INTERNATIONAL STANDARD

ISO 21757-1

First edition
2020-12

Document management — ECMAScript for PDF —

Part 1: Use of ISO 32000-2 (PDF 2.0)

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

Reference number
ISO 21757-1:2020(E)

© ISO 2020

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

COPYRIGHT PROTECTED DOCUMENT

© ISO 2020

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

 © ISO 2020 – All rights reserved

Contents

Page

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/TC 171, *Document management applications*, Subcommittee SC 2, *Document file formats, EDMS systems and authenticity of information*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Introduction

This document specifies a set of ECMAScript object types which define the properties and methods that can be used in ECMAScript scripts embedded in PDF documents to automate and interact with the containing PDF document and the PDF objects within such files.

The goal is to enable the implementation of ECMAScript processors within a broad range of PDF Processors to provide interoperable scripting and automation of PDF documents. This functionality includes the following features, among others:

— processing forms within the document;

— batch processing collections of PDF documents;

— developing and maintaining online collaboration schemes;

— communicating with local databases.

Certain properties and methods that may be discoverable through ECMAScript's introspection facilities are not documented here. Undocumented properties and methods should not be used.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

 © ISO 2020 – All rights reserved

Document management — ECMAScript for PDF —

Part 1:
Use of ISO 32000-2 (PDF 2.0)

1 Scope

This document defines a set of ECMAScript object types for automating and interacting with PDF documents and the contents of such documents.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 32000-2, *Document management — Portable Document Format — Part 2: PDF 2.0*

ISO/IEC 22275:2018, *Information technology — Programming languages, their environments, and system software interfaces — ECMAScript® Specification Suite*

ISO/IEC 22537:2006, *Information technology — ECMAScript for XML (E4X) specification*

3 Terms and definitions

For the purposes of this document, the terms and definitions in ISO 32000-2 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
ECMAscript
means of reference to ISO 22275 and ISO 22537

3.2
SHA
Secure Hash Algorithms, means of reference to FIPS 180-4

4 Notation

ECMAScript objects and properties and other predefined names are written in bold font; values, as well as key terms of interest are written in italic font. Some names can also be used as values, depending on the context, and so the styling of the content will be context specific. Methods, functions, and variables are written in fixed-width font.

EXAMPLE 1 The allowed values for the **projectionType** property are *perspective* and *orthographic*.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Token characters used to delimit objects and describe the structure of PDF files, as defined in ISO 32000-2:2020, 7.2.1, may be identified by their ISO/IEC 646 character name written in upper case in bold font followed by a parenthetic two digit hexadecimal character value with the suffix “h”.

EXAMPLE 2 ***CARRIAGE RETURN*** *(0Dh).*

Text string characters, as defined by ISO 32000-2:2020, 7.9.2, may be identified by their ISO/IEC 10646-1 character name written in uppercase in bold font followed by a parenthetic four digit hexadecimal character code value with the prefix “U+”.

EXAMPLE 3 **EN SPACE** (U+2002).

A property labeled as **R** (read-only) is one whose value cannot be set. An object labeled as **R** (read-only) is one whose reference cannot be modified, though the object itself can be set and its properties may be modified. Unless otherwise indicated, all properties and objects labeled with **R/W** have read/write access.

5 Syntax

5.1 General

Some ECMAScript objects are static objects that can be used as is and must be spelled as indicated. For example, the app object represents the ECMAScript application. There is only one such object and it must be spelled app (case-sensitive).

Other objects are dynamic objects that can be assigned to a variable. For example, a Doc object may be obtained and assigned to a variable:

```
var myDoc = app.newDoc();
```

In this example, myDoc can access all methods and properties of the Doc object. For example:

```
myDoc.closeDoc();
```

5.2 Method arguments

Many of the ECMAScript methods accept either a list of arguments, as is customary in ECMAScript, or a single object argument with properties that contain the arguments. For example, these two calls are equivalent:

```
app.alert( "Multimedia", 3);
```

and

```
app.alert({ cMsg: "Multimedia", nIcon: 3});
```

NOTE The ECMAScript methods defined in support of multimedia do not accept these two argument formats interchangeably. Use the exact argument format described for each method.

6 Paths

Several methods take *device-independent paths* as arguments. See ISO 32000-2 for details about the device-independent path format.

7 Safe path

Developers of PDF Processor software implementing ECMAScript support are encouraged to support the concept of a *safe path* for ECMAScript methods that write data to the local hard drive based on a path passed to it by one of its parameters.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

 © ISO 2020 – All rights reserved

A path cannot point to a system critical folder, for example a root, windows or system directory. A path is also subject to other unspecified tests.

For many methods, the file name must have an extension appropriate to the type of data that is to be saved. Some methods may have a no-overwrite restriction. These additional restrictions are noted in the documentation.

Generally, when a path is judged to not be safe, a **`NotAllowedError`** exception is thrown (see 10.15 Error) and the method fails.

8 Privileged context

An application context in which the application has the permissions necessary to do something on behalf of the current user that is normally restricted. Such permission (or privilege) could be granted by executing a method in a specific way (through the console or batch process), by some PDF property, or because the document was signed with a digital signature trusted by the user. For example, trusting a document certifier's certificate for executing ECMAScript creates a privileged context which enables the ECMAScript to run where it otherwise would not.

9 Privileged versus non-privileged context

Some ECMAScript methods, identified with a **`[Security]`** note, have security restrictions. These methods can be executed only in a privileged context, which includes console, batch and application initialization events. All other events (for example, page open and mouse-up events) are considered non-privileged.

The description of each security-restricted method indicates the events during which the method can be executed.

10 ECMAScript API

10.1 General

This section is a complete reference to the PDF extensions to ECMAScript, its objects, methods, and properties. The section is organized alphabetically by object name.

More information regarding the ECMAScript core can be found in ISO/IEC 22537.

10.2 Annotation

10.2.1 General

This object represents a PDF *markup annotation* (See ISO 32000-2:2020, "12.5.6.2 Markup annotations" for more details). Annotations can be created through ECMAScript by using the **`Doc`** object method **`addAnnot`**.

Before an annotation can be accessed, it must be bound to an ECMAScript variable through a **`Doc`** object method such as **`getAnnot`**:

```
var a = this.getAnnot(0, "Important");
```

The script can then manipulate the annotation named "Important" on page 1 (0-based page numbering system) by means of the variable *a*. For example, the following code first stores the type of the annotation represented by *a* in the variable **`thetype`**, then changes the author to "John Q. Public".

```
var thetype = a.type;           // read property
a.author = "John Q. Public";    // write property
```

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Another way of accessing the `Annotation` object is through the `Doc` object `getAnnots` method.

10.2.2 Annotation types

Annotations are of different types, as reflected in the `type` property. Each type is listed in Table 1: Annotation types and their properties, along with all documented properties returned by the `getProps` method.

While the following Annotation types are defined in ISO 32000-2:2020, "Table 171 — Annotation types", they are not *markup annotations* and are not accessed as Annotation objects:

3D	See the `Annot3D` object for more details
Link	See the `Link` object for more details
Movie	Deprecated in ISO 32000-2
Popup	See the `Annotation popupOpen` and `popupRect` properties
PrinterMark	Deprecated in ISO 32000-2
Projection	Not supported in ECMAScript for PDF
RichMedia	See the `AnnotRichMedia` object for more details
Screen	Deprecated in ISO 32000-2 (replaced by the **AnnotRichMedia** annotation)
TrapNet	Deprecated in ISO 32000-2
Watermark	Not supported in ECMAScript for PDF
Widget	See the `Doc` method `getField(cName)` for more details

Table 1 — Annotation types and their properties

Annotation type	Properties
Caret	`author, borderEffectIntensity, borderEffectStyle, caretSymbol, contents, creationDate, delay, hidden, inReplyTo, intent, lock, modDate, name, noView, opacity, page, popupOpen, popupRect, print, readOnly, rect, refType, richContents, rotate, seqNum, strokeColor, style, subject, toggleNoView, type, width`
Circle	`author, borderEffectIntensity, borderEffectStyle, contents, creationDate, dash, delay, fillColor, hidden, inReplyTo, intent, lock, modDate, name, noView, opacity, page, popupOpen, popupRect, print, readOnly, rect, refType, richContents, rotate, seqNum, strokeColor, style, subject, toggleNoView, type, width`
FileAttachment	`attachIcon, author, borderEffectIntensity, borderEffectStyle, contents, creationDate, delay, hidden, inReplyTo, intent, lock, modDate, name, noView, opacity, page, point, print, readOnly, rect, refType, richContents, rotate, seqNum, strokeColor, style, subject, toggleNoView, type, width`
FreeText	`alignment, author, borderEffectIntensity, borderEffectStyle, callout, contents, creationDate, dash, delay, fillColor, hidden, inReplyTo, intent, lineEnding, lock, modDate, name, noView, opacity, page, print, readOnly, rect, refType, richContents, richDefaults, rotate, seqNum, strokeColor, style, subject, textFont, textSize, toggleNoView, type, width`
Highlight	`author, borderEffectIntensity, borderEffectStyle, contents, creationDate, delay, hidden, inReplyTo, intent, lock, modDate, name, noView, opacity, page, popupOpen, popupRect, print, quads, readOnly, rect, refType, richContents, rotate, seqNum, strokeColor, style, subject, toggleNoView, type, width`

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 1 *(continued)*

Annotation type	Properties
Ink	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `dash`, `delay`, `gestures`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Line	`arrowBegin`, `arrowEnd`, `author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `dash`, `delay`, `doCaption`, `fillColor`, `hidden`, `inReplyTo`, `intent`, `leaderExtend`, `leaderLength`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `points`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Polygon	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `dash`, `delay`, `fillColor`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `vertices`, `width`
PolyLine	`arrowBegin`, `arrowEnd`, `author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `dash`, `delay`, `fillColor`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `vertices`, `width`
Redact	In addition to all the usual properties of a markup type annotation, the properties `alignment`, `overlayText` and `repeat` are specific to the `Redact` annotation.
Sound	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `delay`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `point`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `soundIcon`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Square	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `dash`, `delay`, `fillColor`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Squiggly	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `delay`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `quads`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Stamp	`AP`, `author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `delay`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`
StrikeOut	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `delay`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `quads`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Text	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `delay`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `noteIcon`, `opacity`, `page`, `point`, `popupOpen`, `popupRect`, `print`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `state`, `stateModel`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`
Underline	`author`, `borderEffectIntensity`, `borderEffectStyle`, `contents`, `creationDate`, `delay`, `hidden`, `inReplyTo`, `intent`, `lock`, `modDate`, `name`, `noView`, `opacity`, `page`, `popupOpen`, `popupRect`, `print`, `quads`, `readOnly`, `rect`, `refType`, `richContents`, `rotate`, `seqNum`, `strokeColor`, `style`, `subject`, `toggleNoView`, `type`, `width`

10.2.3 Annotation properties

ISO 32000-2 documents all Annotation properties and specifies how they are stored.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Some property values are stored in the PDF document as names and others are stored as strings (see ISO 32000-2 for details).

Table 2: Annotation object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type *Annotation*.

Table 2 — Annotation object properties

Property	Type	Access	Description
alignment	Number	R/W	*(Valid only for* **`FreeText`** *and* **`Redact`** *annotations)* Controls the alignment of the text for a **`FreeText`** annotation. When the annotation is of type **`Redact`**, this property determines the alignment of the **`overlayText`**. Valid values are: ***`0 Left aligned`*** ***`1 Centered`*** ***`2 Right aligned`***

AP	String	R/W	*(Valid only for* **`Stamp`** *annotations)*
			The named appearance of the stamp to be used in displaying a stamp annotation. The names of the standard stamp annotations are:
			`Approved`
			`AsIs`
			`Confidential`
			`Departmental`
			`Draft`
			`Experimental`
			`Expired`
			`Final`
			`ForComment`
			`ForPublicRelease`
			`NotApproved`
			`NotForPublicRelease`
			`Sold`
			`TopSecret`
			Stamps with **`AP`** values other than those listed above are implementation-defined and may not be interoperable.
			NOTE For a list of stamp names currently in use in the document, see the **`Doc object icons`** property.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description

arrowBegin	String	R/W	*(Valid only for* **`Line`** *and* **`PolyLine`** *objects)*
			Determines the line cap style that specifies the shape to be used at the beginning of a line annotation. Standard values are:
			`None (default)`
			`OpenArrow`
			`ClosedArrow`
			`ROpenArrow`
			`RClosedArrow`
			`Butt`
			`Diamond`
			`Circle`
			`Square`
			`Slash`

arrowEnd	String	R/W	*(Valid only for* **`Line`** *and* **`PolyLine`** *objects)* Determines the line cap style that specifies the shape to be used at the end of a line annotation. Standard values are: **`None`** `(default)` **`OpenArrow`** **`ClosedArrow`** **`ROpenArrow`** **`RClosedArrow`** **`Butt`** **`Diamond`** **`Circle`** **`Square`** **`Slash`**

attachIcon	String	R/W	*(Valid only for* **`FileAttachment`** *objects)*
			The name of an icon to be used in displaying the annotation. Standard values are:
			`Paperclip`
			`PushPin (default)`
			`Graph`
			`Tag`

author	String	R/W	Gets or sets the author of the annotation.
borderEffectIntensity	Number	R/W	The intensity of the border effect, if any. This represents how cloudy a cloudy rectangle, polygon, or oval is.
borderEffectStyle	String	R/W	If non-empty, the name of a border effect style. Currently, the only supported border effects are the empty string (nothing) or “C” for cloudy.
callout	Array	R/W	*(Valid only for* **`FreeText`** *objects)* An array of four or six numbers specifying a callout line attached to the free text annotation. See ISO 32000-2 for additional details.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description
caretSymbol	String	R/W	*(Valid only for* **Caret** *objects)* The symbol associated with a Caret annotation, which is a visual symbol that indicates the presence of text edits. Standard values are: "" (nothing) "P" (paragraph symbol) "S" (space symbol).
contents	String	R/W	Accesses the contents of any annotation that has a pop-up window. For sound and file attachment annotations, specifies the text to be displayed as a description. See also **the Doc** object **addAnnot(...)** method.
creationDate	Date	R	The date and time when the annotation was created.
dash	Array	R/W	*(Valid only for* **FreeText**, **Line**, **PolyLine**, **Polygon**, **Circle**, **Square**, *and* **Ink** *objects)* A dash array defining a pattern of dashes and gaps to be used in drawing a dashed border. For example, a value of [3, 2] specifies a border drawn with 3- point dashes alternating with 2-point gaps. To set the dash array, the **style** property must be set to *D*.
delay	Boolean	R/W	If **true**, property changes to the annotation are queued and then executed when **delay** is set back to *false*. (Similar to the **Field** object **delay** property.)
doc	Doc object	R	The **Doc** object of the document in which the annotation resides.
doCaption	Boolean	R/W	*(Valid only for* **Line** *objects)* If **true**, draws the rich contents in the line appearance itself.
fillColor	Color	R/W	*(Valid only for* **Circle**, **Square**, **Line**, **Polygon**, **PolyLine**, *and* **FreeText** *objects)* Sets the background color for circle, square, line, polygon, polyline, and free text annotations. Values are defined by using **transparent**, **gray**, **RGB** or **CMYK** color. See **color arrays** for information on defining color arrays and how values are used with this property.
gestures	Array	R/W	*(Valid only for* **Ink** *objects)* An array of arrays, each representing a stroked path. Each array is a series of alternating x and y coordinates in default user space, specifying points along the path. When drawn, the points are connected by straight lines or curves in an implementation-dependent way. See ISO 32000-2 for more details.
hidden	Boolean	R/W	If **true**, the annotation is not shown and there is no user interaction, display, or printing of the annotation.
inReplyTo	String	R/W	If non-empty, specifies the name value of the annotation that this annotation is in reply to.
intent	String	R/W	This property allows a markup annotation type to behave differently, depending on the intended use of the annotation. Although this property is defined for all annotation types, setting the **intent** property on annotations of types other than **FreeText**, **Polygon**, or **Line** is implementation-defined, and may not be interoperable.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description
leaderExtend	Number	R/W	*(Valid only for* **`Line`** *objects)* Specifies the length of leader line extensions that extend from both endpoints of the line, perpendicular to the line. These lines extend from the line proper 180 degrees from the leader lines. Values less than zero shall be considered to be an error, and the annotation shall be treated as though the value was set to zero. A value of zero (default) represents no leader line extension.
leaderLength	Number	R/W	*(Valid only for* **`Line`** *objects)* Specifies the length of leader lines that extend from both endpoints of the line, perpendicular to the line. The value may be negative to specify an alternate orientation of the leader lines. A value of zero (default) represents no leader line.

lineEnding	String	R/W	*(Valid only for* **`FreeText`** *objects)*
			This property determines how the end of a callout line is stroked. It is relevant only for a free text annotation when the value of intent is **`FreeTextCallout`**. Standard values are:
			`None`
			`OpenArrow`
			`ClosedArrow`
			`ROpenArrow`
			`RClosedArrow`
			`Butt`
			`Diamond Circle Square`
			`Slash`
			The default is ***`None`***. Values other than these have an implementation-defined meaning and may not be interoperable.

lock	Boolean	R/W	If ***`true`***, the annotation is locked, which is similar to *`readOnly`* with the exception that a PDF Processor may provide a means of viewing (and potentially unlocking) the annotation.
modDate	Date	R/W	The date on which the annotation was last modified.
name	String	R/W	The name of an annotation. This value can be used by the **`Doc`** object **`getAnnot`** method to find and access the properties and methods of the annotation.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description

noteIcon	String	R/W	*(Valid only for* **`Text`** *objects)*
			The name of an icon to be used in displaying the annotation. Standard values are:
			`Check`
			`Circle`
			`Comment`
			`Cross`
			`Help`
			`Insert`
			`Key`
			`NewParagraph`
			`Note`
			`Paragraph`
			`RightArrow`
			`RightPointer`
			`Star`
			`UpArrow`
			`UpLeftArrow`
			The default is ***`Note`***. Values other than these have an implementation-defined meaning, and may not be interoperable.

noView	Boolean	R/W	If ***`true`***, the annotation shall be hidden for interactive presentation of the document. However, if the annotation has an appearance, that appearance shall be used for printing.
opacity	Number	R/W	The constant opacity value to be used in painting the annotation. This value applies to all visible elements of the annotation in its closed state (including its background and border. Permissible values are 0,0 - 1,0. A value of 0,5 makes the annotation semi-transparent.
overlayText	String	R/W	*(Valid only for* ***Redact*** *objects)* A text string specifying the overlay text that should be drawn over the redacted region after the affected content has been removed.
page	Integer	R/W	The 0-based page number of the page containing the annotation.
point	Array	R/W	*(Valid only for* **`Text`**, **`Sound`**, *and* **`FileAttachment`** *objects)* An array of two numbers [*x*, *y*] that specifies the upper left-hand corner in default user space of the icon for a text, sound or file attachment annotation. See also the **`noteIcon`** property and the **`Doc`** object **`addAnnot(...)`** method.
points	Array	R/W	*(Valid only for* **`Line`** *objects)* An array of two points, [[x_1, y_1], [x_2, y_2]], specifying the starting and ending coordinates of the line in default user space.
popupOpen	Boolean	R/W	*(Valid for all objects except* **`FreeText`**, **`Sound`**, *and* **`FileAttachment`***)* If ***`true`***, the pop-up text note appears open when the page is displayed.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description
popupRect	Array	R/W	*(Valid for all objects except* **FreeText**, **Sound**, *and* **FileAttachment***)* An array of four numbers $[x_1, y_1, x_2, y_2]$ specifying the lower-left x, lower-left y, upper-right x, and upper-right y coordinates, in default user space, of the rectangle of the pop-up annotation associated with a parent annotation. It defines the location of the pop-up annotation on the page.
print	Boolean	R/W	Indicates whether the annotation should be printed (***true***) or not (***false***).
quads	Array	R/W	*(Valid only for* **Highlight**, **StrikeOut**, **Underline**, **Squiggly**, *and* **Redact** *objects)* An array of 8 x **n** numbers specifying the coordinates of **n** quadrilaterals in default user space. Each quadrilateral encompasses a word or group of contiguous words in the text underlying the annotation. The **quads** for a word can be obtained through calls to the **Doc** object **getPageNthWordQuads** method. See ISO 32000-2:2020, "Table 182 — Additional entries specific to text markup annotations", for more details.
rect	Array	R/W	The **rect** array consists of four numbers $[x_1, y_1, x_2, y_2]$ specifying the lower-left *x*, lower-left *y*, upper-right *x*, and upper-right *y* coordinates, in default user space, of the rectangle defining the location of the annotation on the page. See also the **popupRect** property.
readOnly	Boolean	R/W	If ***true***, the annotation should display but not interact with the user.
refType	String	R/W	The reference type of the annotation. The property distinguishes whether **inReplyTo** indicates a plain threaded discussion relationship or a group relationship. Standard values are "*R*" and "*Group*". See ISO 32000-2 for additional details.
repeat	Boolean	R/W	*(Valid only for* **Redact** *objects)* If ***true***, the text specified by **overlayText** should be repeated to fill the redacted region after the affected content has been removed. The default is ***false***.
richContents	Array	R/W	*(Valid for all objects except* **Sound** *and* **FileAttachment***)* This property gets the text contents and formatting of an annotation. The rich text contents are represented as an array of **Span** objects containing the text contents and formatting of the annotation. See also the **Field** object **richValue** property and the **event** object properties **richValue**, **richChange**, and **richChangeEx**
richDefaults	Span	R/W	*(Valid only for* **FreeText** *objects)* This property defines the default style attributes for a free text annotation. See the description of the **Field** object **defaultStyle** property for additional details.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description
rotate	Integer	R/W	*(Valid only for* **FreeText** *objects)* The number of degrees the annotation is rotated counter-clockwise relative to the page. PDF Processors implementing this standard shall, at a minimum, support **rotate** values of 0, 90, 180, and 270. Values other than 0, 90, 180, and 270 may not be supported by all PDF Processors, and may be rounded to the nearest supported value.
seqNum	Integer	R	A read-only sequence number for the annotation on the page.
soundIcon	String	R/W	*(Valid only for* **Sound** *objects)* The name of an icon to be used to represent the **Sound** annotation. PDF Processors implementing this standard shall, at a minimum, recognize the standard value "***Speaker***". Other values are implementation-defined, and may not be interoperable.
state	String	R/W	*(Valid only for* **Text** *objects)* The state of the text annotation. Valid standard values for this property depend on the ***stateModel*** value for the annotation. For a state model of ***Marked***, standard values are ***Marked*** and ***Unmarked***. For a ***Review*** state model, the standard values are ***Accepted***, ***Rejected***, ***Cancelled***, ***Completed*** and **None**. Values other than these are implementation-defined, and may not be interoperable.
stateModel	String	R/W	*(Valid only for* **Text** *objects)* **Text** annotations may have a specific state associated with them. The state is specified by a separate text annotation that refers to the original annotation by means of its **inReplyTo** entry (see the **inReplyTo** property). PDF Processors implementing this standard shall support standard values for the **stateModel** property of *Marked* and *Review*. See also the **getStateInModel(cStateModel)** method.
strokeColor	Color	R/W	Sets the appearance color of the annotation. Values are defined by using *transparent, gray, RGB* or *CMYK* color. In the case of a free text annotation, ***strokeColor*** sets the border and text colors. See **color arrays** for information on defining color arrays and how values are used with this property.
style	String	R/W	This property gets and sets the border style. PDF Processors implementing this standard shall support values of *S* (solid) and *D* (dashed). Although this property is defined for all annotation types, setting the **style** property for annotation types other than **Line**, **FreeText**, **Circle**, **Square**, **PolyLine**, **Polygon** and **Ink** annotations is implementation-defined, and may not be interoperable.
subject	String	R/W	Text representing a short description of the subject being addressed by the annotation.
textFont	String	R/W	*(Valid only for* **FreeText** *objects)* Identifies the font that is used when laying out text in a **FreeText** annotation. Valid values for this property defined as properties of the **Font** object. See the **Field** object **textFont** property.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 2 *(continued)*

Property	Type	Access	Description
textSize	Number	R/W	*(Valid only for* **`FreeText`** *objects)* The text size (in points) for a **`FreeText`** annotation. Valid text sizes include zero and the range from 4 to 144, inclusive. Zero indicates the largest point size that allows all the text to fit in the annotation's rectangle.
toggleNoView	Boolean	R/W	If ***`true`***, the **`noView`** flag is toggled when the mouse hovers over the annotation or the annotation is selected. If an annotation has both the **`noView`** and **`toggleNoView`** flags set, the annotation should be invisible. However, when the mouse is over it or it is selected, it should be visible.
type	String	R	The type of annotation. The type of an annotation can only be set within the object-literal argument of the **`Doc`** object **`addAnnot`** method. PDF processors implementing this standard shall support at least the following values: ***`Text`*** ***`FreeText`*** ***`Line`*** ***`Square`*** ***`Circle`*** ***`Polygon`*** ***`PolyLine`*** ***`Highlight`*** ***`Underline Squiggly`*** ***`StrikeOut`*** ***`Stamp`*** ***`Caret`*** ***`Ink`*** ***`FileAttachment`*** ***`Sound`*** Values other than these are implementation-defined, and may not be interoperable.
vertices	Array	R/W	*(Valid only for* **`Polygon`** *and* **`PolyLine`** *objects)* An array of coordinate arrays representing the alternating horizontal and vertical coordinates, respectively, of each vertex, in default user space, of a polygon or polyline annotation. See ISO 32000-2 for details.
width	Number	R/W	*(Valid only for* **`Square`**, **`Circle`**, **`Line`**, **`Ink`**, *and* **`FreeText`** *objects)* The border width in points. If this value is ***`0`***, no border is drawn. The default value is ***`1`***.

10.2.4 Annotation methods

10.2.4.1 General

This section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type **`Annotation`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.2.4.2 destroy()

Parameters

None

Return Value

undefined

Description

Destroys the annotation, removing it from the page. The object becomes invalid.

10.2.4.3 getProps()

Parameters

None

Return Value

An object literal of the properties of the annotation. The object literal is just like the one passed to the Doc object **`addAnnot`** method.

Description

Get the collected properties of an annotation. Can be used to copy an annotation.

10.2.4.4 getStateInModel(cStateModel)

Parameters

Parameters	Type	Description
`cStateModel`	String	The state model to determine the state of the annotation.

Return Value

The result is an array of the identifiers for the current state of the annotation:

If the state model was defined to be ***`exclusive`***, there is only a single state (or no states if the state has not been set).

If the state model is *`non-exclusive`*, there may be multiple states (or no entries if the state has not been set and there is no default).

Description

Gets the current state of the annotation in the context of a state model. See also the **`transitionToState(cStateModel, cState)`** method.

10.2.4.5 setProps(cProps)

Parameters

Parameters	Type	Description
`cProps`	object	A generic object that specifies the properties of the Annotation object to be created (such as **`type`**, **`rect`**, and **`page`**). This object is the same as the parameter of the Doc object **`addAnnot`** method.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Return Value

The **Annotation** object

Description

Sets many properties of an annotation simultaneously.

10.2.4.6 transitionToState(cStateModel, cState)

Parameters

Parameters	Type	Description
cStateModel	String	The state model in which to perform the state transition. **cStateModel** must have been previously added by calling the Collab method **addStateModel**.
cState	String	A valid state in the state model to transition to.

Return Value

undefined

Description

Sets the state of the annotation to ***cState*** by performing a state transition. The state transition should be recorded in the audit trail of the annotation.

See also the **getStateInModel(cStateModel)** method.

For the states to work correctly in a multiuser environment, all users must have the same state model definitions. Therefore, it is best to place state model definitions in a folder-level ECMAScript file that can be distributed to all users or installed on all systems.

10.2.5 Annotation examples

10.2.5.1 Use the destroy() method to remove all FreeText annotations on page 0.

```
var annots = this.getAnnots({ nPage:0 });
for (var i = 0; i < annots.length; i++)
   if (annots[i].type == "FreeText") annots[i].destroy();
```

10.2.5.2 Create an annotation and create a copy of that annotation on each page in the document.

```
var annot = this.addAnnot({
   page: 0,
   type: "Text",
   rect: [40, 40, 140, 140]
});

// Make a copy of the properties of annot
var copy_props = annot.getProps();

// Now create a new annot with the same properties on every page
var numpages = this.numPages;
for (var i=0; i < numpages; i++) {
   var copy_annot = this.addAnnot(copy_props);
   // but move it to page i
   copy_annot.page=i;
}
```

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.2.5.3 Print the status of all annotations on all pages of a document.

```
annots = this.getAnnots()
for ( var i= 0; i< annots.length; i++) {
   states = annots[i].getStateInModel("Review");
   if ( states.length > 0 ) {
      for(j = 0; j < states.length; j++)
      {
         var d = util.printd(2, states[j].modDate);
         var s = states[j].state;
         var a = states[j].author;

         console.println(annots[i].type + ": " + a + " "
         + s + " " + d + "on page "
         + (annots[i].page+1) );
      }
   }
}
```

10.2.5.4 Assuming annot is an Annotation object, this example changes the border to dashed.

```
annot.setProps({ style: "D", dash: [3,2] });
```

10.2.5.5 Set several properties of a Line annotation.

```
var annot = this.addAnnot({type: "Line"})
annot.setProps({
   page: 0,
   points: [[10,40],[200,200]],
   strokeColor: color.red,
   author: "A. C. Robat",
   contents: "Check with Jones on this point.",
   popupOpen: true,
   popupRect: [200, 100, 400, 200], // Place rect at tip of the arrow
   arrowBegin: "Diamond",
   arrowEnd: "OpenArrow"
});
```

10.2.5.6 Define a custom set of transition states, then set the state of an annotation.

```
try {
   // Create a document
   var myDoc = app.newDoc();
   // Create an annot
   var myAnnot = myDoc.addAnnot
   ({
      page: 0,
      type: "Text",
      point: [300,400],
      name: "myAnnot",
   });
   // Create the state model
   var myStates = new Object();
   myStates["initial"] = {cUIName: "Haven't reviewed it"};
   myStates["approved"] = {cUIName: "I approve"};
   myStates["rejected"] = {cUIName: "Forget it"};
   myStates["resubmit"] = {cUIName: "Make some changes"};
   Collab.addStateModel({
      cName: "ReviewStates",
      cUIName: "My Review",
      oStates: myStates,
      cDefault: "initial"
   });
} catch(e) { console.println(e); }
// Change the states
myAnnot.transitionToState("ReviewStates", "resubmit");
myAnnot.transitionToState("ReviewStates", "approved");
```

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.3 AnnotRichMedia

10.3.1 General

An **AnnotRichMedia** object represents a rich media annotation. The **AnnotRichMedia** object can be acquired from the **getAnnotRichMedia** and **getAnnotsRichMedia** methods of the **Doc** object.

10.3.2 AnnotRichMedia properties

Table 3: AnnotRichMedia object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **AnnotRichMedia**.

Table 3 — AnnotRichMedia object properties

Property	Type	Access	Description
activated:	Boolean	R/W	A Boolean value that indicates whether the annotation is enabled and interactive (***true***) or just displaying the poster artwork (***false***). Setting this property to true activates the annotation.
context3D	Global object or undefined	R	If activated is ***true*** and subtype is "*3D*", this property returns the global scripting context of the annotation (a global object containing the 3D scene.) If activated is ***false*** or subtype is not "*3D*", this property returns ***undefined***.
name	String	R	The name of the rich media annotation.
page	Integer	R	The 0-based page number of the page containing the annotation.
rec:	Array	R/W	Returns an array of four numbers $[x_1, y_1, x_2, y_2]$ specifying the lower-left *x*, lower-left *y*, upper-right *x* and upper-right *y* coordinates, in default user space, of the rectangle defining the location of the annotation on the page.
subType	String	R	The subtype of the annotation. The property will have a value of "***3D***", "***Video***", or "***Sound***".

10.4 Annot3D

10.4.1 General

An **Annot3D** object represents a PDF 3D annotation. The **Annot3D** object can be acquired from the **Doc** object methods **getAnnot3D** and **getAnnots3D**.

10.4.2 Annot3D properties

Table 4: Annot3D object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **Annot3D**.

Table 4 — Annot3D object properties

Property	Type	Access	Description
activated :	Boolean	R/W	A Boolean value that indicates whether the annotation is displaying the 3D artwork (***true***) or just the posterboard picture (***false***).
context3D	global Object	R	If activated is *true*, this property returns the context of the 3D annotation (a global object containing the 3D scene.). If activated is ***false***, this property returns undefined.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 4 *(continued)*

Property	Type	Access	Description
innerRect	Array	R	An array of four numbers $[x_1, y_1, x_2, y_2]$ specifying the lower-left x, lower-left y, upper-right x and upper-right y coordinates, in the coordinate system of the annotation (lower-left is [0, 0], top right is [width, height]), of the 3D annotation's bounding box, where the 3D artwork is rendered.
name	String	R	The name of the annotation.
page	Integer	R	The 0-based page number of the page containing the annotation.
rect	Array	R/W	Returns an array of four numbers $[x_1, y_1, x_2, y_2]$specifying the lower-left x, lower-left y, upper-right x and upper-right y coordinates, in default user space, of the rectangle defining the location of the annotation on the page.

10.5 app

10.5.1 General

PDF Processors implementing this standard shall implement support for a static ECMAScript object named **app** that represents the PDF Processor application. The **app** object shall provide the properties defined in **app properties** and methods defined in

app methods. The purpose of the **app** object is to provide a standardized set of functions that enable scripts in PDF documents to interact with PDF Processors in a consistent way. The **app** object also provides a variety of utility routines and convenience functions.

10.5.2 app properties

10.5.2.1 General

Table 5: app object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **app**.

Table 5 — app object properties

Property	Type	Access	Description
activeDocs	Array	R	**[Security]** An array containing the Doc object for each active document. If no documents are active, activeDocs returns nothing; that is, it has the same behavior as **d = new Array(***0***)** in core ECMAScript. The array returned by app.activeDocs includes any documents opened by app.openDoc with the bHidden parameter set to true, subject to the security restrictions described below. NOTE Developers of PDF Processor software implementing ECMAScript support are encouraged to support the document security settings, safe-paths and privileged mode considerations
focusRect	Boolean	R/W	Turns the focus rectangle on and off. The focus rectangle is a faint dotted line around buttons, check boxes, radio buttons, and signatures to indicate that the form field has the keyboard focus. A value of ***true*** turns on the focus rectangle.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 5 *(continued)*

Property	Type	Access	Description
formsVersion	Number	R	The version number of the PDF Processor forms support. Check this property to determine whether objects, properties, or methods in newer versions of the software are available if you want to maintain backward compatibility in your scripts.
fs	Object	R	A **FullScreen** object, which can be used to access the fullscreen properties.
language	String	R	The language of the running PDF Processor. Implementers should use one of the following three-letter language identifier strings to ensure the greatest level of interoperability. ***CHS*** (Chinese Simplified) ***CHT*** (Chinese Traditional) ***DAN*** (Danish) ***DEU*** (German) ***ENU*** (English) ***ESP*** (Spanish) ***FRA*** (French) ***ITA*** (Italian) ***KOR*** (Korean) ***JPN*** (Japanese) ***NLD*** (Dutch) ***NOR*** (Norwegian) ***PTB*** (Brazilian Portuguese) ***SUO*** (Finnish) ***SVE*** (Swedish)
openInPlace	Boolean	R/W	Specifies whether cross-document links are opened in the same window or opened in a new window.
platform	String	R	The platform that the script is currently executing on. There are three valid values: ***WIN*** ***MAC*** ***UNIX***
printerNames	Array of Strings	R	A list of available printers. Each of these values is suitable to use in the **printerName** property of the ***PrintParams*** object. If no printers are installed on the system, an empty array is returned.
runtimeHighlight	Boolean	R/W	If ***true***, the background color and hover color for form fields are shown.
runtimeHighlightColor	Color array	R/W	Sets the color for runtime highlighting of form fields. The value of this property is a **color** array. (See **color arrays** for details.) See the **runtimeHighlight** property.
thermometer	Object	R	A **Thermometer** object, which is a combined status window/progress bar that indicates to the user that a lengthy operation is in progress.
viewerType	String	R	A string that indicates which viewer application is running. Values are implementation defined.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 5 *(continued)*

Property	Type	Access	Description
viewerVariation	String	R	Indicates implementation-defined capabilities of the running viewer application.
viewerVersion	Number	R	Indicates the version number of the current viewer application.

10.5.3 app methods

10.5.3.1 alert(cMsg, nIcon, nType, cTitle, oDoc, oCheckbox)

Parameters

Parameter	Type	Description
cMsg	String	A string containing the message to be displayed.
nIcon	Number	(optional) An icon type. Possible values are these : *0* — Error (default) *1* — Warning *2* — Question *3* — Status NOTE In Mac OS, there is no distinction between warnings and questions.
nType	Number	(optional) A button group type. Possible values are these: *0* — Ok (default) *1* — OK, Cancel *2* — Yes, No *3* — Yes, No, Cancel
cTitle	String	(optional) The dialog box title.
oDoc	Doc object	(optional) The **Doc** object that the alert should be associated with.
oCheckbox	Generic object	(optional) If specified, a check box is created in the lower left region of the alert box. **oCheckbox** is a generic ECMAScript object that has three properties. The first two property values are passed to the alert method; the third property returns a **Boolean** value. **cMsg** (optional) A string to display with the check box. If not specified, the default string is "***Do not show this message again***". **bInitialValue** (optional) If ***true***, the initial state of the check box is checked. The default is ***false***. **bAfterValue**When the alert method exits, contains the state of the check box when the dialog box is closed. If ***true***, the check box was checked when the alert box is closed.

Return Value

Number

Description

Displays an alert dialog box. The return value will be a string representing the button that was pressed to dismiss the dialog. Valid values of the return string are:

0 — Ok

1 — Cancel

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

2 — Yes

3 — No

10.5.3.2 beep(nType)

Parameters

Parameter	Type	Description
nType	Number	(optional) The sound type. Values are associated with sounds as follows: *0* — Error *1* — Warning *2* — Question *3* — Status *4* — Default (default)

Return Value

Undefined

Description

Causes the system to play a sound.

10.5.3.3 clearInterval(oInterval)

Parameters

Parameter	Type	Description
oInterval	Interval object	The registered interval to cancel.

Return Value

Undefined

Description

Cancels a previously registered interval initially set by the **`setInterval`** method.

10.5.3.4 clearTimeOut(oTime)

Parameters

`oTime`

Parameter	Type	Description
oTime	Timeout object	The registered time-out interval to cancel.

Return Value

Undefined

Description

Cancels a previously registered time-out interval. Such an interval is initially set by **`setTimeOut`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.5.3.5 execDialog(monitor[, inheritDialog, parentDoc])

Parameters

Parameters	Type	Description
monitor	Object	An **object** literal. It consists of several handlers (see **Dialog box handlers**) and a description property that describes the dialog box elements (see **description** property).
inheritDialog	Object	(optional) A **Dialog** object that should be reused when displaying this dialog box. It is useful when displaying a series of dialog boxes (such as a wizard) to prevent one from disappearing before the new one is displayed. The default is to not reuse a dialog box.
parentDoc	Object	(optional) A **Doc** object to use as the parent for this dialog box. The default parent is the PDF Processor application.

Return Value

String

The returned value is the **ItemID** of the element that caused the dialog box to be dismissed. The return value is "***ok***" or "***cancel***" if the dismissing element is the ok or cancel button.

Description

Presents a modal dialog box to the user. Modal dialog boxes must be closed by the user before the host application can be directly used again.

The **monitor** parameter specifies a dialog descriptor, which is a generic object literal that consists of a set of handler functions for various events and a set of properties that describe the contents of the dialog box.

Dialog box items are identified by an **ItemID**, which is a unique 4-character string. An **ItemID** is necessary only if the element must be referred to elsewhere in the dialog box description (for example, to set or get a value for the element, to add a handler for the element, or to set a tab order including the element).

The call to **execDialog()** returns a string, which is the ItemID of the element that caused the dialog box to be dismissed. The return value is "**ok**" or "**cancel**" if the dismissing element is the ok or cancel button.

10.5.3.5.1 Dialog box handlers:

The dialog box handlers are called when specific dialog box events occur. Each handler is optional and is passed a Dialog object that can be used to query or set values in the dialog box. The supported handlers are listed in Table 6: Dialog box handlers.

NOTE Application developers are encouraged to provide a means for users to distinguish application dialog boxes from those created by ECMAScript processing.

Table 6 — Dialog box handlers

Dialog box handler	Description
initialize	Called when the dialog box is being initialized.
validate	Called when a field is modified to determine if the value is acceptable (by returning true) or unacceptable (by returning false).
commit	Called when the OK button of the dialog box is clicked.
destroy	Called when the dialog box is being destroyed.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 6 *(continued)*

Dialog box handler	Description

ItemID	Called when the dialog box element ItemID is modified. For a text box, it is when the text box loses focus. For other controls, it is when the selection changes.
	If **`ItemID`** is not an ECMAScript identifier, the name must be enclosed in double quotes when the method is defined, as in the following example.
	`"bt:1": function () { .... }`
	If **`ItemID`** is an ECMAScript identifier, the double quotes are optional. For example, the following are both correct.
	`"butn": function () { .... } butn: function () { .... }`

10.5.3.5.2 description property

The **`description`** property is an object literal that contains properties describing the dialog. Its **`elements`** property specifies the elements of the dialog box, and each of the elements in turn can have an **`elements`** property describing subelements.

The dialog properties at the root level of the **`description`** property are listed in Table 7: Description properties.

Table 7 — Description properties

Property	Type	Description
name	String	The title bar of the dialog box, which should be localized.
first_tab	String	An ItemID for the dialog box item that should be first in the tab order. This dialog box item will also be active when the dialog box is created. This property is required for setting up a tabbing order (see **`elements`** property).
width	Number	The width of the dialog box in pixels. If no width is specified, the combined width of the contents is used.
height	Number	The height of the dialog box in pixels. If no height is specified, the combined height of the contents is used.
char_width	Number	The width of the dialog box in characters. If no width is specified, the combined width of the contents is used.
char_height	Number	The height of the dialog box in characters. If no height is specified, the combined height of the contents is used.
align_children	String	The alignment for all descendants. Must be one of the following values: ***`align_left`*** Left aligned ***`align_center`*** Center aligned ***`align_right`*** Right aligned ***`align_top`*** Top aligned ***`align_fil`*** Align to fill the parent's width; may widen objects ***`align_distribute`*** Distribute the contents over the parent's width ***`align_row`*** Distribute the contents over the parent's width with a consistent baseline ***`align_offscreen`*** Align items one on top of another
elements	Array	An array of object literals that describe the dialog box elements contained within this dialog (see **`elements`** property).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.5.3.5.3 elements property

A dialog box ***elements*** property specifies an object literal with the properties listed in Table 8: Element properties.

Table 8 — Element properties

Property	Type	Description
name	String	The displayed name of the dialog box element, which should be localized. NOTE This property is ignored for the "edit_text" type.
item_id	String	An ItemID for this dialog box, which is a unique 4-character string.
type	String	The type of this dialog box element. It must be one of the following strings ***button*** A push button ***check_box*** A check box ***radio*** A radio button ***list_box*** A list box ***hier_list_box*** A hierarchical list box ***static_text*** A static text box. ***edit_text*** An editable text box ***Popup*** A pop-up control ***ok*** An OK button ***ok_cancel*** An OK and Cancel Button ***ok_cancel_other*** An OK, Cancel, and Other button ***view*** A container for a set of controls ***cluster*** A frame for a set of controls ***gap*** A place holder.
next_tab	String	An ItemID for the next dialog box item in the tab order. NOTE Tabbing does not stop at any dialog box item that is not the target of the next_tab (or first_tab) property. Tabbing should form a circular linked list.
width	Number	Specifies the width of the element in pixels. If no width is specified, the combined width of the contents is used.
height	Number	Specifies the height of the element in pixels. If no height is specified, the combined height of the contents is used.
char_width	Number	Specifies the width of the element in characters. If no width is specified, the combined width of the contents is used.
char_height	Number	Specifies the height of the element in characters. If no height is specified, the combined height of the contents is used.
font	String	The font to use for this element. Must be one of the following strings: ***default*** Default font ***dialog*** Dialog box font ***palette*** Palette (small) Font
bold	Boolean	Specify if the font is bold.
italic	Boolean	Specify if the font is italic.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 8 *(continued)*

Property	Type	Description
alignment	String	Sets the alignment for this element. Must be one of the following values: ***align_left*** Left aligned ***align_center*** Center aligned ***align_right*** Right aligned ***align_top*** Top aligned ***align_fill*** Align to fill the parent's width; may widen objects. ***align_distribute*** Distribute the contents over the parent's width. ***align_***`row` Distribute the contents over the parent's width with a consistent baseline. ***align_***`offscreen` Align items one on top of another
align_children	String	Sets the alignment for all descendants. Possible values are the same as for alignment.
elements	Array	An array of object literals that describe the subelements of this dialog box element. Its properties are the same as those described in this table.

10.5.3.5.4 Additional attributes of some dialog box elements

Some of the element types have additional attributes, as listed in Table 9: Additional element attributes.

Table 9 — Additional element attributes

Element type	Property	Type	Description
static_text	multiline	Boolean	If ***true***, this static text element is multiline. NOTE For Mac OS, the height property must be at least 49 to display the up/down buttons, which allow users to read the whole box content.
edit_text	multiline	Boolean	If ***true***, this edit text element is multiline.
	readonly	Boolean	If ***true***, this text element is read only. NOTE This property is ignored when password is set to true.
	password	Boolean	If ***true***, this text element is a password field.
	PopupEdit	Boolean	If ***true***, it is a pop-up edit text element.
	SpinEdit	Boolean	If ***true***, it is a spin edit text element.
radio	group_id	String	The group name to which this radio button belongs.
ok	ok_name	String	The name for the OK button.
ok_cancel	cancel_name	String	The name for the cancel button.
ok_cancel_other	other_name	String	The name for the other button.

10.5.3.6 goForward()

Parameters

None

Return Value

Undefined

Description

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Goes to the next view on the view stack.

10.5.3.7 launchURL(cUrl[, bNewFrame])

Parameters

Parameter	Type	Description
cURL	String	A string that specifies the URL to launch.
bNewFrame	Boolean	(optional) If ***true***, this method launches the URL in a new window of the browser application. The default is ***false***.

Return Value

Undefined

Description

`[Security]`

Launches a URL in a browser window.

Developers of PDF Processor software implementing ECMAScript support shall observe confirmation dialog operational policy to provide security risk warning to users prior to accessing external URLs.

See also the **`app.openDoc()`** method.

10.5.3.8 mailMsg(bUI, cTo[, cCc, cBcc, cSubject, cMsg])

Parameters

Parameter	Type	Description
bUI	Boolean	Indicates whether user interaction is required. If *true*, the remaining parameters are used to seed the compose-new-message window that is displayed to the user. If *false*, the cTo parameter is required and others are optional. NOTE When this method is executed in a non-privileged context, the bUI parameter is not honored and defaults to true. See Privileged versus non-privileged context.
cTo	String	A semicolon-separated list of addressees.
cCc	String	(optional) A semicolon-separated list of CC addressees.
cBcc	String	(optional) A semicolon-separated list of BCC addressees.
cSubject	String	(optional) Subject line text. The length limit is 64 KB.
cMsg	String	(optional) Mail message text. The length limit is 64 KB.

Return Value

Undefined

Description

Sends out an e-mail message with or without user interaction.

See also the **`Doc`** object **`mailDoc`**(bUI[, cTo, cCc, cBcc, cSubject, cMsg]) and **`mailForm`**(bUI[, cTo, cCc, cBcc, cSubject, cMsg]) methods.

On Windows: The client machine shall have its default mail program configured to be MAPI enabled to use this method.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.5.3.9 newDoc([nWidth, nHeight])

Parameters

Parameter	Type	Description
nWidth	Number	(optional) The width (in points) for the new document. The default value is ***612***.
nHeight	Number	(optional) The height (in points) for the new document. The default value is ***792***.

Return Value

Doc

Description

`[Security]`

Creates a new document and returns its Doc object. The optional parameters specify the media box dimensions of the document in points.

NOTE This method can only be executed during a batch or console event. See Privileged versus non-privileged context. See the **`event`** object for a discussion of ECMAScript events.

10.5.3.10 newCollection()

Parameters

None

Return Value

Doc

Description

Creates a new collection-based PDF, a PDF portfolio, with no content.

To work with an existent PDF portfolio, begin with **`Doc.collection`**.

10.5.3.11 openDoc(cPath[, oDoc, cFS, bHidden, bUseConv, cDest])

Parameters

Parameters	Type	Description
cPath	String	A device-independent path to the document to be opened. If oDoc is specified, the path can be relative to it. The target document must be accessible in the default file system. NOTE When cFS is set to "CHTTP", the cPath string should be escaped, using the core ECMAScript global function encodeURI.
oDoc	Doc	(optional) A **`Doc`** object to use as a base to resolve a relative cPath. Must be accessible in the default file system.
cFS	String	(optional) A string that specifies the source file system name. Two values are supported: "" (the empty string, which is the default), representing the default file system, and "***`CHTTP`***". This parameter is relevant only if the web server supports WebDAV.
bHidden	Boolean	(optional) A Boolean value that if ***`true`***, opens the PDF file with its window hidden. The default is ***`false`***.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameters	Type	Description
bUseConv	Boolean	(optional) A Boolean value that is used when cPath references a non-PDF file. If ***true***, the method tries to convert the non-PDF file to a PDF document. The default is ***false***. NOTE bUseConv can only be set to true during a console or batch event. See also Privileged versus non-privileged context.
cDest	String	(optional) The name of the destination within a document. This parameter forces opening at named destination within the PDF document. For details on named destinations and how to create them, see ISO 32000-2.

Return Value

A `Doc` object or `null`

Description

Opens a specified PDF document and returns its `Doc` object. This object can be used by the script to call methods, or to get or set properties in the newly opened document.

An exception is thrown and an invalid `Doc` object is returned when an HTML document is opened using this method. To catch the exception, enclose `app.openDoc` in a `try/catch` construct.

10.5.3.12 popUpMenu([cItem, Array])

Parameters

Parameter	Type	Description
cItem	String	(optional) If the argument is a string, it is listed in the menu as a menu item. The menu item name "-" is reserved to draw a separator line in the menu.
Array	Array	(optional) If the argument is an array of stings, it appears as a submenu where the first element in the array is the parent menu item. This array can contain further submenus.

Return Value

String or null

Description

Creates a pop-up menu at the current mouse position, containing the specified items. Returns a string representing the name of the menu item that was selected, or null if no item was selected.

10.5.3.13 popUpMenuEx(menuItems[, ...])

Parameters

Property	Type	Description
cName	String	The menu item name, which is the string to appear on the menu item. The value of "-" is reserved to draw a separator line in the menu.
bMarked	Boolean	(optional) A Boolean value specifying whether the item is to be marked with a check. The default is ***false*** (not marked).
bEnabled	Boolean	(optional) A Boolean value specifying whether the item is to appear enabled or grayed out. The default is ***true*** (enabled).
cReturn	String	(optional) A string to be returned when the menu item is selected. The default is the value of `cName`.
oSubMenu	MenuItem	(optional) A MenuItem object representing a submenu item or an array of submenu items, each represented by a MenuItem object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Return Value

String

Description

Creates a pop-up menu at the current mouse position. Each of the parameters is a MenuItem object that describes a menu item to be included in the pop-up menu.

Implementers of this method shall return a string equal to the cReturn value of the menu item that was selected, or its cName if cReturn was not specified for that item. The method returns null if no selection was made.

10.5.3.14 response(cQuestion[, cTitle, cDefault, bPassword, cLabel])

Parameters

Property	Type	Description
cQuestion	String	The question to be posed to the user.
cTitle	String	(optional) The title of the dialog box.
cDefault	String	(optional) A default value for the answer to the question. If not specified, no default value is presented.
bPassword	Boolean	(optional) If ***true***, indicates that the user's response should show as asterisks (*) or bullets (•) to mask the response, which might be sensitive information. The default is ***false***.
cLabel	String	(optional) A short string to appear in front of and on the same line as the edit text field.

Return Value

A string containing the user's response. If the user clicks the Cancel button, the response is the null object. **Description**

Displays a dialog box containing a question and an entry field for the user to reply to the question.

Implementations of this standard shall return a string containing the user's response. If the user clicks the Cancel button, the return value shall be the null object.

10.5.3.15 setInterval(cExpr, nMilliseconds)

Parameters

Property	Type	Description
cExpr	String	The ECMAScript script to be executed.
nMilliseconds	Number	The time period in milliseconds.

Return Value

`Interval` object

Description

Specifies an ECMAScript script and a time period. The script is executed every time the period elapses.

The return value of this method must be held in an ECMAScript variable. Otherwise, the interval object is subject to garbage-collection, which would cause the clock to stop.

To terminate the periodic execution, pass the returned interval object to clearInterval.

See also `clearInterval`(oInterval), `setTimeOut`(cExpr, nilliseconds) and `clearTimeOut`(oTime).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.5.3.16 setTimeOut(cExpr, nilliseconds)

Parameters

Property	Type	Description
cExpr	String	The ECMAScript script to be executed.
nMilliseconds	Number	The time period in milliseconds.

Return Value

`timeout` object

Description

Specifies an ECMAScript script and a time period. The script is executed one time only, after the period elapses.

The return value of this method must be held in an ECMAScript variable. Otherwise, the timeout object is subject to garbage-collection, which would cause the clock to stop.

To cancel the timeout event, pass the returned `timeout` object to `clearTimeOut`.

See also `clearTimeOut`(oTime), `setInterval`(cExpr, nMilliseconds), and `clearInterval`(oInterval).

10.6 Bookmark

10.6.1 General

A `Bookmark` object represents a node in the bookmark tree that PDF Processors may present to users. Bookmarks are typically used as a table of contents allowing the user to navigate quickly to topics of interest. The way in which a PDF Processor presents the Bookmarks tree to users is implementation defined.

10.6.2 Bookmark properties

Table 10: Bookmark object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Bookmark`.

Table 10 — Bookmark object properties

Property	Type	Access	Description
children	Array	R/W	An array of `Bookmark` objects that specify the children of this bookmark in the bookmark tree. If there are no children of this bookmark, this property has a value of ***`null`***.
color	Array	implementation-defined	Specifies the color for a bookmark. Values are defined by using gray, RGB or CMYK color. See `color arrays` for information on defining color arrays and how values are used with this property. See also the `style` property.
doc	Object	R	The `Doc` that the bookmark resides in.
name	String	R/W	The text string for the bookmark that the user sees in the navigational panel.
open	Boolean	implementation-defined	Determines whether the bookmark shows its children in the navigation panel (open) or whether the children subtree is collapsed (closed).
parent	Object	R	The parent bookmark of the bookmark or null if the bookmark is the root bookmark. See also the `children` and `bookmarkRoot` properties.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 10 *(continued)*

Property	Type	Access	Description
style	Integer	implementation-defined	Specifies the style for the bookmark's font: *0* normal *1* italic *2* bold *3* bold-italic See also the `color` property.

10.6.3 Bookmark methods

10.6.3.1 General

This section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type Bookmark.

10.6.3.2 createChild(cName, cExpr, nIndex);

Parameters

Property	Type	Description
cName	String	The name of the bookmark that the user sees in the navigation panel.
cExpr	String	(optional) An expression to be evaluated whenever the user clicks the bookmark. It is equivalent to creating a bookmark with a ECMAScript action, as described in ISO 32000-2. The default is no expression.
nIndex	Number	(optional) The 0-based index into the children array of the bookmark at which to create the new child. The default is *0*. The value of nIndex shall not be negative.

Return Value

Undefined

Description

Creates a new child bookmark at the specified location.

10.6.3.3 execute();

Parameters

None

Return Value

Undefined

Description

Executes the action associated with this bookmark. This can have a variety of behaviors. See ISO 32000-2 for a list of common action types. See also `createChild( cName, cExpr, nIndex)`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.6.3.4 insertChild(oBookmark, nIndex);

Parameters

Property	Type	Description
oBookmark	Bookmark	A bookmark object to add as the child of this bookmark.
nIndex	Number	(optional) The 0-based index into the children array of the bookmark at which to insert the new child. The default is *0*. The value of nIndex shall not be negative.

Return Value

Undefined

Description

Inserts the specified bookmark as a child of this bookmark. If the bookmark already exists in the bookmark tree, it is unlinked before inserting it back into the tree. In addition, the insertion is checked for circularities and disallowed if one exists. This prevents users from inserting a bookmark as a child or grandchild of itself.

10.6.3.5 remove();

Parameters

None

Return Value

Undefined

Description

Removes the bookmark and all its children from the bookmark tree. See also the `children` property and the createChild(cName, cExpr, nIndex) and `insertChild( oBookmark, nIndex)` methods.

10.6.3.6 setAction(cScript);

Parameters

Property	Type	Description
cScript	String	Defines the ECMAScript expression that is to be executed whenever the user clicks the bookmark.

Return Value

Undefined

Description

Sets an ECMAScript action for a bookmark.

10.6.4 Bookmark Examples

The following examples demonstrate the use of some of the properties and methods of `Bookmark` objects. These examples are informative.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.6.4.1 Create a bookmark

This example creates a bookmark at the top of the bookmark panel that takes you to the next page in the document.

```
this.bookmarkRoot.createChild("Next Page", "this.pageNum++");
```

10.6.4.2 List all Bookmarks

This example prints a list of all bookmarks in the document.

```
function DumpBookmark(bkm, nLevel){
   var s = "";
   for (var i = 0; i < nLevel; i++)
      s += " ";
   console.println(s + "+-" + bkm.name);
   if (bkm.children != null)
   for (var i = 0; i < bkm.children.length; i++)
      DumpBookmark(bkm.children[i], nLevel + 1);
}
console.clear();
console.show();
console.println("Dumping all bookmarks in the document.");
DumpBookmark(this.bookmarkRoot, 0);
```

10.6.4.3 Search Bookmarks

This example performs a simple search of the bookmarks for a bookmark with a specific name. If successful, the action associated with the bookmark is executed.

```
// Document-level or folder-level JavaScript
function searchBookmarks(bkm, nLevel, bkmName){
   if ( bkm.name == bkmName )
      return bkm;
   if (bkm.children != null){
      for (var i = 0; i < bkm.children.length; i++){
         var bkMark = searchBookmarks(bkm.children[i], nLevel + 1, bkmName);
         if ( bkMark != null )
            break;
      }
      return bkMark;
   }
   return null;
}
```

The following code initiates the search.

```
var bkmName = app.response({
   cQuestion: "Enter the name of the bookmark to find",
      cTitle: "Bookmark Search and Execute"});
if ( bkmName != null ) {
   var bkm = searchBookmarks(this.bookmarkRoot, 0, bkmName );
   if ( bkm != null )
      bkm.execute();
   else
      app.alert("Bookmark not found");
}
```

10.7 Certificate

10.7.1 General

The **`Certificate`** object provides read-only access to the properties of an X.509 public key certificate.

NOTE There are no security restrictions on this object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.7.2 Certificate properties

10.7.2.1 General

Table 11: Certificate object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **Certificate**. For properties that return objects that are specific to this standard the lists of properties that shall be supported for those object types are provided below.

Table 11 — Certificate object properties

Property	Type	Access	Description
Binary	String	R	The raw bytes of the certificate, as a hex encoded string.
issuerDN	RDN object	R	The distinguished name of the issuer of the certificate, returned as an **RDN** object.
keyUsage	Array of Strings	R	An array of strings indicating the value of the certificate key usage extension. Possible values are: ***kDigitalSignature*** ***kDataEncipherment*** ***kCRLSign*** ***kNonRepudiation*** ***kKeyAgreement*** ***kEncipherOnly*** ***kKeyEncipherment*** ***kKeyCertSign*** ***kDecipherOnly***
MD5Hash	String	R	The MD5 digest of the certificate, represented as a hex-encoded string. This provides a unique fingerprint for this certificate. NOTE The use of MD5 and SHA1 is deprecated with PDF 2.0.
privateKeyValidityEnd	Date object	R	The date before which it's legal to use the private key associated with this certificate. If the PKUP extension is not present or this property isn't present in the extension, this represents the validity end date of the certificate itself. Before a digital ID can be used for signing, PDF Processors should ensure that the signing time is prior to the **privateKeyValidityEnd** date.
privateKeyValidityStart	Date object	R	The date after which it's legal to use the private key associated with this certificate. If the Private Key Usage Period (PKUP) certificate extension is not present, this represents the validity start date of the certificate itself. Before a digital ID can be used for signing, PDF Processors should ensure that the signing time is more recent than the **privateKeyValidityStart** date.
SHA1Hash	String	R	The SHA1 digest of the certificate, represented as a hex-encoded string. This provides a unique fingerprint for this certificate. NOTE The use of MD5 and SHA1 is deprecated with PDF 2.0.
serialNumber	String	R	A unique identifier for this certificate, used in conjunction with issuerDN.
subjectCN	String	R	The common name of the signer.
subjectDN	RDN object	R	The distinguished name of the signer, returned as an **RDN** object.
ubRights	Rights object	R	The application rights that can be enabled by this certificate, returned as a generic **Rights** object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 11 *(continued)*

Property	Type	Access	Description
usage	Usage object	R	The purposes for which this certificate may be used within the PDF Processor environment returned as a **`Usage`** object.
validityEnd	Date object	R	The validity end date of the certificate. Before a digital ID can be used for signing, PDF Processors should ensure that the signing time is prior to the **`validityEnd`** date.
validityStart	Date object	R	The validity start date of the certificate. Before a digital ID can be used for signing, PDF Processors should ensure that the signing time is more recent than the **`validityStart`** date.

10.7.2.1.1 Rights Object

Table 12: Rights object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type *Rights*. A *Rights* object representing the application rights that can be enabled by a specific *Certificate* object can be accessed through the *Certificate.ubRights* property.

Table 12 — Rights object properties

Property	Type	Access	Description
mode	String	R	Possible values are: ***`Evaluation`*** Rights enabled by this certificate for this document are valid as long as this certificate is valid ***`Production`*** Rights enabled by this certificate for this document are valid indefinitely
rights	Array of Strings	R	Array of strings indicating the application rights that can be enabled by this certificate. Possible values are: ***`FormFillInAndSave`*** The right to fill in forms, excluding signature fields, and to save the modified file ***`FormImportExport`*** The right to import and export form data ***`FormAddDelete`*** The right to add or delete a form field ***`SubmitStandalone`*** The right to submit a document outside a browser ***`SpawnTemplate`*** The right to spawn page templates ***`Signing`*** The right to sign existing form fields in a document ***`AnnotModify`*** The right to create, delete, and modify comments ***`AnnotImportExport`*** The right to import and export annotations ***`BarcodePlaintext`*** The right to encode the appearance of a form field as a plain text barcode **AnnotOnline** Allow online commenting. Enables uploading of any annotations in the document to a server and downloading of annotations from a server. Does not enable the addition of these annotations into the document. ***`FormOnline`*** Enable forms-specific online mechanisms such as SOAP or Active Data Object ***`EFModify`*** The right to create, delete, modify, and import named embedded files. Does not apply to file attachment annotations

10.7.2.1.2 Usage Object

Table 13: Usage object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`Usage`**. A **`Usage`** object representing the purposes for

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

which a specific `Certificate` object may be used can be accessed through the `Certificate.usage` property.

Table 13 — Usage object properties

Property	Type	Access	Description
endUserSigning	Boolean	R	true if the certificate is usable for end-user signing.
endUserEncryption	Boolean	R	true if the certificate is usable for end-user encryption.

10.8 color

10.8.1 General

The `color` object is a convenience static object that defines the basic colors. Use this object to set a property or call a method that requires a color array.

10.8.2 color arrays

A color is represented in ECMAScript as an array containing 1, 2, 4, or 5 elements corresponding to a Transparent, Gray, RGB, or CMYK color space, as defined in Table 14: color arrays. The first element in the array is a string denoting the color space type. The subsequent elements are numbers that range between zero and one inclusive.

For example, the color red can be represented as ["RGB", 1, 0, 0].

Invalid strings or insufficient elements in a color array cause the color to be interpreted as the color black.

Table 14 — color arrays

Color space	String	Additional elements	Description
Transparent	*"T"*	0	A transparent color space indicates a complete absence of color and allows those portions of the document underlying the current field to show through.
Gray	*"G"*	1	Colors are represented by a single value — the intensity of achromatic light. In this color space, *0* is black, *1* is white, and intermediate values represent shades of gray. For example, *0.5* represents medium gray.
RGB	*"RGB"*	3	Colors are represented by three values: the intensity of the red, green, and blue components in the output. RGB is commonly used for video displays, which are generally based on red, green, and blue phosphors.
CMYK	*"CMYK"*	4	Colors are represented by four values, the amounts of the cyan, magenta, yellow, and black components in the output. This color space is commonly used for color printers, where they are the colors of the inks used in four-color printing. Only cyan, magenta, and yellow are necessary, but black is generally used in printing because black ink produces a better black than a mixture of cyan, magenta, and yellow inks and because black ink is less expensive than the other inks.

10.8.3 color properties

The color object defines the colors listed in Table 15: color properties.

Table 15 — color properties

Color object	Keyword	Equivalent ECMAScript
Transparent	color.transparent	`[ "T" ]`
Black	color.black	`[ "G", 0 ]`

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 15 *(continued)*

Color object	Keyword	Equivalent ECMAScript
White	color.white	`[ "G", 1 ]`
Red	color.red	`[ "RGB", 1,0,0 ]`
Green	color.green	`[ "RGB", 0,1,0 ]`
Blue	color.blue	`[ "RGB", 0, 0, 1 ]`
Cyan	color.cyan	`[ "CMYK", 1,0,0,0 ]`
Magenta	color.magenta	`[ "CMYK", 0,1 0,0 ]`
Yellow	color.yellow	`[ "CMYK", 0,0,1,0 ]`
Dark Gray	color.dkGray	`[ "G", 0.25 ]`
Gray	color.gray	`[ "G", 0.5 ]`
Light Gray	color.ltGray	`[ "G", 0.75 ]`

10.8.4 color methods

10.8.4.1 convert(colorArray, cColorspace)

Parameters

Property	Type	Description
colorArray	Array	Array of color values.
cColorspace	Colorspace	The color space to which to convert.

Return Value

`color` array

Description

Converts the color space and color values specified by the color object to the specified color space:

Conversion to the gray color space is lossy (in the same fashion that displaying a color TV signal on a black-and-white TV is lossy).

The conversion of RGB to CMYK does not take into account any black generation or undercolor removal parameters.

10.8.4.2 equal(colorArray1, colorArray2)

Parameters

Property	Type	Description
colorArray1	Array	The first color array for comparison.
colorArray2	Array	The second color array for comparison.

Return Value

`boolean`

Description

Compares two Color Arrays to see if they are the same. The routine performs conversions, if necessary, to determine if the two colors are indeed equal (for example, ["RGB",1,1,0] is equal to ["CMYK",0,0,1,0]).

Implementers of this standard shall return ***`true`*** if the two parameters represent the same color, otherwise the return value shall be ***`false`***.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.9 collection

10.9.1 General

A **collection** object is obtained from the **Doc.collection** property. **Doc.collection** returns a *null* value when there is no PDF collection (also called PDF package and PDF portfolio).

The **collection** object is used to set the initial document in the collection, set the initial view of the collection, and to get, add, and remove collection fields (or categories).

10.9.2 collection properties

Table 16: Collection object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **collection**.

Table 16 — Collection object properties

Property	Type	Access	Description
fields	array	R	The **fields** property is an array of **collectionField** objects in the collection.
initialDoc	string	R/W	Use the **initialDoc** property to specify the document in the collection that is initially viewed when the collection is opened. The value of initialDoc is the name property of the **Data** object of the initial document, or ***null*** if no initial document is specified.
initialView	string	R/W	A string describing the initial collection view. Possible values include ***Tile***, ***Details***, ***Hidden***, ***Custom***, or ***null***. The values of ***Tile*** and ***Details*** are built-in views, while ***Custom*** indicates that a document-based Navigator is preferred.

10.9.3 collection methods

10.9.3.1 addField(name, text[, type, order, visible, readOnly])

Parameters

Field	Type	Description
name	String	A string that is the name of the field of the **collectionField** object to be added.
text	String	The text of this field that is displayed to the user in the user interface.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Field	Type	Description
type	String	(Optional) A string that specifies the type of data associated with the field. Possible values are listed below: ***S*** a string type. A field with a string value. ***D*** a date type. A field with a date value. ***N*** a numeric type. A field with a numeric value. The following values identify the types of file-related fields: ***F*** a field reserved for the file name of a member of the collection. ***Desc*** a field reserved for a description string of a member of the collection. ***Size*** a field reserved for the file size of a member of the collection. ***ModDate*** a field reserved for the modification date of a member of the collection. ***CreationDate*** a field reserved for the creation date of a member of the collection. ***CompressedSize*** a field reserved for the compressed file size of a member of the collection. The default is "***S***", a string type.
order	Integer	(Optional) An integer that indicates the order of the field in the display of the collection in the user interface. If this parameter is not specified, the new field is listed last.
visible	Boolean	(Optional) A Boolean value that indicates the visibility of the field. The field is visible if the value is ***true***; not visible if ***false***. The default is ***true***.
readOnly	Boolean	(Optional) A Boolean value, the field is read only if ***true***. The default is ***false***. If a field is read only, the user is not allowed to change the value of this field through the user interface.

Return Value

`collectionField`

Description

Adds a new field (or category) to the collection schema. The method throws an exception if the field already exits.

Implementers of this standard shall return the collectionField object of the newly created field in response to calls to the **`addField`** method.

10.9.3.2 getField(name)

Parameters

Parameter	Type	Description
name	String	A string that is the name of the field of the collectionField object to be retrieved.

Return Value

`collectionField`

Description

The getField method returns the **`collectionField`** object corresponding to the name of a collection field.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Implementers of this standard shall return the `collectionField` object matching the name, or, if a `collectionField` with a matching name is not found, then *`null`* shall be returned.

10.9.3.3 removeField(name)

Parameters

Parameter	Type	Description
name	String	A string that is the name of the field to be removed.

Return Value

undefined

Description

Remove a field from the collection schema.

10.10 collectionField

10.10.1 General

The properties of the `collectionField` object are the fields (or categories) used by the collection. The `text` property (see "Table 17: `collectionField` object properties") is the (display) string the user sees in the user-interface with viewing a listing of the files in the collection. Use the `collectionField` object to set the order the fields are listed in the user-interface, the sort order of the listing of the files, or to set a field as read only.

The collection field values for embedded files in the collection can be got or set through `getFieldValue` and `setFieldValue` methods of the `Data` object.

10.10.2 collectionField properties

10.10.2.1 General

Table 17: collectionField object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `collectionField`.

Table 17 — collectionField object properties

Property	Type	Access	Description
name	string	R	The string that specifies the name of the field.
Order	integer	R/W	Specifies the order the field is listed in relation to other fields. Field order is 0-based. When changing the value, if needed, the order of other fields will be incremented to prevent duplicates.
readOnly	Boolean	R/W	A Boolean value that specifies whether the user may change values of the field in the user interface. When readonly is true, no changes are allowed.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 17 *(continued)*

Property	Type	Access	Description
sort	integer	R/W	The value of sort specifies how this field contributes to the ordering of files in the collection. If the value is 0, this field is not used to sort the files. If the absolute value is 1, this field is the primary sort field. If the absolute value is 2, the field is the secondary sort field (used to break ties on the primary sort field), and so on. Additionally, the sign of value indicates the direction in which the files should be sorted. Positive values indicate that the files are sorted in ascending order, and negative values indicate that files are sorted in descending order.
text	string	R/W	The value of text specifies the display text used by the field in the user-interface.
type	string	R	A text string that specifies the type of data stored in the field. Possible values include: ***S*** the value of this field is a string type. ***D*** the value of this field is a date type. ***N*** the value of this field is a number type. ***F*** the value of this field is the file name of a member of the collection. ***Desc*** the value of this field is a description string of a member of the collection. ***Size*** the value of this field is the file size of a member of the collection. ***ModDate*** the value of this field is the modification date of a member of the collection. ***CreationDate*** the value of this field is the creation date of a member of the collection. ***CompressedSize*** the value of this field is the compressed file size of a member of the collection. The value of this property cannot be changed for a field that already exists. When a new field is created using the ***addField*** method, the type can be set.

10.11 Data

10.11.1 General

The `Data` object is the representation of an embedded file or data stream that is stored in the document. See ISO 32000-2 for details.

Using `Data` objects is a good way to associate and embed source files, metadata, and other associated data with a document. `Data` objects can be inserted from the external file system, queried, and extracted.

See the following `Doc` properties and methods:

`createDataObject`(cName, cValue[, cMediaType, cCryptFilter]), `dataObjects`, `exportDataObject`(cName[, cDIPath, bAllowAuth, nLaunch]), `getDataObject`(cName), `importDataObject`([cDIPath, cCryptFilter]), `removeDataObject`(cName), `openDataObject`(cName), `getDataObjectContents`(cName[, bAllowAuth]), and `setDataObjectContents`(cName, oStream[, cCryptFilter]).

The following objects, properties and methods support PDF portfolios (also called PDF collections or PDF packages):

`app.newCollection`, `Doc.collection`, `Collection object`, and `collectionField` object

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.11.2 Data properties

Table 18: Data object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Data`.

Table 18 — Data object properties

Property	Type	Access	Description
creationDate	Date	R	The creation date of the file that was embedded.
description	String	R/W	The description associated with this data object.
MIMEType	String	R	The MIME type associated with this data object.
modDate	Date	R	The modification date of the file that was embedded.
name	String	R	The name associated with this data object.
path	String	R	The file name and extension of the file that was embedded.
size	Number	R	The size, in bytes, of the uncompressed data object.

10.11.3 Data methods

10.11.3.1 getFieldValue(name[, includePrefix])

Parameters

Parameter	Type	Description
name	String	A string that identifies the field of interest.
includePrefix	Boolean	(Optional) A Boolean value indicating whether the prefix should be included with the returned string. If ***true***, the prefix is included with the return string. The default is ***false***.

Return Value

`Object`

Description

Get the value of the field as specified by the name parameter for this embedded file. When attempting to get a field value that is not defined in the collection schema, null is returned.

10.11.3.2 setFieldValue(name, value[, prefix])

Parameters

Parameter	Type	Description
name	String	A string that is the name of the field of the collectionField object to be retrieved.
value	String	A string, a number, or a date that specifies the value of the field. The value will be type-coerced to match the schema.
prefix	String	(Optional) A string that specifies the field value prefix.

Return Value

`Object`

Description

Type conversion is performed when setting field values. The type conversion is determined by the collectionField in the schema. When attempting to set a field value that is not defined in the schema, an exception is raised.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Returns the value of the field, or ***null*** if field is not present in schema.

10.12 Dialog

10.12.1 General

An instance of this object is passed as a parameter to dialog box handlers (see "Dialog box handlers"). These handlers include the **initialize**, **validate**, **commit**, **destroy** and **ItemID** methods of the dialog box descriptor object literal that is passed to **app.execDialog**. The **Dialog** object allows the current state of the Dialog to be queried and set.

10.12.2 Dialog methods

10.12.2.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type **Dialog**.

10.12.2.2 enable(...)

Parameters

Parameter	Type	Description
object literal	object literal	For each dialog box item to modify, there should be an entry in the object literal with the Dialog ItemID as the label and a Boolean value as the value indicating if it is enabled or not.

Return Value

undefined

Description

Enables or disables various dialog box elements using the object literal passed in.

Typically, enable is called in the **initialize** method (see "Dialog box handlers") of the object literal passed to **app.execDialog** to preset whether various dialog box elements are enabled or not.

10.12.2.3 end([itemId])

Parameters

Parameter	Type	Description
itemId	ItemID	(optional) The ItemID of the dialog box element that will be reported as dismissing the dialog.

Return Value

undefined

Description

Terminates a currently executing dialog box (as if the Cancel button had been pressed). This method takes an optional parameter of the ItemID, a string, of the dialog box element that will be reported as dismissing the dialog. This ItemID will be the return value of the **app.execDialog** call that created the dialog.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.12.2.4 load(...)

Parameters

Parameter	Type	Description
object literal	object literal	For each dialog box item to be modified, there should be an entry in the object literal with the ItemID as the label and the dialog box element setting as the contents. If the dialog box element takes multiple values (for example, a list_box or a popup), the value should be an object literal consisting of the displayed entry as the label and a numeric value as the contents. Similarly, if the dialog box element is hierarchical in nature (for example, a hier_list_box), the value should be a set of nested object literals. If the numeric value is greater than 0, the item is selected, otherwise it is not selected.

Return Value

undefined

Description

Sets the values of dialog box elements using the object literal passed in. Dialog box items are identified by an ItemID which is a unique 4-character string.

Typically, load is called in the **initialize** method (see "Dialog box handlers") of the object literal passed to `app. execDialog` to preset the value of various dialog box elements.

10.12.2.5 store()

Parameters

None

Return Value

`object literal`

Description

Gets the values of dialog box elements as an object literal returned. Dialog box items are identified by an ItemID, which is a unique 4-character string. For each dialog box element, there will be an entry in the object literal with the ItemID as the label and the dialog box element setting as the contents. If the dialog box element takes multiple values (for example, a list_box or a popup), the value should be an object literal consisting of the displayed entry as the label and a numeric value as the contents. If the numeric value is greater than 0, the item was selected, otherwise it was not selected.

Typically, store is called in the commit method (see "Dialog box handlers") of the object literal passed to app.execDialog to extract the value of various dialog box elements.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

10.13 Doc

10.13.1 General

This object provides the interface between a PDF document open in the viewer and the ECMAScript interpreter. It provides methods and properties for accessing the PDF document.

You can access the `Doc` object from ECMAScript in a variety of ways:

The `this` object usually points to the Doc object of the underlying document.

© ISO 2020 – All rights reserved

Some properties and methods, such as `extractPages`, `app.activeDocs`, and `app.openDoc`, return the `Doc` object.

The `Doc` object can often be accessed through event objects, which are created for each event by which an ECMAScript is executed:

For mouse, focus, blur, calculate, validate, and format events, `event.target` returns the `Field` object that initiated the event. You can then access the `Doc` object through the `Doc` method of the `Field` object.

For all other events, `event.target` points to the `Doc`.

10.13.2 Doc properties

Table 19: Doc object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Doc`.

Table 19 — Doc object properties

Property	Type	Access	Description
author	String	R/W	The author of the document. NOTE This property has been superseded by the info property.
baseURL	String	R/W	The base URL for the document is used to resolve relative web links within the document. See also **URL**.
bookmarkRoot	Object	R	The root bookmark for the bookmark tree. This bookmark is not displayed to the user but is a programmatic construct used to access the tree and the child bookmarks.
calculate	Boolean	R/W	If ***true*** (the default value), calculations can be performed for this document. If ***false***, calculations cannot be performed for this document. This property supersedes the app.calculate property, whose use is now discouraged.
collection	Collection or null	R	The value of the collection property is a `Collection` object of the collection in this PDF. Properties of the collection may be changed through the `Collection` object. NOTE A PDF file cannot be converted into a PDF collection. The attribute of a document as a PDF collection cannot be removed. If a new collection is desired, use `app.newCollection`.
creationDate	Date	R	The document's creation date. NOTE This property has been superseded by the info property.
creator	String	R	The creator (application name) of the document. NOTE This property has been superseded by the info property.
dataObjects	Array	R	An array containing all the named `Data` objects in the document.
delay	Boolean	R/W	This property can delay the redrawing of any appearance changes to every field in the document. It is generally used to buffer a series of changes to fields before requesting that the fields regenerate their appearance. If ***true***, all changes are queued until delay is reset to ***false***, at which time all the fields on the page are redrawn. See also the **Field** object **delay** property
dirty	Boolean	R/W	Specifies whether the document needs to be saved as the result of a changes to the document. It is useful to reset the dirty flag when performing changes that do not warrant saving, such as updating a status field.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 19 *(continued)*

Property	Type	Access	Description
disclosed	Boolean	R/W	`[Security]` Specifies whether the document should be accessible to ECMAScript scripts in other documents. The `app.openDoc` and `app.activeDocs` methods check the disclosed property of the document before returning its `Doc`.
docID	Array	R	An array of two strings in hex-encoded binary format. The first string is a permanent identifier based on the contents of the file at the time it was originally created; it does not change when the file is incrementally updated. The second string is a changing identifier based on the file's contents at the time it was last updated. These identifiers are defined by the optional ID entry in a PDF file's trailer dictionary. (See ISO 32000-2.)
documentFileName	String	R	The base file name, with extension, of the document referenced by the `Doc`. The device-independent path is not returned. See also the **path** and **URL** properties. The file size of the document can be obtained from the filesize property.
external	Boolean	R	Specifies whether the current document is being viewed in the PDF Processor application or in an external window (such as a web browser).
filesize	Integer	R	The file size of the document in bytes.
hidden	Boolean	R	This property is true if the document's window is hidden. A window may be hidden, for example, because it is being operated on in batch mode or if it was explicitly opened hidden. The `openDataObject` and app.`openDoc` methods can be used to open a document with a hidden window.
hostContainer	Object	R/W	An instance of the `HostContainer` object if the PDF document is embedded in another container such as a web browser, otherwise undefined.
icons	Array or null	R	An array of named `Icon` objects that are present in the document-level named icons tree. If there are no named icons in the document, the property has a value of null. See also `addIcon`(cName, icon), `getIcon`(cName), `importIcon(cName[, cDIPath, nPage])`, `removeIcon(cName)`, the `Field` object properties `buttonGetIcon([nFace])`, `buttonImportIcon`([cPath, nPage]), `buttonSetIcon`(oIcon[, nFace]), and the `Icon` object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 19 *(continued)*

Property	Type	Access	Description
info	Object	R/W	Specifies an object with properties from the document information dictionary in the PDF file. (See ISO 32000-2). Standard entries are: ***Title*** ***Author*** ***Authors*** ***Subject*** ***Keywords Creator*** ***Producer*** ***CreationDate*** ***ModDate*** ***Trapped*** Properties of this object are writeable and setting a property dirties the document. Additional document information fields can be added by setting non-standard properties. NOTE Standard entries are case insensitive, that is, info.Keywords is the same as info.keywords.
isModal	Object	R	A Boolean value indicating whether the document is currently in a modal state (for example, displaying a modal dialog box using **app.execDialog**).
keywords	Object	R/W	The keywords that describe the document (for example, "***forms***", "***taxes***", "***government***"). NOTE This property has been superseded by the info property.
layout	String	R/W	Changes the page layout of the current document. Valid values are: ***SinglePage*** ***OneColumn*** ***TwoColumnLeft*** ***TwoColumnRight*** ***TwoPageLeft*** ***TwoPageRight***
metadata	String	R/W	Allows you to access the XMP metadata embedded in a PDF document. Returns a string containing the metadata as XML. For information on embedded XMP metadata, see ISO 32000-2. Implementers of this standard shall throw a RaiseError exception if an attempt is made to write a value to **metadata** that is not in XMP format.
modDate	Date	R	The date the document was last modified. NOTE This property has been superseded by the info property.
mouseX	Number	R	Gets the x coordinate of the mouse coordinates in default user space in relation to the current page.
mouseY	Number	R	Gets the y coordinate of the mouse coordinates in default user space in relation to the current page

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 19 *(continued)*

Property	Type	Access	Description
noautocomplete	Boolean	R/W	This property can be used to turn off the auto-complete feature for forms, for this document only: If ***true***, no suggestions are made as the user enters data into a field. If ***false***, auto-complete respects the user preference Forms > Auto-Complete. Setting this property does not change the user's auto-complete preferences. Initially, this property has a value of ***undefined***.
numFields	Integer	R	The total number of fields in the document. See also **getNthFieldName**(nIndex).
numPages	Integer	R	The number of pages in the document.
numTemplates	Integer	R	The number of templates in the document. NOTE This property has been superseded by templates.
path	String	R	The device-independent path of the document, for example: /c/Documents/User/Example.pdf The file name of the document can be acquired by the documentFileName property. See also the URL property.
pageNum	Integer	R/W	Gets or sets the current page of the document. When setting pageNum to a specific page, remember that the values are 0-based.
permStatusReady	Boolean	R	A Boolean value specifying whether the permissions for this document have been resolved. When downloading over a network connection, false can indicate that the document is not available, in the case where permissions must be determined based on a certification signature that covers the entire document.
producer	String	R	The producer of the document. NOTE This property has been superseded by the info property.
requiresFullSave	Boolean	R	This property is ***true*** if the document requires a full save because it is temporary or newly created. Otherwise, it is ***false***.
securityHandler	String	R	The name of the security handler used to encrypt the document. Returns null if there is no security handler (for example, the document is not encrypted).
selectedAnnots	Array	R	An array of **Annotation** objects corresponding to all currently selected annotations. See also **getAnnot**(nPage, cName) and **getAnnots**([nPage, nSortBy, bReverse, nFilterBy]).
subject	String	R/W	The document's subject. NOTE This property has been superseded by the info property.
templates	Array	R	An array of the **Template** objects in the document. See also **createTemplate** (cName, nPage), **getTemplate(cName)**, and **removeTemplate(cName)**.
title	String	R/W	The title of the document. NOTE This property has been superseded by the info property.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 19 *(continued)*

Property	Type	Access	Description
URL	String	R	The document's URL. If the document is local, it returns a URL based on the **file** scheme. This may be different from the baseURL. See also the path and **documentFileName** properties.
viewState	Object	R/W	An opaque object representing the current view state of the document. The state includes, at minimum, information about the current page number, scroll position, zoom level, and field focus. To set this value, you must use what was previously returned from a read of the value. It can be used to restore the view state of a document. NOTE The object is only defined cwithin an embedded PDF.
zoom	Number	R/W	The current page zoom level. Allowed values are between 8,33 % and 6 400 %, specified as a percentage number. For example, a zoom value of 100 specifies 100 %.
zoomType	String	R/W	The current zoom type of the document. The convenience zoomtype object defines all the valid zoom types: **Zoom type** — **Keyword** ***NoVary*** — ***zoomtype.none*** ***FitPage*** — ***zoomtype.fitP*** ***FitWidth*** — ***zoomtype.fitW*** ***FitHeight*** — ***zoomtype.fitH*** ***FitVisibleWidth*** — ***zoomtype.fitV*** ***Preferred*** — ***zoomtype.pref*** ***ReflowWidth*** — ***zoomtype.refW***

10.13.3 Doc methods

10.13.3.1 addAnnot(...)

Parameters

Parameter	Type	Description
object literal	Object	A generic object that specifies the properties of the **Annotation** object, such as **type**, **rect**, and **page**, to be created.

Return Value

Annotation

Description

Creates an **Annotation** object having the specified properties. Properties not specified are given their default values for the specified type of annotation.

10.13.3.2 addField(cName, cFieldType, nPageNum, oCoords)

Parameters

Parameter	Type	Description
cName	String	The name of the new field to create. This name can use the dot separator syntax to denote a hierarchy (for example, name.last creates a parent node, name, and a child node, last).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
cFieldType	String	The type of form field to create. Valid types are: ***text*** ***button*** ***combobox*** ***listbox*** ***checkbox*** ***radiobutton*** ***signature***
nPageNum	Number	The 0-based index of the page to which to add the field.
oCoords	Array	An array of four numbers in rotated user space that specifies the size and placement of the form field. These four numbers are the coordinates of the bounding rectangle, in the following order: upper-left x, upper-left y, lower-right x and lower-right y. See also the **`Field`** object **rect** property. If you use the Info panel to obtain the coordinates of the bounding rectangle, you shall transform them from info space to rotated user space. To do this, subtract the info space y coordinate from the on-screen page height.

Return Value

`Field`

Description

Creates a new form field and returns it as a Field object.

10.13.3.3 addIcon(cName, icon)

Parameters

Parameter	Type	Description
cName	String	The name of the new object
icon	Object	The **`Icon`** object to add.

Return Value

Undefined

Description

Adds a new named Icon object to the document-level icon tree, storing it under the specified name.

See also **`icons`**, **`getIcon`**(cName), **`importIcon`**(cName[, cDIPath, nPage]), **removeIcon**(cName), and the **`Field`** object methods **`buttonGetIcon`**([nFace]), **`buttonImportIcon`**([cPath, nPage]), and **`buttonSetIcon`**(oIcon[, nFace]).

10.13.3.4 addLink(nPage, oCoords)

Parameters

Parameter	Type	Description
nPage	Number	The page on which to add the new link.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
oCoords	Array	An array of four numbers in rotated user space specifying the size and placement of the link. The numbers are the coordinates of the bounding rectangle in the following order: upper-left x, upper-left y, lower-right x and lower-right y.

Return Value

Link

Description

Adds a new link to the specified page with the specified coordinates, if the user has permission to add links to the document. See also **`getLinks`**(nPage, oCoords), **`removeLinks`**(nPage, oCoords) and the **`Link`** object.

10.13.3.5 addRecipientListCryptFilter(cCryptFilter, oGroup)

Parameters

Parameter	Type	Description
cCryptFilter	String	The language-independent name of the crypt filter. This same name should be used as the value of the **`cCryptFilter`** parameter of the **`Doc`** methods **`importDataObject`**, **`createDataObject`**, and **`setDataObjectContents`**. NOTE For some PDF Processors the value of cCryptFilter must be DefEmbeddedFile; for other PDF Processors, the value of cCryptFilter can be any string.
oGroup	Array	An array of Group objects that lists the recipients for whom the data is to be encrypted.

Return Value

undefined

Description

`[Security]`

Adds a crypt filter to the document. The crypt filter is used for encrypting Data objects.

See also the cCryptFilter parameter of the **`importDataObject`**([cDIPath, cCryptFilter]), **`createDataObject`**(cName, cValue[, cMediaType, cCryptFilter]) and **`setDataObjectContents`**(cName, oStream[, cCryptFilter]) methods.

10.13.3.6 addRequirement(cType[, oReq])

Parameters

Parameter	Type	Description
cType	String	The type of document requirement. The types are described by the Requirements Enumerator object.
oReq	Object	(Optional) A Requirement object.

Return Value

undefined

Description

`[Security]`

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Allows a PDF document to be authored so that a certain requirement is needed for the document to properly function in a conforming PDF Processor.

When a PDF Processor opens a document containing a requirement, it shall evaluate whether the requirement can be satisfied before allowing the user to interact with the document. If the requirement cannot be satisfied, the application may limit the user's interaction with the document.

10.13.3.6.1 Requirements enumerator object

This object lists all the possible types of requirements that a document may contain to properly function in PDF Processor.

Table 20: Requirements enumerator property lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`Requirements Enumerator:`**

Table 20 — Requirements enumerator property

Property	Description
requirements.EnableECMAScripts	Some documents may contain data validation scripts that may never run. For example, running of scripts could be controlled by a user preference that a user may disable. This property allows a PDF document to enforce the execution of its scripts in PDF Processor.

10.13.3.6.2 Requirement object

This generic object contains properties that describe the nature of the requirement

Table 21: Requirement object property lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`Requirements object:`**

Table 21 — Requirement object property

Property	Description
aRH	(Optional) An array of **`ReqHandler`** objects.

10.13.3.6.3 ReqHandler object

This generic object contains information about a requirement handler that can be used when a PDF Processor finds an unrecognized requirement. The viewer should delegate requirement checking for the unrecognized requirement to the first handler in the array that supports the type. If no requirement handler can be found to deal with the unrecognized requirement, a generic message should be provided by the viewer.

Table 22: ReqHandler properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`ReqHandler:`**

Table 22 — ReqHandler properties

Property	Description
cType	A string specifying the type of the requirement handler (see the **`ReqHandlers Enumerator object`** for a lists of possible names).
cScriptName	(Optional) A string specifying the name of a document-level ECMAScript present in the document. It may be present if the value of **`cType`** is ***`reqHandlers.JS`***. The named script will not be executed in case the requirement is satisfied.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.6.4 ReqHandlers Enumerator object

This object enumerates the types of requirement handlers a document may contain.

Table 23: ReqHandlers enumerator properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `ReqHandlers Enumerator:`

Table 23 — ReqHandlers enumerator properties

Property	Description
reqHandlers.JS	This handler manages document-level scripts that deal with unrecognized requirements in the PDF document.
reqHandlers.NoOp	This handler allows older viewers to ignore unrecognized requirements.

10.13.3.7 addScript(cName, cScript)

Parameters

Parameter	Type	Description
cName	String	The name of the script. If a script with this name already exists, the new script replaces the old one.
cScript	String	An ECMAScript expression to be executed when the document is opened.

Return Value

undefined

Description

Sets a document-level script for a document. See also `setAction`(cTrigger, cScript), `setPageAction`(nPage, cTrigger, cScript), the `Bookmark` object `setAction`(cScript) method, and the `Field` object `setAction`(cTrigger, cScript) method.

NOTE This method overwrites any script already defined for `cName`.

10.13.3.8 addThumbnails([nStart, nEnd])

Parameters

Parameter	Type	Description
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages. If `nStart` and `nEnd` are not specified, the range of pages is for all pages in the document. If only `nStart` is specified, the range of pages is the single page specified by nStart. If only `nEnd` is specified, the range of a pages is 0 to `nEnd`.
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive range of pages. See `nStart` for details.

Return Value

undefined

Description

Creates thumbnails for the specified pages in the document. See also the `removeThumbnails`([nStart, nEnd]) method.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.9 bringToFront()

Parameters

None

Return Value

undefined

Description

Brings an open document to the front.

10.13.3.10 calculateNow()

Parameters

None

Return Value

undefined

Description

Forces computation of all calculation fields in the current document.

When a form contains many calculations, there can be a significant delay after the user inputs data into a field, even if it is not a calculation field. One strategy is to turn off calculations at some point and turn them back on later.

10.13.3.11 certifyInvisibleSign(oSig[, oInfo, cDIPath, bUI, cLegalAttest])

Parameters

Parameter	Type	Description
oSig	Object	The signature engine object.
oInfo	Object	(optional) Additional signing information.
cDIPath	String	(optional) The file path for saving the signed file. The file is saved over itself if no path is specified.
bUI	Boolean	(optional) Set TRUE to enable UI interaction. May be FALSE if a path and certificate are supplied. The default is FALSE.
cLegalAttest	String	(optional) The signing reason.
bNoSave	Boolean	(optional) A Boolean value indicating whether to close the document without saving: If false (the default), the user is prompted to save the document if it has been modified. If true, the document is closed without prompting the user and without saving, even if the document has been modified. Be careful in using this feature because it can cause data loss without user approval.

Return Value

Boolean

Description

Adds an invisible certification to a document.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Implementers shall return TRUE if the signature is successfully applied to the document or FALSE if the signature is not successfully applied.

10.13.3.12 closeDoc(bNoSave)

Parameters

Parameter	Type	Description
bNoSave	Boolean	If true, close the document without saving. If false, save or prompt user to save before closing the document.

Return Value

undefined

Description

Closes the document.

It is important to use this method carefully, because it is an abrupt change in the document state that can affect any ECMAScript executing after the close. Triggering this method from a `Page` event or `Document` event could cause the application to behave strangely.

In some PDF Processors, a document that closes itself by executing `this.closeDoc` terminates any script that follows it. In other PDF Processors the script is allowed to continue and to terminate naturally. However, if the `Doc` of the closed document is referenced, an exception may be thrown.

10.13.3.13 createDataObject(cName, cValue[, cMediaType, cCryptFilter])

Parameters

Parameter	Type	Description
cName	String	The name to associate with the data object.
cValue	String	A string containing the data to be embedded.
cMediaType	String	(optional) The media type of the data. The default is “text/plain”.
cCryptFilter	String	(optional) The language-independent name of a crypt filter to use when encrypting this data object. This crypt filter must have previously been added to the document’s list of crypt filters, using the Doc **addRecipientListCryptFilter** method, otherwise an exception will be

Return Value

undefined

Description

Creates a Data object.

Data objects can be constructed ad hoc. This is useful if the data is being created in ECMAScript from sources other than an external file.

Related objects, properties, and methods are `dataObjects`, `getDataObject`, `openDataObject`, `importDataObject`, `removeDataObject`, `getDataObjectContents`, and `setDataObjectContents`, and the `Data` object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.14 createTemplate (cName, nPage)

Parameters

Parameter	Type	Description
cName	String	The name to be associated with this page.
nPage	Number	(optional) The 0-based index of the page to operate on. The default is 0, the first page in the document.

Return Value

Template

Description

`[Security]`

Creates a visible template from the specified page. See also the **`templates`** property, the **`getTemplate`**(cName) and **`removeTemplate`**(cName) methods, and the **`Template object`**.

10.13.3.15 deletePages([nStart, nEnd])

Parameters

Parameter	Type	Description
nStart	Number	(optional) The 0-based index of the first page in the range of pages to be deleted. The default is 0, the first page in the document.
nEnd	Number	(optional) The last page in the range of pages to be deleted. If nEnd is not specified, only the page specified by nStart is deleted.

Return Value

undefined

Description

Deletes pages from the document. If neither page of the range is specified, the first page (page 0) is deleted. See also **`extractPages( [nStart, nEnd, cPath])`**.

All pages in the document cannot be deleted. There shall be at least one page remaining.

10.13.3.16 embedDocAsDataObject(cName, oDoc[, cCryptFilter, bUI])

Parameters

Parameter	Type	Description
cName	String	The name to associate with the data object.
oDoc	Object	The document to embed as a data object.
cCryptFilter	String	(optional) The language-independent name of a crypt filter to use when encrypting this data object. This crypt filter must have previously been added to the document's list of crypt filters, using the addRecipientListCryptFilter method, otherwise an exception will be thrown. The predefined Identity crypt filter can be used so that this data object is not encrypted in a file that is otherwise encrypted by the encryptForRecipients method.
bUI	Boolean	(optional) If true, an alert may be shown if oDoc requires saving and the permissions do not allow it to be saved. The default is false.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

undefined

Description

Embeds the specified document as a `Data` Object in the document.

10.13.3.17 exportAsFDF([bAllFields, bNoPassword, aFields, bFlags, cPath, bAnnotations])

Parameters

Parameter	Type	Description
bAllFields	Boolean	(optional) If true, all fields are exported, including those that have no value. If false (the default), excludes those fields that currently have no value.
bNoPassword	Boolean	(optional) If true (the default), do not include text fields that have the password flag set in the exported FDF file.
aFields	Array	(optional) The array of field names to submit or a string containing a single field name:
bFlags	Boolean	If specified, only these fields are exported, except those excluded by bNoPassword.
cPath	String	If aFields is an empty array, no fields are exported. The FDF file might still contain data, depending on the bAnnotations parameter.
bAnnotations	Boolean	If this parameter is omitted or is null, all fields in the form are exported, except those excluded by bNoPassword.

Return Value

Undefined

Description

`[Security]`

Exports form fields as an FDF file to the local hard drive.

10.13.3.18 exportAsFDFStr([bAllFields, bNoPassword, aFields, bAnnotations, cHRef])

Parameters

Parameter	Type	Description
bAllFields	Boolean	(optional) If true, all fields are exported, including those that have no value. If false (the default), excludes those fields that currently have no value.
bNoPassword	Boolean	(optional) If true (the default), do not include text fields that have the password flag set in the exported FDF file.
aFields	Array	(optional) The array of field names to submit or a string containing a single field name:
bFlags	Boolean	If specified, only these fields are exported, except those excluded by bNoPassword.
bAnnotations	Boolean	If aFields is an empty array, no fields are exported. The FDF file might still contain data, depending on the bAnnotations parameter.
cHRef	String	If this parameter is omitted or is null, all fields in the form are exported, except those excluded by bNoPassword.

Return Value

String

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Computes the same results as calling the **doc.exportAsFDF** method, but returns the results as a string instead of saving to a file.

If supplied, the cHRef **parameter** is inserted as the value of the **F** key in the **FDF** dictionary. If not supplied, the **F** key contains the value as **doc.exportAsFDF** would produce.

10.13.3.19 exportAsText([bNoPassword, aFields, cPath])

Parameters

Parameter	Type	Description
bNoPassword	Boolean	(optional) If true (the default), do not include text fields that have the password flag set in the exported text file.
aFields	Array	(optional) The array of field names to submit or a string containing a single field name: If specified, only these fields are exported, except those excluded by bNoPassword. If aFields is an empty array, no fields are exported. If this parameter is omitted or is null, all fields in the form are exported, except those excluded by bNoPassword
cPath	String	(optional) A string specifying the device-independent path for the file. The path may be relative to the location of the current document. If the parameter is omitted, a dialog box is shown to let the user select the file. The parameter cPath shall have a safe path (seeSafe **path**) and have a .txt extension. This method will throw a NotAllowedError (see **Error** object) exception if these security conditions are not met, and the method will fail.

Return Value

undefined

Description

`[Security]`

Exports form fields as a tab-delimited text file to a local hard disk. The text file that is created follows the conventions specified by Microsoft Excel. In particular, **exportAsText** correctly handles quotes and multiline text fields.

This method writes two lines to the text file, the first line is a tab-delimited list of the names of the fields specified by aFields, the second line is a tab-delimited list of the values of the fields.

10.13.3.20 exportAsXFDF([bAllFields, bNoPassword, aFields, cPath, bAnnotations])

Parameters

Parameter	Type	Description
bAllFields	Boolean	(optional) If true, all fields are exported, including those that have no value. If false (the default), excludes those fields that currently have no value.
bNoPassword	Boolean	(optional) If true (the default), do not include text fields that have the password flag set in the exported XFDF.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
aFields	Array	(optional) The array of field names to submit or a string containing a single field name: If specified, only these fields are exported, except those excluded by bNoPassword. If aFields is an empty array, no fields are exported. The XFDF file might still contain data, depending on the bAnnotations parameter. If this parameter is omitted or is null, all fields in the form are exported, except those excluded by bNoPassword. Specify non-terminal field names to export an entire subtree of fields.
cPath	String	(optional) A string specifying the device-independent path for the file. The path may be relative to the location of the current document. If the parameter is omitted, a dialog box is shown to let the user select the file. The parameter cPath shall have a safe path (see **`Safe path`**) and have an XFDF extension. This method will throw a NotAllowedError (see **`Error`** object) exception if these security conditions are not met, and the method will fail.
bAnnotations	Boolean	(optional) If true, annotations are included in the exported XFDF file. The default is false.

Return Value

undefined

Description

`[Security]`

Exports form fields as an XFDF file to the local hard drive.

XFDF is an XML representation of form data. See the document ISO 19444-1, *Document management — XML forms data format for more information*.

There is an import version of this same method, importAnXFDF.

NOTE If the **`cPath`** parameter is specified, this method can only be executed during batch and console events. See Privileged versus non-privileged context for details. The event object contains a discussion of ECMAScript events.

10.13.3.21 exportAsXFDFStr(bAllFields, bNoPassword, aFields, bFlags, bAnnotations, cHRef])

Parameters

Parameter	Type	Description
bAllFields	Boolean	(optional) If true, all fields are exported, including those that have no value. If false (the default), excludes those fields that currently have no value.
bNoPassword	Boolean	(optional) If true (the default), do not include text fields that have the password flag set in the exported XFDF file.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
aFields	Array	(optional) The array of field names to submit or a string containing a single field name: If specified, only these fields are exported, except those excluded by bNoPassword. If aFields is an empty array, no fields are exported. The XFDF file might still contain data, depending on the bAnnotations parameter. If this parameter is omitted or is null, all fields in the form are exported, except those excluded by bNoPassword. Specify non-terminal field names to export an entire subtree of fields.
bAnnotations	Boolean	Shall be false, which is the default. Annotations are not supported.
cHRef	String	When supplied, its value is inserted as the source or target file of the returned XFDF expression (i.e., the value of the href attribute of the f element child of the xfdf element).

Return Value

String

Description

Computes the same results as calling the `doc.exportAsXFDF` method, but returns the results as a string instead of saving to a file.

If supplied, the `cHRef` parameter is inserted as the value of the href attribute of the f element child of the xfdf element. If not supplied, the href attribute of the f element key contains the value as `doc.exportAsXFDF` would produce.

10.13.3.22 exportDataObject(cName[, cDIPath, bAllowAuth, nLaunch])

Parameters

Parameter	Type	Description
cName	String	The name of the data object to extract.
bAllowAuth	Boolean	(optional) If true, a dialog box is used to obtain user authorization. Authorization may be required if the data object was encrypted using the encryptForRecipients method. Authorization dialog boxes are allowed if bAllowAuth is true. The default value is false.
nLaunch	Number	(optional) nLaunch controls whether the file is launched, or opened, after it is saved. Launching may involve opening an external application if the file is not a PDF file. The values of nLaunch are: If the value is 0, the file will not be launched after it is saved. If the value is 1, the file will be saved and then launched. Launching will prompt the user with a security alert warning if the file is not a PDF file. The user will be prompted for a save path. If the value is 2, the file will be saved and then launched. Launching will prompt the user with a security alert warning if the file is not a PDF file. A temporary path is used, and the user will not be prompted for a save path. The temporary file that is created will be deleted by PDF Processor upon application shutdown. The default value is 0.

Return Value

Undefined

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

[Security]

Extracts the specified data object to an external file.

Related objects, properties, and methods are **dataObjects**, **openDataObject**, **createDataObject**, **removeDataObject**, **importDataObject**, **getDataObjectContents**, and **setDataObjectContents**, and the **Data** object.

If **cDIPath** is not passed to this method, a file selection dialog box opens to allow the user to select a save path for the embedded data object.

10.13.3.23 extractPages([nStart, nEnd, cPath])

Parameters

Parameter	Type	Description
nStart	Number	(optional) A 0-based index that defines the start of the range of pages to extract from the source document. If only nStart is specified, the range of pages is the single page specified by nStart.
nEnd	Number	(optional) A 0-based index that defines the end of the range of pages to extract from the source document. If only nEnd is specified, the range of pages is 0 to nEnd.
cPath	String	(optional) The device-independent path to save the new document. The path name may be relative to the location of the current document. The parameter cPath shall have a safe path (see **Safe path**) and have a .pdf extension. This method will throw a NotAllowedError (see 10.15, "**Error**") exception if these security conditions are not met, and the method will fail.

Return Value

Doc

Description

[Security]

Creates a new document consisting of pages extracted from the current document. If a page range is not specified, the method extracts all pages in the document.

See also **deletePages([nStart, nEnd])**.

10.13.3.24 flattenPages([nStart, nEnd, nNonPrint])

Parameters

Parameter	Type	Description
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages in the current document. If only nStart is specified, the page range is the single page specified by nStart.
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive range of pages in the current document.
nNonPrint	Number	(optional) This parameter determines how to handle non-printing annotations. Values are 0 (default) Non-printing annotations are flattened 1 Non-printing annotations are left as is 2 Non-printing annotations are removed from the document

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Return Value

undefined

Description

Converts all annotations in a page range to page contents. If a page range is not specified, all annotations in the document are converted.

Great care shall be used when using this method. All annotations—including form fields, comments, and links — on the specified range of pages are flattened. They may have appearances, but they will no longer be annotations.

10.13.3.25 getAnnot(nPage, cName)

Parameters

Parameter	Type	Description
nPage	Number	The page that contains the Annotation object.
cName	String	The name of the Annotation object.

Return Value

Annotation or null

Description

Returns an `Annotation` object contained on a specific document page.

10.13.3.26 getAnnotRichMedia(nPage, cName)

Parameters

Parameter	Type	Description
nPage	Number	The 0-based page number that contains the AnnotRichMedia object.
cNa	String	The name of the AnnotRichMedia object.

Return Value

AnnotRichMedia or undefined

Description

This method gets an `AnnotRichMedia` object with a given name for a given page.

10.13.3.27 getAnnot3D(nPage, cName)

Parameters

Parameter	Type	Description
nPage	Number	The 0-based page number that contains the Annot3D object.
cName	String	The name of the Annot3D object.

Return Value

Annot3D or undefined

Description

Gets an `Annot3D` object with a given name from a given page.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.28 getAnnots([nPage, nSortBy, bReverse, nFilterBy])

Parameters

Parameter	Type	Description
nPage	Number	(optional) A 0-based page number. If specified, gets only annotations on the given page. If not specified, gets annotations that meet the search criteria from all pages.
nSortBy	Number	(optional) A sort method applied to the array. Values are: ANSB_None (default) Do not sort; equivalent to not specifying this parameter. ANSB_Page Use the page number as the primary sort criteria. ANSB_Author Use the author as the primary sort criteria. ANSB_ModDate Use the modification date as the primary sort criteria. ANSB_Type Use the annotation type as the primary sort criteria.
bReverse	Boolean	(optional) If true, causes the array to be reverse sorted with respect to nSortBy.
nFilterBy	Number	(optional) Gets only annotations satisfying certain criteria. Values are: ANFB_ShouldNone (default) Get all annotations. Equivalent of not specifying this parameter. ANFB_ShouldPrint Only include annotations that can be printed. ANFB_ShouldView Only include annotations that can be viewed. ANFB_ShouldEdit Only include annotations that can be edited. ANFB_ShouldAppearInPanel Only annotations that appear in the annotations pane. ANFB_ShouldSummarize Only include annotations that can be included in a summary. ANFB_ShouldExport Only include annotations that can be included in an export.

Return Value

Array of Annotation or null

Description

Gets an array of Annotation objects satisfying specified criteria. See also `getAnnot`(nPage, cName) and `syncAnnotScan`().

10.13.3.29 getAnnots3D(nPage)

Parameters

Parameter	Type	Description
nPage	Number	The 0-based page number that contains the Annot3D objects.

Return Value

Array of Annot3D or undefined

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

This method returns an array of `Annot3D` objects for a page.

10.13.3.30 getDataObject(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the data object to obtain.

Return Value

Data

Description

Obtains a specific Data object. See also `dataObjects`, `createDataObject`(cName, cValue[, cMediaType, cCryptFilter]), `exportDataObject`(cName[, cDIPath, bAllowAuth, nLaunch]), `importDataObject`([cDIPath, cCryptFilter]), and `removeDataObject`(cName).

10.13.3.31 getDataObjectContents(cName[, bAllowAuth])

Parameters

Parameter	Type	Description
cName	String	The name associated with the Data object to get.
bAllowAuth	Boolean	(optional) The default value is false. If true, a dialog box is used to obtain user authorization. Authorization may be required if the data object was encrypted using encryptForRecipients. Authorization dialog boxes are allowed if bAllowAuth is true.

Return Value

ReadStream

Description

Allows access to the contents of the file attachment associated with a DataObject.

[A NotAllowedError is thrown and the method fails if it attempts to access the content of an embedded file attachment for which any of the following conditions is true (all file name extension matching is case-insensitive):

The attachment's file name extension is ".SWF". If a legitimate .SWF application or module run as part of a Rich Media Annotation or PDF Portfolio navigator is allowed access to the content bytes of .SWF embedded file attachments, it is possible that the legitimate .SWF will load a malicious .SWF.

The attachment's file name extension is ".GIF", ".JPG", ".JPEG", or ".PNG" and the first three bytes of its content have the header signature of a .SWF file ("FWS" or "CWS"). The reason for this security restriction is that the same ActionScriptflash.display.Loader class load() method that can be used to load GIF, JPEG, and PNG images can also be used to load a SWF file. If a malicious SWF file's extension has been altered to that of one of these image types, the SWF could be loaded.

Related objects, properties, and methods are dataObjects, getDataObject, openDataObject, createDataObject, importDataObject, setDataObjectContents, and removeDataObject, and the Data object.]

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.32 getField(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the field of interest.

Return Value

Field

Description

Maps a Field object in the PDF document to an ECMAScript variable.

For more information, see `Field` object.

10.13.3.33 getIcon(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the icon object to obtain.

Return Value

Icon or null

Description

Obtains a specific icon object. See also the `icons` property, the `addIcon`(cName, icon), `importIcon(cName[, cDIPath, nPage])`, and `removeIcon`(cName) methods, and the `Field` object methods `buttonGetIcon`([nFace]), `buttonImportIcon`([cPath, nPage]), and `buttonSetIcon`(oIcon[, nFace]).

10.13.3.34 getLegalWarnings(bExecute)

Parameters

Parameter	Type	Description
bExecute	Boolean	if true, will cause the file to be examined and all detected warnings will be returned. If false, the default value, the warnings that have been embedded in the file, along with the certifier's attestation (if any) will be returned.

Return Value

DocLegalWarning

Description

Returns the legal warnings for this document in the form of an object with entries for each warning that has been found in the document.

If `bExecute` is `true`, the property names correspond to PDF/SigQ level A violations listed below. If `bExecute` is `false`, refer to PDF Reference for a list of possible property names.

A `DocLegalWarning` object containing property names and values of legal warnings. The value of each entry is the number of occurrences of this warning in the document.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.34.1 DocLegalWarning object

The properties listed in Table 24: DocLegalWarning Properties describe the PDF/SigQ1-A Violations.

Table 24 — DocLegalWarning Properties

Property	Description
AlternateImages	Image XObject must not contain an alternate version.
Annotations	The document contains comments. The visual appearances of the comments may change based on external variables.
CatalogHasAA	The document contains hidden actions that may not be intended or known by the end user. Actions include ECMAScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
CatalogHasOpenAction	The document contains hidden actions that will be launched on open. These actions may not be intended or known by the end user. Actions include ECMAScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
DevDepGS_FL	The extended graphic state of the document uses the FL key. The key is a number that indicates how much flatness tolerance should exist when drawing objects.
DevDepGS_TR	The document uses a PDF transfer function that interprets and replaces color. For example, it could replace black with white.
DocHasCryptFilter	Some or all of the content is encrypted and the encryption method is not. For example, the document may be protected. The document contains streams encrypted using the crypt filter.
DocHasNonSigField	The document contains non-signature form fields. The visual appearance of such fields may change based on external variables.
DocHasPresentation	Presentations are not allowed since a presentation may contain animations or other elements that may change document appearance or behavior.
DocHasPSXObj	Visual elements may change based on external variables. For example, a logo may change color based on time or zoom level. No PostScript XObjects allowed.
DocHasXFA	XFA-based (dynamic forms) documents are not allowed since such forms could alter the document's appearance or behavior.
DynamicSigAP	The document contains signed signature fields that may change their visual appearance based on external variables.
ExternalOPIdicts	The document links to images not in the PDF file that are used as alternates. For example, an alternate, high resolution image might be specified for printing. Images and form XObject must not contain an OPI alternate version.
ExternalRefXObjects	Document links to images not in the PDF file. No external XObjects allowed.
ExternalStreams	Document contains external streams. The author has flagged some PDF bytes as a stream which may get data from an external source.
GoTo3DViewActions	The document contains Go To 3D View actions that may be used to change the document's visual appearance through manipulating 3D views without the user's knowledge.
GoToEHasF	The document links to external PDF documents on the Internet, file system, or network and it has no control over the nature of that linked content. Embedded Go To actions must not refer to external hierarchies.
GoToRemoteAction	The document contains Go To actions that may link to external content.
InvalidEOF	The PDF file contains extra bytes after the PDF's end of file marker.
InvalidFileHeader	The PDF file contains extra bytes before the PDF's file header.
ECMAScriptActions	The document contains ECMAScript actions that may be launched without the user's knowledge.
LaunchActions	The document contains Launch File Attachment actions.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 24 *(continued)*

Property	Description
MalformedContentStm	Malformed drawing instructions: Syntax error. The page content violates the grammar for page content definition. For example, the instruction might specify drawing a square but the syntax for doing it is incorrect.
MovieActions	The document contains Launch Movie actions that may be launched without the user's knowledge.
NonEmbeddedFonts	Document contains non-embedded fonts. When the document opens on a system that does not have the requisite fonts, a PDF Processor should replace them with some other font. Users should always turn on font-related warnings.
OptionalContent	The content of the document is divided into layers that can be silently displayed or hidden on the fly.
PageHasAA	A page contains hidden actions that may not be intended or known by the end user. Actions include ECMAScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
RenditionActions	The document contains rendition actions that may be used to launch movies without the user's knowledge.
SetOCStateActions	The document contains SetOCState actions that may be used to change the document's visual appearance by modifying layers' visibility without the user's knowledge.
SigFieldHasAA	A signature field contains actions that could be invoked by mouse over or other user interaction. Actions include ECMAScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
SigFieldHasAction	A signature field contains actions that could be invoked by clicking. Actions include ECMAScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
SoundActions	The document contains launch sound actions.
TrueTypeFonts	This document uses TrueType fonts. TrueType and TrueType-based OpenType fonts are not allowed because they are programs and may change the document's appearance based on external variables. This restriction is not required by PDF/SigQ and is not reported unless the preference setting security\DigSig\bTrueTypeFontPDFSigQWarn is set to 1.
UnknownNamedAction	The document contains named actions that may launch menu items without the user's knowledge.
UnknownPDFContent	Unrecognized PDF content: The document contains PDF content or custom content not supported by the PDF Processor.
UnknownPDFContentStmOp	Unrecognized drawing operator: The document contains PDF content or custom content not supported by the PDF Processor.
URIActions	The document contains Launch URI actions that links to external content.
XObjHasInterpolate	The document author has enabled image interpolation. No image interpolation is allowed.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

10.13.3.35 getLinks(nPage, oCoords)

Parameters

Parameter	Type	Description
nPage	Number	The page that contains the Link objects. The first page is 0.
oCoords	Object	An array of four numbers in rotated user space, the coordinates of a rectangle listed in the following order: upper-left x, upper-left y, lower-right x and lower-right y.

Return Value

Array of Link

© ISO 2020 – All rights reserved

Description

Gets an array of Link objects that are enclosed within specified coordinates on a page. See also `addLink`(nPage, oCoords) and `removeLinks`(nPage, oCoords).

10.13.3.36 getNthFieldName(nIndex)

Parameters

Parameter	Type	Description
nIndex	Number	The index of the field whose name should be obtained.

Return Value

String

Description

Gets the name of the nth field in the document. See also `numFields`.

10.13.3.37 getNthTemplate(nIndex)

Parameters

Parameter	Type	Description
nIndex	Number	The index of the template to obtain.

Return Value

String

Description

NOTE This method is superseded by the templates property, the getTemplate method, and the Template object.

Gets the name of the nth template within the document.

10.13.3.38 getOCGs([nPage])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based page number. If not specified, all the OCGs found in the document are returned. If no argument is passed, returns all OCGs listed in alphabetical order, by name. If nPage is passed, this method returns the OCGs for that page, in the order they were created.

Return Value

Array of OCG

Description

Gets an array of `OCG` objects found on a specified page.

10.13.3.39 getOCGOrder()

Parameters

None

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Return Value

Array or null

Description

Returns this document's `OCGOrder` array. This array represents how layers are displayed in the UI.

Related methods are `getOCGs` and `setOCGOrder`, and the `OCG` object.

Returns

An array containing OCG objects, strings, and similar subarrays, or null if no OCGs are present.

See `setOCGOrder`(oOrderArray) for a description of the order array.

10.13.3.40 getPageBox([cBox, nPage])

Parameters

Parameter	Type	Description
cBox	String	(optional) The type of box. Values are : Art Bleed BBox Crop Trim The default is Crop. For definitions of these boxes see ISO 32000-2.
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.

Return Value

rectangle

Description

Gets a rectangle in rotated user space that encompasses the named box for the page. See also `setPageBoxes`([cBox, nStart, nEnd, rBox]).

10.13.3.41 getPageLabel([nPage])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.

Return Value

Array

Description

Gets page label information for the specified page. If nPage is not passed, implementers shall return the information for the first page of the document.

See `setPageLabels`([nPage, aLabel]) for a description of the contents of the array returned.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.42 getPageNthWord([nPage, nWord, bStrip])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.
nWord	Number	(optional) The 0-based index of the word. The default is 0, the first word on the page.
bStrip	Boolean	(optional) Specifies whether punctuation and white space should be removed from the word before returning. The default is true.

Return Value

String

Description

`[Security]`
Gets the nth word on the page.

See also **`getPageNumWords(nPage)`** and **`selectPageNthWord([nPage, nWord, bScroll])`**.

Implementers shall throw an exception if the document security is set to prevent content extraction.

10.13.3.43 getPageNthWordQuads([nPage, nWord])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.
nWord	Number	(optional) The 0-based index of the word. The default is 0, the first word on the page.

Return Value

Quads list

Description

`[Security]`
Gets the quads list for the nth word on the page. The quads property of the Annotation object can be used for constructing text markup, underline, strikeout, highlight and squiggly annotations. See also **`getPageNthWord`**([nPage, nWord, bStrip]), **`getPageNumWords(nPage)`**, and **`selectPageNthWord([nPage, nWord, bScroll])`**.

NOTE This method throws an exception if the document security is set to prevent content extraction.

10.13.3.44 getPageNumWords(nPage)

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.

Return Value

Integer

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Gets the number of words on the page.

See also `getPageNthWord`([nPage, nWord, bStrip]), `getPageNthWordQuads`([nPage, nWord]), and `selectPageNthWord`([nPage, nWord, bScroll]).

10.13.3.45 getPageRotation([nPage])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.

Return Value

Integer

Description

Gets the rotation of the specified page. See also `setPageRotations`([nStart, nEnd, nRotate]). The return value should be a value of *0*, *90*, *180*, or *270*.

10.13.3.46 getPageTransition([nPage])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page. The default is 0, the first page in the document.

Return Value

Array

Description

Gets the transition of the specified page. See also `setPageTransitions`([nStart, nEnd, aTrans]).

The array returned shall be composed of three values: `[ nDuration, cTransition, nTransDuration ]`.

`nDuration` is the maximum amount of time the page is displayed before the viewer automatically turns to the next page. A duration of –1 indicates that there is no automatic page turning.

`cTransition` is the name of the transition to apply to the page. See the property `app.fs.transitions` for a list of valid transitions.

`cTransDuration` is the duration (in seconds) of the transition effect.

10.13.3.47 getPrintParams()

Parameters

None

Return Value

PrintParams

Description

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Gets a PrintParams object that reflects the default print settings. See the print([bUI, nStart, nEnd, bSilent, bShrinkToFit, bPrintAsImage, bReverse, bAnnotations, printParams])method, which now takes the **printParams** object as its parameter.

10.13.3.48 getTemplate(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the template to retrieve.

Return Value

Template or null

Description

Gets the named template from the document. If the named template does not exist in the document, then implementations shall return `null`. See also `templates`, `createTemplate` (cName, nPage), `removeTemplate`(cName), and the `Template` object.

10.13.3.49 gotoNamedDest(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the destination within a document.

Return Value

undefined

Description

Goes to a named destination within the PDF document. For details on named destinations and how to create them, see ISO 32000-2.

10.13.3.50 importAnFDF([cPath])

Parameters

Parameter	Type	Description
cPath (optional)	String	The device-independent path to the FDF file. It should look like the value of the F entry in an FDF file exported with the submitForm method or with the Forms > Manage Form Data > Export Data From Form menu item. The path may be relative to the location of the current document. If this parameter is omitted, a dialog box is shown to let the user select the file.

Return Value

undefined

Description

Imports the specified FDF file. See also `importAnXFDF`([cPath]) and `importTextData([cPath, nRow])`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.51 importAnXFDF([cPath])

Parameters

Parameter	Type	Description
cPath	String	(optional) The device-independent path to the XFDF file. The path may be relative to the location of the current document. If the parameter is omitted, a dialog box is shown to let the user select the file.

Return Value

undefined

Description

Imports the specified XFDF file containing XML form data.

XFDF is an XML representation of PDFform data. See ISO 19444-1, the XML Form Data Format (XFDF) Specification for more information.

See also **`exportAsXFDF`**([bAllFields, bNoPassword, aFields, cPath, bAnnotations]), **`importAnFDF`**([cPath]) and **`importTextData`**([cPath, nRow]).

10.13.3.52 importDataObject([cDIPath, cCryptFilter])

Parameters

Parameter	Type	Description
cName	String	The name to associate with the data object.
cDIPath	String	(optional) A device-independent path to a data file on the user's hard drive. This path may be absolute or relative to the current document. If not specified, the user is prompted to locate a data file.
cCryptFilter	String	(optional) The language-independent name of a crypt filter to use when encrypting this data object. This crypt filter must have previously been added to the document's list of crypt filters, using the Doc addRecipientListCryptFilter method, otherwise an exception will be thrown. To leave this data object unencrypted in a file that is encrypted by the Doc encryptForRecipients method, the predefined Identity crypt filter can be used.

Return Value

Boolean

Description

`[Security]`

Imports an external file into the document and associates the specified name with the data object. Data objects can later be extracted or manipulated.

Related objects, properties, and methods are **`dataObjects`**, **`getDataObject`**, **`openDataObject`**, **`createDataObject`**, **`exportDataObject`**, **`importDataObject`**, **`getDataObjectContents`**, and **`setDataObjectContents`**, and the **`Data`** object.

When a file attachment is imported using **`importDataObject`**, the value of its **`Data.name`** is assigned by the parameter **`cName`**. However, when a file is attached using the UI, its name is automatically assigned. The attachments are assigned the sequential names "*`Untitled Object`*", "*`Untitled Object 2`*", "*`Untitled Object 3`*", and so on.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Implementations shall return true if the data object is successfully imported, or shall throw an exception if the data object cannot be imported.

10.13.3.53 importIcon(cName[, cDIPath, nPage])

Parameters

Parameter	Type	Description
cName	String	The name to associate with the icon.
cDIPath	String	(optional) A device-independent path to a PDF file on the user's hard drive. This path may be absolute or relative to the current document. cDIPath may only be specified in a batch environment or from the console. If not specified, the nPage parameter is ignored and the user is prompted to locate a PDF file and browse to a particular page.
nPage	Number	(optional) The 0-based index of the page in the PDF file to import as an icon. The default is 0.

Return Value

Integer

Description

`[Security]`

Imports an `icon` into the document and associates it with the specified name.

See also `icons`, `addIcon`(cName, icon), `getIcon`(cName), `removeIcon`(cName), the `Field` object methods `buttonGetIcon`([nFace]), `buttonImportIcon`([cPath, nPage]), `buttonSetIcon(oIcon[, nFace])`, and the `Icon` object.

The return value shall be an integer code indicating whether the import operation was successful or not:

0 No error

1 The user cancelled the dialog box

–1 The selected file could not be opened

–2 The selected page was invalid

10.13.3.54 importTextData([cPath, nRow])

Parameters

Parameter	Type	Description
cPath	String	(optional) A relative device-independent path to the text file. If not specified, the user is prompted to locate the text data file.
nRow	Number	(optional) The 0-based index of the row of the data to import, not counting the header row. If not specified, the user is prompted to select the row to import.

Return Value

The return value shall be an integer return code from Table 25: Doc.importTextData return values.

Table 25 — Doc.importTextData return values

Return code	Description
–3	Warning: Missing Data

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 25 *(continued)*

Return code	Description
-2	Warning: User Cancelled Row Select
-1	Warning: User Cancelled File Select
0	No Error
1	Error: Cannot Open File
2	Error: Cannot Load Data
3	Error: Invalid Row

Description

`[Security]`

Imports a row of data from a text file. Each row must be tab delimited. The entries in the first row of the text file are the column names of the tab delimited data. These names are also field names for text fields present in the PDF file. The data row numbers are 0-based; that is, the first row of data is row zero (this does not include the column name row). When a row of data is imported, each column datum becomes the field value of the field that corresponds to the column to which the data belongs.

See also the **`export version`** of **`this`** method, **`exportAsText`**([bNoPassword, aFields, cPath]).

10.13.3.55 mailDoc(bUI[, cTo, cCc, cBcc, cSubject, cMsg])

Parameters

Parameter	Type	Description
bUI	Boolean	If true (the default), the rest of the parameters are used in a compose-new-message window that is displayed to the user. If false, the cTo parameter is required and all others are optional. [NOTE When this method is executed in a non-privileged context, the bUI parameter is ignored and defaults to true. See Privileged versus non-privileged context.]
cTo	String	(required if bUI is false) The semicolon-delimited list of recipients for the message.
cCc	String	(optional) The semicolon-delimited list of CC recipients for the message.
cBcc	String	(optional) The semicolon-delimited list of BCC recipients for the message.
cSubject	String	(optional) The subject of the message. The length limit is 64 KB.
cMsg	String	(optional) The content of the message. The length limit is 64 KB.

Return Value

Undefined

Description

Saves the current PDF document and mails it as an attachment to all recipients, with or without user interaction.

See also **`mailForm`**(bUI[, cTo, cCc, cBcc, cSubject, cMsg]), and **`mailMsg`**(**`bUI`**, cTo[, cCc, cBcc, cSubject, cMsg]).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.13.3.56 mailForm(bUI[, cTo, cCc, cBcc, cSubject, cMsg])

Parameters

Parameter	Type	Description
bUI	Boolean	If true, the rest of the parameters are used in a compose-new-message window that is displayed to the user. If false, the cTo parameter is required and all others are optional. [NOTE When this method is executed in a non-privileged context, the bUI parameter is not honored and defaults to true. See Privileged versus non-privileged context.
cTo	String	(required if bUI is false) A semicolon-delimited list of recipients for the message.
cCc	String	(optional) A semicolon-delimited list of CC recipients for the message.
cBcc	String	(optional) A semicolon-delimited list of BCC recipients for the message.
cSubject	String	(optional) The subject of the message. The length limit is 64 KB.
cMsg	String	(optional) The content of the message. The length limit is 64 KB.

Return Value

undefined

Description

Exports the form data and mails the resulting FDF file as an attachment to all recipients, with or without user interaction. The method does not support signed signature fields.

See also `mailDoc`(bUI[, cTo, cCc, cBcc, cSubject, cMsg]), and `app.mailMsg(bUI, cTo[, cCc, cBcc, cSubject, cMsg])`.

10.13.3.57 movePage([nPage, nAfter])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page to move. The default is 0.
nAfter	Number	(optional) The 0-based index of the page after which to move the specified page. Use –1 to move the page before the first page of the document. The default is the last page in the document.

Return Value

undefined

Description

Moves a page within the document.

10.13.3.58 newPage([nPage, nWidth, nHeight])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The page after which to add the new page in a 1-based page numbering system. The default is the last page of the document. Use 0 to add a page before the first page. An invalid page range is truncated to the valid range of pages.
nWidth	Number	(optional) The width of the page in points. The default value is 612.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
nHeight	Number	(optional) The height of the page in points. The default value is 792.

Return Value

undefined

Description

`[Security]`

Adds a new page to the active document.

10.13.3.59 openDataObject(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the data object.

Return Value

Doc

Description

Returns the `Doc` of a PDF document that is an embedded `data` object (an attachment) within the document that this method is being called for.

The method shall throw an exception instead of returning a `Doc` if any of the following conditions are true.

— The document that this method is being called for does not contain the requested embedded data object.

— The data object is not a PDF document.

— Permissions forbid opening attachments by means of ECMAScript.

The document should be closed (using `closeDoc`) after it is no longer needed.

The name of a data object is a property of the Data object. A name is given to the object when it is embedded, automatically by a PDF Processor, or programmatically by the ECMAScript methods `createDataObject` or `importDataObject`.

Related objects, properties, and methods are `dataObjects`, `getDataObjectContents`, `setDataObjectContents`, `createDataObject`, and `importDataObject`, and the `Data` object.

10.13.3.60 print([bUI, nStart, nEnd, bSilent, bShrinkToFit, bPrintAsImage, bReverse, bAnnotations, printParams])

Parameters

Parameter	Type	Description
bUI	Boolean	(optional) If true (the default), will cause a UI to be presented to the user to obtain printing information and confirm the action.
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages. If nStart and nEnd are not specified, all pages in the document are printed. If only nStart is specified, the range of pages is the single page specified by nStart. If nStart and nEnd parameters are used, bUI must be false.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive page range. If nStart and nEnd are not specified, all pages in the document are printed. If only nEnd is specified, the range of a pages is 0 to nEnd. If nStart and nEnd parameters are used, bUI must be false.
bSilent	Boolean	(optional) If true, suppresses the cancel dialog box while the document is printing. The default is false.
bShrinkToFit	Boolean	(optional) If true, the page is shrunk (if necessary) to fit within the imageable area of the printed page. If false, it is not. The default is false.
bPrintAsImage	Boolean	(optional) If true, print pages as an image. The default is false.
bReverse	Boolean	(optional) If true, print from nEnd to nStart. The default is false.
bAnnotations	Boolean	(optional) If true (the default), annotations are printed.
printParams	Object	(optional) The PrintParams object containing the settings to use for printing. If this parameter is passed, any other parameters are ignored.

Return Value

undefined

Description

`[Security]`

Prints all or a specific number of pages of the document.

10.13.3.61 removeDataObject(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the data object to remove.

Return Value

undefined

Description

Deletes the **`data`** object corresponding to the specified name from the document.

Related objects, properties, and methods are **`dataObjects`**, **`getDataObject`**, **`openDataObject`**, **`createDataObject`**, **`removeDataObject`**, **`importDataObject`**, **`getDataObjectContents`**, and **`setDataObjectContents`**, and the **`Data`** object.

The name of a data object is a property of the Data object. A name is given to the object when it is embedded, either automatically by a PDF Processor or programmatically by the ECMAScript methods createDataObject or importDataObject.

10.13.3.62 removeField(cName)

Parameters

Parameter	Type	Description
cName	String	The field name to remove.

Return Value

undefined

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Removes the specified field from the document. If the field appears on more than one page, all representations are removed.

10.13.3.63 removeIcon(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the icon to remove.

Return Value

undefined

Description

Removes the specified named icon from the document.

The name of the icon is a property of the `Icon` object. A name is given to the object either by importIcon, when the icon file is imported into the document, or by addIcon, which names an icon that is not in the document-level named icons tree.

See also `icons`, `addIcon`(cName, icon), `getIcon`(cName), and `importIcon`(cName[, cDIPath, nPage]), the `Field` object methods `buttonGetIcon`([nFace]), `buttonImportIcon`([cPath, nPage]), and `buttonSetIcon`(oIcon[, nFace]), and the `Icon` object.

10.13.3.64 removeLinks(nPage, oCoords)

Parameters

Parameter	Type	Description
nPage	Number	The 0-based index of the page from which to remove links.
oCoords	Object	An array of four numbers in rotated user space, the coordinates of a rectangle listed in the following order: upper-left x, upper-left y, lower-right x, and lower-right y.

Return Value

undefined

Description

If the user has permission to remove links from the document, removes all the links on the specified page within the specified coordinates.

Use `getLinks` to help count the number of links removed.

See also `addLink(nPage, oCoords)`, `getLinks`(nPage, oCoords), and the `Link` object.

10.13.3.65 removeRequirement(cType)

Parameters

Parameter	Type	Description
cType	String	The type of requirement to be removed. The types are described by the Requirements Enumerator object.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

undefined

Description

[Security]

Removes an existing requirement present in a PDF document. Removing a requirement frees PDF Processors from having to fulfill it to open the document. The document may not function properly if a requirement is removed.

10.13.3.66 removeScript(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the script to remove.

Return Value

undefined

Description

Removes a document-level ECMAScript, if permissions for script removal is granted.

10.13.3.67 removeTemplate(cName)

Parameters

Parameter	Type	Description
cName	String	The name of the script to remove.

Return Value

undefined

Description

[Security]

Removes the named template from the document.

The template name is a property of the `Template` object. A name is given to a template when it is created, either by a PDF Processor or by the ECMAScript method `getTemplate`.

See also `templates`, `createTemplate` (cName, nPage), `getTemplate`(cName), and the `Template` object.

10.13.3.68 removeThumbnails([nStart, nEnd])

Parameters

Parameter	Type	Description
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages. If nStart and nEnd are not specified, operates on all pages in the document. If only nStart is specified, the range of pages is the single page specified by nStart.
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive range of pages. If nStart and nEnd are not specified, operates on all pages in the document. If only nEnd is specified, the range of pages is 0 to nEnd.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

undefined

Description

Deletes thumbnails for the specified pages in the document. See also `addThumbnails`([nStart, nEnd]).

10.13.3.69 resetForm([aFields])

Parameters

Parameter	Type	Description
aFields	Array	(optional) An array specifying the fields to reset. If not present or null, all fields in the form are reset. You can include non-terminal fields in the array.

Return Value

Undefined

Description

Resets the field values within a document. Resetting a field causes it to take on its default value (which, in the case of text fields, is usually blank).

10.13.3.70 scroll(nX, nY)

Parameters

Parameter	Type	Description
nX	Number	The x coordinate for the point to scroll.
nY	Number	The y coordinate for the point to scroll.

Return Value

undefined

Description

Scrolls the specified point on the current page into the middle of the current view. These coordinates must be defined in rotated user space. See ISO 32000-2 for details.

10.13.3.71 selectPageNthWord([nPage, nWord, bScroll])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page to operate on. The default is 0, the first page in the document.
nWord	Number	(optional) The 0-based index of the word to obtain. The default is 0, the first word on the page.
bScroll	Boolean	(optional) Specifies whether to scroll the selected word into the view if it is not already viewable. The default is true.

Return Value

undefined

Description

Changes the current page number and selects the specified word on the page.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

See also **getPageNthWord**([nPage, nWord, bStrip]), **getPageNthWordQuads**([nPage, nWord]), and **getPageNumWords(nPage)**.

10.13.3.72 setAction(cTrigger, cScript)

Parameters

Parameter	Type	Description
cTrigger	String	The name of the trigger point to which to attach the action. Values are: — WillClose — WillSave — DidSave WillPrint DidPrint
cScript	String	The ECMAScript expression to be executed when the trigger is activated.

Return Value

undefined

Description

Sets the ECMAScript action of the document for a given trigger.

See also **addScript**(cName, cScript), **setPageAction**(nPage, cTrigger, cScript), the **Bookmark** object **setAction**(cTrigger, cScript) method, and the **Field** object **setAction**(cTrigger, cScript) method.

NOTE This method will overwrite any action already defined for the selected trigger.

10.13.3.73 setDataObjectContents(cName, oStream[, cCryptFilter])

Parameters

Parameter	Type	Description
cName	String	The name associated with the **Data** object that is to be replaced with *oStream*.
oStream	Object	A **ReadStream** object representing the contents of the file attachment.
cCryptFilter		(optional) The language-independent name of a crypt filter to use when encrypting this data object.

Return Value

Undefined

Description

Replaces the file attachment specified by the parameter **cName** with the contents of the **oStream** parameter.

10.13.3.74 setOCGOrder(oOrderArray)

Parameters

Parameter	Type	Description
oOrderArray	Object	The array to be used as this document's **OCG Order** array.

Return Value

undefined

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Sets this document's `OCGOrder` array. This array represents how layers are displayed in the UI.

The simplest order array is a flat array of `OCG` objects. In this case, the listed `OCGs` are displayed in the UI as a flat list in the same order. If a subarray is present in the order array and the first element of the array is a string, the string will be listed with the rest of the array nested underneath it. If the first element of the array is not a string, the entire array will appear nested underneath the OCG preceding the subarray.

Related methods are `getOCGs` and `getOCGOrder`, and the `OCG` object.

10.13.3.75 setPageAction(nPage, cTrigger, cScript)

Parameters

Parameter	Type	Description
cTrigger	String	The trigger for the action. Values are: ***Open*** ***Close***
cScript	String	The ECMAScript expression to be executed when the trigger is activated.

Return Value

undefined

Description

Sets the action of a page in a document for a given trigger.

See also `setAction`, `addScript`, the `Bookmark` object `setAction` method, and the `Field` object `setAction` method.

NOTE This method will overwrite any action already defined for the chosen page and trigger.

10.13.3.76 setPageBoxes([cBox, nStart, nEnd, rBox])

Parameters

Parameter	Type	Description
cBox	String	(optional) The box type value, one of : ***Art*** ***Bleed*** ***Crop*** ***Media Trim*** Note that the `BBox` box type is read-only and only supported in getPageBox. For definitions of these boxes, see ISO 32000-2.
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages in the document to be operated on. If `nStart` and `nEnd` are not specified, operates on all pages in the document. If only `nStart` is specified, the range of pages is the single page specified by `nStart`.
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive range of pages in the document to be operated on. If `nStart` and `nEnd` are not specified, operates on all pages in the document.
rBox	Number	(optional) An array of four numbers in rotated user space to which to set the specified box. If not provided, the specified box is removed.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Return Value

undefined

Description

Sets a rectangle that encompasses the named box for the specified pages.

See also `getPageBox`([cBox, nPage]).

10.13.3.77 setPageLabels([nPage, aLabel])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index for the page to be labeled.
aLabel	Array	(optional) An array of three required items [`cStyle`, `cPrefix`, `nStart`]: `cStyle` is the style of page numbering. It can be: *D* decimal numbering *R or r* roman numbering, upper or lower case *A or a* alphabetic numbering, upper or lower case See the PDF Reference for the exact definitions of these styles. *cPrefix* a string to prefix the numeric portion of the page label. *nStart* the ordinal with which to start numbering the pages. If not supplied, any page numbering is removed for the specified page and any others up to the next specified label. The value of `aLabel` cannot be *null*.

Return Value

undefined

Description

Establishes the numbering scheme for the specified page and all pages following it until the next page with an attached label is encountered.

See also `getPageLabel`([nPage]).

10.13.3.78 setPageRotations([nStart, nEnd, nRotate])

Parameters

Parameter	Type	Description
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages in the document to be operated on. If `nStart` and `nEnd` are not specified, operates on all pages in the document. If only `nStart` is specified, the range of pages is the single page specified by `nStart`.
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive range of pages in the document to be operated on. If `nStart` and `nEnd` are not specified, operates on all pages in the document. If only `nEnd` is specified, the range of pages is 0 to nEnd.
nRotate	Number	(optional) The amount of rotation that should be applied to the target pages. Can be *0*, *90*, *180*, or *270*. The default is *0*.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

undefined

Description

Rotates the specified pages in the current document.

See also `getPageRotation`([nPage]).

10.13.3.79 setPageTabOrder(nPage, cOrder)

Parameters

Parameter	Type	Description
nPage	Number	The 0-based index of the page number on which the tabbing order is to be set.
cOrder	String	The order to be used. Values are: ***rows*** ***columns*** ***structure***

Return Value

Undefined

Description

Sets the tab order of the form fields on a page. The tab order can be set by row, by column, or by structure.

10.13.3.80 setPageTransitions([nStart, nEnd, aTrans])

Parameters

Parameter	Type	Description
nStart	Number	(optional) A 0-based index that defines the start of an inclusive range of pages in the document to be operated on. If nStart and nEnd are not specified, operates on all pages in the document. If only nStart is specified, the range of pages is the single page specified by nStart.
nEnd	Number	(optional) A 0-based index that defines the end of an inclusive range of pages in the document to be operated on. If nStart and nEnd are not specified, operates on all pages in the document. If only nEnd is specified, the range of pages is 0 to nEnd.
aTrans	Array	(optional) The page transition array consists of three values : **`nDuration`** **`cTransition`** **`nTransDuration`** **`nDuration`** is the maximum amount of time the page is displayed before the viewer automatically turns to the next page. Set to -1 to turn off automatic page turning. **`cTransition`** is the name of the transition to apply to the page. See fullScreen.transitions for a list of valid transitions. **`nTransDuration`** is the duration (in seconds) of the transition effect. If aTrans is not present, any page transitions for the pages are removed.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

undefined

Description

Sets the page transition for a specific range of pages.

See also `getPageTransition`([nPage]).

10.13.3.81 spawnPageFromTemplate(cTemplate[, nPage, bRename, bOverlay, oXObject])

Parameters

Parameter	Type	Description
cTemplate	String	The template name.
nPage	Number	(optional) The 0-based page number before which or into which the template is spawned, depending on the value of `bOverlay`. If `nPage` is omitted, a new page is created at the end of the document.
bRename	Boolean	(optional) Specifies whether fields should be renamed. The default is ***true***.
bOverlay	Boolean	(optional) If ***false***, the template is inserted before the page specified by `nPage`. If ***true*** (the default) it is overlaid on top of that page.
oXObject	Object	(optional) The value of this parameter is the return value of an earlier call to `spawnPageFromTemplate`.

Return Value

Object

Description

NOTE This method has been superseded by templates, createTemplate, and the Template object spawn method.

Spawns a page in the document using the given template, as returned by `getNthTemplate`.

This method returns an object representing the page contents of the page spawned. This return object can then be used as the value of the optional parameter `oXObject` for subsequent calls to `spawnPageFromTemplate`.

10.13.3.82 submitForm(cURL[, bFDF, bEmpty, aFields, bGet, bAnnotations, bXML, bIncrChanges, bPDF, bCanonical, bExclNonUserAnnots, bExclFKey, cPassword, bEmbedForm, oEcmaScript, cSubmitAs, bInclNMKey, aPackets, cCharset, oXML, cPermID, cInstID, cUsageRights)

Parameters

Parameter	Type	Description
cURL	String	The URL to submit to. This string must end in #FDF if the result from the submission is FDF or XFDF (that is, the value of cSubmitAs is "***FDF***" or "***XFDF***") and the document is being viewed inside a browser window.
bFDF	Boolean	(optional) If ***true*** (the default) form data is submitted as FDF. If ***false***, it is submitted as URL-encoded HTML. [NOTE This option has been deprecated; use cSubmitAs instead.
bEmpty	Boolean	(optional) If ***true***, submit all fields, including those that have no value. If ***false*** (the default), exclude fields that currently have no value. [NOTE If data is submitted as XDP, XML, or XFD (see the `cSubmitAs` parameter, below), this parameter is ignored. All fields are submitted, even fields that are empty. See `aFields`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
aFields	Array	(optional) An array of field names to submit or a string containing a single field name: If supplied, only the fields indicated are submitted, except those excluded by **bEmpty**. If omitted or ***null***, all fields are submitted, except those excluded by **bEmpty**. If an empty array, no fields are submitted. A submitted FDF file might still contain data if **bAnnotations** is true. You can specify non-terminal field names to export an entire subtree of fields. [NOTE If data is submitted as XDP, XML, or XFD (see the **cSubmitAs** parameter), this parameter is ignored. All fields are submitted, even fields that are empty. See **bEmpty**.
bGet	Boolean	(optional) If *true*, submit using the HTTP GET method. If *false* (the default), use a POST. GET is only allowed if the data is sent as HTML (that is, cSubmitAs is HTML).
bAnnotations	Boolean	(optional) If *true*, annotations are included in the submitted FDF or XML file. The default is ***false***. Only applicable if **cSubmitAs** is ***FDF*** or ***XFDF***.
bIncrChanges	Boolean	(optional) If true, include the incremental changes to the PDF document in the submitted FDF file. The default is false. Only applicable if cSubmitAs is FDF.
bCanonical	Boolean	(optional) If ***true***, convert any dates being submitted to standard format (that is, D:YYYYMMDDHHmmSSOHH'mm'; see ISO 32000-2). The default is ***false***.
bExclNonUserAnnots	Boolean	(optional) If ***true***, exclude any annotations that are not owned by the current user. The default is ***false***.
bExclFKey	Boolean	(optional) If ***true***, exclude the F entry. The default is ***false***.
cPassword	String	(optional) The password to use to generate the encryption key, if the FDF file needs to be encrypted before being submitted. Pass the value ***true*** (no quotes) to use the password that the user has previously entered for submitting or receiving an encrypted FDF file. If no password has been entered, the user is prompted to enter a password. Regardless of whether the password is passed in or requested from the user, this new password is remembered within a PDF Processor session for future outgoing or incoming encrypted FDF files. Only applicable if **cSubmitAs** is ***FDF***. Caution: You cannot create password-encrypted FDF files. If this parameter is used, a form trying to submit a password-encrypted FDF will throw an **ESErrorInvalidArgs** exception and the form submission will not occur.
bEmbedForm	Boolean	(optional) If *true*, the call embeds the entire form from which the data is being submitted in the FDF file. Only applicable if **cSubmitAs** is *FDF*.
oJavaScript	Ob ject	(optional) Can be used to include Before, After, and Doc scripts in a submitted FDF file. If present, the value is converted directly to an analogous **CosObj** and used as the ECMAScript attribute in the FDF file. Only applicable if **cSubmitAs** is ***FDF***.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
cSubmitAs	String	(optional) This parameter indicates the format for submission. Values are ***FDF*** Submit as FDF ***XFDF*** Submit as XFDF ***HTML*** Submit as HTML ***XDP*** Submit as XDP ***XML*** Submit as XML. Form data is submitted in XML format unless the parameter oXML contains a valid XMLData object, in which case that is what gets submitted instead. ***XFD*** cSubmitAs Submit as Form Client Data File ***PDF*** Submit the complete PDF document; all other parameters except cURL are ignored. The default is ***FDF***. This parameter supersedes the individual format parameters. However, they are considered in the following priority order, from high to low: ***cSubmitAs***, ***bPDF***, ***bXML***, ***bFDF***.
bInclNMKey	Boolean	(optional) If *true*, include the **NM** entry of any annotations. The default is ***false***.
aPackets	Array	(optional) An array of strings, specifying which packets to include in an XDP submission. This parameter is only applicable if cSubmitAs is XDP. Possible strings are: config datasets sourceSet stylesheet template ***pdf*** The PDF should be embedded; if pdf is not included, only a link to the PDF is included in the XDP. ***xfdf*** Include annotations in the XDP (since that packet uses XFDF format) ***** All packets should be included in the XDP. The default for pdf is to include it as a reference. To embed the PDF file in the XDP, explicitly specify pdf as one of the packets. The default is: ["***datasets***", "***xfdf***"].
cCharset	String	(optional) The encoding for the values submitted. String values are ***utf-8***, ***utf-16***, ***Shift-JIS***, ***BigFive***, ***GBK***, and ***UHC***. If not passed ***utf-8*** should be used. XFDF submission ignores this value and always uses ***utf-8***.
oXML	Object	(optional) This parameter is only applicable if **cSubmitAs** equals ***XML***. It should be an XMLData object, which will get submitted.
cPermID	Sting	(optional) Specifies a permanent ID to assign to the PDF that is submitted if either the value of **cSubmitAs** is *PDF* or **bEmbedForm** is ***true***. This permanent ID is the first entry in the **docID** array (docID[0]). Does not affect the current document.
cInstID	String	(optional) Specifies an instance ID to assign to the PDF that is submitted if either the value of **cSubmitAs** is ***PDF*** or **bEmbedForm** is ***true***. This instance ID is the second entry in the **docID** array (docID[1]). Does not affect the current document.
cUsageRights	String	(optional) Specifies the additional usage rights to be applied to the PDF that is submitted if either the value of **cSubmitAs** is ***PDF*** or **bEmbedForm** is ***true***. The only valid value is ***submitFormUsageRights.RMA***. Does not affect the current document.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

undefined

Description

Submits the form to a specified URL. To call this method, you must be running inside a web browser or have an applicable PDF Processor plug-in installed. (If the URL uses the “mailto” scheme, it will be honored even if not running inside a web browser, as long as a SendMail plug-in is present.)

The https protocol is supported for secure connections.

10.13.3.83 syncAnnotScan()

Parameters

None

Return Value

undefined

Description

Guarantees that all annotations will be scanned by the time this method returns.

To show or process annotations for the entire document, all annotations must have been detected. Normally, a background task runs that examines every page and looks for annotations during idle time, as this scan is a time-consuming task. Much of the annotation behavior works gracefully even when the full list of annotations is not yet acquired by background scanning.

In general, you should call this method if you want the entire list of annotations.

See also `getAnnots`([nPage, nSortBy, bReverse, nFilterBy]).

10.13.3.84 timestampSign([oSig, cDIPath, bUI])

Parameters

Parameter	Type	Description
oSig	Object	(optional) The signature engine object. If this parameter is not specified, the default (internal) signature engine is used.
cDIPath	String	(optional) The file path for saving the signed file. If this parameter is not specified, the file is saved over itself.
bUI	Boolean	(optional) Set TRUE to enable UI interaction. May be FALSE if a path is supplied. The default is FALSE.

Return Value

Boolean

Description

Adds an invisible timestamp to a document.

Returns ***TRUE*** if the signing was successful.

10.14 Embedded PDF

10.14.1 General

This object describes an API exposed to the object model of a container application that allows sending and receiving messages from an embedded PDF document. For example, when a PDF document is

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

embedded in an HTML document using the <OBJECT> tag, the PDF object is scripted in the browser scripting context.

The **HostContainer** object provides the corresponding interface in the PDF scripting model. Both the container and PDF document must explicitly allow this communication to occur for security reasons.

10.14.2 Embedded PDF properties

10.14.2.1 General

Table 26: Embedded PDF object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **Embedded PDF**.

Table 26 — Embedded PDF object properties

Property	Type	Access	Description
messageHandler	Object	R/W	This property allows a script running in the web browser scripting context to register a notification object that will be called if a script in the PDF document calls the Doc hostContainer.postMessage method.

10.14.2.1.1 messageHandler object details

The value of this property is an object that may expose the methods listed in Table 27: messageHandler methods.

Table 27 — messageHandler methods

Method	Description
onMessage	If present, this method will be called in response to the **Doc hostContainer.postMessage** method. The message is delivered asynchronously. The method is passed a single array parameter containing the array passed to the **postMessage** method.
onError	If present, this method will be called in response to an error. It is passed an Error object and an array of strings corresponding to the message that caused the error. If an error occurs and this property is undefined, the error will not be delivered (unlike messages, errors are not queued). The **name** property of the **Error** object will be set to one of the following strings: "***MessageGeneralError***": A general error occurred. "***MessageNotAllowedError***": The operation failed for security reasons. "***MessageDocNotDisclosedError***": The document has not been configured for disclosure to the host container. The hostContainer.messageHandler.**onDisclose** property of the **Doc** must be initialized correctly. "***MessageDocRefusedDisclosureError***": The document has refused to disclose itself to the host container based on the URL because the hostContainer.**messageHandler.onDisclose** method returned false.

When the methods are invoked, this object will be the **messageHandler** instance that the method is being called on. Properties on the ***messageHandler*** property that begin with on are reserved for future use as notification methods.

If the PDF document has had the **postMessage** method called on it prior to this method being registered, all of the queued messages will subsequently be passed to the **messageHandler** object once it is set.

Messages are guaranteed to be delivered in the order in which they are posted and errors are guaranteed to be delivered in the order in which they occurred. However, there is no correspondence between the delivery order of messages and errors.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Exceptions thrown from within the handler methods will be discarded. **Messages** and **errors** will not be delivered while inside an **onMessage** / **onError** handler.

10.14.3 Embedded PDF methods

10.14.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type **Embedded PDF**.

10.14.3.2 postMessage(aMessage)

Parameters

Parameter	Type	Description
aMessage	Array	An array of one or more strings that will be passed to onMessage.

Return Value

undefined

Description

Sends a message asynchronously to the PDF document message handler if the PDF document has disclosed itself by returning true from the onDisclosed method of the **HostContainer** object **messageHandler** property.

The message is passed to the **onMessage** method of the **messageHandler**.

If the PDF document does not disclose itself to the host container, an error will be passed to the **onError** method of the **messageHandler** property at some later point after **postMessage** has returned. If the PDF document has not registered to receive events by setting the **Doc hostContainer.messageHandler** property, the events will be queued until the PDF document sets the property.

The messages will be submitted to a queue of messages until they are delivered. If the queue size exceeds a maximum, an error will be thrown until some of the messages in the queue have been delivered.

10.15 Error

10.15.1 General

Error objects are dynamically created whenever an exception is thrown from methods or properties implemented in ECMAScript. Several subclasses of the Error object can be thrown by core. They all have the Error object as prototype. ECMAScript can throw some of these exceptions or implement subclasses of the Error object at its convenience. If your scripts are using the mechanism of try/catch error handling, the object thrown should be one of the types listed in Table 28: Error objects.

Table 28 — Error objects

Error object	Brief description
RangeError	Argument value is out of valid range.
TypeError	Wrong type of argument value.
ReferenceError	Reading a variable that does not exist.
MissingArgError	Missing required argument.
NumberOfArgsError	Invalid number of arguments to a method.
InvalidSetError	A property set is not valid or possible.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 28 *(continued)*

Error object	Brief description
`InvalidGetError`	A property get is not valid or possible.
`OutOfMemoryError`	Out of memory condition occurred.
`NotSupportedError`	Functionality not supported.
`NotAllowedError`	Method or property is not allowed for security reasons.
`GeneralError`	Unspecified error cause.
`RaiseError`	Internal error.
`DeadObjectError`	Object is dead.
`HelpError`	User requested for help with a method.

Error object types implemented by ECMAScript inherit properties and methods from the core Error object. Some ECMAScript objects may implement their own specific types of exception. A description of the Error subclass (with added methods and properties, if any) should be provided in the documentation for the particular object.

10.15.2 Error properties

Table 29: Error object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Error`.

Table 29 — Error object properties

Property	Type	Access	Description
fileName	String	R	The name of the script that caused the exception to be thrown.
lineNumber	Integer	R	The offending line number from where an exception was thrown in the ECMAScript code.
extMessage	String	R	A message providing additional details about the exception.
message	String	R	The error message providing details about the exception.
name	String	R/W	The name of the Error object subclass, indicating the type of the Error object instance.

10.15.3 Error methods

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type *Error*.

10.15.3.1 toString(...)

Parameters

None

Return Value

String

Description

Gets the error message that provides details about the exception.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.16 event

10.16.1 General

All ECMAScript scripts are executed as the result of a particular event. For each of these events, ECMAScript creates an event object. During the occurrence of each event, you can access this object to get and possibly manipulate information about the current state of the event.

Each event has a type and a name property that uniquely identify the event. This section describes all the events, listed as type/name pairs, and indicates which additional properties, if any, they define.

The `rc` property of an event is its return code. The description for each event describes whether it listens to (is affected by) the return code.

It is important for ECMAScript writers to know when these events occur and in what order they are processed. Some methods or properties can only be accessed during certain events.

For all mouse events, the term mouse denotes a generic pointing devices as defined in ISO 32000-2:2020, "12 Interactive features".

10.16.2 Event type/name combinations

10.16.2.1 App/Init

This event (the application initialization event) occurs when the viewer is started. Script files, called folder-level ECMAScripts, are read in from the application and user ECMAScript folders. They load in the following order: config.js, glob.js, all other files, then any user files.

This event does not listen to the `rc` return code.

10.16.2.2 Batch/Exec

This event occurs during the processing of each document of a batch sequence. Scripts authored as part of a batch sequence can access the event object upon execution.

The target for this event is the `Doc`.

This event listens to the `rc` return code. If `rc` is set to `false`, the batch sequence is stopped.

10.16.2.3 Bookmark/Mouse Up

This event occurs whenever a user clicks a bookmark that executes an ECMAScript.

The target for this event is the bookmark object that was clicked.

This event does not listen to the `rc` return code.

10.16.2.4 Console/Exec

This event occurs whenever a user evaluates a ECMAScript in the console.

This event does not listen to the `rc` return code.

10.16.2.5 Doc/DidPrint

This event is triggered after a document has printed.

The target for this event is the `Doc`.

This event does not listen to the `rc` return code.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.16.2.6 Doc/DidSave

This event is triggered after a document has been saved.

The target for this event is the `Doc`.

This event does not listen to the `rc` return code.

10.16.2.7 Doc/Open

This event is triggered whenever a document is opened. When a document is opened, the document-level script functions are scanned and any exposed scripts are executed.

The target for this event is the `Doc`. This event also defines the `targetName` property.

This event does not listen to the `rc` return code.

10.16.2.8 Doc/WillClose

This event is triggered before a document is closed.

The target for this event is the `Doc`.

This event does not listen to the `rc` return code.

10.16.2.9 Doc/WillPrint

This event is triggered before a document is printed.

The target for this event is the `Doc`.

This event does not listen to the `rc` return code.

10.16.2.10 Doc/WillSave

This event is triggered before a document is saved.

The target for this event is the `Doc`.

This event does not listen to the `rc` return code.

10.16.2.11 External/Exec

This event is the result of an external access, for example, through OLE, AppleScript, or loading an FDF.

This event does not listen to the `rc` return code.

10.16.2.12 Field/Blur

This event occurs after all other events just as a field loses focus. This event is generated regardless of whether a mouse click is used to deactivate the field (for example, a tab key could be used instead).

Additional properties defined:

`target`: The field whose validation script is being executed.

`modifier`, `shift`, `targetName`, and `value`.

This event does not listen to the `rc` return code.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.16.2.13 Field/Calculate

This event is defined when a change in a form requires that all fields that have a calculation script attached to them be executed. All fields that depend on the value of the changed field will now be recalculated. These fields may in turn generate additional Field/Validate, Field/Blur, and Field/Focus events.

Calculated fields may have dependencies on other calculated fields whose values must be determined beforehand. The calculation order array contains an ordered list of all the fields in a document that have a calculation script attached. When a full calculation is needed, each of the fields in the array is calculated in turn starting with the zeroth index of the array and continuing in sequence to the end of the array.

The target for this event is the field whose calculation script is being executed. This event also defines the **source** and **targetName** properties.

This event listens to the **rc** return code. If the return code is set to **false**, the field's value is not changed. If **true**, the field takes on the value found in the value property.

10.16.2.14 Field/Focus

This event occurs after the mouse-down event but before the mouse-up event after the field gains the focus. It occurs regardless of whether a mouse click is used to activate the field (or, for example, the tab key) and is the best place to perform processing that must be done before the user can interact with the field.

The target for this event is the field whose validation script is being executed. This event also defines the **modifier**, **shift**, and **targetName** properties.

This event does not listen to the **rc** return code.

10.16.2.15 Field/Format

This event is triggered once all dependent calculations have been performed. It allows the attached ECMAScript to change the way that the data value appears to a user (also known as its presentation or appearance). For example, if a data value is a number and the context in which it should be displayed is currency, the formatting script can add a dollar sign ($) to the front of the value and limit it to two decimal places past the decimal point.

The target for this event is the field whose format script is being executed. This event also defines the **commitKey**, **targetName**, and **willCommit** properties.

This event does not listen to the **rc** return code. However, the resulting value is used as the field's formatted appearance.

10.16.2.16 Field/Keystroke

This event occurs whenever a user types a keystroke into a text box or combo box (including cut and paste operations) or selects an item in a combo box list or list box field. A keystroke script may limit the type of keys allowed. For example, a numeric field might only allow numeric characters.

A PDF Processor user interface should allow the author to specify a Selection Change script for list boxes. The script is triggered every time an item is selected. This is implemented as the keystroke event where the keystroke value is equivalent to the user selection. This behavior is also implemented for the combo box — the "keystroke" could be thought to be a paste into the text field of the value selected from the drop-down list.

There is a final call to the keystroke script before the validate event is triggered. This call sets the willCommit to true for the event. With keystroke processing, it is sometimes useful to make a final check on the field value (pre-commit) before it is committed. This allows the script writer to gracefully handle particularly complex formats that can only be partially checked on a keystroke-by-keystroke basis.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

The keystroke event of text fields is called in situations other than when the user is entering text with the keyboard or committing the field value. It is also called to validate the default value of a field when set through the UI or by ECMAScript and to validate entries provided by autofill. In these situations not all properties of the event are defined. Specifically, **`event.target`** will be undefined when validating default values and **`event.richChange`** and **`event.richValue`** will be undefined when validating autofill entries.

The target for this event is the field whose keystroke script is being executed. This event also defines the **`commitKey`**, **`change`**, **`changeEx`**, **`keyDown`**, **`modifier`**, **`selEnd`**, **`selStart`**, **`shift`**, **`targetName`**, **`value`**, and **`willCommit`** properties.

This event listens to the **`rc`** return code. If set to **`false`**, the keystroke is ignored. The resulting change is used as the keystroke if the script wants to replace the keystroke code. The resultant **`selEnd`** and **`selStart`** properties can change the current text selection in the field.

10.16.2.17 Field/Mouse Down

This event is triggered when a user starts to click a form field and the mouse button is still down. A mouse-down event does not occur unless a mouse enter event has already occurred. It is advised that you perform very little processing during this event (for example, play a short sound).

The target for this event is the field whose validation script is being executed. This event also defines the **`modifier`**, **`shift`**, and targetName properties.

This event does not listen to the **`rc`** return code.

10.16.2.18 Field/Mouse Enter

This event is triggered when a user moves the pointer inside the rectangle of a field. This is the typical place to open a text field to display help text, for example.

The target for this event is the field whose validation script is being executed. This event also defines the **`modifier`**, **`shift`**, and **`targetName`** properties.

This event does not listen to the **`rc`** return code.

10.16.2.19 Field/Mouse Exit

This event occurs when a user moves the mouse pointer outside of the rectangle of a field. A mouse exit event will not occur unless a mouse enter event has already occurred.

The target for this event is the field whose validation script is being executed. This event also defines the **`modifier`**, **`shift`**, and **`targetName`** properties.

This event does not listen to the **`rc`** return code.

10.16.2.20 Field/Mouse Up

This event is triggered when the user clicks a form field and releases the mouse button. This is the typical place to attach routines such as the submit action of a form. A mouse-up event will not occur unless a mouse-down event has already occurred.

The target for this event is the field whose validation script is being executed. This event also defines the **`modifier`**, **`shift`**, and **`targetName`** properties.

This event does not listen to the **`rc`** return code.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.16.2.21 Field/Validate

Regardless of the field type, user interaction with a field may produce a new value for that field. After the user has either clicked outside a field, tabbed to another field, or pressed the enter key, the user is said to have committed the new data value.

This event is the first event generated for a field after the value has been committed so that an ECMAScript can verify that the value entered was correct. If the validate event is successful, the next event triggered is the calculate event.

The target for this event is the field whose validation script is being executed. This event also defines the **change**, **changeEx**, **keyDown**, **modifier**, **shift**, **targetName**, and **value** properties.

This event does not listen to the **rc** return code. If the return code is set to ***false***, the field value is considered to be invalid and the value of the field is unchanged.

10.16.2.22 Link/Mouse Up

This event is triggered when a link containing an ECMAScript action is activated by the user.

The target for this **event** is the **Doc**.

This event does not listen to the **rc** return code.

10.16.2.23 Menu/Exec

This event occurs whenever ECMAScript that has been attached to a menu item is executed.

The target for this event is the currently active document, if one is open. This event also defines the **targetName** property.

This event listens to the **rc** return code in the case of the enable and marked proc for menu items.A return code of ***false*** will disable or unmark a menu item. A return code of *true* will enable or mark a menu item.

10.16.2.24 Page/Open

This event occurs whenever a new page is viewed by the user and after page drawing for the page has occurred.

The target for this event is the **Doc**.

This event does not listen to the **rc** return code.

10.16.2.25 Page/Close

This event occurs whenever the page being viewed is no longer the current page; that is, the user switched to a new page or closed the document.

The target for this event is the **Doc**.

This event does not listen to the **rc** return code.

10.16.2.26 Screen/Blur

This event occurs after all other events, just as the screen annotation loses focus. This event is generated regardless of whether a mouse click is used to deactivate the screen annotation (for example, the tab key might be used).

The target for this event is the **ScreenAnnot** object that initiated this **event. targetName** is the title of the screen annotation. This event also defines the modifier and shift properties.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

This event does not listen to the **rc** return code.

10.16.2.27 Screen/Close

This event occurs whenever the page being viewed is no longer the current page; that is, the user switched to a new page or closed the document.

The target for this event is the **ScreenAnnot** object that initiated this event. targetName is the title of the screen annotation. This event also defines the **modifier**, **shift**, and **target** properties.

This event does not listen to the **rc** return code.

10.16.2.28 Screen/Focus

This event occurs after the mouse-down event but before the mouse-up after the field gains the focus. This routine is called regardless of whether a mouse click is used to activate the screen annotation (for example, the tab key might be used). It is the best place to perform processing that must be done before the user can interact with the field.

The target for this event is the **ScreenAnnot** object that initiated this **event.targetName** is the title of the screen annotation. This event also defines the **modifier** and **shift** properties.

This event does not listen to the **rc** return code.

10.16.2.29 Screen/InView

This event occurs whenever a new page first comes into view by the user. When the page layout is set to Continuous or Continuous - Facing, this event occurs before the Screen/Open event.

The target for this event is the **ScreenAnnot** object that initiated this event. targetName is the title of the screen annotation. This event also defines the **modifier** and **shift** properties.

This event does not listen to the **rc** return code.

10.16.2.30 Screen/Mouse Down

This event is triggered when a user starts to click a screen annotation and the mouse button is still down. It is advised that you perform very little processing (that is, play a short sound) during this event. A mouse-down event will not occur unless a mouse enter event has already occurred.

The target for this event is the **ScreenAnnot** object that initiated this event. targetName is the title of the screen annotation. This event also defines the **modifier** and **shift** properties.

This event does not listen to the **rc** return code.

10.16.2.31 Screen/Mouse Enter

This event is triggered when a user moves the mouse pointer inside the rectangle of a screen annotation.

The target for this event is the **ScreenAnnot** object that initiated this event. targetName is the title of the screen annotation. This event also defines the **modifier** and **shift** properties.

This event does not listen to the **rc** return code.

10.16.2.32 Screen/Mouse Exit

This event is the opposite of the **Mouse Enter** event and occurs when a user moves the mouse pointer outside of the rectangle of a screen annotation. A **Mouse Exit** event will not occur unless a **Mouse Enter** event has already occurred.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

The target for this event is the `ScreenAnnot` object that initiated this event. targetName is the title of the screen annotation. This event also defines the `modifier` and `shift` properties.

This event does not listen to the `rc` return code.

10.16.2.33 Screen/Mouse Up

This event is triggered when the user clicks a screen annotation and releases the mouse button. This is the typical place to attach routines such as starting a multimedia clip. A mouse-up event will not occur unless a mouse-down event has already occurred.

The target for this event is the ScreenAnnot object that initiated this event. targetName is the title of the screen annotation. This event also defines the `modifier` and `shift` properties.

This event does not listen to the `rc` return code.

10.16.2.34 Screen/Open

This event occurs whenever a new page is viewed by the user and after page drawing for the page has occurred.

The target for this event is the `ScreenAnnot` object that initiated this `event.targetName` is the title of the screen annotation. This event also defines the `modifier` and `shift` properties.

This event does not listen to the `rc` return code.

10.16.2.35 Screen/OutView

This event occurs whenever a page first goes out of view from the user. When the page layout is set to Continuous or Continuous - Facing, this event occurs after the Screen/Close event.

The target for this event is the `ScreenAnnot` object that initiated this `event.targetName` is the title of the screen annotation. This event also defines the `modifier` and `shift` properties.

This event does not listen to the `rc` return code.

10.16.3 Document Event Processing

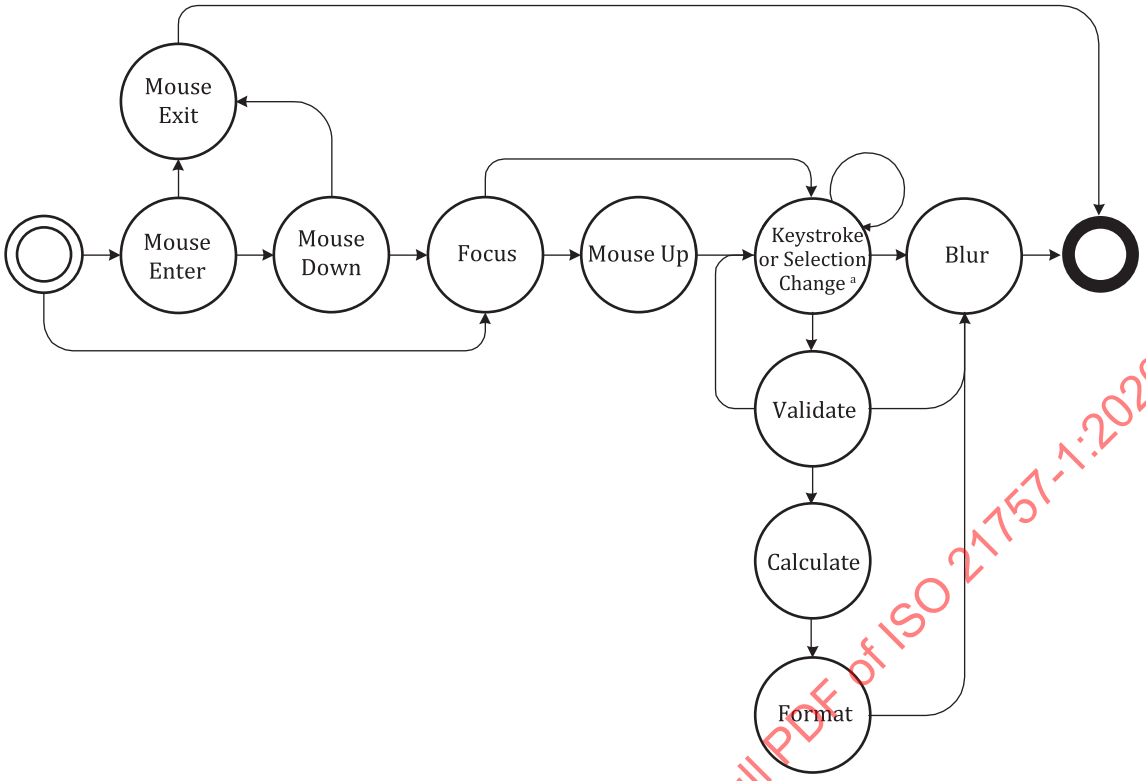
When a document is opened, the Doc/Open event occurs. Functions are scanned and any exposed (top-level) scripts are executed. Next, if the NeedAppearances entry in the PDF file is set to true in the AcroForm dictionary, the formatting scripts of all form fields in the document are executed (See ISO 32000-2). Finally, the Page/Close event occurs.

10.16.4 Form event processing

The order in which the form events occur is shown in the state diagram below. Certain dependencies are worth noting. For example, the mouse-up event cannot occur if the Focus event did not occur.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved



10.16.5 event properties

Table 30: event object propertieslists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **event**.

Table 30 — event object properties

Property	Type	Access	Description
change	String	R/W	A string specifying the change in value that the user has just typed. A ECMAScript may replace part or all of this string with different characters. The change may take the form of an individual keystroke or a string of characters (for example, if a paste into the field is performed).
changeEx	Various	R	Contains the export value of the change and is available only during a Field/Keystroke event for list boxes and combo boxes. For the list box, the keystroke script, if any, is entered under the Selection Change tab in the properties dialog box. For the combo box, changeEx is only available if the pop-up list is used—that is, a selection (with the mouse or the keyboard) is being made from the list. If the combo is editable and the user types in an entry, the Field/Keystroke event behaves as for a text field (that is, there are no changeEx or keyDown event properties). event.changeEx is defined for text fields. When event.fieldFull is true, changeEx is set to the entire text string the user attempted to enter and event.change is the text string cropped to what fits within the field. Use event.richChangeEx (and event.richChange) to handle rich text fields.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 30 *(continued)*

Property	Type	Access	Description

commitKey	Number	R	Determines how a form field will lose focus. Values are:
			0 — Value was not committed (for example, escape key was pressed).
			1 — Value was committed because of a click outside the field using the mouse.
			2 — Value was committed because of pressing the Enter key.
			3 — Value was committed by tabbing to a new field.
			EXAMPLE To automatically display an alert dialog box after a field has been committed, add the following to the field's format script:
			`if (event.commitKey != 0)`
			`app.alert("Thank you for your new field value.");`

fieldFull	Boolean	R	(Only available in **keystroke** events for text fields.) Set to true when the user attempts to enter text that does not fit in the field due to either a space limitation (the Field object property doNotScroll is set to true) or the maximum character limit (the Field object property charLimit is set to a positive value). When fieldFull is true, event.changeEx is set to the entire text string the user attempted to enter and event.change is the text string cropped to what fits within the field. Field properties generally cannot be changed during a keystroke event, so it is necessary for the field to lose focus as a way to commit the data. The user then has to reset the focus and continue entering data
keyDown	Boolean	R	(Available only during a **keystroke** event for list box and combo box.) For a list box or the pop-up part of a combo box, the value is true if the arrow keys were used to make a selection, false otherwise. For the combo box, keyDown is only available if the pop-up part of it is used, that is, a selection (with the mouse or the keyboard) is being made from the pop-up. If the combo is editable and the user types in an entry, the Field/Keystroke event behaves as for a text field (that is, there are no changeEx or keyDown event properties).
modifier	Boolean	R	Specifies whether the modifier key is down during a particular event. The modifier key on the Microsoft Windows platform is Control and on the Mac OS platform is Option or Command. This property is not supported on UNIX.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 30 *(continued)*

Property	Type	Access	Description
name	String	R	The name of the current event as a text string. The type and name together uniquely identify the event. Valid names are: ***Keystroke*** ***Validate*** ***Focus*** ***Blur*** ***Format*** ***Calculate*** ***Mouse Up*** ***Mouse Down*** ***Mouse Enter*** ***Mouse Exit*** ***WillPrint*** ***DidPrint*** ***WillSave*** ***DidSave*** ***Init*** ***Exec*** ***Open*** ***Close***
rc	Boolean	R/W	(Valid only during **keystroke**, **validate**, and **Menu** events.) Used for validation. Indicates whether a particular event in the event chain should succeed. Set to *false* to prevent a change from occurring or a value from committing. The default is *true*.
richChange	Array of Span	R/W	(Valid only for **Keystroke** events.) Specifies the change in value that the user has just typed. The **richChange** property is only defined for rich text fields and mirrors the behavior of the event.change property. The value of **richChange** is an array of **Span** objects that specify both the text entered into the field and the formatting. Keystrokes are represented as single member arrays, while rich text pasted into a field is represented as an array of arbitrary length. When **event.fieldFull** is true, **richChangeEx** is set to the entire rich formatted text string the user attempted to enter and **event.richChange** is the rich formatted text string cropped to what fits within the field. Use **event.changeEx** (and event.change) to handle (plain) text fields. Related objects and properties are **event.richValue**, the **Span** object, the **Field** object **defaultStyle**, **richText**, and **richValue** properties, and the **Annotation** object **richContents** property.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 30 *(continued)*

Property	Type	Access	Description
richChangeEx	Array	R/W	(Valid only for **Keystroke** events.) This property is only defined for rich text fields. It mirrors the behavior of the event.**changeEx** property for **text** fields. Its value is an array of **Span** objects that specify both the text entered into the field and the formatting. Keystrokes are represented as single member arrays, while rich text pasted into a field is represented as an array of arbitrary length. If **event.fieldFull** is true, **richChangeEx** is set to the entire rich formatted text string the user attempted to enter and **event.richChange** is the rich formatted text string cropped to what fits within the field. Use **event.changeEx** (and event.change) to handle (plain) text fields. Related objects and properties include event.**richChange**, **event.richValue**, the **Span** object, the **Field** object defaultStyle, **richText**, and **richValue** properties, and the **Annotation** object **richContents** property.
richValue	Array of Span objects	R/W	(Valid only for **Keystroke** events.) This property mirrors the **richValue** property of the **Field** object and the event.value property for each event. Related objects and properties include the **Span** object, the **Field** object properties **defaultStyle**, **richText**, **richValue**, **event.richChange**, **event.richChangeEx**, and the Annotation object **richContents** property.
selEnd	Integer	R/W	The ending position of the current text selection during a keystroke event.
selStart	Integer	R/W	The starting position of the current text selection during a keystroke event.
shift	Boolean	R	***true*** if the shift key is down during a particular event, ***false*** otherwise.
source	Object	R	The **Field** object that triggered the calculation event. This object is usually different from the target of the event, which is the field that is being calculated.
target	Object	R	The target object that triggered the event. In all **mouse**, **focus**, **blur**, **calculate**, **validate**, and **format** events, it is the **Field** object that triggered the event. In other events, such as **page open** and **close**, it is the **Doc** or this object.
targetName	String	R	Tries to return the name of the ECMAScript being executed. Can be used for debugging purposes to help identify the code causing exceptions to be thrown. Common values of targetName include: The folder-level script file name for App/Init events The document-level script name forDoc/Open events The PDF file name being processed for Batch/Exec events The field name for Field events The menu item name for Menu/Exec events The screen annotation name for Screen events (multimedia events) When an exception is thrown, **targetName** is reported if there is an identifiable name.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 30 *(continued)*

Property	Type	Access	Description
type	String	R	The type of the current event. The type and name together uniquely identify the event. Valid types are: ***Batch*** ***Console*** ***App*** ***Doc*** ***Page*** ***External*** ***Bookmark*** ***Link*** ***Field*** ***Menu***

value	Various	R/W	This property has different meanings for different field events:
			For the Field/Validate event, it is the value that the field contains when it is committed. For a combo box, it is the face value, not the export value (see **changeEx**).
			For example, the following ECMAScript verifies that the field value is between zero and 100:
			`if (event.value < 0 || event.value > 100) {`
			`app.beep(0);`
			`app.alert("Invalid value for field " + event.target.name);`
			`event.rc = false;`
			`}`
			For a Field/Calculate event, ECMAScript should set this property. It is the value that the field should take upon completion of the event.
			For example, the following ECMAScript sets the calculated value of the field to the value of the SubTotal field plus tax.
			`var f = this.getField("SubTotal");`
			`event.value = f.value * 1.0725;`
			For a Field/Format event, ECMAScript should set this property. It is the value used when generating the appearance for the field. By default, it contains the value that the user has committed. For a combo box, this is the face value, not the export value (see **changeEx** for the export value).
			For example, the following ECMAScript formats the field as a currency type of field.
			`event.value = util.printf("$%.2f", event.value);`

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 30 *(continued)*

Property	Type	Access	Description
			For a Field/Keystroke event, it is the current value of the field. If modifying a text field, for example, this is the text in the text field before the keystroke is applied. For Field/Blur and Field/Focus events, it is the current value of the field. During these two events, event.value is read only. That is, the field value cannot be changed by setting event.value. For a list box that allows multiple selections (see **`field.multipleSelection`**), the following behavior occurs. If the field value is an array (that is, multiple items are selected), event.value returns an empty string when getting, and does not accept setting.
willCommit	Boolean	R	Verifies the current keystroke event before the data is committed. It can be used to check target form field values to verify, for example, whether character data was entered instead of numeric data. ECMAScript sets this property to true after the last keystroke event and before the field is validated.

10.17 Field

10.17.1 General

This object represents a form field (that is, a field created using a form tool or the **`Doc addField`** method). In the same manner that a form author can modify an existing field's properties, such as the border color or font, the ECMAScript user can use the Field object to perform the same modifications.

Before a field can be accessed, it must be bound to a ECMAScript variable through a method provided by the Doc. More than one variable may be bound to a field by modifying the field's object properties or accessing its methods. This affects all variables bound to that field.

```
var f = this.getField("Total");
```

This example allows the script to manipulate the form field "Total" by means of the variable f.

Fields can be arranged hierarchically within a document. For example, form fields with names like "***`FirstName`***" and "***`LastName`***" are called flat names and there is no association between them. By changing the field names, a hierarchy of fields within the document can be created.

For example, "***`Name.First`***" and "***`Name.Last`***" forms a tree of fields. The period ('.') separator informs denotes a hierarchy shift. "***`Name`***" in these fields is the parent; "***`First`***" and "***`Last`***" are the children. Also, the field "***`Name`***" is an internal field because it has no visible appearance. "***`First`***" and "***`Last`***" are terminal fields that appear on the page.

Form fields that share the same name also share the same value. Terminal fields can have different presentations of that data. For example, they can appear on different pages, be rotated differently, or have a different font or background color, but they have the same value. Therefore, if the value of one presentation of a terminal field is modified, all others with the same name are updated automatically.

Each presentation of a terminal field is referred to as a widget. An individual widget does not have a name but is identified by index (0-based) within its terminal field. The index is determined by the order in which the individual widgets of this field were created (and is unaffected by tab-order).

`getField` can be used to retrieve the **`Field`** object of one individual widget of a field. This notation consists of appending a "." (period) followed by the widget index to the field name passed. When this approach is used, the Field object returned by **`getField`** encapsulates only one individual widget. You can use the **`Field`** objects returned this way anywhere you would use a **`Field`** object returned by passing

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

the unaltered field name. However, the set of nodes that are affected may vary, as shown in Table 31: Field actions.

Table 31 — Field actions

Action	Field object that represents all widgets	Field object that represents one specific widget
Get a widget property	Gets the property of widget # 0.	Gets the property of the widget.
Set a widget property	Sets the property of all widgets that are children of that field. (The rect property and the setFocus method are exceptions that apply to widget # 0. See example below.)	Sets the property of the widget.
Get a field property	Gets the property of the field.	Gets the property of the parent field.
Set a field property	Sets the property of the field.	Sets the property of the parent field.

The following example changes the rect property of the second radio button (the first would have index 0) of the field "***my radio***".

```
var f = this.getField("my radio.1");
f.rect = [360, 677, 392, 646];
```

10.17.2 Field versus widget attributes

Some properties of the Field object, such as value, apply to all widgets that are children of that field. Other properties, such as rect, are specific to individual widgets.

The following field properties and methods affect field-level attributes:

`calcOrderIndex`, **`charLimit`**, **`comb`**, **`currentValueIndices`**,**`defaultValue`**, **`doNotScroll`**, **`doNotSpellCheck`**, **`delay`**, **`doc`**, **`editable`**, **`exportValues`**, **`fileSelect`**, **`multiline`**, **`multipleSelection`**, **`name`**, **`numItems`**,**`page`**, **`password`**, **`readonly`**, **`required`**,**`submitName`**, **`type`**, **`userName`**, **`value`**, **`valueAsString`**, **`clearItems`**, **`browseForFileToSubmit`**, **`deleteItemAt`**, **`getItemAt`**, **`insertItemAt`**, **`setAction`**, **`setItems`**,**`signatureInfo`**, **`signatureSign`**, and **`signatureValidate`**.

The following field properties and methods affect widget-level attributes:

`alignment`, ***`borderStyle`***, ***`buttonAlignX`***, ***`buttonAlignY`***, ***`buttonPosition`***, ***`buttonScaleHow`***, ***`buttonScaleWhen`***, ***`display`***, ***`fillColor`***, ***`hidden`***, ***`highlight`***, ***`lineWidth`***, ***`print`***, ***`rect`***, ***`strokeColor`***, ***`style`***, ***`textColor`***, ***`textFont`***, ***`textSize`***, ***`buttonGetCaption`***, ***`buttonGetIcon`***, ***`buttonImportIcon`***, ***`buttonSetCaption`***, ***`buttonSetIcon`***, ***`checkThisBox`***, ***`defaultIsChecked`***, ***`isBoxChecked`***, ***`isDefaultChecked`***, ***`setAction`***, and ***`setFocus`***.

NOTE The setAction method can apply at the field or widget level, depending on the event. The **`Keystroke`**, **`Validate`**, **`Calculate`**, **`and Format events apply to fields. The MouseUp`**, **`MouseDown`**, **`MouseEnter`**, **`MouseExit`**, **`OnFocus`**, **`and OnBlur events apply to widgets.`**

The **`checkThisBox`**, **`defaultIsChecked`**, **`isBoxChecked`**, and **`isDefaultChecked`** methods take a widget index, **`nWidget`**, as a parameter. If you invoke these methods on a **`Field`** object f that represents one specific widget, the **`nWidget`** parameter is optional (and is ignored if passed) and the method acts on the specific widget encapsulated by **`f`**.

10.17.3 Field properties

10.17.3.1 General

In general, field properties correspond to those stored in field and annotation dictionaries in the PDF document (see ISO 32000-2).

Some property values are stored in the PDF document as names, while others are stored as strings (see ISO 32000-2). ISO 32000-2 documents all **`Annotation`** properties as well as how they are stored.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32: field object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `field`.

Table 32 — field object properties

<table>
<tr><th>Property</th><th></th><th></th><th colspan="3">Description</th></tr>
<tr><td>alignment</td><td>String</td><td>R/W</td><td colspan="3">(Valid for <code>text</code> fields.)
Controls how the text is laid out within the text field. Values are:
left center right</td></tr>
<tr><td>borderStyle</td><td>String</td><td>R/W</td><td colspan="3">The border style for a field. Valid border styles are
<code>solid</code>
<code>dashed</code>
<code>beveled</code>
<code>inset</code>
<code>underline</code>
The border style determines how the border for the rectangle is drawn. The border object is a static convenience constant that defines all the border styles of a field, as shown in the following table:
<table>
<tr><th>Type</th><th>Keyword</th><th>Description</th></tr>
<tr><td>solid</td><td>border.s</td><td>Strokes the entire perimeter of the rectangle with a solid line.</td></tr>
<tr><td>beveled</td><td>border.b</td><td>Equivalent to the solid style with an additional beveled (pushed-out appearance) border applied inside the solid border.</td></tr>
<tr><td>dashed</td><td>border.d</td><td>Strokes the perimeter with a dashed line.</td></tr>
<tr><td>Inset</td><td>border.i</td><td>Equivalent to the solid style with an additional inset (pushed-in appearance) border applied inside the solid border.</td></tr>
<tr><td>underline</td><td>border.u</td><td>Strokes the bottom portion of the rectangle’s perimeter</td></tr>
</table></td></tr>
<tr><td>buttonAlignX</td><td>Integer</td><td>R/W</td><td colspan="3">(Valid for <code>button</code> fields.)
Controls how space is distributed from the left of the button face with respect to the icon. It is expressed as a percentage between 0 and 100, inclusive. The default value is <code>50</code>.
If the icon is scaled anamorphically (which results in no space differences), this property is not used.</td></tr>
<tr><td>buttonAlignY</td><td>Integer</td><td>R/W</td><td colspan="3">(Valid for <code>button</code> fields.)
Controls how unused space is distributed from the bottom of the button face with respect to the icon. It is expressed as a percentage between 0 and 100, inclusive. The default value is <code>50</code>.
If the icon is scaled anamorphically (which results in no space differences), this property is not used.</td></tr>
</table>

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
buttonFitBounds	Boolean	R/W	(Valid for **button** fields.) If true, the extent to which the icon may be scaled is set to the bounds of the button field. The additional icon placement properties are still used to scale and position the icon within the button face.
buttonPosition	Integer	R/W	(Valid for **button** fields.) Controls how the text and the icon of the button are positioned with respect to each other within the button face. The convenience position object **defines** all of the valid alternatives.
buttonScaleHow	Integer	R/W	(Valid for **button** fields.) Controls how the icon is scaled (if necessary) to fit inside the button face. The convenience scaleHow object defines all of the valid alternatives:
buttonScaleWhen	Integer	R/W	(Valid for **button** fields.) Controls when an icon is scaled to fit inside the button face. The convenience scaleWhen object defines all of the valid alternatives:
calcOrderIndex	Integer	R/W	(Valid for **combobox** and **text** fields.) Changes the calculation order of fields in the document. When a computable text or combo box field is added to a document, the field's name is appended to the calculation order array. The calculation order array determines in what order the fields are calculated.
charLimit	Integer	R/W	(Valid for **text** fields.) Limits the number of characters that a user can type into a text field. See **event.fieldFull** to detect when the maximum number of characters is reached.
comb	Boolean	R/W	(Valid for **text** fields.) If set to ***true***, the field background is drawn as series of boxes (one for each character in the value of the field) and each character of the content is drawn within those boxes. The number of boxes drawn is determined from the charLimit property. It applies only to text fields. The setter will also raise if any of the following field properties are also set multiline, password, and fileSelect. A side-effect of setting this property is that the **doNotScroll** property is also set.
commitOnSelChange	Boolean	R/W	(Valid for **combobox** and **listbox** fields.) Controls whether a field value is committed after a selection change: If ***true***, the field value is committed immediately when the selection is made. If ***false***, the user can change the selection multiple times without committing the field value. The value is committed only when the field loses focus, that is, when the user clicks outside the field.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
currentValueIndices	Integer | Array	R/W	(Valid for **`combobox`** and **`listbox`** fields.) Reads and writes single or multiple values of a list box or combo box. Read Returns the options-array indices of the strings that are the value of a list box or combo box field. These indices are 0-based. If the value of the field is a single string, it returns an integer. Otherwise, it returns an array of integers sorted in ascending order. If the current value of the field is not a member of the set of offered choices (as could happen in the case of an editable combo box) it returns **`-1`**. Write Sets the value of a list box or combo box. It accepts a single integer or array of integers as an argument. To set a single string as the value, pass an integer that is the 0-based index of that string in the options array. Note that in the case of an editable combo box, if the desired value is not a member of the set of offered choices, you must set the value instead. Except for this case, **`currentValueIndices`** is the preferred way to set the value of a list box or combo box. To set a multiple selection for a list box that allows it, pass an array as argument to this property, containing the indices (sorted in ascending order) of those strings in the options array. If passed an array like this: **`Field.value=["item1", "item3"]`**, it also selects the items. The ability for a list box to support multiple selections can be set through multipleSelection. Related methods and properties include **`numItems`**, **`getItemAt`**, **`insertItemAt`**, **`deleteItemAt`** and **`setItems`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

<table>
<tr><th>Property</th><th></th><th></th><th>Description</th></tr>
<tr><td>defaultStyle</td><td>Span object</td><td>R/W</td><td>
(Valid for <code>rich text</code> fields.)

This property defines the default style attributes for the form field. If the user clicks an empty field and begins entering text without changing properties using the property toolbar, these are the properties that will be used. This property is a single Span object without a text property. Some of the properties in the default style span mirror the properties of the Field object. Changing these properties also modifies the defaultStyle property for the field and vice versa.

The following table details the properties of the Field object that are also in the default style and any differences between their values.
<table>
<tr><th>Field properties</th><th>defaultStyle
(Span properties)</th><th>Description</th></tr>
<tr><td>alignment</td><td>alignment</td><td>The alignment property has the same values for both the default style and the Field object.</td></tr>
<tr><td>textFont</td><td>fontFamily
fontStyle
fontWeight</td><td>The value of this field property is a complete font name that represents the font family, weight, and style. In the default style property, each property is represented separately. If an exact match for the font properties specified in the default style cannot be found, a similar font will be used or synthesized</td></tr>
<tr><td>textColor</td><td>textColor</td><td>The textColor property has the same values for both the default style and the Field object.</td></tr>
<tr><td>textSize</td><td>textSize</td><td>The textSize property has the same values for both the default style and the Field object</td></tr>
</table>
NOTE When a field is empty, defaultStyle is the style used for newly entered text. If a field already contains text when defaultStyle is changed, the text will not pick up any changes to defaultStyle. Newly entered text uses the attributes of the text it is inserted into (or specified with the toolbar).

When pasting rich text into a field, unspecified attributes in the pasted rich text are filled with those from defaultStyle.

Superscript and Subscript are ignored in defaultStyle
</td></tr>
</table>

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
defaultValue	String	R/W	(Valid for all fields except **button** and **signature** fields.) The default value of a field—that is, the value that the field is set to when the form is reset. For combo boxes and list boxes, either an export or a user value can be used to set the default. A conflict can occur, for example, when the field has an export value and a user value with the same value but these apply to different items in the list. In such cases, the export value is matched against first.
doNotScroll	Boolean	R/W	(Valid for **text** fields.) If true, the text field does not scroll and the user, therefore, is limited by the rectangular region designated for the field. Setting this property to true or false corresponds to checking or unchecking the Scroll Long Text field in the Options tab of the field.
doNotSpellCheck	Boolean	R/W	(Valid for **text** and **combobox** fields.) If true, spell checking is not performed on this editable text field. Setting this property to ***true*** or ***false*** corresponds to unchecking or checking the Check Spelling attribute in the Options tab of the Field Properties dialog box.
delay	Boolean	R/W	Delays the redrawing of a field's appearance. It is generally used to buffer a series of changes to the properties of the field before requesting that the field regenerate its appearance. Setting the property to ***true*** forces the field to wait until delay is set to ***false***. The update of its appearance then takes place, redrawing the field with its latest settings. There is a corresponding **Doc** delay flag if changes are being made to many fields at once.
display	Integer	R/W	Controls whether the field is hidden or visible on screen and in print. The values for the display property are listed in the table below. This property supersedes the older **hidden** and **print** properties.
doc	object	R	Returns the **Doc** of the document to which the field belongs.
editable	Boolean	R/W	Controls whether a combo box is editable. If ***true***, the user can type in a selection. If ***false***, the user must choose one of the provided selections.
exportValues	Array	R/W	(Valid for **checkbox** and **radiobutton** fields.) An array of strings representing the export values for the field. The array has as many elements as there are annotations in the field. The elements are mapped to the annotations in the order of creation (unaffected by tab-order). For radio button fields, this property is required to make the field work properly as a group. The button that is checked at any time gives its value to the field as a whole. For check box fields, unless an export value is specified, "***Yes***" (or the corresponding localized string) is the default when the field is checked. "***Off***" is the default when the field is unchecked (the same as for a radio button field when none of its buttons are checked).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
fileSelect	Boolean	R/W	(Valid for **`text`** fields.) **[Security]** If ***`true`***, sets the file-select flag in the Options tab of the text field (Field is Used for File Selection). This indicates that the value of the field represents a path of a file whose contents may be submitted with the form. The path may be entered directly into the field by the user, or the user can browse for the file. (See the **`browseForFileToSubmit`**() method.) NOTE The **`fileSelect`** flag is mutually exclusive with the **`multiline`**, **`charLimit`**, **`password`**, and **`defaultValue`** properties.
fillColor	Array	R/W	Specifies the background color for a field. The background color is used to fill the rectangle of the field. Values are defined by using transparent, gray, RGB or CMYK color. See **`color arrays`** for information on defining color arrays and how values are used with this property. In older versions of this specification, this property was named bgColor. The use of bgColor is now discouraged, although it is still valid for backward compatibility.
hidden	Boolean	R/W	NOTE This property has been superseded by the display property and its use is discouraged. If the value is false, the field is visible to the user; if true, the field is invisible. The default value is false.
highlight	String	R/W	(Valid for **`button`** fields.) Defines how a button reacts when a user clicks it. The four highlight modes supported are: none — No visual indication that the button has been clicked. invert — The region encompassing the button's rectangle inverts momentarily. push — The down face for the button (if any) is displayed momentarily. outline — The border of the rectangle inverts momentarily. The convenience highlight object defines each state, as follows:
lineWidth	Integer	R/W	Specifies the thickness of the border when stroking the perimeter of a field's rectangle. If the stroke color is transparent, this parameter has no effect except in the case of a beveled border. Values are: ***`0`*** — none ***`1`*** — thin ***`2`*** — medium ***`3`*** — thick In older versions of this specification, this property was borderWidth. The use of borderWidth is now discouraged, although it is still valid for backward compatibility. The default value for lineWidth is 1 (thin). Any integer value can be used; however, values beyond 5 may distort the field's appearance.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
multiline	Boolean	R/W	(Valid for **text** fields.) Controls how text is wrapped within the field. If ***false*** (the default), the text field can be a single line only. If ***true***, multiple lines are allowed and they wrap to field boundaries.
multipleSelection	Boolean	R/W	(Valid for **listbox** fields.) If true, indicates that a list box allows a multiple selection of items. See also **type**, **value**, and **currentValueIndices**.
name	String	R	This property returns the fully qualified field name of the field as a string object.
numItems	Integer	R	(Valid for **listbox** and **combobox** fields.) The number of items in a combo box or list box. Face names and values of a combo box or list box can be accessed through the getItemAt method.
page	Integer | Array	R	The page number or an array of page numbers of a field. If the field has only one appearance in the document, the page property returns an integer representing the 0-based page number of the page on which the field appears. If the field has multiple appearances, it returns an array of integers, each member of which is a 0-based page number of an appearance of the field. The order in which the page numbers appear in the array is determined by the order in which the individual widgets of this field were created (and is unaffected by tab order). If an appearance of the field is on a hidden template page, page returns a value of -1 for that appearance.
password	Boolean	R/W	(Valid for **text** fields.) Specifies whether the field should display asterisks when data is entered in the field. (Upon submission, the actual data entered is sent.) If this property is true, data in the field is not saved when the document is saved to disk.
radiosInUnison	Boolean	R/W	(Valid for **radiobutton** fields.) If ***false***, even if a group of radio buttons have the same name and export value, they behave in a mutually exclusive fashion, like HTML radio buttons. The default for new radio buttons is false. If ***true***, if a group of radio buttons have the same name and export value, they turn on and off in unison.
readonly	Boolean	R/W	The read-only characteristic of a field. If a field is read-only, the user can see the field but cannot change it.
rect	Array	R/W	An array of four numbers in rotated user space that specify the size and placement of the form field. These four numbers are the coordinates of the bounding rectangle and are listed in the following order: upper-left x, upper-left y, lower-right x and lower-right y. NOTE The **Annotation** object also has a **rect** property. However, the coordinates are not in rotated user space and they are in a different order than in the **Field** object rect property.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
required	Boolean	R/W	(Valid for all fields except **button** fields.) Specifies whether a field requires a value. If true, the field's value must be non-null when the user clicks a submit button that causes the value of the field to be posted. If the field value is null, the user receives a warning message and the submit does not occur.
richText	Boolean	R/W	(Valid for **text** fields.) If ***true***, the field allows rich text formatting. The default is ***false***. Related objects and properties are **richValue**, **defaultStyle**, **event.richValue**, **event.richChange**, **event.richChangeEx**, the **Annotation** object **richContents** property, and the **Span** object.
richValue	Array of Span objects	R/W	(Valid for **rich text** fields.) This property specifies the text contents and formatting of a rich text field. For field types other than rich text, this property is undefined. The rich text contents are represented as an array of **Span** objects containing the text contents and formatting of the field. Related objects and properties are **richText**, **defaultStyle**, **event.richValue**, **event.richChange**, **event.richChangeEx**, the **Annotation** object **richContents** property, and the **Span** object.
rotation	Integer	R/W	The rotation of a widget in counterclockwise increments. Valid values are 0, 90, 180, and 270.
strokeColor	Array	R/W	Specifies the stroke color for a field that is used to stroke the rectangle of the field with a line as large as the line width. Values are defined by using transparent, gray, RGB or CMYK color. See **color arrays** for information on defining color arrays and how values are used with this property. In older versions of this specification, this property was borderColor. The use of borderColor is now discouraged, although it is still valid for backward compatibility.
style	String	R/W	(Valid for **checkbox** and **radiobutton** fields.) Allows the user to set the glyph style of a check box or radio button. The glyph style is the graphic used to indicate that the item has been selected. The style values are associated with keywords as follows.

Style	Keyword
check	style.ch
circle	style.ci
cross	style.cr
diamond	style.di
square	style.sq
star	style.st

submitName	String	R/W	If nonempty, used during form submission instead of name. Only applicable if submitting in HTML format (that is, URL-encoded).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
textColor	Array	R/W	The foreground color of a field. It represents the text color for text, button, or list box fields and the check color for check box or radio button fields. Values are defined the same as the **`fillColor`**. See **`color arrays`** for information on defining color arrays and how values are set and used with this property. In older versions of this specification, this property was **`fgColor`**. The use of fgColor is now discouraged, although it is still valid for backward compatibility. An exception should be thrown if a transparent color space is used to set textColor.
textFont	String	R/W	(valid for **`button`**, **`combobox`**, **`listbox`**, **`text font`** fields.) The font that is used when laying out text in a text field, combo box, list box or button. Valid fonts are defined as properties of the font object. Arbitrary fonts can also be used.
textSize	Number	R/W	Specifies the text size (in points) to be used in all controls. In check box and radio button fields, the text size determines the size of the check. Valid text sizes range from 0 to 32767, inclusive. A value of zero means the largest point size that allows all text data to fit in the field's rectangle.
type	String	R	Returns the type of the field as a string. Valid types are: ***`button`*** ***`checkbox`*** ***`combobox`*** ***`listbox`*** ***`radiobutton`*** ***`signature`*** ***`text`***
userName	String	R/W	The username (short description string) of the field. It is intended to be used as tooltip text whenever the cursor enters a field. It can also be used as a user-friendly name, instead of the field name, when generating error messages.
value	various	R/W	(Valid for all fields **`except button`** fields.) The value of the field data that the user has entered. Depending on the type of the field, may be a **`String`**, **`Date`**, or **`Number`**. Typically, the value is used to create calculated fields. If a field contains rich text formatting, modifying this property will discard the formatting and regenerate the field value and appearance using the **`defaultStyle`** and **`plain text`** value. To modify the field value and maintain formatting use the richValue property. NOTE For signature fields, if the field has been signed, a non-null string is returned as the value. The **`currentValueIndices`** of a list box that has multiple selections is the preferred and most efficient way to get and set the value of this type of field. See also **`valueAsString`** and **`type`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 32 *(continued)*

Property			Description
valueAsString	String	R	(Valid for all fields except **button** fields.) Returns the value of a field as an ECMAScript string. It differs from value, which attempts to convert the contents of a field contents to an accepted format. For example, for a field with a value of "*020*", value returns the integer 20, while **valueAsString** returns the string "*020*".

10.17.3.2 buttonPosition constants

Table 33 — buttonPosition constants

Icon/text placement	Keyword
Text Only	**position.textOnly**
Icon Only	**position.iconOnly**
Icon top, Text bottom	**position.iconTextV**
Text top, Icon bottom	**position.textIconV**
Icon left, Text right	**position.iconTextH**
Text left, Icon right	**position.textIconH**
Text in Icon (overlaid)	**position.overlay**

10.17.3.3 buttonScaleHow keywords

Table 34 — buttonScaleHow keywords

How is icon scaled	Keyword
Proportionally	**scaleHow.proportional**
Non-proportionally	**scaleHow.anamorphic**

10.17.3.4 buttonScaleWhen keywords

Table 35 — buttonScaleWhen keywords

When is icon scaled	Keyword
Always	**scaleWhen.always**
Never	**scaleWhen.never**
If icon is too big	**scaleWhen.tooBig**
If icon is too small	**scaleWhen.tooSmall**

10.17.3.5 defaultStyle related properties

Table 36 — defaultStyle related properties

Field properties	**defaultStyle** (Span properties)	Description
alignment	alignment	The alignment property has the same values for both the default style and the Field object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 36 *(continued)*

Field properties	defaultStyle (Span properties)	Description
textFont	fontFamily fontStyle fontWeight	The value of this field property is a complete font name that represents the font family, weight, and style. In the default style property, each property is represented separately. If an exact match for the font properties specified in the default style cannot be found, a similar font will be used or synthesized.
textColor	textColor	The `textColor` property has the same values for both the default `style` and the `Field` object.
textSize	textSize	The `textSize` property has the same values for both the default `style` and the `Field` object.

10.17.3.6 display keywords

Table 37 — display keywords

Keyword	Effect
`display.visible`	Field is visible on screen and in print
`display.hidden`	Field is hidden on screen and in print
`display.noPrint`	Field is visible on screen but does not print
`display.noView`	Field is hidden on screen but prints

10.17.3.7 Highlight keywords

Table 38 — highlight keywords

Type	Keyword
none	`highlight.n`
invert	`highlight.i`
push	`highlight.p`
outline	`highlight.o`

10.17.3.8 Style keywords

Table 39 — style keywords

Style	Keyword
check	`style.ch`
cross	`style.cr`
diamond	`style.di`
circle	`style.ci`
star	`style.st`
square	`style.sq`

10.17.3.9 textFont keywords

Table 40 — textFont keywords

Text font	Keyword
Times-Roman	`font.Times`

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 40 *(continued)*

Text font	Keyword
Times-Bold	`font.TimesB`
Times-Italic	`font.TimesI`
Times-BoldItalic	`font.TimesBI`
Helvetica	`font.Helv`
Helvetica-Bold	`font.HelvB`
Helvetica-Oblique	`font.HelvI`
Helvetica-BoldOblique	`font.HelvBI`
Courier	`font.Cour`
Courier-Bold	`font.CourB`
Courier-Oblique	`font.CourI`
Courier-BoldOblique	`font.CourBI`
Symbol	`font.Symbol`
ZapfDingbats	`font.ZapfD`

10.17.4 Field methods

10.17.4.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type `Field`.

10.17.4.2 browseForFileToSubmit()

Parameters

None

Return Value

undefined

Description

If invoked on a text field for which the fileSelect flag is set (checked), opens a standard file-selection dialog box. The path entered through the dialog box is automatically assigned as the value of the text field.

If invoked on a text field in which the fileSelect flag is clear (unchecked), an exception is thrown.

10.17.4.3 buttonGetCaption([nFace])

Parameters

Parameter	Type	Description
nFace	Number	(optional) If specified, gets a caption of the given type : *0* —normal caption *1* — down caption *2* — rollover caption The default is *0* (normal caption).

Return Value

string

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Gets the caption associated with a button.

Use buttonSetCaption to set the caption.

10.17.4.4 buttonGetIcon([nFace])

Parameters

Parameter	Type	Description
nFace	Number	(optional) If specified, gets an icon of the given type: *0* —normal icon *1* — down icon *2* — rollover icon The default is *0* (normal icon).

Return Value

Icon

Description

Gets the Icon object of a specified type associated with a button.

See also the `buttonSetIcon(oIcon[, nFace])` method for assigning an icon to a button.

10.17.4.5 buttonImportIcon([cPath, nPage])

Parameters

Parameter	Type	Description
cPath	String	(optional) The device-independent path for the file. PDF Processors should first attempt to open cPath as a PDF file. On failure, PDF Processors should then try to convert the file to PDF from one of the known graphics formats (BMP, GIF, JPEG, PCX, PNG, TIFF) and then import the converted file as a button icon.
nPage	Number	(optional) The 0-based page number from the file to turn into an icon. The default is 0.

Return Value

Integer

Description

[Security]

Imports the appearance of a button from another PDF file. If neither optional parameter is passed, the user is prompted to select a file.

Implementers must return one of the following values, as appropriate:

1 — The user cancelled the dialog

0 — No error

–1 — The selected file could not be opened

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

–2 — The selected page was invalid

See also **buttonGetIcon**([nFace]), **buttonSetIcon**(oIcon[, nFace]), **addIcon**(cName, icon), **getIcon**(cName), **importIcon**(cName[, cDIPath, nPage]), and **removeIcon**(cName).

10.17.4.6 buttonSetCaption(cCaption[, nFace])

Parameters

Parameter	Type	Description
cCaption	String	The caption associated with the button.
nFace	Number	(optional) If specified, sets a caption of the given type: *0* normal caption *1* down caption *2* rollover caption The default is *0* (normal caption)

Return Value

undefined

Description

Sets the caption associated with a button.

Use buttonGetCaption to get the current caption.

See **buttonAlignX**, **buttonAlignY**, **buttonFitBounds**, **buttonPosition**, **buttonScaleHow**, and **buttonScaleWhen** for details on how the icon and caption are placed on the button face.

10.17.4.7 buttonSetIcon(oIcon[, nFace])

Parameters

Parameter	Type	Description
oIcon	Object	The Icon object associated with the button.
nFace	Number	(optional) If specified, sets an icon of the given type: *0* normal icon *1* down icon *2* rollover icon The default is 0 (normal icon)

Return Value

undefined

Description

Sets the icon associated with a button.

See **buttonAlignX**, **buttonAlignY**, **buttonFitBounds**, **buttonPosition**, **buttonScaleHow**, and **buttonScaleWhen** for details on how the icon and caption are placed on the button face.

Use either buttonGetIcon or doc.getIcon to get an Icon object that can be used for the oIcon parameter of this method.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.17.4.8 checkThisBox(nWidget[, bCheckIt])

Parameters

Parameter	Type	Description
nWidget	Number	The 0-based index of an individual check box or radio button widget for this field. The index is determined by the order in which the individual widgets of this field were created (and is unaffected by tab-order). Every entry in the Fields panel has a suffix giving this index; for example, MyField #0.
bCheckIt	Boolean	(optional) Specifies whether the widget should be checked. The default is true.

Return Value

undefined

Description

Checks or unchecks the specified widget.

Only check boxes can be unchecked. A radio button cannot be unchecked using this method, but if its default state is unchecked (see `defaultIsChecked`(...)) it can be reset to the unchecked state using the resetForm. method of the Doc.

For a set of radio buttons that do not have duplicate export values, you can set the value to the export value of the individual widget that should be checked (or pass an empty string if none should be).

10.17.4.9 clearItems()

Parameters

None

Return Value

undefined

Description

defaultIsChecked

10.17.4.10 defaultIsChecked(...)

Parameters

Parameter	Type	Description
nWidget	Number	The 0-based index of an individual radio button or check box widget for this field. The index is determined by the order in which the individual widgets of this field were created (and is unaffected by tab-order). Every entry in the Fields panel has a suffix giving this index (for example, MyField #0).
bIsDefaultChecked	Boolean	(optional) If true (the default), the widget should be checked by default (for example, when the field gets reset). If false, it should be unchecked by default.

Return Value

Boolean

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Sets the specified widget to be checked or unchecked by default.

For a set of radio buttons that do not have duplicate export values, you can set the defaultValue to the export value of the individual widget that should be checked by default (or pass an empty string if none should be checked).

10.17.4.11 deleteItemAt([nIndex])

Parameters

Parameter	Type	Description
nIndex	Number	(optional) The 0-based index of the item in the list to delete. If not specified, the currently selected item is deleted.

Return Value

Undefined

Description

Deletes an item in a combo box or a list box.

For a list box, if the current selection is deleted, the field no longer has a current selection. If this method is invoked again on the same field and no parameter is specified, unexpected behavior can result because there is no current selection to delete. Therefore, it is important to make a new selection (such as by using the currentValueIndices method) for the method to behave as documented.

10.17.4.12 getArray()

Parameters

None

Return Value

Array of Field

Description

Gets the array of terminal child fields (that is, fields that can have a value) for this Field object, the parent field.

10.17.4.13 getItemAt(nIndex[, bExportValue])

Parameters

Parameter	Type	Description
nIndex	Number	The 0-based index of the item in the list or –1 for the last item in the list.
bExportValue	Boolean	(optional) Specifies whether to return an export value: If ***true*** (the default), if the item has an export value, it is returned. If there is no export value, the item name is returned. If ***false***, the method returns the item name.

Return Value

String

Description

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Gets the internal value of an item in a combo box or a list box.

The number of items in a list can be obtained from `numItems`. See also `insertItemAt`(cName[, cExport, nIndex]), `deleteItemAt`([nIndex]), `clearItems`(), `currentValueIndices` and `setItems(oArray)`.

10.17.4.14 getLock()

Parameters

None

Return Value

Lock

Description

Gets a Lock object, a generic object that contains the lock properties of a signature field.

See also `setLock(oLock)`.

10.17.4.15 insertItemAt(cName[, cExport, nIndex])

Parameters

Parameter	Type	Description
cName	String	The item name that will appear in the form.
cExport	String	(optional) The export value of the field when this item is selected. If not provided, the *cName* is used as the export value.
nIdx	Number	(optional) The index in the list at which to insert the item. If *0* (the default), the new item is inserted at the top of the list. If *–1*, the new item is inserted at the end of the list.

Return Value

undefined

Description

Inserts a new item into a combo box or a list box.

Related methods and properties include numItems, getItemAt, deleteItemAt, clearItems, currentValueIndices and setItems.

10.17.4.16 isBoxChecked(nWidget)

Parameters

Parameter	Type	Description
nWidget	Number	The 0-based index of an individual radio button or check box widget for this field. The index is determined by the order in which the individual widgets of this field were created (and is unaffected by tab-order). Every entry in the Fields panel has a suffix giving this index, for example, MyField #0.

Return Value

Boolean

Description

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Determines whether the specified widget is checked.

NOTE For a set of radio buttons that do not have duplicate export values, you can get the value, which is equal to the export value of the individual widget that is currently checked (or returns an empty string, if none is).

10.17.4.17 isDefaultChecked(nWidget)

Parameters

Parameter	Type	Description
nWidget	Number	The 0-based index of an individual radio button or check box widget for this field. The index is determined by the order in which the individual widgets of this field were created (and is unaffected by tab-order). Every entry in the Fields panel has a suffix giving this index, for example, MyField #0.

Return Value

Boolean

Description

Determines whether the specified widget is checked by default (for example, when the field gets reset).

NOTE For a set of radio buttons that do not have duplicate export values, you can get the defaultValue, which is equal to the export value of the individual widget that is checked by default (or returns an empty string, if none is).

10.17.4.18 setAction(cTrigger, cScript)

Parameters

Parameter	Type	Description
cTrigger	String	A string that sets the trigger for the action. Values are: ***MouseUp*** ***MouseDown*** ***MouseEnter*** ***MouseExit*** ***OnFocus*** ***OnBlur*** ***Keystroke Validate*** ***Calculate Format*** For a list box, use the **Keystroke** trigger for the **Selection Change** event.
cScript	String	The ECMAScript code to be executed when the trigger is activated.

Return Value

undefined

Description

Sets the ECMAScript action of the field for a given trigger.

Related methods are the Bookmark object setAction method and the Doc methods setAction, addScript, and setPageAction.

NOTE This method will overwrite any action already defined for the chosen trigger.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.17.4.19 setFocus()

Parameters

None

Return Value

Undefined

Description

Sets the keyboard focus to this field. This can involve changing the page that the user is currently on or causing the view to scroll to a new position in the document. This method brings the document in which the field resides to the front, if it is not already there.

See also `bringToFront`().

10.17.4.20 setItems(oArray)

Parameters

Parameter	Type	Description
oArray	Object	An array in which each element is either an object convertible to a string or another array: For an element that can be converted to a string, the user and export values for the list item are equal to the string. For an element that is an array, the array must have two subelements convertible to strings, where the first is the user value and the second is the export value.

Return Value

undefined

Description

Sets the list of items for a combo box or a list box.

Related methods and properties include `numItems`, `getItemAt`, `deleteItemAt`, `currentValueIndices` and `clearItems`.

10.17.4.21 setLock(oLock)

Parameters

Parameter	Type	Description
oLock	Object	A `Lock` object containing the lock properties.

Return Value

Boolean

Description

[Security]

Controls which fields are to be locked when a signature is applied to this signature field. No modifications can be made to locked fields. If the signature is cleared, all fields that were locked become unlocked. The property settings can be obtained using `getLock`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

This method cannot be applied to a field that is in a document that is already signed.

10.17.4.21.1 Lock Object

A generic ECMAScript object containing lock properties. This object is passed to `setLock` and returned by `getLock` for a `signature` field. It contains the following properties.

Table 41 — Lock object properties

Property			Description
action	R/W	String	The language-independent name of the action. Values are: All — All fields in the document are to be locked. Include — Only the fields specified in fields are to be locked. Exclude — All fields except those specified in fields are to be locked.
fields	R/W	Array of Strings	An array of strings containing the field names. Required if the value of action is Include or Exclude.

10.17.4.22 signatureGetModifications()

Parameters

None

Return Value

The call to signatureGetModifications returns an object containing modification information. The object has the following properties:

Table 42 — Field.signatureGetModifications return properties

Property	Type	Description
`formFieldsCreated`	Array of Field objects	Array of form fields created after signing.
`formFieldsDeleted`	Array of Generic objects	Array of form fields deleted after signing. Each generic object in the array is a string of the form `name : type`.
`formFieldsFilledIn`	Array of Field objects	Array of form fields filled in after signing.
`formFieldsModified`	Array of Field objects	Array of form fields modified after signing. In this context, form field fill-in does not constitute modification.
`annotsCreated`	Array of Annotation objects	Array of annotations created after signing. If the annotation is transient (for example, a dynamically created pop-up), the corresponding element of the array is a string of the form `author : name : type`.
`annotsDeleted`	Array of Generic objects	Array of annotations deleted after signing. Each generic object in the array is of the form `author : name : type`.
`annotsModified`	Array of Annotation objects	Array of annotations modified after signing. If the annotation is transient (for example, a dynamically created pop-up), the corresponding element of the array is a string of the form `author : name : type`.
`numPagesCreated`	Integer	Number of pages added after signing.
`numPagesDeleted`	Integer	Number of pages deleted after signing.
`numPagesModified`	Integer	Number of pages whose content has been modified after signing (add/delete/modify of annotations/form fields are not considered as page modification for this purpose).
`spawnedPagesCreated`	Array of Strings	List of pages spawned after signing. For each spawned page, the name of the source template is provided.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 42 *(continued)*

Property	Type	Description
`spawnedPagesDeleted`	Array of Strings	List of spawned pages deleted after signing. For each spawned page, the name of the source template is provided.
`spawnedPagesModified`	Array of Strings	List of spawned pages modified after signing. For each spawned page, the name of the source template is provided.

Description

Returns an object containing information on modifications that have been made to the document after the signature field was signed. The information includes only the difference between the current and signed state of the document. Transient objects, such as objects added after the signature but then subsequently deleted, are not reported.

10.17.4.23 signatureGetSeedValue()

Parameters

None

Return Value

SeedValue

Description

Returns a `SeedValue` object that contains the seed value properties of a signature field. Seed values are used to control properties of the signature, including the signature appearance, reasons for signing, and the person.

See `signatureSetSeedValue(oSigSeedValue)`.

10.17.4.24 signatureInfo([oSig])

Parameters

Parameter	Type	Description
oSig	Object	(optional) The `SecurityHandler` object to use to retrieve the signature p roperties. If not specified, the security handler is determined by the user preferences. (It is usually the handler that was used to create the signature.)

Return Value

The `signatureInfo` method shall return a `SignatureInfo` object that contains the properties of the signature. This type of object is also used when signing signature fields, signing FDF objects, or with the `FDF` object `signatureValidate` method.

Description

Returns a `SignatureInfo` object that contains the properties of the signature. The object is a snapshot of the signature at the time that this method is called. A security handler may specify additional properties that are specific to the security handler.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.17.4.25 signatureSetSeedValue(oSigSeedValue)

Parameters

Parameter	Type	Description
oSigSeedValue	Object	A SeedValue object containing the signature seed value properties.

Return Value

Undefined

Description

[Security]

Sets properties that are used when signing signature fields. The properties are stored in the signature field and are not altered when the field is signed, the signature is cleared, or when resetForm is called. Use `signatureGetSeedValue` to obtain the property settings.

When setting seed values, keep in mind the following:

Seed values should not be set on signed documents and cannot be set on certified documents. They are primarily used to configure fields on documents that are not yet signed.

10.17.4.25.1 SeedValue Object

A generic ECMAScript object, passed to `signatureSetSeedValue` and returned by `signatureGetSeedValue`, which represents a signature seed value. It has the following properties.

Table 43 — SeedValue object properties

Property	Type	Access	Description
appearanceFilter	String	R/W	A string corresponding to the named appearance that the AppearanceFilter seed value should match.
certspec	object	R/W	A seed value ***`CertificateSpecifier`*** object. see `CertificateSpecifier` Object for a listing of the properties of this object.
digestMethod	Array of strings	R/W	An array of acceptable digest methods to use while signing. The valid string values are ***`SHA1`***, ***`SHA256`***, ***`SHA384`***, ***`SHA512`***, ***`MD5`***, and ***`RIPEMD160`***. NOTE This is only applicable if the DigitalID contains RSA public/private keys. If they contain DSA public/private keys, then the value is always ***`SHA1`***. The flags property indicates whether this is a required constraint.
filter	String	R/W	The language-independent name of the security handler to be used when signing. If filter is specified and the flags property indicates this entry is a required constraint, then the signature handler specified by this entry must be used when signing; otherwise, signing must not take place. If flags indicate that this is an optional constraint, then this handler should be used if it is available. If it is not available, a different handler can be used instead.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 43 *(continued)*

Property	Type	Access	Description
flags	Number	R/W	A set of bit flags controlling which of the following properties of this object are required. The value is the logical OR of the following values, which are set if the corresponding property is required: ***1:*** ***filter*** ***2:*** ***subFilter*** ***4:*** ***version*** ***8:*** ***reasons*** ***16:*** ***legalAttestations*** ***32:*** ***shouldAddRevInfo*** ***64:*** ***digestMethod*** ***128:*** ***lockDocument*** ***256:*** ***appearanceFilter*** Usage: ***1*** specifies filter, ***3*** specifies filter and sub-filter, and ***11*** specifies filter, sub-filter, and reasons. If this field is not present, all properties are optional.
legalAttestations	Array of Strings	R/W	A list of legal attestations that the user can use when creating a Modification Detection and Prevention (MDP) (certified) signature. The value of the corresponding flag in the flags property indicates whether this is a required or optional constraint.
lockDocument	String	R/W	A string corresponding to one of the three **`LockDocument`** seed value states. Values are auto true ***`false`***. Any other value will not be set.
mdp	String	R/W	The MDP setting to use when signing the field. Values are ***`allowNone`*** ***`default`*** ***`defaultAndComments`*** While ***`allowAll`*** is a legal value, it cancels out the effect of MDP and no certification signature can be used for this field, resulting in the signature being an approval signature, not a certification signature. Values are unique identifiers and are described in the table titled Modification Detection and Prevention (MDP) Values.
reasons	Array of Strings	R/W	A list of reasons that the user is allowed to use when signing. If this array has a single element and that element is a single period '.' and if the reasons are marked as required by the flags property, then PDF Processors should suppress the UI for specifying a reason during signing. This overrides the user's preference. If this array is empty and reasons are marked as required, an exception should be thrown.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 43 *(continued)*

Property	Type	Access	Description
shouldAddRevInfo	Boolean	R/W	If set to ***true***, instructs the application to carry out revocation checking of the certificate (and the corresponding chains) used to sign, and include all the revocation information within the signature. Only relevant if **subFilter** is ***adbe.pkcs7.detached*** or ***adbe.pkcs7.sha1***, and it is not applicable for ***adbe.x509.rsa_sha1***. Hence, if the **subFilter** is ***adbe.x509.rsa_sha1*** and it's required that revocation information be added, then the signing operation fails. If **shouldAddRevInfo** is ***true*** and the flags property indicates this is a required constraint, then the tasks described above must be performed. If they cannot be performed, then signing must fail. The default value is ***false***.
subFilter	Array of Strings	R/W	An array of acceptable formats to use for the signature. Refer to the **Signature** Info object's **subFilter** property for a list of known formats. The first name in the array that matches an encoding supported by the signature handler should be the encoding that is actually used for signing. If **subFilter** is specified and the **flags property** indicates that this entry is a required constraint, then the first matching encodings must be used when signing; otherwise, signing must not take place. If the flags property indicates that this is an optional constraint, then the first matching encoding should be used if it is available. If it is not available, a different encoding can be used.
timeStampspec	Object	R/W	A Seed Value **timeStamp** Specifier object. It uses the url and flags properties to specify a timestamp server. See Seed value timeStamp specifier object.
version	Number	R/W	The minimum required version number of the signature handler to be used to sign the signature field. Valid values are ***1*** and ***2***. A value of ***1*** specifies that the parser must be able to recognize all seed value dictionary entries specified in PDF 1.5. A value of ***2*** specifies that it must be able to recognize all seed value dictionary entries specified in PDF 1.7 and earlier. The flags property indicates whether this is a required constraint.

10.17.4.25.2 CertificateSpecifier Object

This generic ECMAScript object contains the certificate specifier properties of a signature seed value. Used in the certSpec property of the SeedValue object. This object contains the following properties:

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 44 — CertificateSpecifier object properties

Property	Type	Access	Description

flags	Number	R/W	A set of bit flags controlling which of the following properties of this object are required. The value is the logical OR of the following values, which are set if the corresponding property is required:
			`1:` `subject`
			`2:` `issuer`
			`4:` `oid`
			`8:` `subjectDN`
			`16:` `Reserved`
			`32:` `keyUsage`
			`64:` `url`
			If this field is not present, all properties are optional.

issuer	Array of Certificate objects	R/W	An array of **`Certificate`** objects that are acceptable for signing. If specified, the signing certificate shall be issued by a certificate that is an exact match with one of the certificates in this array. The value of the corresponding flag in the flags property indicates whether this is a required or optional constraint.
keyUsage	Object	R/W	Array of integers, where each integer specifies the acceptable key-usage extension that must be present in the signing certificate. Each integer is constructed as follows: There are two bits used for each key-usage type (defined in RFC 3280) starting from the least significant bit: digitalSignature(bits 2,1), nonRepudiation(4,3) keyEncipherment(6,5) dataEncipherment(8,7) keyAgreement(10,9) keyCertSign(12,11) cRLSign(14,13) encipherOnly(16,15) decipherOnly(18,17) The value of the two bits have the following semantics: `00`: the corresponding **`keyUsage`** must not be set `01` the corresponding **`keyUsage`** must be set `10` and `11`: the state of the corresponding **`keyUsage`** doesn't matter. For example, if it is required that the key-**`usage`** type digitalSignature be set and the state of all other key-**`usage`** types do not matter, then the corresponding integer would be `0x7FFFFFFD`. That is, to represent digitalSignature, set `01` for bits `2` and `1` respectively, and set `11` for all other key-usage types. The value of the corresponding flag in the flags property indicates whether this is a required constraint.
oid	Array of Strings	R/W	An array of strings that contain Policy OIDs that must be present in the signing certificate. This property is only applicable if the issuer property is present. The value of the corresponding flag in the flags property indicates whether this is a required or optional constraint.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 44 *(continued)*

Property	Type	Access	Description
subject	Array of Certificate objects	R/W	An array of Certificate objects that are acceptable for signing. If specified, the signing certificate shall be an exact match with one of the certificates in this array. The value of the corresponding flag in the flags property indicates whether this is a required or optional constraint.
subjectDN	Array of objects	R/W	Array of objects, where each object specifies a Subject Distinguished Name (DN) that is acceptable for signing. A signing certificate satisfies a DN requirement if it at least contains all the attributes specified in the DN (it can therefore contain additional DN attributes). If this entry is specified, the signing certificate must satisfy at least one of the DNs in the array. DN attribute restrictions are specified in this object by adding them as properties. The properties' key names can either be the corresponding attributes' friendly names or OIDs (as defined in RFC 3280). The properties' value must be of type string. The value of the corresponding flag in the flags property indicates whether this is a required or optional constraint. EXAMPLE 1 {cn:"Alice", ou:"Engineering", o:"Acme Inc"}. The friendly names for the DN attributes (cn, o, ou, and so on) are the ones defined in RDN (see **RDN**). EXAMPLE 2 { cn:"Joe Smith", ou:"Engineering", 2.5.4.43:"JS"}, where OID 2.5.4.43 is used to carry out matching for the "Initials" attribute. The following is sample code to define the above DN: `var subjectDN = {cn:"Joe Smith", ou:"Engineering"}; subjectDN["2.4.5.43"] = "JS";` Attributes whose value is of type DirectoryString or IA5String can be specified as shown in the example above, whereas for all other value types, for example, dateOfBirth whose value is of type GeneralizedTime, the values need to be specified as a hex-encoded binary string. For more information about the various attributes and their types, refer to RFC 3280.
url	String	R/W	A URL that can be used to enroll for a new credential if a matching credential is not found. A degenerate use of this property is when the URL points to a web service that is a DigitalID store (such as a signature web service that can be used for server-based signing). In that case, the URL indicates that as long as the signer has a DigitalID from that web service, it is acceptable for signing. The value of the corresponding flag in the flags property indicates whether this is a required or optional constraint.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 44 *(continued)*

Property	Type	Access	Description
urlType	String	R/W	String indicating the type of URL specified by the url property. Potential values: "***HTML***": The URL points to an HTML website and the PDF Processor should use the system web browser to display its contents. The value of the url bit of the flags property is ignored. "***ASSP***": The URL references a signature web service that can be used for server-based signing. If the url bit of the flags property indicates that this is a required constraint, this implies that the credential used when signing must come from this server. If this attribute isn't present, it's assumed that the URLpoints to a HTML site.

10.17.4.25.3 Seed value timeStamp specifier object

The properties of the seed value timeStamp specifier object are as follows:

Table 45 — SeedValue timeStamp properties

Property	Type	Access	Description
url	String	R/W	URL of the timeStamp server providing an RFC 3161 compliant timeStamp.
flags	Number	R/W	A bit flag controlling whether the time stamp is required (*1*) or not required (*0*). The default is *0*.

10.17.4.26 signatureSign(oSig[, oInfo, cDIPath, bUI, cLegalAttest])

Parameters

Parameter	Type	Description
oSig	Object	Specifies the **SecurityHandler** object to be used for signing. Throws an exception if the specified handler does not support signing operations. Some security handlers require that the user be logged in before signing can occur. If **oSig** is not specified, this method selects a handler based on user preferences or by prompting the user if **bUI** is ***true***.
oInfo	Object	(optional) A **SignatureInfo** object specifying the writable properties of the signature. See also **signatureInfo**([oSig]).
cDIPath	String	(optional) The device-independent path to the file to save to following the application of the signature. If not specified, the file is saved back to its original location.
bUI	Boolean	(optional) A Boolean value specifying whether the security handler should show the user interface when signing. If ***true***, **oInfo** and **cDIPath** are used as default values in the signing dialog boxes. If ***false*** (the default), the signing occurs without a user interface.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
cLegalAttest	String	(optional) A string that can be provided when creating a certification signature. Certification signatures are signatures where the mdp property of the **`SignatureInfo`** object has a value other than allowAll. When creating a certification signature, the document is scanned for legal warnings and these warnings are embedded in the document. A caller can determine what legal warnings are found by first calling the **`Doc getLegalWarnings`** method. If warnings are to be embedded, an author may provide an attestation as to why these warnings are being applied to a document.

Return Value

Boolean

Description

[Security]

Signs the field with the specified security handler. See also getHandler(cName[, bUIEngine]) and login([cPassword, cDIPath, oParams, bUI]).

Signature fields cannot be signed if they are already signed. Use resetForm to clear signature fields.

Return true if the signature was applied successfully, false otherwise.

10.17.4.27 signatureValidate([oSig, bUI])

Parameters

Parameter	Type	Description
oSig	Object	(optional) The security handler to be used to validate the signature. Its value is either a **`SecurityHandler`** object or a **`SignatureParameters`** object. If this handler is not specified, the method uses the security handler returned by the signature's **`handlerName`** property.
bUI	Boolean	(optional) If ***`true`***, allows the UI to be shown when validating, if necessary. The UI may be used to select a validation handler if none is specified. The default is ***`false`***.

Return Value

Integer

Description

Validates and returns the validity status of the signature in a signature field. This routine can be computationally expensive and take a significant amount of time depending on the signature handler used to sign the signature.

The return value shall be an integer representing the validity status of the signature as follows:

— *-1* Not a signature field

— *0* Signature is blank

— *1* Unknown status

— *2* Signature is invalid

— *3* Signature of document is valid, identity of signer could not be verified

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

— *4* Signature of document is valid and identity of signer is valid.

See the **signVisible** and **statusText** properties of the **SignatureInfo** object.

10.17.4.27.1 SignatureParameters object

A generic object with the following properties that specify which security handlers are to be used for validation by signatureValidate:

Table 46 — signatureValidate handlers

Handler	Description
oSecHdlr	The **security handler** object to use to validate this signature.
bAltSecHdlr	If ***true***, an alternate security handler, selected based on user preference settings, may be used to validate the signature. The default is ***false***, which means that the security handler returned by the signature's handlerName property is used to validate the signature. This parameter is not used if oSecHdlr is provided.

10.18 FullScreen

10.18.1 General

The interface to fullscreen (presentation mode) preferences and properties. To acquire a fullScreen object, use **app.fs**.

10.18.2 FullScreen properties

10.18.2.1 General

Table 47: FullScreen object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type *FullScreen*.

Table 47 — FullScreen object properties

Property	Type	Access	Description
backgroundColor	ColorArray	R/W	The background color of the screen in full screen mode. See **color arrays for details.**
clickAdvances	Boolean	R/W	Specifies whether a mouse click anywhere on the page causes the page to advance.
cursor	Number	R/W	Determines the behavior of the pointer in full screen mode. The convenience cursor object defines all the valid cursor behaviors.
defaultTransition	String	R/W	The default transition to use when advancing pages in full screen mode. Use transitions to obtain list of valid transition names supported by the viewer. "***No Transition***" is equivalent to **app.fs.defaultTransition = "";**
escapeExits	Boolean	R/W	**[Security]** A Boolean value specifying the escape key can be used to exit full screen mode.
isFullScreen	Boolean	R/W	If true, the viewer is in full screen mode rather than regular viewing mode, which is possible only if one or more documents are open for viewing. NOTE A PDF document being viewed from within a web browser might not be able to be put into full screen mode.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 47 *(continued)*

Property	Type	Access	Description
loop	Boolean	R/W	Specifies whether the document will loop around to the beginning of the document in response to a page advance (whether generated by mouse click, keyboard, or timer) in full screen mode.
timeDelay	Number	R/W	The default number of seconds before the page automatically advances in full screen mode. See **`useTimer`** to activate or deactivate automatic page turning.
transitions	Array	R	An array of strings representing valid transition names implemented in the viewer. ***`No Transition`*** is equivalent to setting **`defaultTransition`** to the empty string: `app.fs.defaultTransition = "";`
usePageTiming	Boolean	R/W	Specifies whether automatic page turning will respect the values specified for individual pages in full screen mode. Set transition properties of individual pages using **`setPageTransitions`**.
useTimer	Boolean	R/W	Specifies whether automatic page turning is enabled in full screen mode. Use **`timeDelay`** to set the default time interval before proceeding to the next page.

10.18.2.2 Cursor Keywords

Table 48 — Cursor Keywords

Cursor Behavior	Keyword
Always hidden	***`cursor.hidden`***
Hidden after delay	***`cursor.delay`***
Visible	***`cursor.visible`***

10.19 global

10.19.1 General

This is a static ECMAScript object that allows you to share data between documents and to have data be persistent across sessions. Such data is called persistent global data. Global data-sharing and notification across documents is done through a subscription mechanism, which allows you to monitor global data variables and report their value changes across documents.

10.19.2 Creating global properties

You can specify global data by adding properties to the global object. The property type can be a String, a Boolean value, or a Number.

For example, to add a variable called radius and to allow all document scripts to have access to this variable (provided "Enable global object security policy" is not checked), the script defines the property:

```
global.radius = 8;
```

The global variable radius is now known across documents throughout the current viewer session. Suppose two files, A.pdf and B.pdf, are open and the global declaration is made in A.pdf. From within either file (A.pdf or B.pdf), you can calculate the volume of a sphere using global.radius.

```
var V = (4/3) * Math.PI * Math.pow(global.radius, 3);
```

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

In either file, you obtain the same result, 2 144,660 58. If the value of global.radius changes and the script is executed again, the value of V changes accordingly.

10.19.3 Deleting global properties

To delete a variable or a property from the global object, use the delete operator to remove the defined property. Information on the reserved ECMAScript keyword delete can be found in the ECMAScript standard, available from Ecma International (www.ecma-international.org).

10.19.4 Global object security policy

The global security policy places restrictions on document access to global variables. For more information and exceptions, see the Note in **`global`**.

In a document, named docA.pdf, execute the following script in a non-privileged context (mouse-up button):

```
global.x = 1
global.setPersistent("x", true);
```

The path for docA.pdf is the origin saved with the global.x variable; consequently, docA.pdf can access this variable:

```
console.println("global.x = " + global.x);
```

To set this global from docA.pdf, we execute global.x = 3, for example, in any context.

To have a document with a different path get and set this global variable, the getting and setting must occur in a trusted context, with a raised level of privilege. The following scripts are folder ECMAScript.

```
myTrustedGetGlobal = app.trustedFunction( function()
{ app.beginPriv(); var y = global.x; return y app.endPriv();
}) ; myTrustedSetGlobal = app.trustedFunction ( function(value)
{ app.beginPriv(); global.x=value; app.endPriv();
}) ;
```

Another document, docB.pdf can access the global.x variable through the above trusted functions:

```
// Mouse up button action from doc B
console.println("The value of global.x is " + myTrustedGetGlobal());
The global can also be set from docB.pdf:
// Set global.x from docB.pdf myTrustedSetGlobal(2);
Once global.x has been set from docB.pdf, the origin is changed; docA.pdf cannot access
global.x directly unless it is through a trusted function:
// Execute a mouse up button action from docA.pdf
console.println("The value of global.x is " + myTrustedGetGlobal());
```

10.19.5 global object methods

10.19.5.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type *global*.

10.19.5.2 setPersistent(cVariable, bPersist)

Parameters

Parameter	Type	Description
cVariable	String	The variable (global property) for which to set persistence.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
bPersist	Boolean	If true, the property will exist across a PDF Processor's viewer sessions. If false (the default) the property will be accessible across documents but not across the PDF Processor's viewer sessions.

Return Value

undefined

Description

Controls whether a specified variable is persistent across invocations of a PDF Processor.

Persistent global data only applies to variables of type Boolean, Number, or String.

Upon application exit, persistent global variables are stored in the glob.js file located in the user's folder for folder-level scripts and re-loaded at application start. Implementers may place a limit on the size of this file. Any data added to the string after this limit is dropped.

When specifying persistent global variables, you should use a naming convention. For example, you can give your variables the form myCompany_variableName. This prevents collisions with other persistent global variable names throughout the documents.

10.19.5.3 subscribe(cVariable, fCallback)

Parameters

Parameter	Type	Description
cVariable	String	The global property.
fCallback	Function	The function to call when the property is changed.

Return Value

undefined

Description

Allows you to automatically update fields when the value of a global variable changes. If the specified property `cVariable` is changed, even in another document, the specified function `fCallback` is called. Multiple subscribers are allowed for a published property.

10.19.5.3.1 Callback Function

The syntax of the callback function is as follows:

```
function fCallback(newval) {
// newval is the new value of the global variable you // have subscribed to.
< code to process the new value of the global variable >
}
```

The callback will be terminated when the document from which the script was originally registered is closed, or the (global) property deleted.

10.20 HostContainer

10.20.1 General

This object manages communication between a PDF document and a corresponding host container that the document is contained within, such as an HTML page. The host container for a document is specified by the Doc hostContainer property.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

The Embedded PDF object provides the corresponding API for the object model of the container application.

10.20.2 HostContainer properties

10.20.2.1 General

Table 49: HostContainer object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`HostContainer`**.

Table 49 — HostContainer object properties

Property	Type	Access	Description
messageHandler	Object	R/W	A notification object that is called if a script in the host container calls the **`postMessage`** method. This object may expose the methods and properties listed in [TABLEREF]. NOTE Instead of specifying a method, onDisclose can specify the value **`HostContainerDisclosurePolicy.SameOriginPolicy`**, which means that the document will be disclosed to the host container if they both originate from the same location. In this case, the origination is determined by the scheme, server, and port. The scheme must be http, https, or ftp for the document to be disclosed. When these methods are invoked, the **`this`** object will be the **`messageHandler`** instance that the method is being called on. Other methods and properties can be added to **`messageHandler`** provided that they do not begin with on. If **`messageHandler`** is set to ***`null`*** or an object without an **`onMessage`** method, messages sent by **`postMessage`** are queued until the property is set to a valid **`messageHandler`** instance. Messages are guaranteed to be delivered in the order in which they are posted and errors are guaranteed to be delivered in the order in which they occurred. However, there is no correspondence between the delivery order of messages and errors. Exceptions thrown from within the handler methods will be discarded. If an exception is thrown from within **`onDisclose`**, the function will be treated as a failure. Messages and errors will not be delivered while inside an ***`onMessage`*** / ***`onError`*** / ***`onDisclose`*** handler.

10.20.2.2 HostContainer.messageHandler Object Details

Table 50 — HostContainer.messageHandler properties

Method/Property	Description
onMessage	(Optional) A method that is called in response to postMessage. The message is delivered asynchronously. The method is passed a single array parameter containing the array that was passed to postMessage.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 50 *(continued)*

Method/Property	Description
onError	(Optional) A method that is called in response to an error. The method is passed an Error object and an array of strings corresponding to the message that caused the error. The name property of the Error object is set to one of these: ***MessageGeneralError*** A general error occurred. ***MessageNotAllowedError*** The operation failed for security reasons. If an error occurs and this property is undefined, the error will not be delivered (unlike messages, errors are not queued).
onDisclose	A required method that is called to determine whether the host application is permitted to send messages to the document. This allows the PDF document author to control the conditions under which messaging can occur for security reasons. The method should be set during the Doc/Open event. The method is passed two parameters: ***cURL*** The URL indicating the location of the host container (for example, the URL of an HTML page using an <OBJECT> tag). ***cDocumentURL*** he URL of the PDF document that disclosure is being checked for. If the method returns true, the host container is permitted to post messages to the message handler.
allowDeliverWhileDocIsModal	A Boolean value indicating whether messages and errors will be delivered while the document is in a modal state. By default (***false***), messages and errors are queued and not delivered to the onMessage and onError handlers if the application is currently displaying a modal dialog. The app.isModal property can be used to detect this case.

10.20.3 HostContainer methods

10.20.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type *HostContainer*.

10.20.3.2 postMessage(...)

Parameters

Parameter	Type	Description
aMessage	Array	An array of one or more strings that are passed to the onMessage method of the messageHandler property.

Return Value

undefined

Description

Sends a message asynchronously to the message handler for the host container of the PDF document. For this message to be delivered, the host container (for example, an <OBJECT> element in an HTML page) must have registered for notification by setting its messageHandler property. The message is passed to the **onMessage** method of the messageHandler.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

The messages are submitted to a queue until they are delivered. If the queue exceeds a maximum number of messages, a parameter error is thrown until some of the messages in the queue have been delivered.

EXAMPLE var aMessage = [“MyMessageName”, “Message Body”]; this.hostContainer.postMessage(aMessage);

10.21 Icon

10.21.1 General

This generic ECMAScript object is an opaque representation of a `Form` XObject appearance stored in the Doc icons property. It is used with `Field` objects of type button.

10.21.2 icon Properties

The `icon` object contains the following property:

Table 51 — Icon property

Property	Type	Access	Description
name	String	R	The name of the icon. An icon has a name if it exists in the document-level named icons tree.

10.22 Link

10.22.1 General

This object is used to set and get the properties and to set the ECMAScript action of a link.

Link objects can be obtained from the Doc methods addLink or getLinks. (See also `removeLinks(nPage, oCoords)`.)

10.22.2 Link properties

Table 52: Link object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Link`.

Table 52 — Link object properties

Property	Type	Access	Description
borderColor	Array	R/W	The border color of a Link object. See `color arrays` for information on defining color arrays and how colors are used with this property.
borderWidth	Integer	R/W	The border width of the Link object.

highlightMode	String	R/W	The visual effect to be used when the mouse button is pressed or held down inside an active area of a link. The valid values are:
			`None`
			`Invert`
			`Outline`
			`Push`
			The default is `Invert`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 52 *(continued)*

Property	Type	Access	Description
rect	Array	R/W	The rectangle in which the link is located on the page. Contains an array of four numbers, the coordinates in rotated user space of the bounding rectangle, listed in the following order: upper-left x, upper-left y, lower-right x and lower-right y.

10.22.3 Link methods

10.22.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type `Link`.

10.22.3.2 setAction(cScript)

Parameters

Parameter	Type	Description
cScript	String	The ECMAScript action to use.

Return Value

undefined

Description

Sets the specified ECMAScript action for the `MouseUp` trigger for the `Link` object.

NOTE This method will overwrite any action already defined for this link

10.23 Net

10.23.1 General

The Net object allows for the discovery and access to networking services through a variety of protocols.

The Net object forms a namespace for the networking services that currently exist in the global namespace (SOAP). The existing objects will be maintained for backwards-compatibility however any new services should use the new namespace.

10.23.2 Net properties

10.23.2.1 General

Table 53: Net object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Net`.

Table 53 — Net object properties

Property	Type	Access	Description
SOAP	Object	R	The Net.SOAP object allows for communication with network services that use the SOAP Messaging Protocol. The Net.SOAP object has one property and three methods. See the referenced link for full documentation.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 53 *(continued)*

Property	Type	Access	Description
Discovery	Object	R	The Discovery object allows for the dynamic discovery of network services. The Net.Discovery object has two methods. See the referenced link for full documentation.
HTTP	Object	R	The Net.HTTP object allows access to web services that use HTTP. These methods allow communication with a wider variety of services than the Net.SOAP object. The HTTP object can be used to implement any messaging protocol that utilizes HTTP, for example, WebDAV or the ATOM Publishing Protocol. See Net.HTTP for full documentation of the methods of the HTTP object.

10.23.2.1.1 Net.SOAP properties

10.23.2.1.1.1 General

Table 54: Net.SOAP object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Net`.

Table 54 — Net.SOAP object properties

Property	Type	Access	Description
wireDump	Boolean	R/W	If true, synchronous SOAP requests will cause the XML Request and Response to be dumped to the ECMAScript Console. This is useful for debugging SOAP problems.

10.23.2.1.2 Net.SOAP Methods

10.23.2.1.2.1 connect

Parameters

None

Return Value

undefined

Description

Converts the URL of a WSDL document to a ECMAScript object with callable methods corresponding to the web service.

10.23.2.1.2.2 request

Parameters

None

Return Value

undefined

Description

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Initiates a remote procedure call (RPC) or sends an XML message to a SOAP HTTP endpoint. The method either waits for the endpoint to reply (synchronous processing) or calls a method on the notification object (asynchronous processing).

10.23.2.1.2.3 response

Parameters

None

Return Value

undefined

Description

Initiates a remote procedure call (RPC) or sends an XML message to a SOAP HTTP endpoint without waiting for a reply.

10.23.2.1.3 Net.Discovery Methods

10.23.2.1.3.1 queryServices

Parameters

None

Return Value

undefined

Description

Locates network services that have published themselves using DNS Service Discovery (DNS-SD).

10.23.2.1.3.2 resolveService

Parameters

None

Return Value

undefined

Description

Allows a service name to be bound to a network address and port in order for a connection to be made.

10.23.3 Net methods

10.23.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type *Net*.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.23.3.2 streamDecode(oStream, cEncoder)

Parameters

Parameter	Type	Description
oStream	Object	A stream object to be decoded with the specified encoding type.
cEncoder	String	Permissible values for this string are "*hex*" (hex-encoded) and "***base64***" (Base 64- encoded).

Return Value

ReadStream

Description

Allows the oStream object to be decoded with the specified encoding type, cEncoder. It returns a decoded ReadStream object. Typically, it is used to access data returned as part of a SOAP method that was encoded in Base64 or hex encoding.

10.23.3.3 streamDigest(oStream, cEncoder)

Parameters

Parameter	Type	Description
oStream	Object	A stream object to compute the digest of, using the specified message digest algorithm.
cEncoder	String	The digest algorithm to use. The cEncoder parameter must be one of the following values: ***StreamDigest.MD5*** — Digest the content using the MD5 Digest Algorithm (see RFC 1321). ***StreamDigest.SHA1*** — Digest the content using the SHA-1 Digest Algorithm (as defined in FIPS 180-4).

Return Value

ReadStream

Description

Allows the ***oStream*** object to be digested with the specified encoding type, ***cEncoder***. It returns a ***ReadStream*** object containing the computed digest of the ***oStream***. Typically, this is used to compute a digest to validate the integrity of the original data stream or as part of an authentication scheme for a web service.

To be used as a string, the result must be converted to a text format such as Base64 or hex using **SOAP.streamEncode**.

10.23.3.4 streamEncode(oStream, cEncoder)

Parameters

Parameter	Type	Description
oStream	Object	A stream object to be encoded with the specified encoding type.
cEncoder	String	A string specifying the encoding type. Permissible values are "***hex***" (for hex-encoded) and "***base64***" (base 64-encoded).

Return Value

ReadStream

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

This function encodes a stream object. Typically, it is used to pass data as part of a SOAP method when it must be encoded in Base 64 or hex encoding.

10.24 OCG

10.24.1 General

An OCG object represents an optional content group in a PDF file. Content in the file can belong to one or more optional content groups. Content belonging to one or more OCGs is referred to as optional content and its visibility is determined by the states (ON or OFF) of the OCGs to which it belongs. In the simplest case, optional content belongs to a single OCG, with the content being visible when the OCG is on and hidden when the OCG is off. More advanced visibility behavior can be achieved by using multiple OCGs and different visibility mappings.

Use the `Doc getOCGs` method to get an array of OCG objects for a PDF document.

See ISO 32000-2 for additional details on optional content groups.

10.24.2 OCG properties

10.24.2.1 General

Table 55: OCG object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `OCG`.

Table 55 — OCG object properties

Property	Type	Access	Description
constants.intents	Object	R	An OCG's intent array can contain arbitrary strings, but those contained in this object are the only ones that should be recognized by a PDF Processor.
constants.states	Object	R	The states object is used to set the initial state of the OCG (see `initState`).
initState	Boolean	Implementation-defined	This property is used to determine whether this OCG is on or off by default. See the `OCG.constants.states Object details` for possible values.
locked	Boolean	R/W	This property is used to determine whether this OCG is locked. If an OCG is locked, its on/off state cannot be toggled through the UI.
name	String	R/W	The text string seen in the UI for this OCG. It can be used to identify OCGs, although it is not necessarily unique.
state	Boolean	R/W	Represents the current on/off state of this OCG. Changing the state of an OCG does not modify the document, the OCGs revert to their initial state when the document is loaded again.

10.24.2.2 OCG.constants Details

10.24.2.2.1 General

Each instance of an OCG object inherits the `constants` property, which is a wrapper object for holding various constant values.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.24.2.2.2 OCG.constants.intents Object details

Table 56 — OCG.constants.intents properties

Property	Description
design	Designates a design intent in an OCG object.
view	Designates a view intent in an OCG object.

10.24.2.2.3 OCG.constants.states Object details

Table 57 — OCG.constants.states properties

Property	Description
on	Designates an OCG state of ON.
off	Designates an OCG state of OFF.

10.24.3 OCG methods

10.24.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type *OCG*.

10.24.3.2 getIntent()

Parameters

None

Return Value

Array of strings

Description

Returns this OCG's intent array.

An OCG will affect the visibility of content only if it has onstants.intents.view as an intent.

See also setIntent(aIntentArray) and theOCG.constants.intents Object details Object.

10.24.3.3 setAction(cExpr)

Parameters

Parameter	Type	Description
cExpr	String	The expression to be evaluated after the OCG state changes.

Return Value

undefined

Description

Registers an ECMAScript expression to be evaluated after every state change for this OCG.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Setting an action for an OCG does not dirty the document, therefore, the assignment of action must be made when the document is first opened, as document level script, or dynamically through user interaction.

NOTE This method will overwrite any action already defined for this OCG.

10.24.3.4 setIntent(aIntentArray)

Parameters

Parameter	Type	Description
aIntentArray	Array	An array of strings to be used as this OCG's intent array.

Return Value

undefined

Description

Sets this OCG's intent array. An OCG should only affect the visibility of content if this array contains constants.intents.view. See the `ocg`.constants.intents Object details Object for possible values.

See also **`getIntent`**() and the`ocg`.constants.intents Object details **`Object`**.

10.25 PrintParams

10.25.1 General

This is a generic object that controls printing parameters that affect documents printed using ECMAScript. Changing this object shall not change persistent user preferences or make permanent changes to the document.

The Doc **`print`** method takes a PrintParams object as its argument. You can obtain a **`PrintParams`** object from the Doc **`getPrintParams`** method. The returned object can then be modified.

Many of the **`PrintParams`** properties take integer constants as values, which you can access using the **`constants`** property. For example:

```
// Get the printParams object of the default printer var pp = this.getPrintParams();
// Set some properties pp.interactive = pp.constants.interactionLevel.automatic;
pp.colorOverride = pp.colorOverrides.mono;
// Print this.print(pp);
```

The **`constants`** properties are all integers allowing read access only.

10.25.2 PrintParams properties

10.25.2.1 General

Table 58: PrintParams properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type ***`PrintParams`***.

Table 58 — PrintParams properties

Property	Type	Access	Description
bitmapDPI	Integer	R/W	The dots per inch (DPI) to use when producing bitmaps or rasterizing transparency. The valid range is ***`1`*** to ***`9600`***. If the document protections specify a maximum printing resolution, the lower of the two values is used. The default is 300. Illegal values are treated as ***`300`***. See also **`gradientDPI`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 58 *(continued)*

Property	Type	Access	Description
constants	object	R	Each instance of a **`PrintParams`** object inherits this property, which is a wrapper that holds various constant values. The values are all integers allowing read access only. They are used as option values of some of the other properties of the **`PrintParams`** object, and their values are listed with the properties to which they apply.
DuplexType	Integer	R/W	Sets the printing duplex mode to simplex, duplex long edge, or duplex short edge feed. The duplex options are accessed through the **`constants.duplexType`** object. The properties of this object are listed in the table that follows. The selected printer should support duplex printing, if either **`DuplexFlipLongEdge`** or **`DuplexFlipShortEdge`** is used.
fileName	String	R/W	The device-independent path for a file name to be used instead of sending the print job to the printer (Print to File). The path may be relative to the location of the current document. The default value is the empty string (no file name). NOTE Printing to a file produces output suitable for the printer, for example, Postscript or GDI commands. When printerName is an empty string andfileName is specified, the current document is saved as a PostScript file.
firstPage	Integer	R/W	The first page number of the document to print. The number is 0-based. The first page of any document is ***0***, regardless of page number labels. The default value is ***0***, and values that are out of the document's page range are treated as ***0***. See also **`lastPage`**.
flags	Integer	R/W	A bit field of flags to control printing. These flags can be set or cleared using bitwise operations through the **`constants.flagValues`** object. Zero or more flags can be set; unsupported flags are ignored. The flags default to those set by user preferences. [TABLEREF] lists the base properties of the **`flagValues`** object.
gradientDPI	Integer	R/W	The dots per inch to use when rasterizing gradients. This value can generally be set lower than bitmapDPI because it affects areas to which the eye is less sensitive. It must be set from ***1*** to ***9600***. Illegal values are treated as ***150***. If the document protections specify a maximum printing resolution, the lower of the two values will be used. The default value is ***150***.
lastPage	Integer	R/W	The last 0-based page number of the document to print. The term "0-based" means the first page of any document is 0, regardless of page number labels. If the value is less than firstPage or outside the legal range of the document, this reverts to the default value. The default value is the number of pages in the document less one. See **`firstPage`** for an example.
NumCopies	Integer	R/W	Sets the number of copies to be printed for this print job.
pageHandling	Integer	R/W	Takes one of seven values specified by the **`constants.handling`** object. If set to an illegal value, it is treated as shrink. The default is ***`shrink`***.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 58 *(continued)*

Property	Type	Access	Description
pageSubset	Integer	R/W	Select even, odd, or all the pages to print. The value of `pageSubset` is set through the `constants.subsets` object. The default is `all`.
printAsImage	Boolean	R/W	Set to true to send pages as large bitmaps. This can be slow and more jagged looking but can work around problems with a printer's PostScript interpreter. Set `bitmapDPI` to increase or decrease the resolution of the bitmap. The default is `false`.
printContent	Integer	R/W	Sets the contents of the print job. The value of the printContents property is set through the `constants.printContents` object. The default is `doc`.
printerName	String	R/W	The name of the destination printer. By default, printerName is set to the name of the default printer. If printerName is set to an empty string, the printer used is either the last printer used by the PDF Processor (if printing occurred during the current session) or the System Default printer. When printerName is an empty string and fileName is a nonempty string, the current document is saved as a PostScript file. See also `printerNames`.
reversePages	Boolean	R/W	Set to `true` to print pages in reverse order (last to first). The default value is `false`.

10.25.2.2 PrintParams.constants Properties

10.25.2.2.1 General

Table 59 — PrintParams.constants properties

Constant object	Contains constant values for this property
duplexTypes	DuplexType
handling	pageHandling
printContents	printContent
flagValues	flags
subsets	pageSubset

10.25.2.2.2 constants.duplexTypes Object

Table 60 — constants.duplexTypes properties

Property	Description
Simplex	Prints on one side of the paper.
DuplexFlipLongEdge	Prints on both sides of the paper; the paper flips along the long edge.
DuplexFlipShortEdge	Prints on both sides of the paper; the paper flips along the short edge.

10.25.2.2.3 constants.flagValues object

Table 61 — constants.flagValues properties

Property	Description
setPageSize	Enable `setPageSize`. Choose the paper tray by the PDF page size.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 61 *(continued)*

Property	Description
suppressCenter	Do not center the page.
suppressRotate	Do not rotate the page.

10.25.2.2.4 constants.handling Object

Table 62 — constants.handling object properties

Property	Reader	Description
none		No page scaling is applied.
fit		Pages are enlarged or shrunk to fit the printer's paper.
shrink		Small pages are printed small, and large pages are shrunk to fit on the printer's paper.
tileAll	No	All pages are printed using tiling settings. This can be used to turn a normal-sized page into a poster by setting the tileScale property greater than 1.
tileLarge	No	Small or normal pages are printed in the original size and large pages are printed on multiple sheets of paper.
nUp		Pages are rescaled to print multiple pages on each printer page. Properties related to Multiple Pages Per Sheet printing are **`nUpAutoRotate`**, **`nUpNumPagesH`**, **`nUpNumPagesV`**, **`nUpPageBorder`** and **`nUpPageOrder`**.
booklet		Prints multiple pages on the same sheet of paper in the order required to read correctly when folded. The properties of the booklet object are used to set the parameters for booklet printing.

10.25.2.2.5 constants.subsets Object

Table 63 — constants.subsets object properties

Property	Description
all	Print all pages in the page range.
even	Print only the even pages. Page labels are ignored, and the document is treated as if it were numbered 1 through n, the number of pages.
odd	Print only the odd pages.

10.25.2.2.6 constants.printContents Object

Table 64 — constants.printContents object properties

Property	Description
doc	Print the document contents, not comments.
docAndComments	Print the document contents and comments.
formFieldsOnly	Print the contents of form fields only. Useful for printing onto pre-preprinted forms.

10.26 RDN

10.26.1 General

This generic object represents a Relative Distinguished Name. It is used by **`securityHandler.newUser`** and the **`certificate.issuerDN`** and **`certificate.subjectCN`** properties.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.26.2 RDN properties

Table 65: RDN properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `RDN`.

Table 65 — RDN properties

Property	Type	Access	Description
businessCategory	String	R	Business category
c	String	R	Country or region. Shall be a two-character upper case ISO 3166 standard string (for example, "*`US`*").
Cn	String	R	Common name (for example, "*`John Smith`*")
countryOfCitizenship	String	R	Country of citizenship
countryOfResidence	String	R	Country of residence
dateOfBirth	String	R	Date of birth (not supported by newUser)
Dc	String	R	Domain Component
dnQualifier	String	R	Distinguished Name Qualifier
e	String	R	Email address (for example, "*`jsmith@example.com`*")
gender	String	R	Gender
generationQualifier	String	R	Generation qualifier
givenName	String	R	Given name
initials	String	R	Initials
l	String	R	Locality or province
name	String	R	Name
nameAtBirth	String	R	Name at birth
o	String	R	Organization name
ou	String	R	Organizational unit
placeOfBirth	String	R	Place of birth
postalAddress	String	R	Postal address (not supported by `newUser`)
postalCode	String	R	Postal code
pseudonym	String	R	Pseudonym
serialNumber	String	R	Serial number

10.27 ReadStream

10.27.1 General

A ReadStream object is an object literal that represents a stream of data. It contains a method to allow reading the data stream.

10.27.2 ReadStream methods

Table 66: ReadStream parameter the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type `ReadStream`.

Table 66 — ReadStream parameter

Name	Syntax	Returns	Description
read	`read(nBytes)`	String	The read method takes the number of bytes to read and returns a hex-encoded string with the data from the stream. The `read` method is a destructive operation on the stream and returns a zero length string to indicate the end of the stream.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.28 security

10.28.1 General

The `security` object is a static ECMAScript object that exposes security-related PDF functions such as encryption and digital signatures. Security functions are performed using a `SecurityHandler` object, which is obtained from the `security` object using the `getHandler` method.

10.28.2 security constants

10.28.2.1 General

Several convenience strings are defined within the security object. The constants are held as properties of the wrapper objects listed below.

10.28.2.2 security.HandlerName Object

These are constants used when determining which handler to use.

Table 67 — security.HandlerName object properties

Property	Type	Access	Description
StandardHandler	String	R	This value can be specified in the handler property for a `SecurityPolicy` object that is based on the Standard (password-based) security handler.
PPKLiteHandler	String	R	This value can be specified in the handler property for a `SecurityPolicy` object that is based on the PPKLite (certificate-based) security handler. This value can also be passed to `security.getHandler` to create a new security context.
APSHandler	String	R	This the value specified in the handler property for a `SecurityPolicy` object that is based on the Adobe Policy Server security handler. This value can also be passed to `security.getHandler` to create a new security context.

10.28.2.3 security.EncryptTarget Object

These constants are used when determining what data a policy is encrypting. They can be used in the target property of the `SecurityPolicy` object.

Table 68 — security.EncryptTarget object properties

Property	Type	Access	Description
EncryptTargetDocument	String	R	The Security Policy encrypts the entire document when applied.
EncryptTargetAttachments	String	R	The Security Policy encrypts only the file attachments embedded within the document. This means the document can be opened, but attachments cannot be opened without providing the correct security authentication. It is used for eEnvelope workflows.

10.28.3 security Properties

Table 69: security object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `security`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 69 — security object properties

Property	Type	Access	Description
handlers	Array	R	An array containing the language-independent names of the available security handlers that can be used for encryption or signatures.
validateSignaturesOnOpen	Boolean	R/W	`[Security]` Gets or sets the user-level preference that causes signatures to be automatically validated when a document is opened.

10.28.4 security Methods

10.28.4.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type `security`. The details of parameters for each method that accepts parameters are provided in separate tables below.

10.28.4.2 chooseRecipientsDialog(oOptions)

Parameters

Parameter	Type	Description
oOptions	Object	A `DisplayOptions` object containing the parameters for the display options.

Return Value

Array of Group

Description

[Security]

10.28.4.2.1 Opens a dialog box that allows a user to choose a list of recipients. Returns an array of generic `Group` objects. DisplayOptions object

The DisplayOptions object contains the following properties.

Table 70 — DisplayOptions properties

Property	Description
bAllowPermGroups	Controls whether permissions can be set for entries in the recipient list. Default value is ***true***.
bPlaintextMetadata	If this property is specified, a check box is displayed to allow a user to select whether metadata is plaintext or encrypted. Its default value is the value of this property (***true*** or ***false***). If this property is not specified, the check box is not shown.
cTitle	The title to be displayed in the dialog box. The default is "***Choose Recipients***".
cNote	A note to be displayed in the dialog box. The default is not to show any note.
bAllowImportFromFile	Specifies whether the option is displayed that allows a user to import recipients from a file. The default value is ***true***.
bRequireEncryptionCert	If true, recipients will be required to include an encryption certificate. The default value is ***true***.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 70 *(continued)*

Property	Description
bRequireEmail	If true, recipients will be required to include an email address. The default value is ***false***.
bUserCert	If ***true***, the user will be prompted to provide his or her own certificate so that he or she can be included in the list of recipients. Setting this flag to ***true*** results in a prompt but does not require that the user provide a certificate.

10.28.4.3 chooseSecurityPolicy([oOptions])

Parameters

Parameter	Type	Description
oOptions	Object	(optional) A **`SecurityPolicyOptions`** object containing the parameters used for filtering the list of security policies returned. If not specified, all policies found are displayed.

Return Value

SecurityPolicy or null

Description

[Security]

Displays a dialog box to allow a user to choose from a list of security policies, filtered according to the options.

Returns a single **`SecurityPolicy`** object or **`null`** if the user cancelled the selection.

10.28.4.4 exportToFile(oObject, cDIPath)

Parameters

Parameter	Type	Description
oObject	Object	The Certificate object that is to be exported to disk.
cDIPath	String	The device-independent save path. The parameter cDIPath must be a safe path (see **`Safe path)`** and must end with the extension .cer.

Return Value

String

Description

[Security]

Exports a Certificate object to a local disk as a raw certificate file.

Returns the path of the file that was written, if successful.

See also **`importFromFile(cType`**[, cDIPath, bUI, cMsg]).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.28.4.5 getHandler(cName[, bUIEngine])

Parameters

Parameter	Type	Description
cName	String	The language-independent name of the security handler, as returned by the ***handlers*** property. Constant strings are defined for each of the valid handlers. See security constants, in particular the **security.HandlerName Object** object.
bUIEngine	Boolean	(optional) If ***true***, the method returns the existing security handler instance that is associated with the PDF Processor user interface (so that, for example, a user can log in through the user interface). If ***false*** (the default), returns a new engine.

Return Value

SecurityHandler

Description

[Security]

Obtains a **SecurityHandler** object. The caller can create as many new engines as desired and each call to getHandler creates a new engine. However, there is only one UI engine.

Returns the **SecurityHandler** object specified by **cName**. If the handler is not present, returns a **null** object.

10.28.4.6 getSecurityPolicies([bUI, oOptions])

Parameters

Parameter	Type	Description
bUI	Boolean	(optional) A flag controlling whether UI can be displayed. Default value is false.
oOptions	Object	(optional) A SecurityPolicyOptions object containing the parameters used for filtering the list of security policies returned. If not specified, all policies found are returned.

Return Value

Array of **SecurityPolicyOptions**

Description

[Security]

Returns the list of security policies currently available, filtered according to the options specified. The master list of security policies will be updated prior to filtering. The default SecurityHandler objects are used to retrieve the latest policies. If no policies are available or none meet the filtering restrictions, **null** will be returned.

Returns an array of **SecurityPolicyOptions** objects or **null**.

10.28.4.6.1 SecurityPolicyOptions object

The SecurityPolicyOptions object has the following properties:

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 71 — SecurityPolicyOptions object properties

Property	Description
bFavorites	If not passed, policies are not filtered based on whether a policy is a Favorite. If ***true***, only policies that are Favorites are returned. If ***false***, only policies that are not Favorites are returned.
cHandler	If not passed, policies are not filtered based on security filter. If defined, only policies that match the specified security filter are returned. The valid values are defined in security Constants (see the **security.HandlerName** Object). Only a single value can be passed.
cTarget	If not defined, policies are not filtered based on the target. If defined, only policies that match the specified target are returned. The valid values are defined in the **EncryptTarget** object of security constants. Only a single value can be passed.

10.28.4.7 importFromFile(cType[, cDIPath, bUI, cMsg])

Parameters

Parameter	Type	Description
cType	String	The type of object to be returned by this method. The only supported type is "***Certificate***".
cDIPath	String	(optional) When **bUI** is ***false***, this parameter is required and specifies the device-independent path to the file to be opened. If **bUI** is ***true***, this is the seed path used in the open dialog box.
bUI	Boolean	(optional) ***true*** if the user should be prompted to select the file that is to be imported. The default is ***false***.
cMsg	String	(optional) If **bUI** is ***true***, the title to use in the open dialog box. If cMsg is not specified, the default title is used.

Return Value

Certificate

Description

[Security]

Reads a raw data file and returns the data as an object with a type specified by cType. The file being imported must be a valid certificate. See also exportToFile(oObject, cDIPath).

10.29 SecurityHandler

10.29.1 General

SecurityHandler objects are used to access security handler capabilities such as signatures, encryption, and directories. Different security handlers have different properties and methods. This section documents the full set of properties and methods that security objects may have. Individual **SecurityHandler** objects may or may not implement these properties and methods.

SecurityHandler objects can be obtained using the **security.getHandler** method.

10.29.2 SecurityHandler properties

10.29.2.1 General

Table 72: SecurityHandler object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **SecurityHandler**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 72 — SecurityHandler object properties

Property	Type	Access	Description
appearances	Array	R	`[Security]` An array containing the language-dependent names of the available user-configured appearances for the specified security handler. Appearances are used to create the on-page visual representation of a signature when signing a signature field. The name of an appearance can be specified as a signature info object property when signing a signature field using the `Field` object `signatureSign` method. If a security handler does not support selection of appearances, this property will return ***`null`***.
digitalIDs	Object	R	**[Security]** The certificates that are associated with the currently selected digital IDs for this security handler.
directories	Array	R	**[Security]** An array of the available `Directory` objects for this `Security Handler`. `New Directory` objects can be created using the `newDirectory` method.
directoryHandlers	Array	R	**[Security]** An array containing the language-independent names of the available directory handlers for the specified security handler. Valid directory handler names are required when activating a new `Directory` object using its info property.
docDecrypt	Boolean	R	**[Security]** Returns *true*, if the security handler is capable of decrypting PDF files.
docEncrypt	Boolean	R	**[Security]** Returns *true*, if the security handler is capable of encrypting PDF files.
isLoggedIn	Boolean	R	**[Security]** Returns *true* if currently logged into this `SecurityHandler` object. See the `login([cPassword, cDIPath, oParams, bUI])` method. Different security handlers will have their own rules for determining the value of this property
loginName	String	R	**[Security]** The name associated with the actively selected signing digital ID for the security handler. This may require that the login method be called to select a signing credential. The return value is ***`null`*** if a signing credential is not selected or if the security handler does not support this property.
loginPath	String	R	**[Security]** The device-independent path to the user's profile file used to log in to the security handler. The return value is ***`null`*** if no one is logged in, if the security handler does not support this property, or if this property is irrelevant for the currently logged in user.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 72 *(continued)*

Property	Type	Access	Description
name	String	R	**[Security]** The language-independent name of the security handler. Example values for the Default Certificate, Windows Certificate, and Entrust Security Handlers are ***`Adobe.PPKLite`***, ***`Adobe.PPKMS`***, and ***`Entrust.PPKEF`***. All security handlers must support this property.
signAuthor	Boolean	R	**[Security]** Specifies whether the security handler is capable of generating certified documents. A certified document is signed with both a byte-range signature and an object signature. Object signatures, which are generated by traversing the document's object tree, are used to detect and prevent modifications to a document. See the **`mdpproperty`** of the ***`SignatureInfo`*** object for details regarding modification detection and prevention (MDP) settings.
signFDF	Boolean	R	**[Security]** Indicates that the security handler is capable of signing FDF files.
signInvisible	Boolean	R	**[Security]** Specifies whether the security handler is capable of generating invisible signatures.
signValidate	Boolean	R	**[Security]** Indicates whether the security handler is capable of validating signatures.
signVisible	Boolean	R	**[Security]** Specifies whether the security handler is capable of generating visible signatures.
uiName	String	R	**[Security]** The language-dependent string for the security handler. This string is suitable for user interfaces. All security handlers must support this property.
validateFDF	Boolean	R	**[Security]** Returns ***`true`***, if the security handler is capable of validating signatures over FDF files.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

10.29.2.2 SecurityHandler.digitalIDs Details

The return value is a generic object with the following properties:

Table 73 — SecurityHandler.digitalIDs object properties

Property	Type	Description
oEndUserSignCert	Certificate object	The certificate that is associated with the currently selected digital IDs that is to be used by this **`SecurityHandler`** object when signing. The property is undefined if there is no current selection.
oEndUserCryptCert	Certificate object	The certificate that is associated with the currently selected digital IDs that is to be used when encrypting a document with this **`SecurityHandler`** object. The property is undefined if there is no current selection.

© ISO 2020 – All rights reserved

Table 73 *(continued)*

Property	Type	Description
certs	Array of Certificate objects	An array of certificates corresponding to the list of all digital IDs that are available for this `SecurityHandler` object.
stores	Array of Strings	An array of strings, one for every `Certificate` object, identifying the store where the digital ID is stored. The string values are up to the security handler. For `Adobe.PPKLite` the valid values are '`PKCS12`' and '`MSCAPI`'.

10.29.3 SecurityHandler methods

10.29.3.1 login([cPassword, cDIPath, oParams, bUI])

Parameters

Parameter	Type	Description
cPassword	String	(optional) The password necessary to access the password-protected digital ID.
cDIPath	String	(optional) A device-independent path to the password-protected digital ID file or a PKCS#11 library.
oParams	Object	(optional) A `LoginParameters` object with parameters that are specific to a particular `SecurityHandler` object. The common fields in this object are described below. These fields include the `cDIPath` and `cPassword` values, thus allowing the parameter list to be expressed in different ways.
bUI	Boolean	(optional) Set to `true` if the user interface should be used to log the user in. If set to `false`, the default engine associated with the user interface is not used and the logged in state for the digital ID is not shown in the Security Settings dialog box. This attribute should be supported by all security handlers that support this method.

Return Value

boolean

Description

`[Security]`

Provides a mechanism by which digital IDs can be accessed and selected for a particular Security Handler. Through the user interface, a default digital ID can be selected that persists either eternally or for as long as the application is running. If such a selection has been made through the UI, it might not be necessary to log into a Security Handler prior to using the digital ID.

Parameters tend to be specific to a particular handler.

The parameters **cPassword** and **cDIPath** are provided for backward compatibility, or they can be included as properties of the **oParams** object. This latter method is the preferred calling convention.

See also `logout`(), `newUser`([cPassword,cDIPath, oRDN, oCPS, bUI, cStore]), and `loginName`.

Returns ***true*** if the login succeeded, ***false*** otherwise.

10.29.3.2 logout()

Parameters

None

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Return Value

boolean

Description

[Security]

Logs out for the **SecurityHandler** object.

Returns ***true*** if the logout succeeded, ***false*** otherwise.

10.29.3.3 newDirectory()

Parameters

None

Return Value

Directory

Description

[Security]

Returns a new **Directory** object. This object must be activated using its **info** property before it is marked for persistence and before it can be used for searches. Existing **directory** objects can be discovered using the **directories** property.

10.29.3.3.1 LoginParameters Object

This generic ECMAScript object contains parameters for the **login** method. It has the following properties:

Table 74 — LoginParameters object properties

Property	Type	Description
cDIPath	String	The path to a file that contains the digital ID or a PKCS#11 library.
cPassword	String	A password that is used to authenticate the user. This password may be used to access a password-protected digital ID file or a PKCS#11 device.
oEndUserSignCert	generic object	Selects a digital ID for the purpose of performing end-user signing. The value of this property is a **Certificate** object (or a generic object with the same property names as a **Certificate** object), which defines the certificate that is being selected. It may not be necessary to call this method for a particular handler All security handlers must be able to process the binary and SHA1Hash properties of this object. This object can be empty if **bUI** is ***true***.
cMsg	String	A message to display in the login dialog box, if **bUI** is ***true***.
cURI	String	URI used to connect to a server.
cUserId	String	Username used when connecting to a server.
cDomain	String	Domain name used when connecting to a server.
iSlotID	Integer	Specifies the slot ID of a PKCS#11 device to log into.
cTokenLabel	String	Specifies the token label of a PKCS#11 device to log into.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.29.3.4 newUser([cPassword,cDIPath, oRDN, oCPS, bUI, cStore])

Parameters

Parameter	Type	Description
cPassword	String	(optional) The password necessary to access the password-protected digital ID file.
cDIPath	String	(optional) The device-independent path to the password-protected digital ID file.
oRDN	Object	The relative distinguished name (RDN) as an RDN object, containing the issuer or subject name for a certificate. The only required field is cn. If the country c is provided, it shall be two characters, using the ISO 3166 standard (for example, "US").
oCPS	Object	(optional) A generic object containing certificate policy information that will be embedded in the Certificate Policy extension of the certificate. The object has these properties: **`oid`** is required and indicates the certificate policy object identifier. **`url`** (optional) is a URL that points to detailed information about the policy under which the certificate has been issued **`notice`** (optional) is an abridged version of the information, embedded in the certificate.
bUI	Boolean	(optional) If ***`true`***, the user interface can be used to enroll. This parameter is supported by all security handlers that support this method.
cStore	String	(optional) A string identifying the store where the generated credential has to be stored.

Return Value

Boolean

Description

`[Security]`

Supports enrollment with some security handlers by creating a new self-sign credential.

Returns *`true`* if successful, throws an exception if not successful.

10.29.3.5 setPasswordTimeout(cPassword, iTimeout)

Parameters

Parameter	Type	Description
cPassword	String	The password needed to set the timeout value.
iTimeout	Integer	The timeout value, in seconds. Set to *`0`* for always expire (that is, password always required). Set to ***`0x7FFFFFFF`*** for never expire.

Return Value

undefined

Description

`[Security]`

Sets the number of seconds after which password should expire between signatures. For this handler the default timeout value for a new user is *`0`* (password always required).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Throws an exception if the user has not logged in to the security handler, or if the call is unsuccessful for any other reason.

10.30 SecurityPolicy

10.30.1 General

[Security]

The **Security Policy** object represents a group of security settings used to apply encryption to a document. It is acquired as the return value of both **getSecurityPolicies** and **chooseSecurityPolicy**.

10.30.2 SecurityPolicy properties

Table 75: SecurityPolicy object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **SecurityPolicy**.

Table 75 — SecurityPolicy object properties

Property	Type	Acces	Description
policyID	String	R	A generated string that uniquely identifies the Security Policy. It is not intended to be human-readable. This may be set to a known policy ID on a newly created **SecurityPolicy** object to force any method using this policy to retrieve the correct security settings before applying the policy.
Name	String	R	The policy name used in the UI. It may be localized.
Description	String	R	The policy description used in the UI. It may be localized.
Handler	String	R	An enumerated value representing the security handler implementing this Security Policy. The possible values are defined in the **HandlerName** object.
target	String	R	An enumeration value representing the target data to be encrypted. The possible values are defined in the **EncryptTarget**.

10.31 SignatureInfo

10.31.1 General

A generic ECMAScript object that contains the properties of a digital signature. Some properties are supported by all handlers, and additional properties can be supported.

The **SignatureInfo** object is returned by the **Field** object methods **signatureValidate** and **signatureInfo** and is passed to the **FDF** object methods **signatureSign** and **signatureValidate**.

Writable properties can be specified when signing the object.

10.31.2 SignatureInfo Base properties

All handlers define the following properties.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 76 — SignatureInfo base object properties

Property	Type	Access	Description
buildInfo	Object	R	An object containing software build and version information for the signature. This reference needs to be eliminated. The format of this object is described in the technical note PDF Signature Build Dictionary Specification on the Adobe Solutions Network website.
date	Date object	R	The date and time that the signature was created, returned as an ECMAScript **`Date`** object.
dateTrusted	Boolean	R	A Boolean value that indicates whether the date is from a trusted source. If this value is not present, the date should be assumed to be from an untrusted source (for example, the signer's computer system time).
digestMethod	String	R/W	The digest method to be used for signing. The valid values are **`SHA1`**, **`SHA256`**, **`SHA384`**, **`SHA512`** and **`RIPEMD160`**. It is important that the caller knows that a particular signature handler can support this digest method.
handlerName	String	R	The language-independent name of the security handler that was specified as the Filter attribute in the signature dictionary. It is usually the name of the security handler that created the signature, but can also be the name of the security handler that the creator wants to be used when validating the signature.
handlerUserName	String	R	The language-dependent name corresponding to the security handler specified by handlerName. It is available only when the named security handler is available.
handlerUIName	String	R	The language-dependent name corresponding to the security handler specified by handlerName. It is available only when the named security handler is available.
location	String	R/W	Optional user-specified location when signing. It can be a physical location (such as a city) or hostname.
mdp	String	R/W	The Modification Detection and Prevention (MDP) setting that was used to sign the field or FDF object being read, or the MDP setting to use when signing. Values are: **`allowNone allowAll default defaultAndComments`** See Modification Detection and Prevention (MDP) Values for details. The value of allowAll, the default, means that MDP is not used for the signature, resulting in this not being a certification signature.
name	String	R	Name of the user that created the signature.
numRevisions	Number	R	The number of revisions in the document. Used only for signature fields.
reason	String	R/W	The user-specified reason for signing.
revision	Number	R	The signature revision to which this signature field corresponds. Used only for signature fields.
sigValue	String	R	Raw bytes of the signature, as a hex encoded string.
status	Number	R	The validity status of the signature, computed during the last call to **`signatureValidate`**. See **`status and idValidity`** properties.
statusText	String	R	The language-dependent text string, suitable for user display, denoting the signature validity status, computed during the last call to the **`signatureValidate`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 76 *(continued)*

Property	Type	Access	Description
subFilter	String	R/W	The format to use when signing. Consult ISO 32000-2 for a complete list of supported values. The known values used for public key signatures include ***adbe.pkcs7.sha1***, ***adbe.pkcs7.detached***, and ***adbe.x509.rsa_sha1***. It is important that the caller know that a particular signature handler can support this format.
timeStamp	String	W	The URL of the server for time-stamping the signature. The only schemes and transport protocols supported for fetching time stamps are http or https. This property is write-only. If the signature is time stamped, during verification the property **dateTrusted** will be set to ***true*** (provided the time stamp signature is trusted) and the **verifyDate** and the signing date will be the same.
verifyDate	Date object	R	The date and time that the signature was either signed or verified (if the signature was verified). If the signature is signed with a timestamp, **verifyDate** should be the signing date; if the signature is signed without a timestamp, **verifyDate** will be the verifying date. The property returns an ECMAScript **Date** object.
verifyHandlerName	String	R	The language-independent name of the security handler that was used to validate this signature. This will be ***null*** if the signature has not been validated (that is, if the status property has a value of ***1***).
verifyHandlerUIName	String	R	The language-dependent name corresponding to the security handler specified by **verifyHandlerName**. This will be ***null*** if the signature has not been validated, that is, if the status property has a value of ***1***).

10.31.3 SignatureInfo object public key security handler properties

10.31.3.1 General

Public key security handlers may define the following additional properties.

Table 77: SignatureInfo object properties lists additional properties that PDF Processors supporting this standard shall implement for objects of type **SignatureInfo** that represent public key security handlers.

Table 77 — SignatureInfo object properties

Property	Type	Access	Description
appearance	String	W	The name of the user-configured appearance to use when signing this field. PPKLite and PPKMS use the standard appearance handler. In this situation, the appearance names can be found in the signature appearance configuration dialog box of the Security user preferences. The default, when not specified, is to use the Standard Text appearance. Used only for visible signature fields.
certificates	Array	R	Array containing a hierarchy of certificates that identify the signer. The first element in the array is the signer's certificate. Subsequent elements include the chain of certificates up to the certificate authority that issued the signer's certificate. For self-signed certificates this array will contain only one entry.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 77 *(continued)*

Property	Type	Access	Description
contactInfo	String	R/W	The user-specified contact information for determining trust. For example, it can be a telephone number that recipients of a document can use to contact the author. This is not recommended as a scalable solution for establishing trust.
byteRange	Array	R	An array of numbers indicating the bytes that are covered by this signature.
docValidity	Number	R	The validity status of the document byte range digest portion of the signature, computed during the last call to **`signatureValidate`**. All PDF document signature field signatures include a byte range digest. See **`Validity Values`** for details of the return codes.
idPrivValidity	Number	R	The validity of the identity of the signer. This value is specific to the handler. This value is ***0*** unless the signature has been validated, that is, if the status property has a value of ***1***.
idValidity	Number	R	The validity of the identity of the signer as number. See the return codes of the **`idValidity`** property in the table status and idValidity propertiesstatus and idValidity properties.
objValidity	Number	R	The validity status of the object digest portion of the signature, computed during the last call to **`signatureValidate`**. For PDF documents, signature field certification signatures and document-level application rights signatures include object digests. All FDF files are signed using object digests. See **`Validity Values`** for details of the return codes.
revInfo	Object	R	A generic object containing two properties: **`CRL`** and **`OCSP`**. These properties are arrays of hex-encoded strings, where each string contains the raw bytes of the revocation information that was used to carry out revocation checking of a certificate. For **`CRL`**, the strings represent CRLs. For **`OCSP`**, the strings represent OCSP responses. These properties are populated only if the application preference to populate them is turned on, because this data can potentially get very large.
trustFlags	Number	R	The bits in this number indicate what the signer is trusted for. The value is valid only when the value of the **`status`** property is ***4***. These trust settings are derived from the trust setting in the recipient's trust database). Bit assignments are: ***1*** Trusted for signatures ***2*** Trusted for certifying documents ***3*** Trusted for dynamic content such as multimedia ***4*** Reserved ***5*** The ECMAScript in the PDF file is trusted to operate outside the normal PDF restrictions
password	String	W	A password required as authentication when accessing a private key that is to be used for signing. This may or may not be required, dependent on the policies of the security handler.

10.31.3.2 status and idValidity properties

Table 78: SignatureInfo object status and idValidity properties lists the codes returned by the **`SignatureInfo`** Object, **`status`** and **`idValidity`** properties.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 78 — SignatureInfo object status and idValidity properties

Status code	Description
-1	Not a signature field.
0	Signature is blank or unsigned.
1	Unknown status. This occurs if the signature has not yet been validated; for example, if the document has not completed downloading over a network connection.
2	Signature is invalid.
3	Signature is valid but the identity of the signer could not be verified.
4	Signature is valid and the identity of the signer is valid.

10.31.3.3 Validity Values

The following codes are returned by the **docValidity** and **objValidity** properties. (See **SignatureInfo** object public key security handler properties). They provide a finer granularity of the validity of the signature than the status property.

Table 79 — SignatureInfo Validity Values

Status code	Description
kDSSigValUnknown	Validity not yet determined.
kDSSigValUnknownTrouble	Validity could not be determined because of errors encountered during the validation process.
kDSSigValUnknownBytesNotReady	Validity could not be determined because all bytes are not available, for example, when viewing a file in a web browser. Even when bytes are not immediately available, this value may not be returned if the underlying implementation blocks when bytes are not ready.
kDSSigValInvalidTrouble	Validity for this digest was not computed because there were errors in the formatting or information contained in this signature. There is sufficient evidence to conclude that the signature is invalid.
kDSSigValUnused	The validity for this digest is not used (for example, no document validity if no byte range).
kDSSigValJustSigned	The signature was just signed, so it is implicitly valid.
kDSSigValFalse	The digest or validity is invalid.
kDSSigValTrue	The digest or validity is valid.

10.31.3.4 Private Validity Values

Verification of the validity of the signer's identity is specific to the handler that is being used to validate the identity. This value may contain useful information regarding an identity. The identity is returned in the **idPrivValidity** property. This value is mapped to an **idValidity** value that is common across all handlers.

Table 80 — SignatureInfo Private Validity Values

Status Code	idValidity Mapping	Description
kIdUnknown	**1 (unknown)**	Validity not yet determined.
kIdTrouble	**1 (unknown)**	Could not determine validity because of errors, for example, internal errors, or could not build the chain, or could not check basic policy.
kIdInvalid	**2 (invalid)**	Certificate is invalid: not time nested, invalid signature, invalid/unsupported constraints, invalid extensions, chain is cyclic.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 80 *(continued)*

Status Code	idValidity Mapping	Description
`kIdNotTimeValid`	`2 (invalid)`	Certificate is outside its time window (too early or too late).
`kIdRevoked`	`2 (invalid)`	Certificate has been revoked.
`kIdUntrustedRoot`	`1 (unknown)`	Certificate has an untrusted root certificate.
`kIdBrokenChain`	`2 (invalid)`	Could not build a certificate chain up to a self-signed root certificate.
`kIdPathLenConstraint`	`2 (invalid)`	Certificate chain has exceeded the specified length restriction. The restriction was specified in Basic Constraints extension of one of the certificates in the chain.
`kIdCriticalExtension`	`1 (unknown)`	One of the certificates in the chain has an unrecognized critical extension.
`kIdJustSigned`	`4 (valid)`	Just signed by user (similar to kIdIsSelf)
`kIdAssumedValid`	`3 (idunknown)`	Certificate is valid to a trusted root, but revocation could not be checked and was not required.
`kIdIsSelf`	`4 (valid)`	Certificate is my credential (no further checking was done).
`kIdValid`	`4 (valid)`	Certificate is valid to a trusted root (in the Windows or PDF Processor Address Book).
`kIdRevocationUnknown`	`?`	Certificate is valid to a trusted root, but revocation could not be checked and was required by the user.

10.31.4 Modification Detection and Prevention (MDP) Values

Modification detection and prevention (MDP) settings control which changes are allowed to occur in a document before the signature becomes invalid. Changes are recorded outside of the byte range, for signature fields, and can include changes that have been incrementally saved as part of the document or changes that have occurred in memory between the time that a document is opened and when the signature is validated. MDP settings may only be applied to the first signature in a document. Use of MDP will result in a certification signature. MDP has one of the following four values.

— ***`allowAll`*** — Allow all changes to a document without any of these changes invalidating the signature. This results in MDP not being used for the signature.

— ***`allowNone`*** — Do not allow any changes to the document without invalidating the signature. Note that this will also lock down the author's signature.

— ***`default`*** — Allow form field fill-in if form fields are present in the document. Otherwise, do not allow any changes to the document without invalidating the signature.

— ***`defaultAndComments`*** — Allow form field fill-in if form fields are present in the document and allow annotations (comments) to be added, deleted or modified. Otherwise, do not allow any changes to the document without invalidating the signature. Note that annotations can be used to obscure portions of a document and thereby affect the visual presentation of the document.

10.32 SOAP

10.32.1 General

The `SOAP` object allows remote procedure calls to be made to, or sends an XML Message to, a remote server from ECMAScript.

The SOAP 1.1 protocol (see http://www.w3.org/TR/SOAP/) is used to marshal ECMAScript parameters to a remote procedure call (either synchronously or asynchronously) and to unmarshal the result as a ECMAScript object. The `SOAP` object also has the ability to communicate with web services, described by the Web Services Description Language (WSDL—see http://www.w3.org/TR/wsdl).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.32.2 SOAP properties

Table 81: SOAP object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type SOAP.

Table 81 — SOAP object properties

Property	Type	Access	Description
wireDump	Boolean	R/W	If true, synchronous SOAP requests will cause the XML Request and Response to be dumped to the ECMAScript Console. This is useful for debugging SOAP problems.

10.32.3 SOAP methods

10.32.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type SOAP.

10.32.3.2 connect(cURL)

Parameters

Parameter	Type	Description
cURL	String	The URL of a WSDL document. It must be an HTTP or HTTPS URL.

Return Value

WSDL service proxy object

Description

Converts the URL of a WSDL document (cURL) to a ECMAScript object with callable methods corresponding to the web service.

The parameters to the method calls and the return values obey the rules specified for the SOAP. request method.

10.32.3.2.1 WSDL Service Proxy Object

Calls to SOAP.connect returns a WSDL Service Proxy object with ECMAScript methods corresponding to each operation in the WSDL document provided at the URL.

The parameters required for the method depend on the WSDL operation you are calling and how the operation encodes its parameters.

— If the WSDL operation is using the SOAP RPC encoding (as described in Section 7 of the SOAP 1.1 Specification), the arguments to the service method are the same as the parameter order in the WSDL document.

— If the WSDL service is using the SOAP document/literal encoding, the function will have a single argument indicating the request message. The argument may be an ECMAScript object literal describing the message or it may be either a string or a ReadStream object with an XML fragment describing the message. The return value of the service method will correspond to the return value of the WSDL operation.

The ECMAScript function objects corresponding to each web service method use the properties given in Table 82 if they are set. The default is for none of the properties to be set.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 82 — WSDL Service Proxy object properties

Property	Description
asyncHandler	Indicates that the web service method should be performed asynchronously. The property corresponds to the **`oAsync`** parameter of SOAP.request.
requestHeader	Indicates that the web service method should include a SOAP Header in the request. The property corresponds to the **`oReqHeader`** parameter of SOAP.request.
responseHeader	Indicates that the web service method should return a SOAP Header from the response. The property corresponds to the **`oRespHeader`** parameter of SOAP.request.
authenticator	Indicates how authentication should be handled for the web service method. The property corresponds to the **`oAuthenticate`** parameter of SOAP.request.

10.32.3.2.2 Exceptions

SOAP Faults cause a **`SOAPError`** exception to be thrown. If there is a problem at the networking level, such as an unavailable endpoint, a **`NetworkError`** is thrown. See the *request*(...) method for more information.

10.32.3.3 queryServices(cType, oAsync[, aDomains])

Parameters

Parameter	Type	Description
cType	String	The DNS SRV Service Name to search for. Some possible examples are: "***`http`***" — Locate web servers "***`ftp`***" — Locate FTP servers
oAsync	Object	A **`notification`** object that is notified when services are located on the network or when services that had previously been reported are removed. The notification methods are not called until the **`queryServices`** method returns and are called during idle processing. The **`oAsync`** parameter should implement the following methods: ***`addServices`*** — This method is called when available services matching the query are located. The parameter is an array of **`Service Description`** objects for the services that have been added. ***`removeServices`*** — This method is called when services that had previously been introduced by calling the addServices notification method are no longer available. The parameter is an array of **`Service Description`** objects for the services that have been removed.
aDomains	Array	(optional) An array of domains that the query should be made for. The only valid domains are: ***`ServiceDiscovery.local`*** — Search for services in the local networking link using Multicast DNS (mDNS). This is useful for finding network services in an ad hoc networking environment, but network services will only be located within the scope of the current network router. ***`ServiceDiscovery.DNS`*** — Search for services in the default DNS domain using unicast DNS. This is useful for locating network services in the context of a DNS server, but typically requires IS assistance to register a service and is less dynamic.

Return Value

Service query object

Description

`[Security]`

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Locates network services that have published themselves using DNS Service Discovery (DNS-SD). This method can locate services that have registered using Multicast DNS (mDNS) for location on a local networking link or through unicast DNS for location within an enterprise. The results of service location are always returned asynchronously and the query continues (with notification as services become available or unavailable) until it is stopped.

The result of querying for services is a set of service names that can be bound when needed by calling resolveService.

Services can either use a third-party mDNS responder to be located in the local network link or register themselves in a DNS server (either statically or dynamically) to be located within an enterprise networking environment.

10.32.3.3.1 Service Query Object

Calls to `SOAP.queryServices` return a service query object that manages the duration of the query. The query will continue until one of the following conditions is met.

— The `service query` object returned from `queryServices` is garbage collected.

— The stop `method` of the `service query` object returned from `queryServices` is called.

The `service query` object provides a single method, described in Table 83: SOAP.queryServices object methods.

Table 83 — SOAP.queryServices object methods

Method	Description
`stop`	Causes the query to terminate. This method can be called from a notification callback but the operation will not stop until idle processing time.

10.32.3.3.2 Exceptions

Implementers of this standard should throw appropriate exceptions if calls to `SOAP.queryServices`, or any of its associated processes, fail to complete normally. For example, if a network connection is not available, implementers should throw an exception indicating that no network connection is available.

10.32.3.3.3 Service Description Object

The `service description` object passed to `addServices` and `removeServices` have the properties given in Table 84:

Table 84 — Service description properties

Property	Description
name	The Unicode display name of the service.
domain	The DNS domain in which the service was located. If the service was located in the local networking link, the domain name will be "`local`".
type	The DNS SRV Service Name of the service that was located – this will be the same as the cType parameter passed to queryServices. This can be useful when the same notification callback is being used for multiple queries.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.32.3.4 resolveService(cType, cDomain, cService, oResult)

Parameters

Parameter	Type	Description
cType	String	The DNS SRV Service Name to resolve.
cDomain	String	The domain that the service was located in.
cService	String	The service name to resolve.
oResult	Object	An object that will be called when the service is resolved. See **`Additional notes on the oResult parameter`**.

Return Value

Service query object

Description

`[Security]`

Allows a service name to be bound to a network address and port in order for a connection to be made. The connection information is returned asynchronously and should be treated as temporary since the network location of a service may change over time (for example, if a DHCP lease expires or if a service moves to a new server).

NOTE This method is deprecated, new services should use **`Net.Discovery.resolveService`**.

10.32.3.4.1 Service Query Object

Calls to **`SOAP.resolveService`** return a service query object that manages the duration of the resolve. The resolve will continue until one of the following conditions is met.

— The **`service query`** object returned from **`resolveService`** is garbage collected.

— The resolve method of the **`oResult`** object is called indicating that the operation completed (either by resolving the service, error, or a timeout).

— The stop method of the service query object returned from **`resolveService`** is called.

The **`service query`** object provides a single method, described in Table 85: Service query object methods.

Table 85 — Service query object methods

Method	Description
`stop`	Causes the resolve to terminate. This method can be called from a notification callback but the operation will not stop until idle time.

10.32.3.4.2 Exceptions

Implementers of this standard should throw appropriate exceptions if calls to **`SOAP.queryServices`**, or any of its associated processes, fail to complete normally. For example, if a network connection is not available, implementers should throw an exception indicating that no network connection is available.

10.32.3.4.3 Additional notes on the oResult parameter

The **`oResult`** object is a notification object that will be called when the service is resolved. The notification methods will not be called until the **`resolveService`** method returns and are called during idle processing. The **`oResult`** parameter should implement the method given in Table 86:

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

 © ISO 2020 – All rights reserved

Table 86 — oResult parameter method

Method	Description
resolve	This method is called with two parameters (**`nStatus`** and **`oInfo`**) when the service is resolved or if it cannot be resolved. The parameter **`nStatus`** is the state indicating if the service could be resolved (see below). If the service was successfully resolved, the **`oInfo`** object, an instance of the **`ServiceInfo`** object (see below), specifies the connection information.

The **`nStatus`** parameter passed to the resolve method can have one of the values given in Table 87:

Table 87 — nStatus parameters

Value	Description
0	The service was successfully resolved.
1	The service timed out before being resolved. The default timeout in a PDF Processor should be 60 seconds.
-1	There was a networking error trying to resolve the service.

The ServiceInfo object passed to the resolve method has the properties Table 88.

Table 88 — ServiceInfo object properties

Property	Description
target	The IP address or DNS name of the machine supplying the service.
port	The port on the machine supplying the service.
info	An object with name - value pairs that the service has supplied. For example, in the case of an HTTP service, the path property will contain the path on the web service so that the service URL would be http://<target>:<port>/<info["path"]>).

10.32.3.5 request(...)

Parameters

Parameter	Type	Description
cURL	String	The URL for a SOAP HTTP endpoint. The URL method must be one of: *`http`* — Connect to a server at a URI on a port. For example, *`http://serverName:portNumber/URI`* *`https`* — Connect to a secured (SSL) server at a URI on a port. For example, *`https://serverName:portNumber/URI`*
oRequest	Object	An object that specifies the remote procedure name and parameters or the XML message to send. See **`Additional notes on the oResult parameter.`**

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
oAsync	Object	(optional) An object that specifies that the method invocation will occur asynchronously. The default is for the request to be made synchronously. The **`oAsync`** object literal must have a function called response that will be called with two parameters (**`oResult`** and **`cURI`**) when the response returns. oResult is the same result object that would have been returned from the request call if it was called synchronously. **`cURI`** is the URI of the endpoint that the request was made to. The **`oAsync`** object response callback has the following parameters: ***`response`*** — The response object from the SOAP request. uri — The URI that the SOAP request was made to. ***`exception`*** — An exception object (see the exceptions below) if there was an error, null otherwise. ***`header`*** — A response SOAP header (see the description of the **`oRespHeader`** parameter) or null if there are no response headers.
cAction	String	(optional) In SOAP 1.1, this parameter is passed as the SOAPAction header. In SOAP 1.2, this parameter is passed as the **`action`** parameter in the Content-Type header. The default is for the action to be an empty string. The SOAPAction is a URN written to an HTTP header used by firewalls and servers to filter SOAP requests. The WSDL file for the SOAP service or the SOAP service description will usually describe the SOAPAction header required (if any).
bEncoded	Boolean	(optional) Encoded the request using the SOAP Encoding described in the SOAP Specification. Otherwise, the literal encoding is used. The default is ***`true`***.
cNamespace	String	(optional) A namespace for the message schema when the request does not use the SOAP Encoding. The default is to omit the schema declaration.
oReqHeader	Object	(optional) An object that specifies a SOAP header to be included with the request. The default is to send a request with only a SOAP Body. The object is specified in the same way as the **`oRequest`** object except for two additional properties that can be specified in the request description: ***`soapActor`*** — The recipient (or actor specified as a URI) that the SOAP header should be processed by. The default is the first recipient to process the request. ***`soapMustUnderstand`*** — A Boolean value indicating that the request body cannot be interpreted if this header type is not understood by the recipient. The default is that understanding the header is optional.
oRespHeader	Object	(optional) An object that will be populated with the SOAP headers returned when the method completes if the function is being called synchronously (the header will be passed to the **`oAsync`** callback method otherwise). The default is for the headers not to be returned. See the description of the **`cResponseStyle`** parameter for the object format.
cVersion	String	(optional) The version of the SOAP protocol to use when generating the XML Message – either 1.1 or 1.2. The default is to use "***`SOAPVersion.version_1_1`***".

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
oAuthenticate	Object	(optional) An object that specifies how to handle HTTP authentication or credentials to use for Web Service Security. The default is to present a user interface to the user to handle HTTP authentication challenges for BASIC and DIGEST authentication modes. The **`oAuthenticate`** object can have the following properties: **`Username`** — A string containing the username to use for authentication. **`Password`** — A string containing the credential to use. **`UsePlatformAuth`** — A Boolean value indicating that platform authentication should be used. If ***`true`***, Username and Password are ignored and the underlying platform networking code is used. This may cause an authentication UI to be shown to the user and/or the credentials of the currently logged in user to be used. The default is ***`false`*** and is only supported on the Windows platform
cResponseStyle	String	(optional) An enumerated type indicating how the return value (in the case of the SOAP Body) and the oRespHeader object (in the case of a SOAP header) will be structured: ***`SOAPMessageStyle.JS`*** — (Default) The response will be an object describing the SOAP Body (or SOAP Header) of the returned message. This is recommended when using the SOAP encoding for the request but is not ideal when using the literal encoding – using the XML or Message style is better. ***`SOAPMessageStyle.XML`*** — The response will be a stream object containing the SOAP Body (or SOAP Header) as an XMLfragment. If there are any attachments associated with the response, the Stream object will have an object property oAttachments. The object keys are the unique names for the attachment parts and the value must be a Stream object containing the attachment contents. ***`SOAPMessageStyle.Message`*** — The response will be an object describing the SOAP Body (or SOAP Header) corresponding to the XML Message. This differs from the ECMAScript response style in the following ways: XML Elements are returned as an array of objects rather than an object to maintain order and allow elements with the same name. XML Attributes are preserved using the **`soapAttributes`** property. Namespaces are processed and returned in the **`soapName`** and **`soapQName`** properties. The content of an element is in the **`soapValue`** property.
cRequestStyle	String	(optional) Allows the interpretation of oRequest to be altered. The following values are permitted: ***`SOAPRequestStyle.SOAP`*** — (the default) The request is made using the SOAP messaging model. ***`SOAPRequestStyle.RawPost`*** — The oRequest parameter is used as the request body for an HTTP Post. oRequest must be a ReadStream object. If this method is called within the context of a document, the document must be open in the browser. Additionally, the origination URL of the document (scheme, server, and port) must match the origination of the **`cURL`** parameter. The response is a **`ReadStream`** object with the response from the request.
cContentType	String	(optional) Allows the HTTP content-type header to be specified. The default is to use the SOAP messaging HTTP content-type.

Return Value

Response or nothing

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Initiates a remote procedure call (RPC) or sends an XML message to a SOAP HTTP endpoint. The method either waits for the endpoint to reply (synchronous processing) or calls a method on the notification object (asynchronous processing).

Returns a **`response`** object if the method was called synchronously (that is, there is no **`oAsync`** parameter) or nothing if the method was called asynchronously. See the description of **`cResponseStyle`** for the object description.

10.32.3.5.1 oRequest Parameter Details

The **`oRequest`** object supports the properties given in Table 89:

Table 89 — oRequest parameters

Parameter	Type	Description
soapType	String	The SOAP type that will be used for the value when generating the SOAP message. It is useful when a datatype is needed other than the automatic datatype binding described above. The type should be namespace qualified using the <namespace>:<type> notation, for example http://mydomain/types:myType However, the xsd (XMLSchema Datatypes), xsi (XMLSchema Instance), and SOAP-ENC (SOAP Encoding) namespaces are implicitly defined in the SOAP message, so the soapType can use them, as in xsd:int for the XMLSchema Datatype Integer type.
soapValue	String or ReadStream	The value that will be used when generating the SOAP message. It can be a string or a **`ReadStream`** object. soapValue is passed unescaped (that is, not XML Entity escaped). For example, "<" is not converted to "<" in the XML Message. Consequently, soapValue can be a raw XML fragment that will be passed to the XML Message. **`soapValue`** can now also be an array of nodes that are an ordered set of children of the node in the request message.

soapName	String	The element name that will be used when generating the SOAP message instead of the key name in the object literal.
		
		For example, integers are not supported in ECMAScript, but an integer parameter to a **`SOAP`** method can be constructed as follows:
		
		`var oIntParameter = { soapType: "xsd:int", soapValue: "1" };`
		`Later, the oRequest parameter for the SOAP. request method might be this:`
		`oRequest: { "http://soapinterop.org/:echoInteger": { inputInteger: oIntParameter } }`

soapAttributes	Object	An object specifying XML attributes to be included when building the element corresponding to the request node. The object keys are the attribute names and the corresponding value is the attribute value.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 89 *(continued)*

Parameter	Type	Description
soapQName	Object	An object specifying the namespace qualified name (QName) of the request node. For example, in the element `<ns:local xmlns:ns="urn:example.org">`, the element name is a QName consisting of a local name ("`local`") and a namespace ("`urn:example.org`"). This object has two properties: **`localName`** — A string indicating the local name of the QName. **`nameSpace`** — A string indicating the namespace of the QName.
soapAttachment	Boolean	Specifies whether the **`soapValue`** contents of the node should be encoded as an attachment according to the SwA specification. The **`soapValue`** must be a stream if the corresponding soapAttachment property is **`true`**, otherwise an exception will be thrown.
soapParamOrder	Array	An array indicating the order in which RPC parameters should be sent to the server. The array is a set of strings with the parameter names. This value is only applicable when **`bEncoding`** is ***`true`***.

10.32.3.5.2 Mapping of Data Types

The SOAP types in the result are mapped to ECMAScript types as given in Table 90:

Table 90 — SOAP ECMAScript types

SOAP type	ECMAScript type
xsd:string	**`String`**
xsd:integer	**`Number`**
xsd:float	**`Number`**
xsd:dateTime	**`Date`**
xsd:boolean	**`Boolean`**
xsd:hexBinary	**`ReadStream object`**
xsd:base64Binary	**`ReadStream object`**
SOAP-ENC:base64	**`ReadStream object`**
SOAP-ENC:Array	**`Array`**
No Type Information	**`String`**

When passing parameters to a remote procedure, ECMAScript types are bound to SOAP types automatically as listed in Table 91: ECMAScript SOAP types:

Table 91 — ECMAScript SOAP types

ECMAScript type	SOAP type
`String`	xsd:string
`Number`	xsd:float
`Date`	xsd:dateTime
`Boolean`	xsd:boolean
`ReadStream object`	SOAP-ENC:base64
`Array`	SOAP-ENC:Array

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 91 *(continued)*

ECMAScript type	SOAP type
`Other`	No type information

NOTE The xsd namespace refers to the XML Schema Datatypes namespace http://www.w3.org/2001/XMLSchema. The SOAP-ENC namespace refers to the SOAP Encoding namespace http://schemas.xmlsoap.org/soap/encoding/.

10.32.3.5.3 Exceptions

— `SOAPError` is thrown when the SOAP endpoint returns a `SOAPFault`. The `SOAPError` Exception object has the properties given in Table 92:

Table 92 — SOAPError properties

Property	Description
`faultCode`	A string indicating the SOAP Fault Code for this fault.
`faultActor`	A string indicating the SOAP Actor that generated the fault.
`faultDetail`	A string indicating detail associated with the fault.

— `NetworkError` is thrown when there is a failure from the underlying HTTPtransport layer or in obtaining a Network connection. The `NetworkError Exception` object has the `property statusCode`, which is an HTTP Status code or –1 if the network connection could not be made.

Other standard ECMAScript exceptions may also be thrown as appropriate.

If the method was called asynchronously, any exception object shall be passed to the response callback method.

10.32.3.5.4 Additional notes on the oRequest parameter

The `oRequest` parameter is an object literal that specifies the remote procedure name and the parameters to call. The object literal uses the fully qualified method name of the remote procedure as the key. The namespace should be separated from the method name by a colon.

For example, if the namespace for the method is http://mydomain/methods and the method name is echoString, the fully qualified name would be http://mydomain/methods:echoString. The value of this key is an object literal, each key is a parameter of the method, and the value of each key is the value of the corresponding parameter of the method. For example:

```
oRequest: {
"http://soapinterop.org/:echoString":{inputString: "Echo!"}
}
```

10.32.3.6 response(...)

Parameters

Parameter	Type	Description
cURL	String	The URL for a SOAP HTTP endpoint. The URL method must be one of these: *http* — Connect to a server at a URI on a port. For example, *http://serverName:portNumber/URI* *https* — Connect to a secured (SSL) server at a URI on a port. For example, *https://serverName:portNumber/URI* See the `cURL` parameter of **SOAP.request**(...).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
oRequest	Object	An object that specifies the remote procedure name and parameters or the XML message to send. See the `oRequest` parameter of **SOAP.request**(...).
cAction	String	(optional) The SOAP Action header for this request as specified by the SOAP Specification. The default is for the SOAP Action to be empty. See the `cAction` parameter of **SOAP.request**(...).
bEncoded	Boolean	(optional) A Boolean value specifying whether the request was encoded using the SOAP Encoding described in the SOAP Specification. The default is ***true***.
cNamespace	String	(optional) A namespace for the message schema when the request does not use the SOAP Encoding (the `bEncoded` flag is ***false***). The default is to have no namespace.
oReqHeader	Object	(optional) An object that specifies a SOAP header to be included with the request. The default is to send a request with only a SOAP Body. See the `oReqHeader` parameter of **SOAP.request**(...).
cVersion	String	(optional) The version of the SOAP protocol to use. The default is to use "***SOAPVersion.version_1_1***". See the `cVersion` parameter of **SOAP.request**(...).
oAuthenticate	Object	(optional) An object that specifies the type of authentication scheme to use and to provide credentials. The default is for an interactive UI to be displayed if HTTP authentication is encountered. See the `oAuthenticate` parameter of SOAP.request(...).
cRequestStyle	String	(optional) Same as **cRequestStyle** for SOAP.request(...), except that there is no response from the server.
cContentType	String	(optional) Same as **cContentType**for SOAP.request(...), except that there is no response from the server.

Return Value

Boolean

Description

Initiates a remote procedure call (RPC) or sends an XML message to a SOAP HTTP endpoint without waiting for a reply.

This method is deprecated, services should use Net.SOAP.response.

10.32.3.6.1 Exceptions

If there is a problem at the networking level, such as an unavailable endpoint, a `NetworkError` will be thrown.

10.32.3.7 streamDecode(oStream, cEncoder)

Parameters

Parameter	Type	Description
oStream	Object	A stream object to be decoded with the specified encoding type.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Parameter	Type	Description
cEncoder	String	Permissible values for this string are "***hex***" (hex-encoded) and "***base64***" (Base 64- encoded).

Return Value

ReadStream

Description

Allows the oStream object to be decoded with the specified encoding type, cEncoder. It returns a decoded **ReadStream** object. Typically, it is used to access data returned as part of a SOAP method that was encoded in Base64 or hex encoding.

This method is deprecated, services should use Net.streamDecode (see Net methods).

10.32.3.8 streamDigest(oStream, cEncoder)

Parameters

Parameter	Type	Description
oStream	Object	A stream object to compute the digest of, using the specified message digest algorithm.
cEncoder	String	The digest algorithm to use. The cEncoder parameter must be one of the following values: ***StreamDigest.MD5*** — Digest the content using the MD5 Digest Algorithm (see RFC 1321). ***StreamDigest.SHA1*** — Digest the content using the SHA-1 Digest Algorithm (as defined in FIPS 180-4).

Return Value

ReadStream

Description

Allows the oStream object to be digested with the specified encoding type, cEncoder. It returns a ReadStream object containing the computed digest of the **oStream**. Typically, this is used to compute a digest to validate the integrity of the original data stream or as part of an authentication scheme for a web service.

To be used as a string, the result must be converted to a text format such as Base64 or hex using **SOAP.streamEncode**.

This method is deprecated, services should use Net.streamDigest (see Net methods).

10.32.3.9 streamEncode(oStream, cEncoder)

Parameters

Parameter	Type	Description
oStream	Object	A stream object to be encoded with the specified encoding type.
cEncoder	String	A string specifying the encoding type. Permissible values are "***hex***" (for hex-encoded) and "***base64***" (base 64-encoded).

Return Value

ReadStream

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

This function encodes a stream object. Typically, it is used to pass data as part of a SOAP method when it must be encoded in Base 64 or hex encoding.

This method is deprecated, new services should use Net.streamEncode (see Net methods).

10.32.3.10 streamFromString(cString)

Parameters

Parameter	Type	Description
cString	String	The string to be converted.

Return Value

ReadStream

Description

This function converts a string to a **`ReadStream`** object. Typically, this is used to pass data as part of a **`SOAP`** method.

10.32.3.11 stringFromStream(oStream)

Parameters

Parameter	Type	Description
oStream	Object	The **`ReadStream`** object to be converted.

Return Value

String

Description

This function converts a **`ReadStream`** object to a string. Typically, this is used to examine the contents of a **`stream`** object returned as part of a response to a SOAP method.

10.33 Span

10.33.1 General

A generic object that represents a length of text and its associated properties in a rich text form field or annotation. A rich text value consists of an array of span objects representing the text and formatting.

NOTE **`Span`** objects are a copy of the rich text value of the field or annotation. To modify and reset the rich text value to update the field, use the **`Field`** object **`richValue`** property, or the **`Annotation`** object property **`richContents`**, and the **`event`** object properties **`richValue`**, **`richChange`**, and **`richChangeEx`**.

10.33.2 Span properties

Table 93: Span object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`Span`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 93 — Span object properties

Property	Type	Access	Description
alignment	String	R/W	The horizontal alignment of the text. Alignment for a line of text is determined by the first span on the line. The values of alignment are ***left*** ***center*** ***right*** The default value is ***left***.
fontFamily	Array	R/W	The font family used to draw the text. It is an array of family names to be searched for in order. The first entry in the array is the font name of the font to use. The second entry is an optional generic family name to use if an exact match of the first font is not found. The generic family names are ***symbol*** ***serif*** ***sans-serif*** ***cursive*** ***monospace*** ***fantasy*** The default generic family name is ***sans-serif***.
fontStretch	String	R/W	Specifies the normal, condensed or extended face from a font family to be used to draw the text. The values of **fontStretch** are: ***ultra-condensed*** ***extra-condensed*** ***condensed*** ***semi-condensed*** ***normal*** ***semi-expanded*** ***expanded*** ***extra-expanded*** ***ultra-expanded*** The default value is ***normal***.
fontStyle	String	R/W	Specifies the text is drawn with an italic or oblique font. The values of **fontStyle** are: *italic* *normal* The default is *normal*.
fontWeight	Number	R/W	The weight of the font used to draw the text. For the purposes of comparison, normal is anything under 700 and bold is greater than or equal to ***700***. The values of fontWeight are ***100,200,300,400,500,600,700,800,900*** The default value is ***400***.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 93 *(continued)*

Property	Type	Access	Description
strikethrough	Boolean	R/W	If strikethrough is ***true***, the text is drawn with a strike-through. The default is ***false***.
subscript	Boolean	R/W	Specifies the text is subscript. If ***true***, subscript text is drawn with a reduced point size and a lowered baseline. The default is ***false***.
superscript	Boolean	R/W	Specifies the text is superscript. If ***true***, superscript text is drawn with a reduced point size and a raised baseline. The default is ***false***.
text	String	R/W	The text within the span.
textColor	Color Array	R/W	A color array representing the RGB color to be used to draw the text (see the `color` object). The default color is ***black***.
textSize	Number	R/W	The point size of the text. The value of textSize can be any number between ***0*** and ***32767***, inclusive. A text size of zero means to use the largest point size that will allow all text data to fit in the field's rectangle. The default text size is ***12.0***.
underline	Boolean	R/W	If underline is ***true***, the text is underlined. The default is ***false***.

10.34 Template

10.34.1 General

`Template` objects are named pages within the document. These pages may be hidden or visible and can be copied or spawned. They are typically used to dynamically create content (for example, to add pages to an invoice on overflow).

See also the `Doc` templates property, and `createTemplate` (cName, nPage), `getTemplate(cName)`, and `removeTemplate`(cName) methods.

10.34.2 Template properties

Table 94: Template object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Template`.

Table 94 — Template object properties

Property	Type	Access	Description
Hidden	Boolean	R/W	Determines whether the template is hidden. Hidden templates cannot be seen by the user until they are spawned or are made visible. When an invisible template is made visible, it is appended to the document.
name	String	R	The name of the template that was supplied when the template was created.

10.34.3 Template methods

10.34.3.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type `Template`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.34.3.2 spawn([nPage, bRename, bOverlay, oXObject])

Parameters

Parameter	Type	Description
nPage	Number	(optional) The 0-based index of the page number after which or on which the new page will be created, depending on the value of **bOverlay**. The default is *0*.
bRename	Boolean	(optional) Specifies whether form fields on the page should be renamed. The default is ***true***.
bOverlay	Boolean	(optional) If ***true*** (the default), the template is overlaid on the specified page. If ***false***, it is inserted as a new page before the specified page. To append a page to the document, set **bOverlay** to ***false*** and set **nPage** to the number of pages in the document. NOTE For certified documents or documents with "Advanced Form Features rights", the bOverlay parameter is disabled. A template cannot be overlaid for these types of documents.
oXObject	Object	(optional) The value of this parameter is the return value of an earlier call to spawn.

Return Value

Object

Description

Creates a new page in the document based on the template. Calls to this method shall return an object representing the page contents of the page spawned. This return object can then be used as the value of the optional parameter **oXObject** for subsequent calls to spawn.

10.35 Thermometer

10.35.1 General

This object is a combined status window and progress bar that indicates to the user that a lengthy operation is in progress. To acquire a **Thermometer** object, use **app.thermometer**.

10.35.2 Thermometer properties

Table 95: Thermometer object properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **Thermometer**.

Table 95 — Thermometer object properties

Property	Type	Access	Description
Cancelled	Boolean	R	Specifies whether the user wants to cancel the current operation. The user can indicate the desire to terminate the operation by pressing the Esc key on the Windows and UNIX platforms and Command-period on the Mac OS platform.
Duration	Number	R/W	Sets the value that corresponds to a full thermometer display. The thermometer is subsequently filled in by setting its value. The default duration is *100*.
Text	String	R/W	Sets the text string that is displayed by the thermometer.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 95 *(continued)*

Property	Type	Access	Description
value	Number	R/W	Sets the current value of the thermometer and updates the display. The value can range from *0* (empty) to the value set in duration. For example, if the thermometer's duration is *10*, the current value must be between *0* and *10*, inclusive. If the value is less than zero, it is set to zero. If the value is greater than duration, it is set to duration.

10.35.3 Thermometer methods

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type `Thermometer`.

10.35.3.1 begin()

Parameters

None

Return Value

undefined

Description

Initializes the thermometer and displays it with the current value as a percentage of the duration.

10.35.3.2 end()

Parameters

None

Return Value

undefined

Description

Draws the thermometer with its current value set to the thermometer's duration (a full thermometer), then removes the thermometer from the display.

10.36 this

10.36.1 General

In ECMAScript, the special keyword `this` refers to the current object. The current object is defined as follows.

— In an object method, it is the object to which the method belongs.

— In a constructor function, it is the object being constructed.

— In a document-level script or field-level script, it is the `Doc` and therefore can be used to set or get document properties and functions.

— In a function defined in one of the folder-level ECMAScripts files, it is undefined. Calling functions should pass the `Doc` to any function at this level that needs it.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

For example, assume that the following function was defined at the plug-in folder level:

```
function PrintPageNum(doc)
{ /* Print the current page number to the console. */
      console.println("Page = " + doc.pageNum);
}
```

Variables and functions that are defined in scripts are parented off of the **this** object. For example:

```
var f = this.getField("Hello");
```

is equivalent to

```
this.f = this.getField("Hello");
```

with the exception that the variable **f** can be garbage collected at any time after the script is run.

10.36.2 Variable and function name conflicts

ECMAScript programmers should avoid using property and method names from the **Doc** as variable names. Using method names after the reserved word var will throw an exception, as the following line shows:

```
var getField = 1; // TypeError: redeclaration of function getField
```

Use of property names will not throw an exception, but the value of the property may not be altered if the property refers to an object:

```
// "title" will return "1", but the document will now be named "1".
var title = 1;
// Property not altered, info still an object
var info = 1; // "info" will return [object Info]
```

10.37 util

10.37.1 General

A static ECMAScript object that defines a number of utility methods and convenience functions for string and date formatting and parsing.

10.37.2 util methods

10.37.2.1 General

The following section lists the minimum set of methods, including their syntax and return values, that PDF Processors supporting this standard shall implement for objects of type **util**.

10.37.2.2 crackURL(cURL)

Parameters

Parameter	Type	Description
cURL	String	A string specifying the URL.

Return Value

UrlParts object

Description

Breaks a URL into its component parts.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

`util.crackURL` can break a URL that uses Internet Protocol version 4 (IPv4) or version 6 (IPv6) addressing. The return object for `util.crackURL` includes the nURLType property to help determine the IP type of the URL being passed as the `cURL` argument to `util.crackURL`.

No effort is made to determine the IP type if cURL contains a DNS name rather than an IP address as the hostname.

This method throws a parameter error if the parameter is missing, the URL is not well-formed, or the URL scheme is not file, http, or https.

10.37.2.2.1 UrlParts object properties

Calls to `util.crackURL` return a `UrlParts` object containing the properties given in Table 96:

Table 96 — UrlParts object properties

Property	Description
cScheme	The scheme of the URL. Valid values are file, http or https.
cUser	(Optional) The username specified in the URL.
cPassword	(Optional) The password specified in the URL.
cHost	The hostname of the URL.
nPort	The port number of the URL.
cPath	(Optional) The path portion of the URL.
cQuery	(Optional) The parameter string portion of the URL.
cFragment	(Optional) The fragments of the URL.
nURLType	(Version) An integer valued property that takes a value of 0 if `cURL` uses IPv4 and a value of *1* if cURL uses IPv6. (IPv4 and IPv6 refer to the Internet Protocol versions 4 and 6, respectively.)

10.37.2.3 iconStreamFromIcon(oIcon)

Parameters

Parameter	Type	Description
oIcon	Object	An Icon object to be converted into an `Icon Stream` object.

Return Value

IconStream

Description

Converts an XObject-based `Icon` object into an `Icon Stream` object.

This method allows an icon obtained from the `Doc importIcon` or `getIcon` methods to be used in a method such as `app.addToolButton`, which would otherwise accept only an `IconStream` object as an input parameter.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.37.2.4 printd(cFormat, oDate[, bXFAPicture])

Parameters

Parameter	Type	Description
cFormat	String	The date and time format. It can be one of the following types: A string that is a pattern of supported substrings that are place-holders for date and time data. Recognized date and time strings are shown in 10.37.2.4.1 cFormat String Patterns. A number specifying the format. Supported values (along with examples of each format) are : 0 — *PDF date format*. EXAMPLE *D:20000801145605+07'00'* 1 — *Universal*. EXAMPLE *D:20000801145605+07'00'* 2 — *Localized string*. EXAMPLE *2000/08/01 14:56:05*
oDate	Object	A **Date** object to format. **Date** objects can be obtained from the **Date** constructor of core ECMAScript or from the **util.scand** method.

Return Value

String

Description

Returns a date using a specified format.

10.37.2.4.1 cFormat String Patterns

String	Effect	Example
mmmm	Long month	*September*
mmm	Abbreviated month	*Sep*
mm	Numeric month with leading zero	*09*
m	Numeric month without leading zero	*9*
dddd	Long day	*Wednesday*
ddd	Abbreviated day	*Wed*
dd	Numeric date with leading zero	*03*
d	Numeric date without leading zero	*3*
yyyy	Long year	*1997*
yy	Abbreviated Year	*97*
HH	24 hour time with leading zero	*09*
H	24 hour time without leading zero	*9*
hh	12 hour time with leading zero	*09*
h	12 hour time without leading zero	*9*
MM	minutes with leading zero	*08*
M	minutes without leading zero	*8*
ss	seconds with leading zero	*05*
s	seconds without leading zero	*5*
tt	am/pm indication	*am*
t	single digit am/pm indication	*a*
\	use as an escape character	

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.37.2.5 printf(cFormat[, ...])

Parameters

Parameter	Type	Description
cFormat	String	The format string to use.
arguments	String	The optional arguments(s) that contain the data to be inserted in place of the % tags specified in the first parameter, the format string. The number of optional arguments must be the same as the number of % tags.

Return Value

String

Description

Formats one or more arguments as a string according to a format string. It is similar to the C function of the same name. This method converts and formats incoming arguments into a result string according to a format string (cFormat).

NOTE The **`util.printf`** function does not accept an object literal with properties that contain the arguments. Arguments are entered in the usual comma-delimited list.

10.37.2.5.1 Format String specification

The format string consists of two types of objects:

1) Ordinary characters, which are copied to the result string.

2) Conversion specifications, each of which causes conversion and formatting of the next successive argument to printf.

Each conversion specification is constructed as follows:

```
%[,nDecSep][cFlags][nWidth][.nPrecision]cConvChar
```

Table 97: Format String Conversions describes the components of a conversion specification.

Table 97 — Format String Conversions

Conversion	Description
nDecSep	A comma character (,) followed by a digit that indicates the decimal/separator format: ***0*** — Comma separated, period decimal point ***1*** — No separator, period decimal point ***2*** — Period separated, comma decimal point ***3*** — No separator, comma decimal point
cFlags	Only valid for numeric conversions and consists of a number of characters (in any order), which will modify the specification: ***+*** — Specifies that the number will always be formatted with a sign. ***space*** — If the first character is not a sign, a space will be prefixed. ***0*** — Specifies padding to the field with leading zeros. ***#*** — Specifies an alternate output form. For f, the output will always have a decimal point.
nWidth	A number specifying a minimum field width. The converted argument is formatted to be at least this many characters wide, including the sign and decimal point, and may be wider if necessary. If the converted argument has fewer characters than the field width, it is padded on the left to make up the field width. The padding character is normally a space, but is 0 if the zero padding flag is present (**`cFlags`** contains *0*).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 97 *(continued)*

Conversion	Description
nPrecision	A period character (.) followed by a number that specifies the number of digits after the decimal point for float conversions.
cConvChar	Indicates how the argument should be interpreted: ***d*** — Integer (truncating if necessary) ***f*** — Floating-point number ***s*** — String ***x*** — Integer (truncating if necessary) and formatted in unsigned hexadecimal notation

10.37.2.6 printx(cFormat, cSource)

Parameters

Parameter	Type	Description
cFormat	String	The formatting string to use.
cSource	String	The source string to use.

Return Value

String

Description

Formats a source string, **`cSource`**, according to a formatting string, **`cFormat`**, and returns the formatted string.

10.37.2.6.1 Format Masking Characters

A valid format for **`cFormat`** is any string that may contain special masking characters as given in Table 98:

Table 98 — Format Masking Characters

Value	Effect
?	Copy next character.
X	Copy next alphanumeric character, skipping any others.
A	Copy next alpha character, skipping any others.
9	Copy next numeric character, skipping any others.
*******	Copy the rest of the source string from this point on.
\	Escape character.
>	Upper case translation until further notice.
<	Lower case translation until further notice.
=	Preserve case until further notice (default).

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

10.37.2.7 readFileIntoStream([cDIPath, bBase64])

Parameters

Parameter	Type	Description
cDIPath	String	(optional) A device-independent path to an arbitrary file on the user's hard drive. This path may be absolute or relative to the current document. If not specified, the user is presented with the File Open dialog to locate the file. If the cDIPath parameter is specified, this method can only be executed in privileged context, during a batch or console event (or when the document is certified with a certificate trusted to execute "embedded high privileged ECMAScript").
bEncodeBase64	Boolean	(optional) If **`true`**, base64-encode the file content.

Return Value

ReadStream

Description

Loads an external file into an ECMAScript stream, optionally base64 encodes it, and returns the content as an encoded stream.

10.37.2.8 scand(cFormat, cDate)

Parameters

Parameter	Type	Description
cFormat	String	The rules to use for formatting the date. cFormat uses the same syntax as found in printd.
cDate	String	The date to convert.

Return Value

Date or null

Description

Converts a date into an ECMAScript **`Date`** object according to the rules of a format string. This routine is much more flexible than using the date constructor directly.

NOTE Given a two-digit year for input, scand uses the date horizon heuristic to resolve the ambiguity. If the year is less than 50, it is assumed to be in the 21st century (that is, add 2000). If it is greater than or equal to 50, it is in the 20th century (add 1900).

The supplied date cDate should be in the same format as described by cFormat.

Returns the converted Date object, or null if the conversion fails.

10.37.2.9 spansToXML(aSpans)

Parameters

Parameter	Type	Description
An array of Span objects	Array	An array of **`Span`** objects to be converted into an XML string.

Return Value

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

String

Description

Converts an array of Span objects into an XML(XFA) String as described in ISO 32000-2.

10.37.2.10 streamFromString(cString[, cCharSet])

Parameters

Parameter	Type	Description
cString	String	The string to be converted into a `ReadStream` object.
cCharSet	String	(optional) The encoding for the string in cString. The options are ***utf-8***, ***utf-16***, ***Shift-JIS***, ***BigFive***, ***GBK***, ***UHC***. The default is ***utf-8***.

Return Value

ReadStream

Description

Converts a string to a ReadStream object.

10.37.2.11 stringFromStream(oStream[, cCharSet])

Parameters

Parameter	Type	Description
oStream	Object	ReadStream object to be converted into a string.
cCharSet	String	(optional) The encoding for the string in oStream. The options are ***utf-8***, ***utf-16***, ***Shift-JIS***, ***BigFive***, ***GBK***, ***UHC***. The default is ***utf-8***.

Return Value

String

Description

Converts a ReadStream object to a string.

10.37.2.12 xmlToSpans(cXml)

Parameters

Parameter	Type	Description
aString	String	An XML (XFA) string to be converted to an array of `Span` objects.

Return Value

Array of `Span` objects

Description

Converts an XML (XFA) string, as described in ISO 32000-2, to an array of `Span` objects suitable for specifying as the `richValue` or `richContents` of a field or annotation.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

11 ECMAScript 3D API

11.1 General

This section specifies a set of ECMAScript object types which define the properties and methods that can be used to automate and interact with 3D objects. This 3D ECMAScript engine, which may be distinct from the ECMAScript engine for a PDF Processor, can be accessed in one of two ways. The primary way is by attaching a default script to the 3D annotation. This script will be run directly by the 3D ECMAScript engine.

In addition, PDF Processors may provide a mechanism to directly access the 3D ECMAScript engine API from within the PDF Processor's scripting engine by means of the Annot3D properties.

11.1.1 Basic Objects

There are several basic objects, such as **`Color`**, **`Matrix4x4`**, and **`Vector3`** that are used to create general-purpose objects. The basic objects are used throughout this specification and are only meaningful when attached to objects such as **`Scene`** or **`Runtime`**. For example, a **`Color`** object could be created and used to set the **`Background`** color of a **`Canvas`**.

Vector3 Examples

```
v1 = new Vector3( 1.2, 3, 4.5 );
v2 = new Vector3( 5, 8, 13 );
v3 = new Vector3();
```

Matrix4x4 Examples

```
m1 = new Matrix4x4().rotateAboutX(Math.PI/1.5).rotateAboutY(Math.PI/3);
m2 = new Matrix4x4().rotateAboutZ(Math.PI/4).translate(new Vector3(0,5,0));
m3 = new Matrix4x4(m1);
```

Color Examples

```
c1 = new Color( 0.6, 0.8, 1.0 );    // light blue
c2 = new Color( 0.5, 0.5, 0.5 );    // middle grey
c3 = new Color();    //black
// A function to blend two Colors
Color.prototype.blend = function( color, amount )
{
red = ( this.r * ( 1 - amount ) ) + ( color.r * amount );
green = ( this.g * ( 1 - amount ) ) + ( color.g * amount );
blue = ( this.b * ( 1 - amount ) ) + ( color.b * amount );
return( new Color( red, green, blue ) );
}
c4 = c1.blend( c2, 0.25 );
```

11.1.2 Scene object

The **`Scene`** is an object that contains all of the 3D-related content. It can be accessed using the global variable scene, which is a reference to the main **`Scene`** object. Most of the contents of the **`Scene`** are structured into a hierarchy of **`Node`** objects and maintains lists of all these objects in the form of a **`SceneObjectList`**.

For more information, see Scene.

11.1.3 Canvas object

Represents a rectangular region into which a Scene is rendered from a particular viewpoint. For more information, see Canvas.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

11.1.4 Runtime object

The **Runtime** object is used to represent the instance of the playback engine. It manages all event processing and places where the graphic and textual content is rendered. It is accessed via the global variable runtime, which is a reference to the main **Runtime** object.

For more information, see Runtime.

11.1.5 Resource objects

Some objects, such as Image, are driven by content that is streamed from a file or over a network. To create an image, load a .png, .jpg, or .gif file as a **Resource**, which subsequently may be used to create a new **Image** object, as shown in the following example:

EXAMPLE

```
faceRes = new Resource("pdf://picture.jpg");
faceImage = new Image( faceRes );
aMaterial = scene.meshes.getByIndex(0).material;
aMaterial.diffuseTexture.setImage( faceImage );
```

For more information, see **Resource** and **Image**.

11.2 Event handlers

11.2.1 General

There are several types of event handlers. Each one responds to a different type of event during simulation. They use a callback mechanism to run a function when an event occurs. The event is passed as an argument to the event handler's **onEvent** function so that it can be queried when the function runs. Event handlers are registered via the **addEventHandler** method, of the **Runtime** object.

11.2.2 CameraEvent

A **CameraEvent** is created when a **View** is selected. For more information, see section 12.6.2, **CameraEvent** objects.

11.2.3 KeyEvent

A **KeyEvent** is created when a key is pressed or released while the 3D Canvas is in focus. The following example illustrates how to handle a key event:

EXAMPLE

```
myKeyHandler    = new KeyEventHandler();
myKeyHandler.onEvent    = function( event )
{
   console.print( "Key pressed with code: " + event.characterCode );
}
runtime.addEventHandler( myKeyHandler );
```

For more information, see KeyEvent properties.

11.2.4 MouseEvent

A **MouseEvent** is created when the mouse is clicked on an active 3D Canvas or the cursor moves over an active 3D Canvas. The following syntax could be used to handle a mouse event:

EXAMPLE

```
myMouseHandler    = new MouseEventHandler();
myMouseHandler.onMouseDown    = true;
myMouseHandler.target    = scene.meshes.getByIndex(0);
```

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

```
myMouseHandler.onEvent   = function( event )
{
   console.print( "Mouse down at pixel " + event.mouseX );
   console.print( ", " + event.mouseY );
}
runtime.addEventHandler( myMouseHandler );
```

For more information, see MouseEvent.

11.2.5 RenderEvent

A **RenderEvent** is created immediately before an instance of the **Canvas** is drawn. If there is a split view in the PDF Processor resulting in two visible 3D rendered areas, a unique **RenderEvent** will be called for each of them. This is necessary in the case of a camera-aligned image (sprite) in the 3D content that needs to be pixel-aligned. Since the pixel dimensions of the two areas are possibly different, there are two callbacks that pass the different dimensions. This makes it possible to modify the **Scene** in the appropriate manner before it is drawn.

For more information, see RenderEvent.

11.2.6 ScrollWheelEvent

A **ScrollWheelEvent** object is created when the mouse scroll wheel is activated over an active 3D Canvas object.

For more information, see ScrollWheelEvent.

11.2.7 SelectionEvent

A **SelectionEvent** object is created when an object is selected from an active 3D Canvas object or from a model tree. If the selection is made from a **Canvas** object, a **MouseEvent** is also created.

For more information, see SelectionEvent.

11.2.8 TimeEvent

A **TimeEvent** is created when the 3D annotation is enabled and simulation is active. The **time** and **deltaTime** properties are measured in terms of simulation time, not real time. **TimeEvent** objects are used to drive animation. For applications requiring accurate, real-time measurement, use the ECMAScript **Date** object. The following syntax is used to handle a time event:

EXAMPLE

```
myTimeHandler   = new TimeEventHandler();
myTimeHandler.onEvent   = function( event )
{
console.print( "Current simulation time is:" + event.time );
console.print( " second(s)" );
}
runtime.addEventHandler( myTimeHandler );
```

For more information, see TimeEvent.

11.2.9 ToolEvent

A **ToolEvent** is created when a tool is clicked in the PDF Processor's toolbar. The **Runtime** object's **addCustomToolButton** method may be used to add a custom tool to the toolbar which will also be generated, and allows a script to be attached to the tool selection event.

For more information, see ToolEvent.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

12 Object overview

12.1 General

This section provides an overview of the objects in the 3D ECMAScript for PDF API.

12.2 Animation

12.2.1 General

A type of `SceneObject`, used to store keyframe animation sequences of `Node` objects in the `Scene`. In addition to the methods and properties below, it also contains the same methods and properties as a `SceneObject`.

12.2.2 Animation properties

Table 99: Animation properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **Animation**.

Table 99 — Animation properties

Property	Type	Access	Description
currentTime	number	R/W	The current time measured in seconds.
endTime	number	R	The end time of the sequence, measured in seconds.
framesPerSecond	number	R	The number of frames per second used to author the sequence.
Length	number	R	The length of the **Animation**, measured in seconds.
startTime	number	R	The start time of the sequence, measured in seconds.

12.3 Background

12.3.1 General

Represents the background of a `Canvas`. It can be used as a target of a `MouseEventHandler`. (See Canvas and MouseEventHandlerf.)

12.3.2 Background object properties

Table 100: Background properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `Background`.

Table 100 — Background properties

Property	Type	Access	Description
image	Image	R/W	The image to be used by the **Background**.

12.3.3 Background object methods

12.3.3.1 getColor()

Parameters

None

Return Value

A `Color` object representing the background color of the `Canvas`.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Description

Obtains the background color.

12.3.3.2 setColor(topColor, bottomColor)

Parameters

Parameter	Type	Description
topColor	Color object	A `Color` object representing the desired background color. If `bottomColor` is used, `topColor` represents the top background color used in a linear gradient.
bottomColor	Color object	*(optional)* A `Color` object representing the bottom background color used in a linear gradient.

Return Value

undefined

Description

Sets the background `Color`. If only one color is passed to this method, the background is a constant color. If two colors are passed to this method, the background is a linear gradient from top to bottom, with the first color argument representing the top color and the second representing the bottom color.

12.4 BoundingBox

12.4.1 General

`BoundingBox` objects represent axis-aligned read-only bounding boxes. Table 101: BoundingBox properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type `BoundingBox`. There are no methods defined in this standard for `BoundingBox` objects.

12.4.2 BoundingBox properties

Table 101: BoundingBox properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type BoundingBox.

Table 101 — BoundingBox properties

Property	Type	Access	Description
center	Vector3	R	The coordinates of the `BoundingBox` center.
max	Vector3	R	The coordinates of the `BoundingBox` corner with the greatest x, y, and z values.
min	Vector3	R	The coordinates of the `BoundingBox` corner with the smallest x, y, and z values

12.5 Camera

12.5.1 General

A `Camera` object represents a `Node` that controls the projection from world space to screen space. In addition to the methods and properties below, it also contains the same methods and properties as a Node.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

12.5.2 Camera properties

Table 102: Camera properties lists the properties that PDF Processors supporting this standard shall implement for objects of type `Camera`.

Table 102 — Camera properties

Property	Type	Access	Description
absoluteBindingScale	number	R/W	The absolute binding scale value for the camera.
binding	string	R/W	The view plane calculation type, which can take one of the following values: ***min*** ***max*** ***horizontal*** ***vertical***
BINDING_HORIZONTAL	string	R	A string constant for the binding value of "***horizontal***".
BINDING_MAX	string	R	A string constant for the binding value of "***max***".
BINDING_MIN	string	R	A string constant for the binding value of "***min***".
BINDING_VERTICAL	string	R	A string constant for the binding value of "***vertical***".
far	number	R/W	The distance from the `Camera` to the far clipping plane. A value of -1 for both near and far signifies to use auto clipping plane calculations.
fov	number	R/W	The size of the field of view for perspective `Camera` objects, measured in radians.
near	number	R/W	The distance from the `Camera` to the near clipping plane. A value of -1 for both near and far signifies to use auto clipping plane calculations.
position	Vector3	R	The position of the origin of the `Camera` in world space.
positionLocal	Vector3	R	The position of the origin of the `Camera` in local space.
projectionType	string	R/W	The type of projection, which can take one of the following values: ***perspective*** ***orthographic***
roll	number	R/W	The roll angle of the `Camera`, measured in radians.
target	Node	R	The current `Node` used as the `Camera` object's target.
targetPosition	Vector3	R	The position of the `Camera` object's target in world space.
targetPositionLocal	Vector3	R/W	The position of the `Camera` object's target in local space.
TYPE_ORTHOGRAPHIC	string	R	A string constant for the camera projection type of "***orthographic***".
TYPE_PERSPECTIVE	string	R	A string constant for the camera projection type of "***perspective***".
up	Vector3	R	The up direction in world space.
upLocal	Vector3	R	The up direction in local space.
useAbsoluteBinding	Boolean	R	Determines whether the `Camera` uses absolute binding for its projection.
viewPlaneSize	number	R/W	The size of the view plane for orthographic `Camera` objects, measured in scene units.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

12.5.3 Camera methods

12.5.3.1 getScreenFromPosition(position, canvasWidth, canvasHeight)

Parameters

Property	Type	Description
position	Vector3	A **`Vector3`** object representing the 3D position.
canvasWidth	Number	The width of the Canvas, measured in pixels.
canvasHeight	Number	The height of the Canvas, measured in pixels

Return Value

`Vector3`

Description

Obtains the screen coordinates of the provided 3D position.

Implementers shall return a **`Vector3`** object representing the screen coordinates, with x and y as pixel positions and z equal to zero. See Vector3 for more information on the return object.

12.5.3.2 getDirectionFromScreen(x, y, canvasWidth, canvasHeight)

Parameters

Property	Type	Description
x	Number	The x-coordinate, measured in pixels.
y	Number	The y-coordinate, measured in pixels.
canvasWidth	Number	The width of the Canvas, measured in pixels.
canvasHeight	Number	The height of the Canvas, measured in pixels

Return Value

`Vector3`

Description

Obtains the direction from the normalized coordinates.

12.6 CameraEvent

12.6.1 General

The object that is passed as an argument to the **`onEvent`** method of the **`CameraEventHandler`** object is a **`CameraEvent`** object.

12.6.2 CameraEvent properties

Table 103: CameraEvent properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`CameraEvent`**.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

Table 103 — CameraEvent properties

Property	Type	Access	Description
binding	string	R	The view plane calculation type, which can take one of the following values: ***min*** ***max*** ***horizontal*** ***vertical***
canvas	Canvas	R	The **Canvas** in which the event took place.
currentTool	string	R	The name of the current tool.
far	number	R	The distance from the **Camera** to the far clipping plane. A value of -1 for both near and far signifies to use auto clipping plane calculations.
fov	number	R	The size of the field of view for perspective **Camera** objects, measured in radians.
near	number	R	The distance from the **Camera** to the near clipping plane. A value of -1 for both near and far signifies to use auto clipping plane calculations.
projectionType	string	R	The type of projection, which can take one of the following values: ***perspective*** ***orthographic***
targetDistance	number	R	The distance from the **Camera** to its target.
transform	Matrix4x4	R	The **Camera** object's transformation matrix.
view	View object	R	The name of the view being activated.
viewPlaneSize	number	R	The size of the view plane, measured in scene units.

12.7 CameraEventHandler

12.7.1 General

The **CameraEventHandler** object exposes a callback mechanism that allows a function to be evaluated when a camera event occurs. Event handlers are registered with the **Runtime addEventHandler** method.

12.7.2 CameraEventHandler methods

12.7.2.1 new CameraEventHandler()

Parameters

None

Return Value

CameraEventHandler

Description

A constructor that creates a new **CameraEventHandler** object.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved

12.7.2.2 onEvent(event)

Parameters

Property	Type	Description
`event`	Object	**A `CameraEvent` object representing the event.**

Return Value

Undefined

Description

A method that is called when a view is selected from the list of views on the 3D toolbar or in the context menu for an active 3D annotation.

12.8 Canvas

12.8.1 General

Represents a rectangular region into which the **`Scene`** is rendered from the viewpoint of the attached **`Camera`**.

See related objects, `Scene` and `Camera`.

12.8.2 Canvas properties

Table 104: Canvas properties lists the minimum set of properties that PDF Processors supporting this standard shall implement for objects of type **`Canvas`**.

Table 104 — Canvas properties

Property	Type	Access	Description
background	Background	R	The **`Background`** object associated with the **`Canvas`**.

12.8.3 Canvas methods

12.8.3.1 getCamera

Parameters

None

Return Value

Camera

Description

Obtains the **`Camera`** object attached to the **`Canvas`**.

This method shall return a **`Camera`** object.

12.8.3.2 setCamera(camera)

Parameters

Property	Type	Description
camera	Camera	The **`Camera`** object used to set the object's value.

STANDARDSISO.COM : Click to view the full PDF of ISO 21757-1:2020

© ISO 2020 – All rights reserved