icons, toolbar elements, and image viewing panes typically should implement <atk-image>.

<atk-image> primarily provides two types of information: coordinate information (useful for screen review mode of screenreaders, and for use by onscreen magnifiers), and descriptive information. The descriptive information is provided for alternative, text-only presentation of the most significant information present in the image.

10.2 Usage

atk-image-get-image-position (self <atk-image*>) (coord_type <atk-coord-type>) ⇒ (x int) (y int)

Get the position of the image in the form of a point specifying the images top-left corner.

image a <gobject> instance that implements AtkImageIface

x address of <gint> to put x coordinate position; otherwise, -1 if value cannot be obtained.

y address of <gint> to put y coordinate position; otherwise, -1 if value cannot be obtained.

coord-type specifies whether the coordinates are relative to the screen or to the components top level window

atk-image-get-image-description (self <atk-image*>) ⇒ (ret mchars)

Get a textual description of this image.

image a <gobject> instance that implements AtkImageIface

ret a string representing the image description

atk-image-set-image-description (self <atk-image*>) (description mchars) ⇒ (ret bool)

Sets the textual description for this image.

image a <gobject> instance that implements AtkImageIface

description a string description to set for image

ret boolean TRUE, or FALSE if operation could not be completed.
**atk-image-get-image-size** *(self <atk-image*>) ⇒ (width int) ⇒ (height int)*

Get the width and height in pixels for the specified image. The values of *width* and *height* are returned as -1 if the values cannot be obtained (for instance, if the object is not onscreen).

- **image** a <gobject> instance that implements AtkImageIface
- **width** filled with the image width, or -1 if the value cannot be obtained.
- **height** filled with the image height, or -1 if the value cannot be obtained.

**atk-image-get-image-locale** *(self <atk-image*>) ⇒ (ret mchars)*

Since ATK 1.12

- **image** An <atk-image>
- **ret** a string corresponding to the POSIX LC_MESSAGES locale used by the image description, or NULL if the image does not specify a locale.
11 AtkNoOpObjectFactory

The AtkObjectFactory which creates an AtkNoOpObject.

11.1 Overview

The AtkObjectFactory which creates an AtkNoOpObject. An instance of this is created by an AtkRegistry if no factory type has not been specified to create an accessible object of a particular type.

11.2 Usage

\texttt{atk-no-op-object-factory-new} \Rightarrow (ret <atk-object-factory>) \quad \textbf{[Function]}

Creates an instance of an <atk-object-factory> which generates primitive (non-functioning) <atk-objects>.

\texttt{ret} \quad \text{an instance of an <atk-object-factory>}
12 AtkNoOpObject

An AtkObject which purports to implement all ATK interfaces.

12.1 Overview

An AtkNoOpObject is an AtkObject which purports to implement all ATK interfaces. It is the type of AtkObject which is created if an accessible object is requested for an object type for which no factory type is specified.

12.2 Usage

atk-no-op-object-new (obj <gobject>) ⇒ (ret <atk-object>)  [Function]

Provides a default (non-functioning stub) <atk-object>. Application maintainers should not use this method.

obj a <gobject>

ret a default (non-functioning stub) <atk-object>
13 AtkObjectFactory

The base object class for a factory used to create accessible objects for objects of a specific GType.

13.1 Overview

This class is the base object class for a factory used to create an accessible object for a specific GType. The function `atk-registry-set-factory-type` is normally called to store in the registry the factory type to be used to create an accessible of a particular GType.

13.2 Usage

```lisp
atk-object-factory-invalidate (self <atk-object-factory>) [Function]
 invalidate [Method]

Inform factory that it is no longer being used to create accessibles. When called, factory may need to inform <atk-objects> which it has created that they need to be re-instantiated. Note: primarily used for runtime replacement of <atk-object-factories> in object registries.

factory an <atk-object-factory> to invalidate
```


## 14 AtkObject

The base object class for the Accessibility Toolkit API.

### 14.1 Overview

This class is the primary class for accessibility support via the Accessibility ToolKit (ATK). Objects which are instances of `<atk-object>` (or instances of AtkObject-derived types) are queried for properties which relate basic (and generic) properties of a UI component such as name and description. Instances of `<atk-object>` may also be queried as to whether they implement other ATK interfaces (e.g. `<atk-action>`, `<atk-component>`, etc.), as appropriate to the role which a given UI component plays in a user interface.

All UI components in an application which provide useful information or services to the user must provide corresponding `<atk-object>` instances on request (in GTK+, for instance, usually on a call to `#gtk-widget-get-accessible`), either via ATK support built into the toolkit for the widget class or ancestor class, or in the case of custom widgets, if the inherited `<atk-object>` implementation is insufficient, via instances of a new `<atk-object>` subclass.

### 14.2 Usage

atk-implementor-ref-accessible (self <atk-implementor*>) ⇒ [Function]

- (ret <atk-object>)
  - Gets a reference to an object’s `<atk-object>` implementation, if the object implements `<atk-object-iface>

  implementor
  - The `<gobject>` instance which should implement `<atk-implementor-iface>` if a non-null return value is required.

  ret
  - a reference to an object’s `<atk-object>` implementation

atk-object-get-name (self <atk-object>) ⇒ (ret mchars) [Function]

get-name
  - Gets the accessible name of the accessible.

  accessible
  - an `<atk-object>`

  ret
  - a character string representing the accessible name of the object.

atk-object-get-description (self <atk-object>) ⇒ (ret mchars) [Function]

get-description
  - Gets the accessible description of the accessible.

  accessible
  - an `<atk-object>`

  ret
  - a character string representing the accessible description of the accessible.

atk-object-get-parent (self <atk-object>) ⇒ (ret <atk-object>) [Function]

get-parent
  - Gets the accessible parent of the accessible.
atk-object-ref-accessible-child \( \text{(self <atk-object>) (i int)} \Rightarrow \) \[\text{Function}\]
\( \right) \text{(ret <atk-object>)} \]
\( \text{ref-accessible-child} \) \[\text{Method}\]

Gets a reference to the specified accessible child of the object. The accessible children are 0-based so the first accessible child is at index 0, the second at index 1 and so on.

\( \text{accessible} \text{ an <atk-object>} \)
\( i \text{ a gint representing the position of the child, starting from 0} \)
\( \text{ret} \text{ an <atk-object> representing the specified accessible child of the accessible.} \)

atk-object-ref-relation-set \( \text{(self <atk-object>)} \Rightarrow \) \[\text{Function}\]
\( \text{<atk-relation-set>} \)
\( \text{ref-relation-set} \) \[\text{Method}\]

Gets the \(<atk-relation-set>\) associated with the object.

\( \text{accessible} \text{ an <atk-object>} \)
\( \text{ret} \text{ an <atk-relation-set> representing the relation set of the object.} \)

atk-object-get-layer \( \text{(self <atk-object>)} \Rightarrow \) \[\text{Function}\]
\( \text{<atk-layer>} \)
\( \text{get-layer} \) \[\text{Method}\]

‘atk_object_get_layer’ is deprecated and should not be used in newly-written code. Use atk_component_get_layer instead.

Gets the layer of the accessible.

Returns:

\( \text{accessible} \text{ an <atk-object>} \)
\( \text{ret} \text{ an <atk-layer> which is the layer of the accessible} \)

atk-object-get-mdi-zorder \( \text{(self <atk-object>)} \Rightarrow \) \[\text{Function}\]
\( \text{(ret int)} \)
\( \text{get-mdi-zorder} \) \[\text{Method}\]

‘atk_object_get_mdi_zorder’ is deprecated and should not be used in newly-written code. Use atk_component_get_mdi_zorder instead.

Gets the zorder of the accessible. The value G_MININT will be returned if the layer of the accessible is not ATK_LAYER_MDI.

Returns:

\( \text{accessible} \text{ an <atk-object>} \)
\( \text{ret} \text{ a gint which is the zorder of the accessible, i.e. the depth at which the component is shown in relation to other components in the same container.} \)
atk-object-get-role (self <atk-object>) ⇒ (ret <atk-role>) [Function]
get-role [Method]
    Gets the role of the accessible.
    accessible  an <atk-object>
    ret  an <atk-role> which is the role of the accessible

atk-object-ref-state-set (self <atk-object>) ⇒ (ret <atk-state-set>) [Function]
ref-state-set [Method]
    Gets a reference to the state set of the accessible; the caller must unreference it when
    it is no longer needed.
    accessible  an <atk-object>
    ret  a reference to an <atk-state-set> which is the state set of the accessible

atk-object-get-index-in-parent (self <atk-object>) ⇒ (ret int) [Function]
get-index-in-parent [Method]
    Gets the 0-based index of this accessible in its parent; returns -1 if the accessible does
    not have an accessible parent.
    accessible  an <atk-object>
    ret  an integer which is the index of the accessible in its parent

atk-object-set-name (self <atk-object>) (name mchars) [Function]
set-name [Method]
    Sets the accessible name of the accessible.
    accessible  an <atk-object>
    name  a character string to be set as the accessible name

atk-object-set-description (self <atk-object>) (description mchars) [Function]
set-description [Method]
    Sets the accessible description of the accessible.
    accessible  an <atk-object>
    description  a character string to be set as the accessible description

atk-object-set-parent (self <atk-object>) (parent <atk-object>) [Function]
set-parent [Method]
    Sets the accessible parent of the accessible.
    accessible  an <atk-object>
    parent  an <atk-object> to be set as the accessible parent
atk-object-set-role (self <atk-object>) (role <atk-role>) [Function]

set-role [Method]

Sets the role of the accessible.

accessible an <atk-object>

role an <atk-role> to be set as the role

atk-object-notify-state-change (self <atk-object>) (state unsigned-int64) (value bool) [Function]

notify-state-change [Method]

Emits a state-change signal for the specified state.

accessible an <atk-object>

state an <atk-state> whose state is changed

value a gboolean which indicates whether the state is being set on or off

atk-object-initialize (self <atk-object>) (data <gpointer>) [Function]

initialize [Method]

This function is called when implementing subclasses of <atk-object>. It does initialization required for the new object. It is intended that this function should called only in the ...-new functions used to create an instance of a subclass of <atk-object>

accessible a <atk-object>

data a <gpointer> which identifies the object for which the AtkObject was created.

atk-object-add-relationship (self <atk-object>) (relationship <atk-relation-type>) (target <atk-object>) ⇒ (ret bool) [Function]

add-relationship [Method]

Adds a relationship of the specified type with the specified target.

object The <atk-object> to which an AtkRelation is to be added.

relationship The <atk-relation-type> of the relation

target The <atk-object> which is to be the target of the relation.

ret TRUE if the relationship is added.

atk-object-remove-relationship (self <atk-object>) (relationship <atk-relation-type>) (target <atk-object>) ⇒ (ret bool) [Function]

remove-relationship [Method]

Removes a relationship of the specified type with the specified target.

object The <atk-object> from which an AtkRelation is to be removed.

relationship The <atk-relation-type> of the relation

target The <atk-object> which is the target of the relation to be removed.

ret TRUE if the relationship is removed.
**atk-object-get-attributes** (\texttt{self <atk-object>}) \Rightarrow (\texttt{ret <atk-attribute-set*>})

[Function]

**get-attributes**

Get a list of properties applied to this object as a whole, as an \texttt{<atk-attribute-set>} consisting of name-value pairs. As such these attributes may be considered weakly-typed properties or annotations, as distinct from strongly-typed object data available via other get/set methods. Not all objects have explicit "name-value pair" \texttt{<atk-attribute-set>} properties.

Returns:

\texttt{accessible} An \texttt{<atk-object>}.

\texttt{ret} an \texttt{<atk-attribute-set>} consisting of all explicit properties/annotations applied to the object, or an empty set if the object has no name-value pair attributes assigned to it.

Since ATK 1.12

\texttt{atk-role-get-name} (\texttt{role <atk-role>}) \Rightarrow (\texttt{ret mchars})

[Function]

Gets the description string describing the \texttt{<atk-role>} role.

\texttt{role} The \texttt{<atk-role>} whose name is required

\texttt{ret} the string describing the AtkRole

\texttt{atk-role-get-localized-name} (\texttt{role <atk-role>}) \Rightarrow (\texttt{ret mchars})

[Function]

Gets the localized description string describing the \texttt{<atk-role>} role.

\texttt{role} The \texttt{<atk-role>} whose localized name is required

\texttt{ret} the localized string describing the AtkRole

\texttt{atk-role-for-name} (\texttt{name mchars}) \Rightarrow (\texttt{ret <atk-role>})

[Function]

Get the \texttt{<atk-role>} type corresponding to a role name.

\texttt{name} a string which is the (non-localized) name of an ATK role.

\texttt{ret} the \texttt{<atk-role>} enumerated type corresponding to the specified name, or \texttt{<atk-role-invalid>} if no matching role is found.
15 AtkRegistry

An object used to store the GType of the factories used to create an accessible object for an object of a particular GType.

15.1 Overview

The AtkRegistry is normally used to create appropriate ATK "peers" for user interface components. Application developers usually need only interact with the AtkRegistry by associating appropriate ATK implementation classes with GObject classes via the atk_registry_set_factory_type call, passing the appropriate GType for application custom widget classes.

15.2 Usage

atk-registry-set-factory-type (self <atk-registry>) (type <gtype>) (factory_type <gtype>)

set-factory-type

Associate an <atk-object-factory> subclass with a <g-type>. Note: The associated factory-type will thereafter be responsible for the creation of new <atk-object> implementations for instances appropriate for type.

registry the <atk-registry> in which to register the type association
type an <atk-object> type
factory-type an <atk-object-factory> type to associate with type. Must implement AtkObject appropriate for type.

atk-registry-get-factory-type (self <atk-registry>) (type <gtype>) ⇒ (ret <gtype>)

get-factory-type

Provides a <g-type> indicating the <atk-object-factory> subclass associated with type.

registry an <atk-registry>
type a <g-type> with which to look up the associated <atk-object-factory> subclass
ret a <g-type> associated with type type

atk-registry-get-factory (self <atk-registry>) (type <gtype>) ⇒

get-factory

Gets an <atk-object-factory> appropriate for creating <atk-objects> appropriate for type.

registry an <atk-registry>
type a <g-type> with which to look up the associated <atk-object-factory>
ret an `<atk-object-factory>` appropriate for creating `<atk-objects>` appropriate for `type`.

`atk-get-default-registry ⇒ (ret `<atk-registry>`)` [Function]

Gets a default implementation of the `<atk-object-factory>/type` registry. Note: For most toolkit maintainers, this will be the correct registry for registering new `<atk-object>` factories. Following a call to this function, maintainers may call `atk-registry-set-factory-type` to associate an `<atk-object-factory>` subclass with the GType of objects for whom accessibility information will be provided.

ret a default implementation of the `<atk-object-factory>/type` registry
Chapter 16: AtkRelationSet

A set of AtkRelations, normally the set of AtkRelations which an AtkObject has.

16.1 Overview

The AtkRelationSet held by an object establishes its relationships with objects beyond the normal "parent/child" hierarchical relationships that all user interface objects have. AtkRelationSets establish whether objects are labelled or controlled by other components, share group membership with other components (for instance within a radio-button group), or share content which "flows" between them, among other types of possible relationships.

16.2 Usage

atk-relation-set-new ⇒ (ret <atk-relation-set>)

Creates a new empty relation set.

ret a new <atk-relation-set>

atk-relation-set-contains (self <atk-relation-set>)
  (relationship <atk-relation-type>) ⇒ (ret bool)

contains

Determine whether the relation set contains a relation that matches the specified type.

set an <atk-relation-set>

relationship an <atk-relation-type>

ret 't' if relationship is the relationship type of a relation in set, 'f' otherwise

atk-relation-set-remove (self <atk-relation-set>) (relation <atk-relation>)

remove

Removes a relation from the relation set. This function unref's the <atk-relation> so it will be deleted unless there is another reference to it.

set an <atk-relation-set>

relation an <atk-relation>

atk-relation-set-add (self <atk-relation-set>) (relation <atk-relation>)

add

Add a new relation to the current relation set if it is not already present. This function ref's the AtkRelation so the caller of this function should unref it to ensure that it will be destroyed when the AtkRelationSet is destroyed.

set an <atk-relation-set>

relation an <atk-relation>
atk-relation-set-get-n-relations \( (self \ <\text{atk-relation-set}\> \) \Rightarrow \) \[\text{[Function]}\]
\( (ret \ int) \)

get-n-relations

Determines the number of relations in a relation set.

\( set \quad \text{an} \ <\text{atk-relation-set}\> \)

\( ret \quad \text{an integer representing the number of relations in the set.} \)

atk-relation-set-get-relation \( (self \ <\text{atk-relation-set}\>) (i \ int) \) \[\text{[Function]}\]
\( \Rightarrow (ret \ <\text{atk-relation}\> \)

get-relation

Determines the relation at the specified position in the relation set.

\( set \quad \text{an} \ <\text{atk-relation-set}\> \)

\( i \quad \text{a gint representing a position in the set, starting from 0.} \)

\( ret \quad \text{a} \ <\text{atk-relation}\>, \text{which is the relation at position} \ i \ \text{in the set.} \)
17 AtkRelation

An object used to describe a relation between a object and one or more other objects.

17.1 Overview

An AtkRelation describes a relation between an object and one or more other objects. The actual relations that an object has with other objects are defined as an AtkRelationSet, which is a set of AtkRelations.

17.2 Usage

atk-relation-type-register (name mchars) ⇒ (ret <atk-relation-type>) [Function]
Associate name with a new <atk-relation-type>
name a name string
ret an <atk-relation-type> associated with name

atk-relation-type-get-name (type <atk-relation-type>) ⇒ (ret mchars) [Function]
Gets the description string describing the <atk-relation-type> type.
type The <atk-relation-type> whose name is required
ret the string describing the AtkRelationType

atk-relation-type-for-name (name mchars) ⇒ (ret <atk-relation-type>) [Function]
Get the <atk-relation-type> type corresponding to a relation name.
name a string which is the (non-localized) name of an ATK relation type.
ret the <atk-relation-type> enumerated type corresponding to the specified name, or <atk-relation-null> if no matching relation type is found.

atk-relation-new (targets <atk-object**>) (n_targets int) (relationship <atk-relation-type>) ⇒ (ret <atk-relation>) [Function]
Create a new relation for the specified key and the specified list of targets.
targets an array of pointers to <atk-objects>
n-targets number of <atk-objects> pointed to by targets
relationship an <atk-relation-type> with which to create the new <atk-relation>
ret a pointer to a new <atk-relation>

atk-relation-get-relation-type (self <atk-relation>) ⇒ (ret <atk-relation-type>) [Function]
get-relation-type
Gets the type of relation
relation  an <atk-relation>
ret    the type of relation

atk-relation-get-target (self <atk-relation>) ⇒ (ret <g-ptr-array*>)  [Function]

get-target  [Method]
   Gets the target list of relation
   relation  an <atk-relation>
   ret    the target list of relation

atk-relation-add-target (self <atk-relation>) (target <atk-object>)  [Function]
add-target  [Method]
   Adds the specified AtkObject to the target for the relation, if it is not already present.
   relation  an <atk-relation>
   target   an <atk-object>

Since ATK 1.9
18 AtkSelection

The ATK interface implemented by container objects whose children can be selected.

18.1 Overview

<atk-selection> should be implemented by UI components with children which are exposed by <atk-object-ref-child> and <atk-object-get-n-children>, if the use of the parent UI component ordinarily involves selection of one or more of the objects corresponding to those <atk-object> children - for example, selectable lists.

Note that other types of "selection" (for instance text selection) are accomplished a other ATK interfaces - <atk-selection> is limited to the selection/deselection of children.

18.2 Usage

atk-selection-add-selection (self <atk-selection*>(i int) ⇒ ret bool)

Adds the specified accessible child of the object to the object’s selection.

| selection  | a <gobject> instance that implements AtkSelectionIface |
| i          | a <gint> specifying the child index. |
| ret        | TRUE if success, FALSE otherwise. |

atk-selection-clear-selection (self <atk-selection*>( ) ⇒ ret bool)

Clears the selection in the object so that no children in the object are selected.

| selection  | a <gobject> instance that implements AtkSelectionIface |
| ret        | TRUE if success, FALSE otherwise. |

atk-selection-ref-selection (self <atk-selection*>(i int) ⇒ ret <atk-object>)

Gets a reference to the accessible object representing the specified selected child of the object. Note: callers should not rely on ‘#f’ or on a zero value for indication of whether AtkSelectionIface is implemented, they should use type checking/interface checking macros or the atk-get-accessible-value convenience method.

| selection  | a <gobject> instance that implements AtkSelectionIface |
| i          | a <gint> specifying the index in the selection set. (e.g. the ith selection as opposed to the ith child). |
| ret        | an <atk-object> representing the selected accessible, or ‘#f’ if selection does not implement this interface. |

atk-selection-get-selection-count (self <atk-selection*>( ) ⇒ ret int)

Gets the number of accessible children currently selected. Note: callers should not rely on ‘#f’ or on a zero value for indication of whether AtkSelectionIface is implemented, they should use type checking/interface checking macros or the atk-get-accessible-value convenience method.
selection  a <gobject> instance that implements AtkSelectionIface
ret       a gint representing the number of items selected, or 0 if selection does not implement this interface.

atk-selection-is-child-selected (self <atk-selection*>)(i int)  [Function]
            ⇒ (ret bool)
Determines if the current child of this object is selected Note: callers should not rely on ‘#f’ or on a zero value for indication of whether AtkSelectionIface is implemented, they should use type checking/interface checking macros or the atk-get-accessible-value convenience method.

selection  a <gobject> instance that implements AtkSelectionIface
i          a <gint> specifying the child index.
ret        a gboolean representing the specified child is selected, or 0 if selection does not implement this interface.

atk-selection-remove-selection (self <atk-selection*>)(i int)  [Function]
            ⇒ (ret bool)
Removes the specified child of the object from the object’s selection.

selection  a <gobject> instance that implements AtkSelectionIface
i          a <gint> specifying the index in the selection set. (e.g. the ith selection as opposed to the ith child).
ret        TRUE if success, FALSE otherwise.

atk-selection-select-all-selection (self <atk-selection*>)  [Function]
            ⇒ (ret bool)
Causes every child of the object to be selected if the object supports multiple selections.

selection  a <gobject> instance that implements AtkSelectionIface
ret        TRUE if success, FALSE otherwise.
19 AtkStateSet

An AtkStateSet determines a component’s state set.

19.1 Overview

An AtkStateSet determines a component’s state set. It is composed of a set of AtkStates.

19.2 Usage

atk-state-set-new ⇒ (ret <atk-state-set>)  [Function]

Creates a new empty state set.

ret a new <atk-state-set>

atk-state-set-is-empty (self <atk-state-set>) ⇒ (ret bool)  [Function]

is-empty  [Method]

Checks whether the state set is empty, i.e. has no states set.

set an <atk-state-type>

ret ‘#t’ if set has no states set, otherwise ‘#f’

atk-state-set-add-state (self <atk-state-set>) (type <atk-state-type>) ⇒ (ret bool)  [Function]

add-state  [Method]

Add a new state for the specified type to the current state set if it is not already present.

set an <atk-state-set>

type an <atk-state-type>

ret ‘#t’ if the state for type is not already in set.

atk-state-set-add-states (self <atk-state-set>) (types <atk-state-type*>) (n_types int)  [Function]

add-states  [Method]

Add the states for the specified types to the current state set.

set an <atk-state-set>

types an array of <atk-state-type>

n-types The number of elements in the array

atk-state-set-clear-states (self <atk-state-set>)  [Function]

clear-states  [Method]

Removes all states from the state set.

set an <atk-state-set>
atk-state-set-contains-state (self <atk-state-set>) (type <atk-state-type>) ⇒ (ret bool) [Function]
contains-state
Checks whether the state for the specified type is in the specified set.

  set   an <atk-state-set>
  type  an <atk-state-type>
  ret   ‘#t’ if type is the state type is in set.

atk-state-set-contains-states (self <atk-state-set>) (types <atk-state-type*>) (n_types int) ⇒ (ret bool) [Function]
contains-states
Checks whether the states for all the specified types are in the specified set.

  set      an <atk-state-set>
  types    an array of <atk-state-type>
  n-types  The number of elements in the array
  ret      ‘#t’ if all the states for type are in set.

atk-state-set-remove-state (self <atk-state-set>) (type <atk-state-type>) ⇒ (ret bool) [Function]
remove-state
Removes the state for the specified type from the state set.

  set     an <atk-state-set>
  type    an <atk-type>
  ret     ‘#t’ if type was the state type is in set.

and-sets
Constructs the intersection of the two sets, returning ‘#f’ if the intersection is empty.

  set     an <atk-state-set>
  compare-set another <atk-state-set>
  ret     a new <atk-state-set> which is the intersection of the two sets.

atk-state-set-or-sets (self <atk-state-set>) (compare_set <atk-state-set>) ⇒ (ret <atk-state-set>) [Function]
or-sets
Constructs the union of the two sets.

  set     an <atk-state-set>
  compare-set another <atk-state-set>
  ret     a new <atk-state-set> which is the union of the two sets, returning ‘#f’ if is empty.

xor-sets

Constructs the exclusive-or of the two sets, returning ‘#f’ is empty. The set returned by this operation contains the states in exactly one of the two sets.

set an <atk-state-set>

compare-set another <atk-state-set>

ret a new <atk-state-set> which contains the states which are in exactly one of the two sets.
20 AtkState

An AtkState describes a component’s particular state.

20.1 Overview

An AtkState describes a component’s particular state. The actual state of an component is described by its AtkStateSet, which is a set of AtkStates.

20.2 Usage

atk-state-type-get-name (type <atk-state-type>) ⇒ (ret mchars)  [Function]

Gets the description string describing the <atk-state-type> type.

type The <atk-state-type> whose name is required
ret the string describing the AtkStateType

atk-state-type-for-name (name mchars) ⇒ (ret <atk-state-type>)  [Function]

Gets the <atk-state-type> corresponding to the description string name.

name a character string state name
ret an <atk-state-type> corresponding to name
21 AtkStreamableContent

The ATK interface which provides access to streamable content.

21.1 Overview

An interface whereby an object allows its backing content to be streamed to clients. Typical implementors would be images or icons, HTML content, or multimedia display/rendering widgets.

Negotiation of content type is allowed. Clients may examine the backing data and transform, convert, or parse the content in order to present it in an alternate form to end-users.

The AtkStreamableContent interface is particularly useful for saving, printing, or post-processing entire documents, or for persisting alternate views of a document. If document content itself is being serialized, stored, or converted, then use of the AtkStreamableContent interface can help address performance issues. Unlike most ATK interfaces, this interface is not strongly tied to the current user-agent view of the a particular document, but may in some cases give access to the underlying model data.

21.2 Usage

atk-streamable-content-get-stream (self <atk-streamable-content*>) (mime_type mchars) ⇒ (ret <gio-channel*>)

Gets the content in the specified mime type.

streamable a GObject instance that implements AtkStreamableContentIface

mime-type a gchar* representing the mime type

ret A <gio-channel> which contains the content in the specified mime type.

atk-streamable-content-get-uri (self <atk-streamable-content*>) (mime_type mchars) ⇒ (ret mchars)

Get a string representing a URI in IETF standard format (see http://www.ietf.org/rfc/rfc2396.txt) from which the object’s content may be streamed in the specified mime-type, if one is available. If mime_type is NULL, the URI for the default (and possibly only) mime-type is returned.

Note that it is possible for get_uri to return NULL but for get_stream to work nonetheless, since not all GIOChannels connect to URIs.

streamable a GObject instance that implements AtkStreamableContentIface

mime-type a gchar* representing the mime type, or NULL to request a URI for the default mime type.
Returns a string representing a URI, or NULL if no corresponding URI can be constructed.

Since ATK 1.12
22 AtkTable

The ATK interface implemented for UI components which contain tabular or row/column information.

22.1 Overview

<atk-table> should be implemented by components which present elements ordered via rows and columns. It may also be used to present tree-structured information if the nodes of the trees can be said to contain multiple "columns". Individual elements of an <atk-table> are typically referred to as "cells", and these cells are exposed by <atk-table> as child <atk-objects> of the <atk-table>. Both row/column and child-index-based access to these children is provided.

Children of <atk-table> are frequently "lightweight" objects, that is, they may not have backing widgets in the host UI toolkit. They are therefore often transient.

Since tables are often very complex, <atk-table> includes provision for offering simplified summary information, as well as row and column headers and captions. Headers and captions are <atk-objects> which may implement other interfaces (<atk-text>, <atk-image>, etc.) as appropriate. <atk-table> summaries may themselves be (simplified) <atk-tables>, etc.

22.2 Usage

ltk-table-ref-at (self <atk-table*>)(row int)(column int) ⇒ (ret <atk-object>) [Function]

Get a reference to the table cell at row, column.

table a GObject instance that implements AtkTableIface
row a <gint> representing a row in table
column a <gint> representing a column in table
ret a AtkObject* representing the referred to accessible

ltk-table-get-index-at (self <atk-table*>)(row int)(column int) ⇒ (ret int) [Function]

Gets a <gint> representing the index at the specified row and column.

table a GObject instance that implements AtkTableIface
row a <gint> representing a row in table
column a <gint> representing a column in table
ret a <gint> representing the index at specified position. The value -1 is returned if the object at row,column is not a child of table or table does not implement this interface.

ltk-table-get-column-at-index (self <atk-table*>)(index int) ⇒ (ret int) [Function]

Gets a <gint> representing the column at the specified index.
**Table** a GObject instance that implements AtkTableInterface

**Index** a <gint> representing an index in **Table**

**Ret** a gint representing the column at the specified index, or -1 if the table does not implement this interface

**Atk-table-get-row-at-index** (*self <atk-table*>)(index int) ⇒ [Function]

Gets a <gint> representing the row at the specified index.

**Table** a GObject instance that implements AtkTableInterface

**Index** a <gint> representing an index in **Table**

**Ret** a gint representing the row at the specified index, or -1 if the table does not implement this interface

**Atk-table-get-n-columns** (*self <atk-table*>) ⇒ (ret int)

Gets the number of columns in the table.

**Table** a GObject instance that implements AtkTableIface

**Ret** a gint representing the number of columns, or 0 if value does not implement this interface.

**Atk-table-get-n-rows** (*self <atk-table*>) ⇒ (ret int)

Gets the number of rows in the table.

**Table** a GObject instance that implements AtkTableIface

**Ret** a gint representing the number of rows, or 0 if value does not implement this interface.

**Atk-table-get-column-extent-at** (*self <atk-table*>)(row int) ⇒ (ret int)

Gets the number of columns occupied by the accessible object at the specified row and column in the table.

**Table** a GObject instance that implements AtkTableIface

**Row** a <gint> representing a row in **Table**

**Column** a <gint> representing a column in **Table**

**Ret** a gint representing the column extent at specified position, or 0 if value does not implement this interface.

**Atk-table-get-row-extent-at** (*self <atk-table*>)(row int) ⇒ (ret int)

Gets the number of rows occupied by the accessible object at a specified row and column in the table.

**Table** a GObject instance that implements AtkTableIface

**Row** a <gint> representing a row in **Table**

**Column** a <gint> representing a column in **Table**
**ret** a gint representing the row extent at specified position, or 0 if value does not implement this interface.

**atk-table-get-caption** (*self <atk-table*>) ⇒ (*ret <atk-object>* ) [Function]

Gets the caption for the table.

*table* a GObject instance that implements AtkTableInterface

*ret* a AtkObject* representing the table caption, or ‘#f’ if value does not implement this interface.

**atk-table-get-column-description** (*self <atk-table*>) (*column int*) ⇒ (*ret mchars*) [Function]

Gets the description text of the specified column in the table

*table* a GObject instance that implements AtkTableIface

*column* a <gint> representing a column in table

*ret* a gchar* representing the column description, or ‘#f’ if value does not implement this interface.

**atk-table-get-row-description** (*self <atk-table*>) (*row int*) ⇒ (*ret mchars*) [Function]

Gets the description text of the specified row in the table

*table* a GObject instance that implements AtkTableIface

*row* a <gint> representing a row in table

*ret* a gchar* representing the row description, or ‘#f’ if value does not implement this interface.

**atk-table-get-column-header** (*self <atk-table*>) (*column int*) ⇒ (*ret <atk-object>* ) [Function]

Gets the column header of a specified column in an accessible table.

*table* a GObject instance that implements AtkTableIface

*column* a <gint> representing a column in the table

*ret* a AtkObject* representing the specified column header, or ‘#f’ if value does not implement this interface.

**atk-table-get-row-header** (*self <atk-table*>) (*row int*) ⇒ (*ret <atk-object>* ) [Function]

Gets the row header of a specified row in an accessible table.

*table* a GObject instance that implements AtkTableIface

*row* a <gint> representing a row in the table

*ret* a AtkObject* representing the specified row header, or ‘#f’ if value does not implement this interface.
atk-table-get-summary \((self <atk-table\*)\) \(\Rightarrow (ret <atk-object>)\) [Function]

Get the summary description of the table.

\(table\) a GObject instance that implements AtkTableIface

\(ret\) a AtkObject* representing a summary description of the table, or zero if value does not implement this interface.

atk-table-set-caption \((self <atk-table\*)\) \((caption <atk-object>)\) [Function]

Sets the caption for the table.

\(table\) a GObject instance that implements AtkTableIface

\(caption\) a <atk-object> representing the caption to set for \(table\)

atk-table-set-row-description \((self <atk-table\*)\) \((row \text{ int})\) \((description \text{ mchars})\) [Function]

Sets the description text for the specified \(row\) of \(table\).

\(table\) a GObject instance that implements AtkTableIface

\(row\) a <gint> representing a row in \(table\)

\(description\) a <gchar> representing the description text to set for the specified \(row\) of \(table\)

atk-table-set-column-description \((self <atk-table\*)\) \((column \text{ int})\) \((description \text{ mchars})\) [Function]

Sets the description text for the specified \(column\) of the \(table\).

\(table\) a GObject instance that implements AtkTableIface

\(column\) a <gint> representing a column in \(table\)

\(description\) a <gchar> representing the description text to set for the specified \(column\) of the \(table\)

atk-table-set-row-header \((self <atk-table\*)\) \((row \text{ int})\) \((header \text{ <atk-object>})\) [Function]

Sets the specified row header to \(header\).

\(table\) a GObject instance that implements AtkTableIface

\(row\) a <gint> representing a row in \(table\)

\(header\) an <atk-table>

atk-table-set-column-header \((self <atk-table\*)\) \((column \text{ int})\) \((header \text{ <atk-object>})\) [Function]

Sets the specified column header to \(header\).

\(table\) a GObject instance that implements AtkTableIface

\(column\) a <gint> representing a column in \(table\)

\(header\) an <atk-table>
atk-table-set-summary

```haskell
(atk-table-set-summary (self <atk-table*>) (accessible <atk-object>))
```

Sets the summary description of the table.

- **table**: a GObject instance that implements AtkTableIface
- **accessible**: an <atk-object> representing the summary description to set for table

atk-table-is-column-selected

```haskell
(atk-table-is-column-selected (self <atk-table*>)) (column int)
```

⇒ (ret bool)

Gets a boolean value indicating whether the specified `column` is selected.

- **table**: a GObject instance that implements AtkTableIface
- **column**: a <gint> representing a column in `table`
- **ret**: a gboolean representing if the column is selected, or 0 if value does not implement this interface.

atk-table-is-row-selected

```haskell
(atk-table-is-row-selected (self <atk-table*>)) (row int)
```

⇒ (ret bool)

Gets a boolean value indicating whether the specified `row` is selected.

- **table**: a GObject instance that implements AtkTableIface
- **row**: a <gint> representing a row in `table`
- **ret**: a gboolean representing if the row is selected, or 0 if value does not implement this interface.

atk-table-is-selected

```haskell
(atk-table-is-selected (self <atk-table*>)) (row int) (column int)
```

⇒ (ret bool)

Gets a boolean value indicating whether the accessible object at the specified `row` and `column` is selected.

- **table**: a GObject instance that implements AtkTableIface
- **row**: a <gint> representing a row in `table`
- **column**: a <gint> representing a column in `table`
- **ret**: a gboolean representing if the cell is selected, or 0 if value does not implement this interface.

atk-table-add-column-selection

```haskell
(atk-table-add-column-selection (self <atk-table*>)) (column int)
```

⇒ (ret bool)

Adds the specified `column` to the selection.

- **table**: a GObject instance that implements AtkTableIface
- **column**: a <gint> representing a column in `table`
- **ret**: a gboolean representing if the column was successfully added to the selection, or 0 if value does not implement this interface.
**atk-table-add-row-selection** *(self <atk-table*>) (row int) ⇒ (ret bool)*

 Adds the specified row to the selection.

 *table* a GObject instance that implements AtkTableIface
 *row* a <gint> representing a row in table
 *ret* a gboolean representing if row was successfully added to selection, or 0 if value does not implement this interface.

**atk-table-remove-column-selection** *(self <atk-table*>) (column int) ⇒ (ret bool)*

 Adds the specified column to the selection.

 *table* a GObject instance that implements AtkTableIface
 *column* a <gint> representing a column in table
 *ret* a gboolean representing if the column was successfully removed from the selection, or 0 if value does not implement this interface.

**atk-table-remove-row-selection** *(self <atk-table*>) (row int) ⇒ (ret bool)*

 Removes the specified row from the selection.

 *table* a GObject instance that implements AtkTableIface
 *row* a <gint> representing a row in table
 *ret* a gboolean representing if the row was successfully removed from the selection, or 0 if value does not implement this interface.
23 AtkText

The ATK interface implemented by components with text content.

23.1 Overview

<atk-text> should be implemented by <atk-objects> on behalf of widgets that have text content which is either attributed or otherwise non-trivial. <atk-objects> whose text content is simple, unattributed, and very brief may expose that content via <atk-object-get-name> instead; however if the text is editable, multi-line, typically longer than three or four words, attributed, selectable, or if the object already uses the 'name' ATK property for other information, the <atk-text> interface should be used to expose the text content. In the case of editable text content, <atk-editable-text> (a subtype of the <atk-text> interface) should be implemented instead.

<atk-text> provides not only traversal facilities and change notification for text content, but also caret tracking and glyph bounding box calculations. Note that the text strings are exposed as UTF-8, and are therefore potentially multi-byte, and caret-to-byte offset mapping makes no assumptions about the character length; also bounding box glyph-to-offset mapping may be complex for languages which use ligatures.

23.2 Usage

atk-text-get-text (self <atk-text*>) (start_offset int) (end_offset int) ⇒ (ret mchars) [Function]

Gets the specified text.

text an <atk-text>
start-offset start position
end-offset end position
ret the text from start-offset up to, but not including end-offset.

atk-text-get-character-at-offset (self <atk-text*>) (offset int) ⇒ (ret unsigned-int32) [Function]

Gets the specified text.

text an <atk-text>
offset position
ret the character at offset.

atk-text-get-text-after-offset (self <atk-text*>) (offset int) ⇒ (ret mchars) (start_offset int) (end_offset int) [Function]

Gets the specified text.

If the boundary_type if ATK_TEXT_BOUNDARY_CHAR the character after the offset is returned.
If the boundary_type is ATK_TEXT_BOUNDARY_WORD_START the returned string is from the word start after the offset to the next word start.
The returned string will contain the word after the offset if the offset is inside a word.
If the boundary_type is ATK_TEXT_BOUNDARY_WORD_END the returned string is from the word end at or after the offset to the next work end.
The returned string will contain the word after the offset if the offset is inside a word and will contain the word after the offset if the offset is not inside a word.
If the boundary_type is ATK_TEXT_BOUNDARY_SENTENCE_START the returned string is from the sentence start after the offset to the next sentence start.
The returned string will contain the sentence after the offset if the offset is inside a sentence or if the offset is not inside a sentence.
If the boundary_type is ATK_TEXT_BOUNDARY_SENTENCE_END the returned string is from the sentence end at or after the offset to the next sentence end.
The returned string will contain the sentence after the offset if the offset is inside a sentence and will contain the sentence after the sentence after the offset if the offset is not inside a sentence.
If the boundary_type is ATK_TEXT_BOUNDARY_LINE_START the returned string is from the line start after the offset to the next line start.
If the boundary_type is ATK_TEXT_BOUNDARY_LINE_END the returned string is from the line end at or after the offset to the next line start.

text an <atk-text>
offset position boundary-type
boundary-type
An <atk-text-boundary>
start-offset the start offset of the returned string
end-offset the offset of the first character after the returned substring
ret the text after offset bounded by the specified boundary-type.

atk-text-get-text-at-offset (self <atk-text*>)(offset int) [Function]
  (boundary_type <atk-text-boundary>) ⇒ (ret mchars) (start_offset int)
  (end_offset int)

Gets the specified text.
If the boundary_type if ATK_TEXT_BOUNDARY_CHAR the character at the offset is returned.
If the boundary_type is ATK_TEXT_BOUNDARY_WORD_START the returned string is from the word start at or before the offset to the word start after the offset.
The returned string will contain the word at the offset if the offset is inside a word and will contain the word before the offset if the offset is not inside a word.
If the boundary_type is ATK_TEXT_BOUNDARY_WORD_END the returned string is from the word end before the offset to the word end at or after the offset.

The returned string will contain the word at the offset if the offset is inside a word and will contain the word after to the offset if the offset is not inside a word.

If the boundary type is ATK_TEXT_BOUNDARY_SENTENCE_START the returned string is from the sentence start at or before the offset to the sentence start after the offset.

The returned string will contain the sentence at the offset if the offset is inside a sentence and will contain the sentence before the offset if the offset is not inside a sentence.

If the boundary_type is ATK_TEXT_BOUNDARY_SENTENCE_END the returned string is from the sentence end before the offset to the sentence end at or after the offset.

The returned string will contain the sentence at the offset if the offset is inside a sentence and will contain the sentence after the offset if the offset is not inside a sentence.

If the boundary_type is ATK_TEXT_BOUNDARY_LINE_START the returned string is from the line start at or before the offset to the line start after the offset.

If the boundary_type is ATK_TEXT_BOUNDARY_LINE_END the returned string is from the line end before the offset to the line end at or after the offset.

\[
\begin{align*}
\text{text} & \quad \text{an <atk-text>} \\
\text{offset} & \quad \text{position} \\
\text{boundary-type} & \quad \text{An <atk-text-boundary>} \\
\text{start-offset} & \quad \text{the start offset of the returned string} \\
\text{end-offset} & \quad \text{the offset of the first character after the returned substring} \\
\text{ret} & \quad \text{the text at offset bounded by the specified boundary-type.}
\end{align*}
\]

\textbf{atk-text-get-text-before-offset (self <atk-text*>)(offset int)} \quad \text{[Function]}

\((\text{boundary_type <atk-text-boundary*>}) \Rightarrow (\text{ret mchars})(\text{start_offset int}) (\text{end_offset int})\)

Gets the specified text.

If the boundary_type if ATK_TEXT_BOUNDARY_CHAR the character before the offset is returned.

If the boundary_type is ATK_TEXT_BOUNDARY_WORD_START the returned string is from the word start before the word start before the offset to the word start before the offset.

The returned string will contain the word before the offset if the offset is inside a word and will contain the word before the word before the offset if the offset is not inside a word.
If the boundary_type is ATK_TEXT_BOUNDARY_WORD_END the returned string is from the word end before the word end at or before the offset to the word end at or before the offset.

The returned string will contain the word before the offset if the offset is inside a word or if the offset is not inside a word.

If the boundary type is ATK_TEXT_BOUNDARY_SENTENCE_START the returned string is from the sentence start before the sentence start before the offset to the sentence start before the offset.

The returned string will contain the sentence before the offset if the offset is inside a sentence and will contain the sentence before the sentence before the offset if the offset is not inside a sentence.

If the boundary_type is ATK_TEXT_BOUNDARY_SENTENCE_END the returned string is from the sentence end before the sentence end at or before the offset to the sentence end at or before the offset.

The returned string will contain the sentence before the offset if the offset is inside a sentence or if the offset is not inside a sentence.

If the boundary type is ATK_TEXT_BOUNDARY_LINE_START the returned string is from the line start before the line start ar or before the offset to the line start ar or before the offset.

If the boundary_type is ATK_TEXT_BOUNDARY_LINE_END the returned string is from the line end before the line end before the offset to the line end before the offset.

\[
\begin{align*}
\text{text} & \quad \text{an } \texttt{<atk-text>} \\
\text{offset} & \quad \text{position} \\
\text{boundary-type} & \quad \text{An } \texttt{<atk-text-boundary>} \\
\text{start-offset} & \quad \text{the start offset of the returned string} \\
\text{end-offset} & \quad \text{the offset of the first character after the returned substring} \\
\text{ret} & \quad \text{the text before offset bounded by the specified boundary-type.}
\end{align*}
\]

\[
\texttt{atk-text-get-caret-offset (self <atk-text*>)} \Rightarrow (\text{ret int}) \quad [\text{Function}]
\]

Gets the offset position of the caret (cursor).

\[
\begin{align*}
\text{text} & \quad \text{an } \texttt{<atk-text>} \\
\text{ret} & \quad \text{the offset position of the caret (cursor)}.
\end{align*}
\]

\[
\texttt{atk-text-get-character-extents (self <atk-text*>)(offset int)} \quad [\text{Function}]
\]

\[
\begin{align*}
(coords <\texttt{atk-coord-type}>) \Rightarrow (x \text{ int})(y \text{ int})(width \text{ int})(height \text{ int})
\end{align*}
\]

Get the bounding box containing the glyph representing the character at a particular text offset.

\[
\begin{align*}
\text{text} & \quad \text{an } \texttt{<atk-text>}
\end{align*}
\]
offset  The offset of the text character for which bounding information is required.

x  Pointer for the x coordinate of the bounding box

y  Pointer for the y coordinate of the bounding box

width  Pointer for the width of the bounding box

height  Pointer for the height of the bounding box

coords  specify whether coordinates are relative to the screen or widget window

**atk-text-get-run-attributes** *(self <atk-text*>)(offset int) ⇒ [Function]*

(ret <atk-attribute-set*>)(start_offset int)(end_offset int)

Creates an <atk-attribute-set> which consists of the attributes explicitly set at the position offset in the text. start-offset and end-offset are set to the start and end of the range around offset where the attributes are invariant. Note that end-offset is the offset of the first character after the range. See the enum AtkTextAttribute for types of text attributes that can be returned. Note that other attributes may also be returned.

text  an <atk-text>

offset  the offset at which to get the attributes

start-offset  the address to put the start offset of the range

end-offset  the address to put the end offset of the range

ret  an <atk-attribute-set> which contains the attributes explicitly set at offset. This <atk-attribute-set> should be freed by a call to atk-attribute-set-free.

**atk-text-get-default-attributes** *(self <atk-text*>) ⇒ [Function]*

<atk-attribute-set>

Creates an <atk-attribute-set> which consists of the default values of attributes for the text. See the enum AtkTextAttribute for types of text attributes that can be returned. Note that other attributes may also be returned.

text  an <atk-text>

ret  an <atk-attribute-set> which contains the default values of attributes at offset. This <atk-attribute-set> should be freed by a call to atk-attribute-set-free.

**atk-text-get-character-count** *(self <atk-text*>) ⇒ [Function]*

(ret int)

Gets the character count.

text  an <atk-text>

ret  the number of characters.
atk-text-get-offset-at-point (self <atk-text*>) (x int) (y int)  [Function]
(coords <atk-coord-type>) ⇒ (ret int)
Gets the offset of the character located at coordinates x and y. x and y are interpreted as being relative to the screen or this widget’s window depending on coords.

- **text**: an <atk-text>
- **x**: screen x-position of character
- **y**: screen y-position of character
- **coords**: specify whether coordinates are relative to the screen or widget window
- **ret**: the offset to the character which is located at the specified x and y coordinates.

atk-text-get-bounded-ranges (self <atk-text*>) (rect <atk-text-rectangle*>) (coord-type <atk-coord-type>) (x-clip-type <atk-text-clip-type>) (y-clip-type <atk-text-clip-type>) ⇒ (ret <atk-text-range**>)
Get the ranges of text in the specified bounding box.

- **text**: an <atk-text>
- **rect**: An AtkTextRectangle giving the dimensions of the bounding box.
- **coord-type**: Specify whether coordinates are relative to the screen or widget window.
- **x-clip-type**: Specify the horizontal clip type.
- **y-clip-type**: Specify the vertical clip type.
- **ret**: Array of AtkTextRange. The last element of the array returned by this function will be NULL.

Since ATK 1.3

atk-text-get-range-extents (self <atk-text*>) (start_offset int) (end_offset int) (coord-type <atk-coord-type>) (rect <atk-text-rectangle**>)
Get the bounding box for text within the specified range.

- **text**: an <atk-text>
- **start-offset**: The offset of the first text character for which boundary information is required.
- **end-offset**: The offset of the text character after the last character for which boundary information is required.
- **coord-type**: Specify whether coordinates are relative to the screen or widget window.
rect A pointer to a AtkTextRectangle which is filled in by this function.

Since ATK 1.3

atk-text-free-ranges (ranges <atk-text-range**>) [Function]
Frees the memory associated with an array of AtkTextRange. It is assumed that the array was returned by the function atk_text_get_bounded_ranges and is NULL terminated.

ranges A pointer to an array of <atk-text-range> which is to be freed.

Since ATK 1.3

atk-text-get-n-selections (self <atk-text*>) ⇒ (ret int) [Function]
Gets the number of selected regions.

text an <atk-text>
ret The number of selected regions, or -1 if a failure occurred.

atk-text-get-selection (self <atk-text*>)(selection_num int) ⇒ [Function]
(ret mchars) (start_offset int) (end_offset int)
Gets the text from the specified selection.

text an <atk-text>

selection-num The selection number. The selected regions are assigned numbers that correspond to how far the region is from the start of the text. The selected region closest to the beginning of the text region is assigned the number 0, etc. Note that adding, moving or deleting a selected region can change the numbering.

start-offset passes back the start position of the selected region
end-offset passes back the end position of (e.g. offset immediately past) the selected region
ret the selected text.

atk-text-add-selection (self <atk-text*>)(start_offset int) ⇒ (ret bool)
Adds a selection bounded by the specified offsets.

text an <atk-text>

start-offset the start position of the selected region
end-offset the offset of the first character after the selected region.
ret ‘#t’ if success, ‘#f’ otherwise
atk-text-remove-selection (self <atk-text*>)(selection_num int) ⇒ (ret bool)
Removes the specified selection.

text an <atk-text>
selection-num
The selection number. The selected regions are assigned numbers that correspond to how far the region is from the start of the text. The selected region closest to the beginning of the text region is assigned the number 0, etc. Note that adding, moving or deleting a selected region can change the numbering.

ret ‘#t’ if success, ‘#f’ otherwise

atk-text-set-selection (self <atk-text*>)(selection_num int)(start_offset int)(end_offset int) ⇒ (ret bool)
Changes the start and end offset of the specified selection.

text an <atk-text>
selection-num
The selection number. The selected regions are assigned numbers that correspond to how far the region is from the start of the text. The selected region closest to the beginning of the text region is assigned the number 0, etc. Note that adding, moving or deleting a selected region can change the numbering.

start-offset
the new start position of the selection

end-offset
the new end position of (e.g. offset immediately past) the selection

ret ‘#t’ if success, ‘#f’ otherwise

atk-text-set-caret-offset (self <atk-text*>)(offset int) ⇒ (ret bool)
Sets the caret (cursor) position to the specified offset.

text an <atk-text>
offset position

ret ‘#t’ if success, ‘#f’ otherwise.

atk-text-attribute-get-name (attr <atk-text-attribute>) ⇒ (ret mchars)
Gets the name corresponding to the <atk-text-attribute>

attr
The <atk-text-attribute> whose name is required

ret
a string containing the name; this string should not be freed

atk-text-attribute-for-name (name mchars) ⇒ (ret <atk-text-attribute>)
Get the <atk-text-attribute> type corresponding to a text attribute name.
name a string which is the (non-localized) name of an ATK text attribute.

ret the <atk-text-attribute> enumerated type corresponding to the specified name, or <atk-text-attribute-invalid> if no matching text attribute is found.

atk-text-attribute-get-value (attr <atk-text-attribute>) [Function]
(index int) ⇒ (ret mchars)

Gets the value for the index of the <atk-text-attribute>

attr The <atk-text-attribute> for which a value is required

index The index of the required value

ret a string containing the value; this string should not be freed; NULL is returned if there are no values maintained for the attr value.
# Chapter 24: AtkUtil

A set of ATK utility functions for event and toolkit support.

## 24.1 Overview

A set of ATK utility functions which are used to support event registration of various types, and obtaining the 'root' accessible of a process and information about the current ATK implementation and toolkit version.

## 24.2 Usage

**atk-add-focus-tracker**

```c
(atk-add-focus-tracker (focus_tracker <atk-event-listener>) ⇒ [Function]
  (ret unsigned-int))
```

Adds the specified function to the list of functions to be called when an object receives focus.

- `focus_tracker`: Function to be added to the list of functions to be called when an object receives focus.
- `ret`: added focus tracker id, or 0 on failure.

**atk-remove-focus-tracker**

```c
(atk-remove-focus-tracker (tracker_id unsigned-int))
```

Removes the specified focus tracker from the list of functions to be called when any object receives focus.

- `tracker-id`: the id of the focus tracker to remove

**atk-focus-tracker-init**

```c
(atk-focus-tracker-init (init <atk-event-listener-init>) ⇒ [Function]
```

Specifies the function to be called for focus tracker initialization. This function should be called by an implementation of the ATK interface if any specific work needs to be done to enable focus tracking.

- `init`: Function to be called for focus tracker initialization

**atk-focus-tracker-notify**

```c
(atk-focus-tracker-notify (object <atk-object>) ⇒ [Function]
```

Cause the focus tracker functions which have been specified to be executed for the object.

- `object`: an <atk-object>

**atk-add-global-event-listener**

```c
(atk-add-global-event-listener (listener <g-signal-emission-hook>) (event_type mchars) ⇒ [Function]
  (ret unsigned-int))
```

Adds the specified function to the list of functions to be called when an event of type event_type occurs.

- `listener`: the listener to notify
- `event-type`: the type of event for which notification is requested
- `ret`: added event listener id, or 0 on failure.
atk-remove-global-event-listener \( (\text{listener\_id unsigned-int}) \) [Function]
- Removes the specified event listener
  \( \text{listener\_id} \)  the id of the event listener to remove

atk-remove-key-event-listener \( (\text{listener\_id unsigned-int}) \) [Function]
- Removes the specified event listener
  \( \text{listener\_id} \)  the id of the event listener to remove

atk-get-root  \( \Rightarrow (\text{ret <atk-object>}) \) [Function]
- Gets the root accessible container for the current application.
  \( \text{ret} \)  the root accessible container for the current application

atk-get-focus-object  \( \Rightarrow (\text{ret <atk-object>}) \) [Function]
- Gets the currently focused object.
  Returns:
  \( \text{ret} \)  the currently focused object for the current application
- Since ATK 1.6

atk-get-toolkit-name  \( \Rightarrow (\text{ret mchars}) \) [Function]
- Gets name string for the GUI toolkit implementing ATK for this application.
  \( \text{ret} \)  name string for the GUI toolkit implementing ATK for this application

atk-get-toolkit-version  \( \Rightarrow (\text{ret mchars}) \) [Function]
- Gets version string for the GUI toolkit implementing ATK for this application.
  \( \text{ret} \)  version string for the GUI toolkit implementing ATK for this application
25 AtkValue

The ATK interface implemented by valuators and components which display or select a value from a bounded range of values.

25.1 Overview

<atk-value> should be implemented for components which either display a value from a bounded range, or which allow the user to specify a value from a bounded range, or both. For instance, most sliders and range controls, as well as dials, should have <atk-object> representations which implement <atk-value> on the component’s behalf. <at-kvalues> may be read-only, in which case attempts to alter the value return FALSE to indicate failure.

25.2 Usage

atk-value-get-current-value (self <atk-value*>)(value <gvalue>)

[Function]

Gets the value of this object.

obj a GObject instance that implements AtkValueIface

value a <gvalue> representing the current accessible value

atk-value-get-maximum-value (self <atk-value*>)(value <gvalue>)

[Function]

Gets the maximum value of this object.

obj a GObject instance that implements AtkValueIface

value a <gvalue> representing the maximum accessible value

atk-value-get-minimum-value (self <atk-value*>)(value <gvalue>)

[Function]

Gets the minimum value of this object.

obj a GObject instance that implements AtkValueIface

value a <gvalue> representing the minimum accessible value

atk-value-set-current-value (self <atk-value*>)(value <gvalue>) ⇒ (ret bool)

[Function]

Sets the value of this object.

obj a GObject instance that implements AtkValueIface

value a <gvalue> which is the desired new accessible value.

ret ‘#t’ if new value is successfully set, ‘#f’ otherwise.

atk-value-get-minimum-increment (self <atk-value*>)(value <gvalue>)

[Function]

Gets the minimum increment by which the value of this object may be changed. If zero, the minimum increment is undefined, which may mean that it is limited only by the floating point precision of the platform.
obj a GObject instance that implements AtkValueIface

value a <gvalue> representing the minimum increment by which the accessible value may be changed

Since ATK 1.12
Concept Index

(Index is nonexistent)
Function Index

A
add.................................................. 29
add-relationship.............................. 25
add-state.......................................... 35
add-states....................................... 35
add-target....................................... 32
and-sets.......................................... 36
atk-action-do-action......................... 2
atk-action-get-description.................. 2
atk-action-get-keybinding................... 3
atk-action-get-localized-name.............. 3
atk-action-get-n-actions.................... 2
atk-action-get-name.......................... 3
atk-action-set-description................. 3
atk-add-focus-tracker......................... 56
atk-add-global-event-listener............... 56
atk-component-add-focus-handler.......... 4
atk-component-contains..................... 4
atk-component-get-alpha..................... 7
atk-component-get-attributes............... 4
atk-component-get-layer..................... 5
atk-component-get-mdi-zorder.............. 5
atk-component-get-position............... 5
atk-component-get-size..................... 5
atk-component-grab-focus.................... 6
atk-component-remove-focus-handler........ 6
atk-component-set-extends.................. 6
atk-component-set-position................. 6
atk-component-set-size..................... 7
atk-document-get-attribute-value......... 8
atk-document-get-attributes............... 9
atk-document-get-document................. 8
atk-document-get-document-type........... 8
atk-document-get-local........................ 9
atk-document-get-attribute-value......... 8
atk-editable-text-copy-text............... 10
atk-editable-text-cut-text................ 10
atk-editable-text-delete-text............. 11
atk-editable-text-insert-text............. 10
atk-editable-text-paste-text.............. 11
atk-editable-text-set-text-contents....... 10
atk-focus-tracker-init.......................... 56
atk-focus-tracker-notify.................... 56
atk-get-default-registry................... 28
atk-get-focus-object........................ 57
atk-get-root.................................. 57
atk-get-toolkit-name........................ 57
atk-get-toolkit-version...................... 57
atk-gobject-accessible-for-object........ 12
atk-gobject-accessible-get-object........ 12
atk-hyperlink-get-end-index................ 14
atk-hyperlink-get-n-anchors................ 15
atk-hyperlink-get-object................... 14
atk-hyperlink-get-start-index.............. 14
atk-hyperlink-get-uri........................ 14
atk-hyperlink-impl-get-hyperlink......... 13
atk-hyperlink-is-inline..................... 15
atk-hyperlink-is-selected-link............... 15
atk-hyperlink-is-valid...................... 15
atk-hypertext-get-link........................ 16
atk-hypertext-get-link-index................ 16
atk-hypertext-get-n-links................... 16
atk-image-get-image-description........... 17
atk-image-get-image-locale................... 18
atk-image-get-image-position................ 17
atk-image-get-image-size.................... 18
atk-image-set-image-description............ 17
atk-implementor-ref-accessible............. 22
atk-no-op-object-factory-new............... 19
atk-no-op-object-new.......................... 20
atk-object-add-relationship................ 25
atk-object-factory-invalidate.............. 21
atk-object-get-attributes................... 26
atk-object-get-description................... 22
atk-object-get-index-in-parent............... 24
atk-object-get-layer........................... 23
atk-object-get-mdi-zorder.................... 23
atk-object-get-name........................... 22
atk-object-get-parent.......................... 22
atk-object-get-role............................ 24
atk-object-initialize.......................... 25
atk-object notify-state-change............... 25
atk-object-ref-accessible-child............ 23
atk-object-ref-relationship-set........... 23
atk-object-ref-state-set.................... 24
atk-object-remove-relationship............... 25
atk-object-set-description................... 24
atk-object-set-name........................... 24
atk-object-set-parent.......................... 24
atk-object-set-role............................ 25
atk-registry-get-factory.................... 27
atk-registry-get-factory-type............... 27
atk-registry-get-factory-type................ 27
atk-relation-add-target..................... 32
atk-relation-get-relation-type............... 31
atk-relation-get-target..................... 32
atk-relation-get-new.......................... 31
atk-relation-set-add......................... 29
atk-relation-set-contains................... 29
atk-relation-set-get-n-relations............ 30
atk-relation-set-get-relationship........... 30
atk-relation-set-new......................... 29
atk-relation-set-remove...................... 29
atk-relation-type-for-name................... 31
atk-relation-type-get-name................... 31
atk-relation-type-register.................... 31
atk-remove-focus-tracker..................... 56
atk-remove-global-event-listener........... 57
atk-remove-key-event-listener............... 57
Function Index

atk-role-for-name ..................................... 26
atk-role-get-localized-name ....................... 26
atk-role-get-name .................................. 26
atk-selection-add-selection ....................... 33
atk-selection-clear-selection ..................... 33
atk-selection-get-selection-count ................. 33
atk-selection-is-child-selected ................... 34
atk-selection-ref-selection ....................... 34
atk-selection-remove-selection .................... 34
atk-selection-select-all-selection ................. 34
atk-state-set-add-state ................................ 35
atk-state-set-add-states .............................. 35
atk-state-set-and-sets ................................ 36
atk-state-set-clear-states ......................... 35
atk-state-set-contains-state ....................... 36
atk-state-set-contains-states ...................... 36
atk-state-set-is-empty ................................ 35
atk-state-set-new .................................... 35
atk-state-set-or-sets ................................ 36
atk-state-set-remove-state ......................... 36
atk-state-set-xor-sets ................................ 37
atk-state-type-for-name .............................. 38
atk-state-type-get-name ................................ 38
atk-streamable-content-get-stream ................. 39
atk-streamable-content-get-uri .................... 39
atk-table-add-column-selection .................... 45
atk-table-add-row-selection ......................... 46
atk-table-get-caption ................................ 43
atk-table-get-column-at-index ..................... 41
atk-table-get-column-description .................. 43
atk-table-get-column-extent-at .................... 42
atk-table-get-column-header ....................... 43
atk-table-get-index-at ................................ 41
atk-table-get-n-columns .................................... 42
atk-table-get-n-rows .................................... 42
atk-table-get-row-at-index ......................... 42
atk-table-get-row-description ..................... 43
atk-table-get-row-extent-at ....................... 42
atk-table-get-row-header ............................ 43
atk-table-get-summary ................................ 44
atk-table-is-column-selected ....................... 45
atk-table-is-row-selected ............................ 45
atk-table-ref-at ...................................... 41
atk-table-remove-column-selection ................ 46
atk-table-remove-row-selection .................... 46
atk-table-set-caption ................................ 44
atk-table-set-column-description .................. 44
atk-table-set-column-header ....................... 44
atk-table-set-row-description ..................... 44
atk-table-set-row-header ............................ 44
atk-table-set-summary ................................ 45
atk-text-add-selection ................................ 53
atk-text-attribute-for-name ...................... 54
atk-text-attribute-get-name ....................... 54
atk-text-attribute-get-value ...................... 55
atk-text-free-ranges ................................ 53
atk-text-get-bounded-ranges ....................... 52
atk-text-get-caret-offset ......................... 50
atk-text-get-character-at-offset ................. 47
atk-text-get-character-count ..................... 51
atk-text-get-character-extends ................. 50
atk-text-get-default-attributes ................. 51
atk-text-get-n-selections ......................... 53
atk-text-get-offset-at-point ...................... 52
atk-text-get-range-extends ....................... 52
atk-text-get-run-attributes ....................... 51
atk-text-get-selection ............................ 53
atk-text-get-text .................................... 47
atk-text-get-text-after-offset ..................... 47
atk-text-get-text-at-offset ....................... 48
atk-text-get-text-before-offset ................... 49
atk-text-remove-selection ......................... 54
atk-text-set-caret-offset ......................... 54
atk-text-set-selection .............................. 54
atk-value-get-current-value ....................... 58
atk-value-get-maximum-value ....................... 58
atk-value-get-minimum-increment .................. 58
atk-value-get-minimum-value ....................... 58
atk-value-set-current-value ....................... 58

C
clear-states ........................................... 35
contains ................................................. 29
contains-state .......................................... 36
contains-states ......................................... 36

G
get-attributes ........................................ 26
get-description ........................................ 22
get-end-index .......................................... 14
get-factory ............................................ 27
get-factory-type ....................................... 27
get-index-in-parent .................................... 24
get-layer ................................................ 23
get-mdi-zorder .......................................... 23
get-n-anchors .......................................... 15
get-n-relations ......................................... 30
get-name ................................................ 22
get-object ............................................... 12, 14
get-parent ............................................... 22
get-relation ............................................. 30
get-relation-type ....................................... 31
get-role ................................................ 24
get-start-index ......................................... 14
get-target ............................................... 32
get-uri ................................................ 14

I
initialize ............................................... 25
invalidate .............................................. 21
is-empty ................................................. 35
is-inline ................................................ 15
is-selected-link.............................. 15
is-valid......................................... 15

N
notify-state-change......................... 25

O
or-sets........................................ 36

R
ref-accessible-child....................... 23
ref-relation-set............................ 23

S
set-description.............................. 24
set-factory-type............................ 27
set-name...................................... 24
set-parent.................................... 24
set-role...................................... 25

X
xor-sets....................................... 37

ref-state-set............................... 24
remove......................................... 29
remove-relationship....................... 25
remove-state................................. 36