# Acrobat and PDFL SDKs and Accessibility

This PDF is programmatically generated: Review copy only

# Contents

CHAPTER

ONE

# ACROBAT AND PDFL SDKS AND ACCESSIBILITY

Adobe provides methods to make the content of a PDF file available to assistive technology such as screen readers:

- On the Microsoft® Windows® operating system, Acrobat and Adobe Reader export PDF content as COM objects. Accessibility applications such as screen readers can interface with Acrobat or Adobe Reader in two ways:
  - Through the Microsoft Active Accessibility (MSAA) interface, using MSAA objects that Acrobat or Adobe Reader exports
  - Directly through exported COM objects that allow access to the PDF document's internal structure, called the document object model (DOM).

The DOM and MSAA models are related, and developers can use either or both. Acrobat issues notifications to accessibility clients about interesting events occurring in the PDF file window and responds to requests from such clients.

> **Warning:** This document assumes that you are familiar with the ATK architecture.

## 1.1 Determining rendering order and logical order

When rendering documents on the screen, Acrobat provides visual fidelity in a device-independent manner. However, the order in which Acrobat renders characters is not necessarily the same as the order in which they are to be read. Acrobat does not use standard system services that are used by assistive technology to capture content displayed on the screen.

Tagged PDF , introduced in PDF 1.4, defines a logical structure for the document that corresponds to the logical order of the content, regardless of the order in which the content is rendered. Acrobat uses the logical structure of a Tagged PDF document to determine word order. Through the accessibility interfaces, Acrobat can deliver the text of the PDF file as Unicode and can also make active elements such as links and form fields accessible.

Note: Acrobat can determine the logical structure of an untagged PDF file to some extent, but the results may be less satisfactory.

### 1.1.1 Accessing documents and pages

Through the accessibility interfaces, Acrobat can deliver contents of the entire PDF document contents or only the current visible pages, regardless of what part of the document is visible on the screen:

- Delivering the entire document permits assistive technology to search the document for the next link or next instance of text.
- Delivering individual pages is necessary for very large documents that might exhaust the resources of the assistive technology.

The user controls the delivery method using the reading preferences.

## 1.2 Processing inaccessible documents

A document can be inaccessible for one of the following reasons:

- It is protected by security settings
- It is, or appears, empty
- It is temporarily unavailable

The interfaces treat inaccessible documents as follows:

- Acrobat exports an MSAA object from the document, whose type indicates the reason for the inaccessibility.
- In Acrobat 6.0, inaccessible documents do not export any PDF DOM objects; attempts to retrieve PDF DOM objects from it fail without indicating the reason.
- In Acrobat 7.0 and later, the DOM interface returns objects that represent the document, and DOM methods can be used to find out why the document is inaccessible.

### 1.2.1 Processing protected documents

A document may have security settings that make it inaccessible. This can occur under the following conditions:

- It uses 40-bit RC4 encryption, and the author has forbidden copying text and graphics.
- It uses 128-bit RC4 encryption, and the author has forbidden making the contents accessible.
- It uses a non-standard security handler, and the document settings forbid making the contents accessible.

In these cases, the user must contact the document author to provide a version that permits accessibility.

The following occurs when a document has security settings that make it inaccessible:

- Acrobat exports an MSAA `IAccessible` object warning of a possible error. This object has the role `ROLE_SYSTEM_TEXT` and the name "`Alert: Protection Failure`". For more information, see PDF Protected Document.
- When using the DOM interface in Acrobat 7, `GetDocInfo` returns the status `DocState_Protected`.

You can become an Adobe Trusted Partner and create Trusted Assistive Technology. Trusted Partners are developers of assistive products that respect the copy protection of encrypted PDF files, and can gain access to 40-bit encrypted files. For more information on becoming a Trusted Partner, see http://www.adobe.com/go/acrobat_developer .

### 1.2.2 Processing empty documents

A document can be inaccessible because it is empty, or it can appear empty because of the way the PDF was created. For instance, scanned images that have not been run through an optical character recognition (OCR) tool appear to be empty. Malformed structure trees can also make a document appear empty.

The following occurs when a document appears to be empty:

- Acrobat exports an MSAA `IAccessible` object warning of a possible error. This object has the role `ROLE_SYSTEM_TEXT` and the name "`Alert: Empty document`". If Acrobat is delivering information a page at a time, a genuinely empty page also generates this warning. For more information, see Empty PDF Document.
- When using the DOM in Acrobat 7, `GetDocInfo` returns the status `DocState_Empty`.

 

### 1.2.3 Processing unavailable documents

When a document is unavailable, Acrobat returns similar objects from MSAA and DOM. A document may be unavailable for one of several reasons:

- If Acrobat is still preparing the document for access and the assistive technology attempts to read the document, the MSAA object name is "`Alert: Document being processed`".
- If Acrobat is waiting for a document on the web to download to the disk, the MSAA object name is "`Alert: Document downloading`".
- If the user cancels processing so that the document will never be available, the MSAA object name is "`Alert: Document unavailable`".

In all these cases, when using the DOM, the status returned in `GetDocInfo` is `DocState_Unavailable`.

## 1.3 Handling event notifications

Each open document in Acrobat is associated with its own window handle. All `WinNotifyEvent` notifications for any part of the document use that window handle. For the PDF window:

- If `childID == CHILDID_SELF` (that is, 0), the event is for the entire document or page.
- If the `childID` parameter of the notification is non-zero, the event is for an object within the window, such as a form field, link, comment, or some part of the page content such a line or paragraph of text.

For Acrobat 7.0 and later, the following occurs:

- If the selection is set or changed, `VALUECHANGE` is notified, with the `childID` of the `IAccessible` object containing the beginning of the selection.
- If the selection is set, `SELECTION` is notified on the document (with a `childID` of `0` ).
- If the selection is cleared, `SELECTIONREMOVE` is notified on the document.
- If the selection is extended, `SELECTIONADD` is notified, except when it is extended via keyboard commands (in that case `SELECTIONREMOVE` followed by `SELECTION` is notified).
- A `LOCATIONCHANGE` notification is issued when the caret moves. `SHOW` and `HIDE` notifications are issued when the caret is activated and deactivated.

### 1.3.1 Retrieving an MSAA object for an event

You can retrieve an `IAccessible` object from event notifications by using the MSAA function `AccessibleObjectFromEvent`. This object represents the document or an element within the document.

Some events always return an object of a particular type. For others, you must determine the type of the object from the role and specific `childID`. The meaning of the event can be different for different types of objects. For more information, see Identifying IAccessible objects in a document.

Acrobat posts the following `WinEvent` notifications:

| Notification | Description |
|---|---|
| `EVENT_OBJECT_FOCUS` | The document window, a link, a comment, or a form field has received keyboard focus. |
| `AccessibleObjectFromEvent` | Returns the appropriate `IAccessible` object, either for the document or page itself or for the link, comment, or form field. The `childID` parameter identifies the object. |
| `EVENT_OBJECT_LOCATIONCHANGE` | The caret (text cursor) has moved. If the caret is in a text edit field containing keyboard focus, the value of the text field may also have changed.<br>The `idObjectType` parameter for this event is `objid_caret`. `AccessibleObjectFromEvent` returns an `IAccessible` object for the caret. |
| `EVENT_OBJECT_STATECHANGE` | If the `idObject` parameter is `CHILDID_SELF`, the current document or page has changed its state by opening or closing a comment. The client should update its copy of the document content. Only the `IAccessible` object for the comment changes when this occurs.<br>If `childID` is non-zero, it is the UID of the `IAccessible` object for a form field, such as a checkbox or radio button, whose state has changed. |
| `EVENT_OBJECT_VALUECHANGE` | If the `idObject` parameter is `CHILDID_SELF`, a new document or page has been opened or the current content has changed. The client should update its cached value of the document or page.<br>If the `childID` parameter is not `CHILDID_SELF`, it identifies the content on the page to which the user has turned his or her attention. For instance, if a page has scrolled or Acrobat has followed a link to a new page, it identifies the first visible content on the page. The client may wish to update its internal state about where it is reading the document. |

### 1.3.2 Retrieving a PDF DOM object for an event

To retrieve a DOM object, you can do one of the following actions:

- Call the MSAA library function `AccessibleObjectFromEvent` to get an `IAccessible` object (as described above). Then call that `IAccessible` object's `get_PDDomNode` method to get the corresponding DOM object. For more information, see IGetPDDomNode interface.
- Call the MSAA library function `AccessibleObjectFromWindow` on the window containing the document and pass `OBJID_NATIVEOM` as the second parameter. This returns the DOM object for the root of the document.

CHAPTER

TWO

# READING PDF FILES THROUGH MSAA

Microsoft Active Accessibility defines the `IAccessible` interface to applications. This interface consists of a set of methods and properties that are defined in the MSAA documentation.

Acrobat implements and exports a set of `IAccessible` objects of different types to represent a document, its pages, and other elements of the document hierarchy.

An MSAA client can retrieve an `IAccessible` object for a user interface element in the following four ways:

- Set a `WinEvent` hook, receive a notification, and call `AccessibleObjectFromEvent` to retrieve an `IAccessible` interface pointer for the user interface element that generated the event. See Handling event notifications for details.
- Call `AccessibleObjectFromWindow` and pass the user interface element's window handle. Each open document in Acrobat is associated with its own window handle.
- Call `AccessibleObjectFromPoint` and pass a screen location that lies within the user interface element's bounding rectangle.
- Call an `IAccessible` method such as `accNavigate` or `get_accParent` to move to a different `IAccessible` object.

## 2.1 Acrobat implementation of IAccessible objects

Each type of `IAccessible` object has a different implementation of the standard methods:

- Links, tables, and form fields are explicitly identified through MSAA.
- Headers, paragraphs, and other elements of document structure are only represented implicitly.

---

**Note:** These elements are explicit in the DOM interface; see Reading PDF Files Through the DOM Interface.

---

For each document, Acrobat builds a tree of `IAccessible` objects representing the document and its internal structure. Because there is just one window handle associated with the document, Acrobat posts all event notifications to that window. In each notification, a `childID` identifies an `IAccessible` object for an element in the document. For example, when the user tabs to the next link, the `EVENT_OBJECT_FOCUS` notification includes a `childID` that is the UID of the link object. See Handling event notifications.

The following interfaces are exported from the `IAccessible` object by Acrobat:

## 2.2 IGetPDDomNode interface

This interface exports one function, `get_PDDomNode` , which returns a DOM object. The methods described in Reading PDF Files Through the DOM Interface" can then be used on this object.

### 2.2.1 get_PDDomNode

Returns a DOM object. For more information, see Reading PDF Files Through the DOM Interface.

`varID` is the same as for the other MSAA methods (see Descriptive properties and methods)

```
HRESULT get_PDDomNode(
VARIANT varID,
IPDDomNode **ppDispDoc);
```

## 2.3 ISelectText interface

In Acrobat 7.0, the `ISelectText` interface is an interface exported by the `IAccessible` objects. It exports one function, `selectText` , that sets the text selection, but specifies the end location via `IAccessible` objects instead of DOM nodes. The `ISelectText` interface is available from the root `IAccessible` object.

### 2.3.1 selectText

Sets the text selection. `startAccID` and `endAccID` are the `accID` identifiers for the starting and ending `IAccessible` elements, and `startIndex` and `endIndex` are zero-based indexes into the text of those `IAccessible` objects.

```
LRESULT selectText(
long startAccID,
```

(continues on next page)

 

(continued from previous page)

```
long startIndex,
long endAccID,
long endIndex);
```

## 2.4 Identifying IAccessible objects in a document

You can identify the type of an `IAccessible` object by using the `get_accRole` method to get its Role attribute. However, you must also distinguish individual objects from others of the same type. You can do this by means of a unique identifier (UID) defined by Acrobat.

The `IAccessible` objects defined by Acrobat export a private interface, `IAccID` , defined in the file `IAccID.h` . It contains one function, `get_accID` . Use this UID to determine when two `IAccessible` objects refer to the same element in the document.

When a value-change notification or a focus notification has a non-zero `childID` , the value of `childID` is the UID of one of the objects on the page or document. Use the UID to uniquely identify the object that is the target of the notification.

### 2.4.1 get_accID

Returns an identifier that is unique within the open document or page.

```
HRESULT get_accID(long *id);
```

**Parameters**

| | |
|---|---|
| id | (Filled by the method) Returns the unique identifier of the `IAccessible` object. Must not be `NULL` . |

**Returns**

Always returns `s_ok` .

**Example**

```
IAccID *pID;
 long uid;
 /* query for the IAccID interface */
 RESULT hr = pObj->QueryInterface (IID_IAccID,
                                   reinterpret_cast<void **>(&
↪pID));
 if (!FAILED(hr))
```

(continues on next page)

(continued from previous page)

```
{
        pID->get_accID(&uid);
        pID->Release();
}
```

---

**Note:** If you obtained the `IAccessible` object via a call to `AccessibleObjectFrom` XXX, it is not possible to query directly for this private interface. In that case, you must use this alternate code:

---

```
IServiceProvider *sp = NULL;
 hr = n->QueryInterface(IID_IServiceProvider, (LPVOID*)&sp);
 if (SUCCEEDED(hr) && sp) {
        hr = sp->QueryService(SID_AccID, IID_IAccID, (LPVOID*)&pID);
        sp->Release();
 }
```

## 2.5 IAccessible method summary

This section provides a brief syntax summary of the `IAccessible` interface methods as defined by MSAA. All methods return `HRESULT` . The methods and properties are organized into the following groups:

- Navigation and hierarchy
- Descriptive properties and methods
- Selection and focus
- Spatial mapping

## 2.6 Navigation and hierarchy

This section provides information on the APIs used in navigation and to traverse the hierarchy.

 

### 2.6.1 accNavigate

Traverses to another user interface element within a container and retrieves the object. All visual objects support this method.

```
accNavigate (long navDir, VARIANT varStart, VARIANT* pvarEnd);
```

Properties

| | |
|---|---|
| navDir [in] | The direction to navigate, in spatial order or logical order. These are the spatial navigation constants:<br>NAVDIR_UP NAVDIR_DOWN<br>NAVDIR_RIGHT<br>NAVDIR_LEFT<br>These are the logical navigation constants:<br>NAVDIR_FIRSTCHILD<br>NAVDIR_LASTCHILD<br>NAVDIR_NEXT<br>NAVDIR_PREVIOUS<br>• All `accNavigate` methods in PDF objects support the logical navigation directions. Only a few (PDF Structure Element, PDF ComboBox Form Field, and PDF ListBox Form Field) support the spatial navigation directions. Spatial navigation is only supported where it is explicitly noted. |
| varStart [in] | `CHILDID_SELF` to start navigation at the object itself, a child ID to start at one of the object's child elements. |
| pvarEnd [out, retval] | Returns a structure that contains information about the destination object. See MSAA documentation for details. |

Returns

```
HRESULT
```

### 2.6.2 get_accChild

Retrieves an `IDispatch` interface pointer for the specified child, if one exists. All objects support this property.

```
get_accChild (VARIANT varChildID, IDispatch** ppdispChild);
```

Properties

| | |
|---|---|
| varChildID [in] | The child ID for which to obtain a pointer. This can be a UID or the 1-based index of the child to retrieve. |
| ppdispChild [out, retval] | Returns the address of the child's `IDispatch` interface. |

Returns

```
HRESULT
```

### 2.6.3 get_accChildCount

Retrieves the number of children that belong to this object. All objects support this property.

```
get_accChildCount (long* pcountChildren);
```

Properties

| | |
|---|---|
| pcountChildren [out, retval] | Returns the number of children. The children are accessible objects or child elements. If the object has no children, this value is zero. |

Returns

```
HRESULT
```

### 2.6.4 get_accParent

Retrieves an `IDispatch` interface pointer for the parent of this object. All objects support this property.

```
get_accParent (IDispatch** ppdispParent);
```

Properties

| | |
|---|---|
| ppdispParent<br>[out, retval] | Returns the address of the parent's `IDispatch` interface. |

Returns

```
HRESULT
```

## 2.7 Descriptive properties and methods

This section provides information on the descriptive APIs.

### 2.7.1 accDoDefaultAction

Performs the object's default action. Not all objects have a default action.

```
accDoDefaultAction (VARIANT varID);
```

Properties

| | |
|---|---|
| varID [in] | `CHILDID_SELF` to perform the action for the object itself, a child ID to perform the action for one of the object's child elements. |

Returns

```
HRESULT
```

### 2.7.2 get_accDefaultAction

Retrieves a string that describes the object's default action. Not all objects have a default action.

```
get_accDefaultAction(VARIANT varID, BSTR* pszDefaultAction);
```

Properties

| | |
|---|---|
| varID [in] | CHILDID_SELF to get information for the object itself, a child ID to get information for one of the object's child elements. |
| pszDefaultAction<br>[out, retval] | Returns a localized string that describes the default action for the object, or NULL if this object has no default action. |

Returns

```
HRESULT
```

### 2.7.3 get_accDescription

Retrieves a string that describes the visual appearance of the object. Not all objects have a description.

```
get_accDescription (VARIANT varID, BSTR* pszDescription);
```

Properties

| | |
|---|---|
| varID [in] | CHILDID_SELF to get information for the object itself, a child ID to get information for one of the object's child elements. |
| pszDescription<br>[out, retval] | Returns a localized string that describes the object, or NULL if this object has no description. |

Returns

```
HRESULT
```

### 2.7.4 get_accName

Retrieves the name of the object. All objects have a name.

```
get_accName (VARIANT varID, BSTR* pszName );
```

Properties

| | |
|---|---|
| varID [in] | `CHILDID_SELF` to get information for the object itself, a child ID to get information for one of the object's child elements. |
| pszName<br>[out, retval] | Returns a localized string that contains the name of the object. |

Returns

| |
|---|
| `HRESULT` |

### 2.7.5 get_accRole

Retrieves the role of the object. All objects have a role.

```
get_accRole (VARIANT varID, VARIANT* pvarRole );
```

Properties

| | |
|---|---|
| varID [in] | `CHILDID_SELF` to get information for the object itself, a child ID to get information for one of the object's child elements. |
| pvarRole<br>[out, retval] | Returns a structure that contain an object role constant in its `IVal` member. |

Returns

| |
|---|
| `HRESULT` |

### 2.7.6 get_accState

Retrieves the state of the object. All objects have a state.

```
get_accState (VARIANT varID, VARIANT* pvarState );
```

Properties

| | |
|---|---|
| varID [in] | `CHILDID_SELF` to get information for the object itself, a child ID to get information for one of the object's child elements. |
| pvarRole<br>[out, retval] | Returns a structure that contain an object state constant in its `IVal` member. |

Returns

```
HRESULT
```

### 2.7.7 get_accValue

Retrieves the value of the object. Not all objects have a value.

```
get_accValue (VARIANT varID, BSTR* pszValue );
```

Properties

| | |
|---|---|
| varID [in] | `CHILDID_SELF` to get information for the object itself, a child ID to get information for one of the object's child elements. |
| pszValue<br>[out, retval] | Returns a localized string that contains the current value of the object. |

Returns

```
HRESULT
```

## 2.8 Selection and focus

This section provides information on the selection and focus APIs.

### 2.8.1 accSelect

Modifies the selection or moves the keyboard focus of the object. All objects that support selection or receive the keyboard focus support this method.

```
accSelect (long flagsSelect, VARIANT varID);
```

Properties

| | |
|---|---|
| flagsSelect [in] | Flags that control how the selection or focus operation is performed. A logical OR of these SELFLAG constants:<br>SELFLAG_NONE<br>SELFLAG_TAKEFOCUS<br>SELFLAG_TAKESELECTION<br>SELF-LAG_EXTENDSELECTION<br>SELFLAG_ADDSELECTION<br>SELF-LAG_REMOVESELECTION |
| varID [in] | CHILDID_SELF to select the object itself, a child ID to select one of the object's child elements. |

Returns

```
HRESULT
```

## 2.8.2 get_accFocus

Retrieves the object that has the keyboard focus. All objects that receive the keyboard focus support this property.

```
get_accFocus (VARIANT* pvarID);
```

Properties

| | |
|---|---|
| pvarID [out, retval] | Returns the address of a VARIANT structure that contains information about the object that has the focus. See MSAA documentation for details. |

Returns

```
HRESULT
```

### 2.8.3 get_accSelection

Retrieves the selected children of the object. All objects that support selection support this property.

```
get_accSelection (VARIANT* pvarChildren);
```

Properties

| | |
|---|---|
| pvarChildren [out, retval] | Returns the address of a `VARIANT` structure that contains information about the selected children. See the MSAA documentation for details. |

Returns

```
HRESULT
```

## 2.9 Spatial mapping

### 2.9.1 accLocation

Retrieves the object's current screen location. All visual objects support this method.

```
accLocation (long* pxLeft, long* pyTop, long* pcxWidth, long*␣
↪pcyHeight, VARIANT varID );
```

Properties

| | |
|---|---|
| pxLeft, pxTop [out] | Return the x and y screen coordinates of the upper-left boundary of the object's location. (The origin is the upper left corner of the screen.) |
| pxWidth, pxHeight [in] | Return the object's width and height in pixels. |
| varID [in] | `CHILDID_SELF` to get information for the object itself, a child ID to get information for one of the object's child elements. |

Returns

```
HRESULT
```

### 2.9.2 accHitTest

Retrieves the object at a specific screen location. All visual objects support this method.

```
accHitTest (long, long, VARIANT* pvarID);
```

Properties

| | |
|---|---|
| pxLeft, pxTop [in] | The x and y screen coordinates of the point to test. (The origin is the upper left corner of the screen.) |
| pvarID [out, retval] | Address of a `VARIANT` structure that identifies the object at the specified point. The information returned depends on the location of the specified point in relation to the object whose `accHitTest` method is being called. You can use this method to determine whether the object at that point is a child of the object for which the method is called. For details, see the MSAA documentation.<br>• For PDF objects, hit testing has been implemented in a very basic way; it does not identify the boundaries of the object itself with fine granularity, but reports whether or not the tested location is within the bounding box of an element or subtree. |

Returns

```
HRESULT
```

## 2.10 IAccessible object types for PDF

This section describes the MSAA `IAccessible` object types that are defined to represent PDF documents and their elements. For each object, its methods are listed along with notes on how the implementation is specific to the object type.

**Note:** Methods that are not listed are not implemented for a given object type.

The objects are:

- PDF Document
- PDF Page
- PDF Protected Document
- Empty PDF Document
- PDF Structure Element
- PDF Content Element
- PDF Comment
- PDF Link
- PDF Text Form Field
- PDF Button Form Field
- PDF CheckBox Form Field
- PDF RadioButton Form Field
- PDF ComboBox Form Field
- PDF List Box Form Field
- PDF Digital Signature Form Field
- PDF Caret

The following are some general notes:

- PDF form fields generally correspond closely to standard user interface elements described in the MSAA SDK document. The `IAccessible` objects of form fields attempt to match the behavior described in Appendix A, "Supported User Interface Elements," of the MSAA document. An exception is the PDF combo box, which has a much simpler structure.
- Form fields, links, and comments, as well as the document as a whole, can take keyboard focus. Subparts of the document (sections, paragraphs, and so on) cannot take focus.
- A document's contents may be only partially visible on the screen. The `get_accLocation` method for a given object returns the screen location of the visible part of the object only. You can use this method to determine which portions of the content are visible.

### 2.10.1 PDF Document

Represents the contents of an entire PDF document. The subtree of `IAccessible` objects beneath the PDF Document object reflects the logical structure of the document.

---

**Note:** Content that is not part of the logical structure, such as page headers and footers, is not presented through the MSAA interface.

---

| Method | Implementation notes |
|---|---|
| accHitTest | Returns the object at a given location if the location is within the document's bounding box. |
| accLocation | Returns the screen coordinates of the visible part of the document. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT` , `NAVDIR_LEFT` ). |
| accSelect | For `SELFLAG_TAKEFOCUS` , the focus is set to the window containing the document and the document is positioned at the beginning. The other `SELFLAG` values are not supported. |
| get_accChild | Returns a child object. |
| get_accChildCount | Returns the number of child objects beneath this one. |
| get_accDescription | The description contains the full path name of the document and the number of pages it contains: "fileName, XXX pages". |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accParent | The parent is `NULL` . |
| get_accRole | The role is `ROLE_SYSTEM_DOCUMENT` . |
| get_accSelection | Returns `NULL` . |
| get_accState | The state is `STATE_SYSTEM_READONLY` . |
| get_accValue | If the root of the structure tree has an `Alt` attribute, the value is the contents of the `Alt` attribute. |

### 2.10.2 PDF Page

Represents the contents of one page of a PDF document. The subtree of `IAccessible` objects beneath the PDF Page node reflects the logical structure of the page.

---

**Note:** Content that is not part of the logical structure, such as page headers and footers, is not presented through the MSAA interface.

---

| Method | Implementation notes |
|---|---|
| accHitTest | Returns the object at the given location if the location is within the page's bounding box. |
| accLocation | Returns the screen coordinates of the visible part of the page. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT` , `NAVDIR_LEFT` ). |
| accSelect | For `SELFLAG_TAKEFOCUS` , the focus is set to the window containing the page and the page is positioned at the top. The other `SELFLAG` values are not supported. |
| get_accChild | Returns a child object. |
| get_accChildCount | Returns the number of child objects beneath this one. |
| get_accDescription | The description contains the full path name of the document and the page number of the page: "fileName, page XXX". |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accParent | The parent is `NULL` . |
| get_accRole | A custom role, `Page` , is defined for this object. |
| get_accSelection | Returns `NULL` . |
| get_accState | The state is `STATE_SYSTEM_READONLY` . |
| get_accValue | If the root of the structure tree has an `Alt` attribute, the value is the contents of the `Alt` attribute |

### 2.10.3 PDF Protected Document

Represents a protected document. When the permissions associated with a document disable accessibility, the contents are not exported through the MSAA interface. The `IAccessible` object for such a document informs the client that the document is protected.

| Method | Implementation notes |
|---|---|
| accHitTest | Returns `NULL`. |
| accLocation | The screen coordinates of the visible part of the document. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP`, `NAVDIR_DOWN`, `NAVDIR_RIGHT`, `NAVDIR_LEFT`). |
| accSelect | Returns `NULL`. |
| get_accChildCount | The child count is 0. |
| get_accFocus | Returns `NULL`. |
| get_accName | The name is "Alert: Protection Failure". |
| get_accParent | The parent is `NULL`. |
| get_accRole | The role is `ROLE_SYSTEM_TEXT`. |
| get_accSelection | Returns `NULL`. |
| get_accState | The state is `STATE_SYSTEM_ALERT_MEDIUM + STATE_SYSTEM_UNAVAILABLE + STATE_SYSTEM_READONLY`. |
| get_accValue | The value is "This document's security settings prevent access." |

### 2.10.4 Empty PDF Document

Represents an empty or apparently empty document. A PDF file may have no contents to export through MSAA if, for instance, the file is a scanned image that has not been run through an optical character recognition (OCR) tool. The `IAccessible` object for empty documents and pages informs the client that there may be a problem, even if the document or page is genuinely empty.

| Method | Implementation notes |
|---|---|
| accHitTest | Returns `NULL`. |
| accLocation | Returns the screen coordinates of the visible part of the document. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP`, `NAVDIR_DOWN`, `NAVDIR_RIGHT`, `NAVDIR_LEFT`). |
| accSelect | Returns `NULL`. |
| get_accChildCount | The child count is 0. |
| get_accFocus | Returns `NULL`. |
| get_accName | The name is "Alert: Empty document". |
| get_accParent | The parent is `NULL`. |
| get_accRole | The role is `ROLE_SYSTEM_TEXT`. |
| get_accSelection | Returns `NULL`. |
| get_accState | The state is `STATE_SYSTEM_READONLY`. |
| get_accValue | The value is "This document appears to be empty. It may be a scanned image that needs OCR or it may have malformed structure." |

### 2.10.5 PDF Structure Element

Represents a subtree of the logical structure tree for the document. It might correspond to a paragraph, a heading, a chapter, a span of text within a word, or a figure.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | If the element has state `STATE_SYSTEM_LINKED`, performs the action associated with the link. |
| accHitTest | Returns this object or any child at the given location if the location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the subtree. |
| accNavigate | Only spatial navigation (`NAVDIR_UP`, `NAVDIR_DOWN`, `NAVDIR_RIGHT`, `NAVDIR_LEFT`) is supported for table elements (`ROLE_SYSTEM_CELL`, `ROLE_SYSTEM_ROW`, `ROLE_SYSTEM_ROWHEADER`, `ROW_SYSTEM_COLUMNHEADER`). |
| accSelect | For `SELFLAG_TAKEFOCUS`, sets focus to the document window and positions the document to the beginning of the structure element content. The other `SELFLAG` values are not supported. |
| get_accChild | Returns a child object. |
| get_accChildCount | Returns the number of child objects beneath this one.<br>If the node has an `Alt` or `ActualText` attribute, the child count is always zero. |
| get_accDefaultAction | If the element has state `STATE_SYSTEM_LINKED`, returns a text description of the action associated with the link (such as "go to page 5" or "play movie"). |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accParent | The parent is either another structure element or the document structure root. |
| get_accRole | The role is one of:<br>ROLE_SYSTEM_GROUPING<br>ROLE_SYSTEM_TABLE<br>ROLE_SYSTEM_CELL<br>ROLE_SYSTEM_ROW<br>ROLE_SYSTEM_ROWHEADER<br>ROW_SYSTEM_COLUMNHEADER |
| get_accSelection | Returns `NULL`. |
| get_accState | The state is a logical OR of one or more of the following:<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_LINKED |

### 2.10.6 PDF Content Element

Corresponds to a leaf node of the logical structure tree for the document. It corresponds to marking commands in the page content stream.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | If the element has state `STATE_SYSTEM_LINKED`, performs the action associated with the link. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the element. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP`, `NAVDIR_DOWN`, `NAVDIR_RIGHT`, `NAVDIR_LEFT`). |
| accSelect | For `SELFLAG_TAKEFOCUS`, sets focus to the document window and positions the document to the beginning of the content. The other `SELFLAG` values are not supported. |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | If the element has state `STATE_SYSTEM_LINKED`, describes the action associated with the link. |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accParent | The parent is either a structure element or the document structure root. |
| get_accRole | The role is one of:<br>ROLE_SYSTEM_TEXT<br>ROLE_SYSTEM_GRAPHIC<br>ROLE_SYSTEM_CLIENT |
| get_accSelection | Returns `NULL`. |
| get_accState | The state is a logical OR of one or more of the following:<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_LINKED<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED<br>• `STATE_SYSTEM_READONLY` is always set.<br>• If the element is part of a link (that is, if it has an ancestor of role `ROLE_SYSTEM_LINK`) then both `STATE_SYSTEM_LINKED` and `STATE_SYSTEM_FOCUSABLE` are set, and `STATE_SYSTEM_FOCUSED` can also be set. |
| get_accValue | If this node has an `Alt` or `ActualText` attribute, the value is the content of that attribute. Otherwise, the value is all of the text |

### 2.10.7 PDF Comment

Corresponds to a comment, such as a text note or highlight comment, attached to the document.

---

**Note:** PDF comments cover a range of objects, many of which do not map into the standard MSAA roles. The `IAccessible` object captures the most important properties of comments.

---

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | The default action depends on the type of comment. It can, for example, open or close a popup. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT` , `NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS` (that is, selecting the comment gives it the keyboard focus). |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | Describes the default action, which depends on the type of comment. |
| get_accDescription | For file attachment and sound comments, a description of the icon for the comment. |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | • The name indicates the type of comment; for example, Text Comment or Underline Comment.<br>• If the comment is open and has a title, the name also contains the title of the comment.<br>• If the comment is a Free Text comment or modifies a span of text (such as an Underline or Strikeout Comment), the name also contains the text. |
| get_accParent | The parent is either a structure element or the document structure root. |
| get_accRole | The role is one of:<br>ROLE_SYSTEM_TEXT<br>ROLE_SYSTEM_WHITESPACE<br>ROLE_SYSTEM_PUSHBUTTON |
| get_accSelection | Returns `NULL` . |
| get_accState | The state is a logical OR of one or more of the following:<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_LINKED<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_EXPANDED |

### 2.10.8 PDF Link

Corresponds to a link in the document.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | Performs the link's action. |
| accHitTest | Returns this object or any child at the given location if the location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT` , `NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS` |
| get_accChild | Returns a child object. |
| get_accChildCount | Returns the number of children. If the node has an `Alt` or `ActualText` attribute, the child count is always zero. |
| get_accDefaultAction | Describes the action defined for this link. |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | If there is an `Alt` or `ActualText` attribute associated with this link, the name is the associated `Alt` text or `ActualText` . Otherwise, the name is the value of the first content child. |
| get_accParent | The parent is either a structure element or the document structure root. |
| get_accRole | The role is `ROLE_SYSTEM_LINK` . |
| get_accSelection | Returns `NULL` . |
| get_accState | The state is a logical OR of the following:<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_LINKED<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED |
| get_accValue | The value is a unique identifier for each link. |

### 2.10.9 PDF Text Form Field

Corresponds to a text form field in the document.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | Sets focus to the text field for editing. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT` , `NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS` (that is, selecting the field gives it the keyboard focus). |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | The default action is "DoubleClick", which sets the keyboard focus to this field. |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | The user name (short description) of the form field. |
| get_accParent | Returns the parent object. |
| get_accRole | The role is `ROLE_SYSTEM_TEXT` . |
| get_accState | The state of the text field is a logical OR of one of more of:<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_UNAVAILABLE<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_SELECTABLE<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED<br>• STATE_SYSTEM_PROTECTED |
| get_accValue | The value is the text in the text field. |

### 2.10.10 PDF Button Form Field

Corresponds to a button form field in the document.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | Presses the button. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT`, `NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS` (that is, selecting the field gives it the keyboard focus). |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | The default action is "Press". |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | The user name of the form field (short description). |
| get_accParent | Returns the parent object. |
| get_accRole | The role is `ROLE_SYSTEM_PUSHBUTTON`. |
| get_accState | The state of the button is a logical OR of one or more of:<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_UNAVAILABLE<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED |

### 2.10.11 PDF CheckBox Form Field

Corresponds to a checkbox form field in the document.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | Checks or unchecks the box. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT` , `NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS` (that is, selecting the field gives it the keyboard focus). |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | • If the check box has been selected, the default action is "UnCheck".<br>• If the check box has not been selected, the default action is "Check". |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | The user name (short description) of the form field. |
| get_accParent | Returns the parent object. |
| get_accRole | The role is `ROLE_SYSTEM_CHECKBUTTON`. |
| get_accState | The state of the check box is a logical OR of one or more of:<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_UNAVAILABLE<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED<br>• STATE_SYSTEM_CHECKED |

### 2.10.12 PDF RadioButton Form Field

Corresponds to a radio button form field in the document.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | Clicks the radio button. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT`,`NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS` (that is, selecting the field gives it the keyboard focus). |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | The default action is "Check". |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | The user name (short description) of the form field. |
| get_accParent | Returns the parent object. |
| get_accRole | The role is `ROLE_SYSTEM_RADIOBUTTON`. |
| get_accState | The state of the radio button is a logical OR of one or more of:<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_UNAVAILABLE<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED<br>• STATE_SYSTEM_CHECKED |

### 2.10.13 PDF ComboBox Form Field

Corresponds to a combo box form field in the document. It can represent either the combo box itself, or a list item in a combo box.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | • The combo box does not have a default action.<br>• For a list item, the default action is "DoubleClick", which selects the list item. |
| accHitTest | • For a combo box, returns this object or any child at the given location if the location is within the bounding box of this object.<br>• For a list item, returns this object if the given location is within the bounding box of this object. |
| accLocation | • For a combo box, returns the screen coordinates of the visible part of the object.<br>• For a list item, the location is always reported as 0,0,0,0. |
| accNavigate | • Spatial directions `NAVDIR_UP` and `NAVDIR_DOWN` are available for list items. |
| accSelect | • The combo box supports `SELFLAG_TAKEFOCUS` (that is, selecting the field gives it the keyboard focus).<br>• For a list item, sets the combo box to the list item value. |
| get_accChild | • For a combo box, gets the child items.<br>• A list item has no children. |
| get_accChildCount | • For a combo box, the child count is the number of items in the list.<br>• For a list item, the child count is 0. |
| get_accDefaultAction | • The combobox does not have a default action.<br>• For a list item, the default action is "DoubleClick", which selects the list item. |
| get_accFocus | • Returns the object that has the keyboard focus if it is this object or its child. |

### 2.10.14 PDF List Box Form Field

Corresponds to a list box form field in the document. It can represent either the list box itself or a list item in a list box.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | • The list box does not have a default action.<br>• For a list item, the default action is "Double Click," which selects the item. |
| accHitTest | • For a list box, returns this object or any child at the given location if the location is within the bounding box of this object.<br>• For a list item, returns this object if the given location is within the bounding box of this object. |
| accLocation | • For a list box, returns the screen coordinates of the visible part of the object.<br>• For a list item, the location is always reported as 0,0,0,0. |
| accNavigate | • Spatial directions `NAVDIR_UP` and `NAVDIR_DOWN` are available for list items. |
| accSelect | • The list box supports `SELFLAG_TAKEFOCUS` (that is, selecting the field gives it the keyboard focus).<br>• For a list item, sets the list box selection to the list item value. |
| get_accChild | • For a list box, gets the child items.<br>• A list item has no children. |
| get_accChildCount | • For a list box, the child count is the number of items in the list box.<br>• For a list item, the child count is 0. |
| get_accDefaultAction | • The list box does not have a default action.<br>• For a list item, the default action is "Double Click," which selects the item. |
| get_accFocus | • Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | |

### 2.10.15 PDF Digital Signature Form Field

Corresponds to a digital signature form field in the document.

| Method | Implementation notes |
|---|---|
| accDoDefaultAction | Signs the document if the signature field is unsigned and has either been opened with Acrobat or the document has permissions that allow signing. If the document is signed, the default action brings up a dialog box containing the signature information. |
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the visible part of the object. |
| accNavigate | Does not support spatial navigation (`NAVDIR_UP` , `NAVDIR_DOWN` , `NAVDIR_RIGHT`,`NAVDIR_LEFT` ). |
| accSelect | Supports `SELFLAG_TAKEFOCUS`. |
| get_accChildCount | The child count is 0. |
| get_accDefaultAction | Returns `NULL`. |
| get_accFocus | Returns the object that has the keyboard focus if it is this object or its child. |
| get_accName | The user name (short description) of the form field. |
| get_accParent | Returns the parent object. |
| get_accRole | The Digital Signature form field does not map to any of the existing roles, and a custom role, `Signature`, has been defined for it. |
| get_accState | The `State` attribute of the digital signature is a logical OR of one of more of these values:<br>• STATE_SYSTEM_INVISIBLE<br>• STATE_SYSTEM_UNAVAILABLE<br>• STATE_SYSTEM_READONLY<br>• STATE_SYSTEM_FOCUSABLE<br>• STATE_SYSTEM_FOCUSED<br>• STATE_SYSTEM_CHECKED<br>• STATE_SYSTEM_TRAVERSED<br>• If `STATE_SYSTEM_CHECKED` is set, but not `STATE_SYSTEM_TRAVERSED` , the signature is unverified.<br>• If `STATE_SYSTEM_TRAVERSED` is set, but not `STATE_SYSTEM_CHECKED` , the signature is invalid.<br>• If both `STATE_SYSTEM_CHECKED` and `STATE_SYSTEM_TRAVERSED` are set, the signature is valid. |
| get_accValue | The `Value` attribute is the name and date of the signature, if that information is present. |

### 2.10.16 PDF Caret

Represents a caret (text cursor). If a document contains the system caret because focus is within an editable text field or an editable ComboBox field, clients can obtain an `IAccessible` object for the caret to determine where it is located.

| Method | Implementation notes |
|---|---|
| accHitTest | Returns this object if the given location is within the bounding box of this object. |
| accLocation | Returns the screen coordinates of the caret, both when the caret is in a form field and when it is in the document. |
| get_accChildCount | The child count is 0. |
| get_accDescription | The description is a string containing the index of the character in the field that follows the caret.<br>If the caret is at the beginning of the field, the description string is "0". If the caret follows the first character, the description string is "1". |
| get_accParent | The parent is the field containing the caret. However, the caret `IAccessible` object is not listed among the children of that field's `IAccessible` object. |
| get_accRole | The role is `ROLE_SYSTEM_CARET`. |
| get_accState | The state is 0. |
| get_accValue | The value is the current value of the Text field or ComboBox form field containing the caret. |

CHAPTER

THREE

# READING PDF FILES THROUGH THE DOM INTERFACE

Acrobat 6.0 and later defines a document object model (DOM) that provides more complete access to the document structure than the MSAA interface. The Accessibility plug-in defines and exports five COM interfaces in `AcrobatAccess.lib` that expose Acrobat's document hierarchy:

- `IPDDomNode` defines methods that apply to all elements of the document hierarchy.
- `IPDDomDocument` interface is exported by the root object for the page or document.
- `IPDDomNodeExt` interface is exported by every object that exports `IPDDomNode`.
- `IPDDomElement` defines additional methods that apply only to structure elements.
- `IPDDomWord` defines additional methods that apply only to individual words in the document.
- `IPDDomGroupInfo` defines an additional method that applies to radio buttons, list boxes, and combo boxes.

Clients of these interfaces must include the files `AcrobatAccess.h`, `AcrobatAccess_i.c` and `IPDDom.h`.

## 3.1 IPDDomNode data types

This section describes the data types for the PDF DOM hierarchy.

### 3.1.1 CPDDomNodeType

Defines the type of a node in the PDF DOM hierarchy returned by `GetType`.

```
typedef enum {
   CPDDomNode_Document = 1,
   CPDDomNode_Page = 2,
   CPDDomNode_StructElement = 3,
```

(continues on next page)

 

(continued from previous page)

```
   CPDDomNode_Text = 4,
   CPDDomNode_Word = 5,
   CPDDomNode_Char = 6,
   CPDDomNode_Graphic = 7,
   CPDDomNode_Link = 8,
   CPDDomNode_PushButtonField = 9,
   CPDDomNode_TextEditField =10,
   CPDDomNode_StaticTextField =11,
   CPDDomNode_ListboxField =12,
   CPDDomNode_ComboboxField =13,
   CPDDomNode_CheckboxField =14,
   CPDDomNode_RadioButtonField =15,
   PDDomNode_SignatureField =16,
   CPDDomNode_OtherField =17,
   CPDDomNode_Comment =18,
   CPDDomNode_TextComment =19,
   CPDDomNode_Other =20,
   CPDDomNode_LineSeg =21,
   CPDDomNode_WordSeg =22
} CPDDomNodeType;
```

### 3.1.2 PDDom_FontStyle

Constants for font styles returned by `GetFontInfo`.

```
typedef enum {
   PDDOM_FONTATTR_ITALIC = 0x1,
   PDDOM_FONTATTR_SMALLCAP = 0x2,
   PDDOM_FONTATTR_ALLCAP = 0x4,
   PDDOM_FONTATTR_SCRIPT = 0x8,
   PDDOM_FONTATTR_BOLD = 0x10,
   PDDOM_FONTATTR_LIGHT = 0x20
} PDDOM_FontStyle;
```

### 3.1.3 FontInfoState

Constants for font status returned by `GetFontInfo`.

```
typedef enum {
   FontInfo_Unchecked =1,
   FontInfo_NoInfo =2,
   FontInfo_MixedInfo =3,
```

(continues on next page)

(continued from previous page)

```
   FontInfo_Valid =4
} FontInfoState;
```

### 3.1.4 DocState

Constants for document status returned by `GetDocInfo` in the IPDDomDocument interface.

```
enum DocState {
   DocState_OK =0,
   DocState_Protected =1,
   DocState_Empty =2,
   DocState_Unavailable =3
};
```

### 3.1.5 NodeRelationship

Constants returned by `Relationship` in the IPDDomNodeExt interface.

```
enum NodeRelationship {
   NodeRelationship_Descendant =0,
   NodeRelationship_Ancestor =1,
   NodeRelationship_Before =2,
   NodeRelationship_After =3
   NodeRelationship_Equal =4,
   NodeRelationship_None =5
};
```

## 3.2 IPDDomNode methods

`IPDDomNode` defines methods that apply to all elements of the document hierarchy.

 

### 3.2.1 Words and lines in text

An `IPDDomNode` that represents a text node has the role `CPDDomNode_Text`. By default, the children of text nodes are word nodes. To get the word children of a text node, call the `IPDomNode` method `GetChild`. An `IPDDomNode` that represents a word has the role `CPDDomNode_Word`.

---

Note: When a word is hyphenated and thus appears on two lines, each segment of the word is returned as a child that has the role `CPDDom_WordSeg`.

---

Text can also be thought of as having lines as children. To get the line children of a text node, call the `IPDomNode` method `GetTextInLines`. This method returns a new object for the text node. Subsequently, calling `getChild` on this object returns lines as children. An `IPDDomNode` that represents a line has the role `CPDDomNode_LineSeg`. The children of that line node will be the words in that line.

### 3.2.2 GetParent

`ppDispParent` returns the `IPDDomNode` for the parent of this element if there ís a parent element in the DOM hierarchy, or `NULL` if this element is the root element of the hierarchy.

```
LRESULT GetParent (IDispatch **ppDispParent)
```

### 3.2.3 GetType

`nodeType` returns the `CPDDomNodeType` of this element.

```
LRESULT GetType (long *nodeType)
```

### 3.2.4 GetChild

`ppDispChild` returns the `IPDDomNode` for the child of this element at position `index`, or `NULL` if there is no child at position `index`.

For a text node, this returns child words; see Words and lines in text.

```
LRESULT GetChild (ASInt32 index,  IDispatch **ppDispChild)
```

### 3.2.5 GetChildCount

`pCountChildren` returns the number of children of this element.

```
LRESULT GetChildCount (long *pCountChildren)
```

### 3.2.6 GetName

`pszName` returns the name of this element.

- For individual words, this is `NULL`.
- For form fields, it is the short description associated with the field.
- For comments, it is a combination of the comment type and subject (if any).

```
LRESULT GetName (BSTR *pszName)
```

### 3.2.7 GetValue

`pszValue` returns the value of this element.

- For individual words, this is the word itself.
- For form fields, it is the current text content of the field.
- For links, it is a description of the associated action.
- For comments, it is the contents.
- For a signature field, it is the name of the signer and the date signed.

```
LRESULT GetValue (BSTR *pszValue)
```

### 3.2.8 IsSame

If `pNode` refers to the same node as this element, `isSame` returns `true`.

```
LRESULT IsSame (IPDDomNode *pNode,  BOOL *isSame)
```

### 3.2.9 GetTextContent

`pszText` returns the value of all text in the document subtree rooted at this element. Alternate text, actual text, and expansion attributes are included and may override text within the document.

```
LRESULT GetTextContent (BSTR *pszText)
```

### 3.2.10 GetFontInfo

These values describe the font characteristics for the text content of this element.

- `fontStatus` returns a value of type `FontInfoState`.
  - If value is `FontInfo_NoInfo`, the text is not rendered, so it has no font characteristics. For example, alternate text has no font characteristics.
  - If value is `FontInfo_Valid`, the rest of the values describe the font characteristics for all of the text in the subtree. That is, each word of the text either has these characteristics or has no font characteristics.
  - If value is `FontInfo_MixedInfo`, different words of the text have different font characteristics, and the document subtree must be examined more closely to determine which text has which font characteristics.
- `pszName` returns the name of the font.
- `fontSize` returns the point size.
- `fontAttr` returns the set of `PDDom_FontStyle` values.

`red, green, blue` return the RGB components of the color of the text. Each component is a value between 0 and 1.

```
LRESULT GetFontInfo (long* fontStatus,  BSTR* pszName,  float*␣
↪fontSize,  long* fontAttr, float* red,  float* green,  float* blue)
```

### 3.2.11 GetLocation

Returns the screen coordinates of the upper left corner, width, and height of the content of the element. Note that this is not exactly the same as the bounding box. If the element spans multiple pages, this method returns only the location on the first visible page. If none of the element's contents are visible, this method returns an empty location.

```
LRESULT GetLocation (long *pxLeft, ong *pyTop,  long *pcxWidth,  long␣
↪*pcyHeight)
```

 

### 3.2.12 GetFromID

`ppDispNode` returns the `IPDDomNode` for the element in the same document with the matching ID attribute, or `NULL` if there is no such element.

The `id` value is not the same as the UID returned by `IAccID` in the MSAA interface; it is an optional attribute of the PDF file itself, as returned by `GetID` in `IPDDomElement`.

```
LRESULT GetFromID (BSTR id,  IDispatch **ppDispNode)
```

### 3.2.13 GetIAccessible

Returns the MSAA `IAccessible` element corresponding to this element. (Acrobat exports an MSAA interface to the document, as well as a DOM interface.)

Not all DOM elements have corresponding MSAA elements, because the DOM tree breaks the content down into much smaller pieces. If `ppIAccessible` is `NULL`, search for an ancestor with a non-`NULL` value for `GetIAccessible` to find the corresponding MSAA interface.

Use the method `get_PDDomNode` to find the `IPDDomNode` corresponding to a PDF document `IAccessible` object.

```
LRESULT GetIAccessible (IDispatch **ppIAccessible)
```

### 3.2.14 ScrollTo

Makes the contents of the node visible. If the contents cover more than one page, only the contents on the first page are made visible. If the entire contents do not fit, the upper left portion is shown.

```
LRESULT ScrollTo()
```

### 3.2.15 GetTextInLines

`ppDispTextLines` returns an `IPDDomNode` whose children (obtained by calling `GetChild`) have the role `CPDDomNode_LineSeg`; see Words and lines in text.

`visibleOnly` controls whether the children include only lines that contain at least some visible text.

If the role the node is not `CPDDomNode_Text`, this method returns `E_FAIL`.

```
LRESULT GetTextInLines (BOOL visibleOnly,  IDispatch** ppDispTextLines)
```

## 3.3 IPDDomNodeExt methods

The `IPDDomNodeExt` interface is exported by every object that exports `IPDDomNode`. For Acrobat 7.0 and later, the following methods are available from all objects.

### 3.3.1 Navigate

Traverses to another user interface element within a container and retrieves the object. `navDir` indicates which type of navigation is desired, and the node in that direction is returned in `next`. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT Navigate(
long navDir,
IPDDomNode* next);
```

### 3.3.2 ScrollToEx

Determines where to scroll when the item is too large to fit in the window. If both parameters are `true`, this method is equivalent to `ScrollTo`. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT ScrollToEx(
BOOL favorLeft,
BOOL favorTop);
```

### 3.3.3 SetFocus

Sets the focus to this node, if it can take focus. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT SetFocus();
```

### 3.3.4 GetState

Returns a set of state flags identical to those returned by `get_accState` for the corresponding `IAccessible` object. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT GetState(
long* state);
```

### 3.3.5 GetIndex

Returns the child index of this node in its parent. The root node returns -1. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT GetIndex(
long* pIndex);
```

### 3.3.6 GetPageNum

Returns the first and last pages on which the node appears. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT GetPageNum(
long* firstPage,
long* lastPage);
```

### 3.3.7 DoDefaultAction

Executes the default action for a node. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT DoDefaultAction();
```

### 3.3.8 Relationship

Returns the relationship of the `node` parameter to this node. The value is of type `NodeRelationship`, defined in IPDDom.h. This method is defined in the `IPDDomNodeExt` interface on any node.

```
HRESULT Relationship(
PDDomNode* node,
long* pRel);
```

## 3.4 IPDDomDocument methods

The root object for the page or document exports the `IPDDomDocument` interface. For Acrobat 7.0 and later, the following methods are available from the root object.

### 3.4.1 SetCaret

Sets the caret to the specified index in the word. If the index is 0, it is placed at the beginning of the word.

```
HRESULT SetCaret(
IPDDomWord* node,
long index);
```

### 3.4.2 GetCaret

Returns the screen location of the caret, the node containing the caret, and the zero-based index of the caret within the node. The node may be a word node or a form field. If there is no active caret, the call returns `S_FALSE` .

```
HRESULT GetCaret(
long* pxLeft,
long* pyTop,
long* pcxWidth,
long* pcyHeight,
IPDDomNode** node,
long* index);
```

### 3.4.3 NextFocusNode

Gets the next or previous focusable `IPDDomNode` . If `forward` is `true` , it gets the next focusable node. Returns `NULL` if there is not another focusable node in the selected direction. Searches only the current DOM tree, which means that in page mode it will only return results within the page tree instead of the entire document.

```
HRESULT NextFocusNode(
BOOL forward,
IPDDomNode* node);
```

### 3.4.4 GetFocusNode

Returns the `IPDDomNode` with focus. The node is set to `NULL` if the focus is on the document (rather than an annotation) or if the focus is not within the document.

```
HRESULT GetFocusNode(
IPDDomNode* node);
```

### 3.4.5 SelectText

Sets the text selection by identifying the start and end locations of the selection.

```
HRESULT SelectText(
IPDDomWord* startNode,
long startIndex,
IPDDomWord* endNode,
long endIndex);
```

### 3.4.6 GetTextSelection

Retrieves the value of the selected text.

```
HRESULT GetTextSelection(
BSTR* selection);
```

### 3.4.7 GetSelectionBounds

**Not implemented** . This procedure always returns `S_FALSE` .

```
HRESULT GetSelectionBounds(
IPDDomWord** start,
long* startIndex,
IPDDomWord** stop,
long* stopIndex);
```

### 3.4.8 GetDocInfo

Returns the full pathname of the file, how many pages it contains, and the range of pages that are at least partially visible. The `status` indicates whether there are issues with this document or page, such as access controls prohibiting access or an apparently empty page or document. If `lang` is not `NULL`, it is the default language used in the document.

---

**Note:** The `GetDocInfo` and `GoToPage` methods use different numbering systems. The page numbers returned as `firstVisiblePage` and `lastVisiblePage` by `GetDocInfo` are based on page 1 as the first page of the document. However, the `GoToPage` method treats page 0 as the first page of the document. Therefore, you must adjust accordingly when passing the value of `pageNum` to `GoToPage`.

---

```
HRESULT GetDocInfo(
BSTR* fileName,
long* nPages,
long* firstVisiblePage,
long* lastVisiblePage,
long* status,
BSTR* lang);
```

### 3.4.9 GoToPage

Positions the document so that the requested page is visible.

---

**Note:** The `GetDocInfo` and `GoToPage` methods use different numbering systems. The page numbers returned as `firstVisiblePage` and `lastVisiblePage` by `GetDocInfo` are based on page 1 as the first page of the document. However, the `GoToPage` method treats page 0 as the first page of the document. Therefore, you must adjust accordingly when passing the value of `pageNum` to `GoToPage`.

---

```
HRESULT GoToPage(
long pageNum);
```

## 3.5 IPDDomElement Methods

`IPDDomElement` defines additional methods that apply only to structure elements.

### 3.5.1 GetTagName

`pszTagName` returns the structural element tag for this element.

```
LRESULT GetTagName (BSTR *pszTagName)
```

### 3.5.2 GetStdName

`pszStdName` returns the standard role for this element. The standard roles are:

```
Document, Part, Art, Sect, Div, BlockQuote, Caption, TOC, TOCI, Index,␣
↪NonStruct, Private, Table, TR, TH, TD, L, LI, Lbl, LBody, P, H, H1,␣
↪H2, H3, H4, H5, H6, Span, Quote, Note, Reference, BibEntry, Code,␣
↪Link, Figure, Formula,Form
```

For details, see the PDF Reference, version 1.6 , section 10.7.3.

```
LRESULT GetStdName (BSTR *pszStdName)
```

### 3.5.3 GetID

`pszId` returns the ID string associated with this element, if it has been defined.

The `id` value is not the same as the UID returned by `IAccID` in the MSAA interface; it is an optional attribute of the PDF file itself. See Table 10.10 of section 10.6 of the PDF Reference, version 1.6 .

```
LRESULT GetID (BSTR *pszId)
```

### 3.5.4 GetAttribute

`pszAttrVal` returns the value of the specified attribute for specified owner for this element. Owner can be `NULL` or an empty string.

If the element does not have the requested attribute, the method returns `S_FALSE` .

The set of owners and attributes is open-ended, but the standard structure attributes for Tagged PDF are defined in section 10.7.4 of the PDF Reference, version 1.6 . See the table below for accessibility attributes.

```
LRESULT GetAttribute (BSTR pszAttr, BSTR pszOwner,  BSTR *pszAttrVal)
```

### Accessibility attributes

Some of the attributes that are useful for assistive technology are listed here. For a complete list, see section 10.8 of the PDF Reference, version 1.6 .

| Attribute | Owner | Value |
|---|---|---|
| Lang | | ISO language code for text within this element. |
| Alt | | Text containing an equivalent replacement for the content of this element. Automatically incorporated into the value or text content of the element or any of its ancestor elements. |
| ActualText | | Text which is an exact replacement for the content of this element, for example, the text of an illuminated character. Automatically incorporated into the value or text content of the element or any of its ancestor elements. |
| E | | The expanded form of the element's content, when it is an abbreviation or acronym. |
| RowSpan | Table | Number of rows spanned by the table cell. |
| ColSpan | Table | Number of columns spanned by the table cell. |
| Headers | Table | Array of IDs of Table Header (TH) cells associated with this table cell (TD or TH). |
| Scope | Table | The scope of this table header cell: `Row` , `Column` , or `Both` . |
| Summary | Table | Text that describes the table's purpose and structure, for use in non-visual rendering such as speech or Braille. |

## 3.6 IPDDomWord methods

`IPDDomWord` defines additional methods that apply only to individual words in the document.

### 3.6.1 LastWordOfLine

If this is the last word in a line on the page, `isLast` returns `true`. Use this function to determine where the line breaks occur in text. Note that line breaks are inserted into the text content for elements.

```
LRESULT LastWordOfLine (BOOL *isLast)
```

## 3.7 IPDDomGroupInfo method

`IPDDomGroupInfo` defines an additional method that applies to radio buttons, list boxes, and combo boxes.

### 3.7.1 GetGroupPosition

`groupSize` returns the number of items in the radio button set, the list, or the combo box drop-down list. `position` returns the 1-based index of the node in that set of values. That is, a value of 1 for `position` indicates the first value in the set.

```
GetGroupPosition (long *groupSize, long *position)
```

 