BS ISO 21757-1:2020

BSI Standards Publication

Document management — ECMAScript for PDF

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

Documents
Document Sharing Hub

Full document details: BS ISO 21757-1:2020

National foreword

This British Standard is the UK implementation of ISO 21757-1:2020.

The UK participation in its preparation was entrusted to Technical Committee IDT/1, Document Management Applications.

A list of organizations represented on this committee can be obtained on request to its committee manager.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© The British Standards Institution 2021
Published by BSI Standards Limited 2021

ISBN 978 0 580 98751 9

ICS 35.240.30

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 31 January 2021.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: BS ISO 21757-1:2020

INTERNATIONAL STANDARD

ISO 21757-1

First edition
2020-12

Document management — ECMAScript for PDF —

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

Reference number
ISO 21757-1:2020(E)

Documents
Document Sharing Hub

© ISO 2020

Full document details: BS 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

Documents
Document Sharing Hub

Full document details: BS ISO 21757-1:2020

Contents

Page

Foreword ... **ix**
Introduction ... **x**
1 Scope ... **1**
2 Normative references ... **1**
3 Terms and definitions ... **1**
4 Notation ... **1**
5 Syntax ... **2**
5.1 General ... 2
5.2 Method arguments ... 2
6 Paths ... **2**
7 Safe path ... **2**
8 Privileged context ... **3**
9 Privileged versus non-privileged context ... **3**
10 ECMAScript API ... **3**
10.1 General ... 3
10.2 Annotation ... 3
10.2.1 General ... 3
10.2.2 Annotation types ... 4
10.2.3 Annotation properties ... 5
10.2.4 Annotation methods ... 13
10.2.5 Annotation examples ... 15
10.3 AnnotRichMedia ... 17
10.3.1 General ... 17
10.3.2 AnnotRichMedia properties ... 17
10.4 Annot3D ... 17
10.4.1 General ... 17
10.4.2 Annot3D properties ... 17
10.5 app ... 18
10.5.1 General ... 18
10.5.2 app properties ... 18
10.5.3 app methods ... 20
10.6 Bookmark ... 30
10.6.1 General ... 30
10.6.2 Bookmark properties ... 30
10.6.3 Bookmark methods ... 31
10.6.4 Bookmark Examples ... 32
10.7 Certificate ... 33
10.7.1 General ... 33
10.7.2 Certificate properties ... 34
10.8 color ... 36
10.8.1 General ... 36
10.8.2 color arrays ... 36
10.8.3 color properties ... 36
10.8.4 color methods ... 37
10.9 collection ... 38
10.9.1 General ... 38
10.9.2 collection properties ... 38
10.9.3 collection methods ... 38
10.10 collectionField ... 40
10.10.1 General ... 40

© ISO 2020 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

10.10.2 collectionField properties ... 40
10.11 Data ... 41
10.11.1 General ... 41
10.11.2 Data properties ... 42
10.11.3 Data methods ... 42
10.12 Dialog ... 43
10.12.1 General ... 43
10.12.2 Dialog methods ... 43
10.13 Doc ... 44
10.13.1 General ... 44
10.13.2 Doc properties ... 45
10.13.3 Doc methods ... 49
10.14 Embedded PDF ... 89
10.14.1 General ... 89
10.14.2 Embedded PDF properties ... 90
10.14.3 Embedded PDF methods ... 91
10.15 Error ... 91
10.15.1 General ... 91
10.15.2 Error properties ... 92
10.15.3 Error methods ... 92
10.16 event ... 93
10.16.1 General ... 93
10.16.2 Event type/name combinations ... 93
10.16.3 Document Event Processing ... 99
10.16.4 Form event processing ... 99
10.16.5 event properties ... 100
10.17 Field ... 105
10.17.1 General ... 105
10.17.2 Field versus widget attributes ... 106
10.17.3 Field properties ... 106
10.17.4 Field methods ... 118
10.18 FullScreen ... 135
10.18.1 General ... 135
10.18.2 FullScreen properties ... 135
10.19 global ... 136
10.19.1 General ... 136
10.19.2 Creating global properties ... 136
10.19.3 Deleting global properties ... 137
10.19.4 Global object security policy ... 137
10.19.5 global object methods ... 137
10.20 HostContainer ... 138
10.20.1 General ... 138
10.20.2 HostContainer properties ... 139
10.20.3 HostContainer methods ... 140
10.21 Icon ... 141
10.21.1 General ... 141
10.21.2 icon Properties ... 141
10.22 Link ... 141
10.22.1 General ... 141
10.22.2 Link properties ... 141
10.22.3 Link methods ... 142
10.23 Net ... 142
10.23.1 General ... 142
10.23.2 Net properties ... 142
10.23.3 Net methods ... 144
10.24 OCG ... 146
10.24.1 General ... 146
10.24.2 OCG properties ... 146

© ISO 2020 – All rights reserved
AVA Documents Document Sharing Hub
Full document details: BS ISO 21757-1:2020

10.24.3 OCG methods 147
10.25 PrintParams 148
10.25.1 General 148
10.25.2 PrintParams properties 148
10.26 RDN 151
10.26.1 General 151
10.26.2 RDN properties 152
10.27 ReadStream 152
10.27.1 General 152
10.27.2 ReadStream methods 152
10.28 security 153
10.28.1 General 153
10.28.2 security constants 153
10.28.3 security Properties 153
10.28.4 security Methods 154
10.29 SecurityHandler 157
10.29.1 General 157
10.29.2 SecurityHandler properties 157
10.29.3 SecurityHandler methods 160
10.30 SecurityPolicy 163
10.30.1 General 163
10.30.2 SecurityPolicy properties 163
10.31 SignatureInfo 163
10.31.1 General 163
10.31.2 SignatureInfo Base properties 163
10.31.3 SignatureInfo object public key security handler properties 165
10.31.4 Modification Detection and Prevention (MDP) Values 168
10.32 SOAP 168
10.32.1 General 168
10.32.2 SOAP properties 169
10.32.3 SOAP methods 169
10.33 Span 181
10.33.1 General 181
10.33.2 Span properties 181
10.34 Template 183
10.34.1 General 183
10.34.2 Template properties 183
10.34.3 Template methods 183
10.35 Thermometer 184
10.35.1 General 184
10.35.2 Thermometer properties 184
10.35.3 Thermometer methods 185
10.36 this 185
10.36.1 General 185
10.36.2 Variable and function name conflicts 186
10.37 util 186
10.37.1 General 186
10.37.2 util methods 186

11 ECMAScript 3D API 193
11.1 General 193
11.1.1 Basic Objects 193
11.1.2 Scene object 193
11.1.3 Canvas object 193
11.1.4 Runtime object 194
11.1.5 Resource objects 194
11.2 Event handlers 194
11.2.1 General 194
11.2.2 CameraEvent 194

AVA Documents
Document Sharing Hub

© ISO 2020 – All rights reserved

Full document details: BS ISO 21757-1:2020

11.2.3 KeyEvent ... 194
11.2.4 MouseEvent ... 194
11.2.5 RenderEvent ... 195
11.2.6 ScrollWheelEvent ... 195
11.2.7 SelectionEvent ... 195
11.2.8 TimeEvent ... 195
11.2.9 ToolEvent ... 195

12 Object overview ... 196
12.1 General ... 196
12.2 Animation ... 196
12.2.1 General ... 196
12.2.2 Animation properties ... 196
12.3 Background ... 196
12.3.1 General ... 196
12.3.2 Background object properties ... 196
12.3.3 Background object methods ... 196
12.4 BoundingBox ... 197
12.4.1 General ... 197
12.4.2 BoundingBox properties ... 197
12.5 Camera ... 197
12.5.1 General ... 197
12.5.2 Camera properties ... 198
12.5.3 Camera methods ... 199
12.6 CameraEvent ... 199
12.6.1 General ... 199
12.6.2 CameraEvent properties ... 199
12.7 CameraEventHandler ... 200
12.7.1 General ... 200
12.7.2 CameraEventHandler methods ... 200
12.8 Canvas ... 201
12.8.1 General ... 201
12.8.2 Canvas properties ... 201
12.8.3 Canvas methods ... 201
12.9 ClippingPlane ... 202
12.9.1 General ... 202
12.9.2 ClippingPlane Methods ... 202
12.10 Color ... 202
12.10.1 General ... 202
12.10.2 Color properties ... 202
12.10.3 Color methods ... 202
12.11 HitInfo ... 203
12.11.1 General ... 203
12.11.2 HitInfo properties ... 204
12.12 Host ... 204
12.12.1 General ... 204
12.13 Image ... 204
12.13.1 General ... 204
12.13.2 Image properties ... 204
12.13.3 Image methods ... 204
12.14 KeyEvent ... 205
12.14.1 General ... 205
12.14.2 KeyEvent properties ... 205
12.15 KeyEventHandler ... 207
12.15.1 General ... 207
12.15.2 KeyEventHandler methods ... 208
12.16 Light ... 208
12.16.1 General ... 208
12.16.2 Light properties ... 208

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

12.17 Material 209
12.17.1 General 209
12.17.2 Material properties 209
12.18 Matrix4x4 210
12.18.1 General 210
12.18.2 Matrix4x4 Properties 210
12.18.3 Matrix4x4 Methods 211
12.19 Mesh 219
12.19.1 General 219
12.19.2 Mesh properties 219
12.19.3 Mesh methods 219
12.20 MouseEvent 220
12.20.1 General 220
12.20.2 MouseEvent properties 220
12.21 MouseEventHandler 221
12.21.1 General 221
12.21.2 MouseEventHandler properties 221
12.21.3 MouseEventHandler methods 222
12.22 Node 222
12.22.1 General 222
12.22.2 Node properties 223
12.22.3 Node methods 223
12.23 Quaternion 224
12.23.1 General 224
12.23.2 Quaternion methods 224
12.24 RenderEvent 226
12.24.1 General 226
12.24.2 RenderEvent properties 226
12.25 RenderEventHandler 226
12.25.1 General 226
12.25.2 RenderEventHandler methods 226
12.26 Resource 228
12.26.1 General 228
12.26.2 Resource properties 228
12.26.3 Resource methods 228
12.27 Runtime 228
12.27.1 General 228
12.27.2 Runtime properties 228
12.27.3 Runtime methods 230
12.28 Scene 235
12.28.1 General 235
12.28.2 Scene methods 238
12.29 SceneObject 240
12.29.1 General 240
12.30 SceneObjectList 240
12.30.1 General 240
12.30.2 SceneObjectList methods 240
12.31 ScrollWheelEvent 241
12.31.1 General 241
12.31.2 ScrollWheelEvent 241
12.32 ScrollWheelEventHandler 242
12.32.1 General 242
12.32.2 ScrollWheelEventHandler methods 242
12.33 SelectionEvent 242
12.33.1 General 242
12.33.2 SelectionEvent properties 242
12.34 SelectionEventHandler 243
12.34.1 General 243

© ISO 2020 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

12.34.2 SelectionEventHandler methods ... 243
12.35 StateEvent ... 243
12.35.1 General ... 243
12.35.2 StateEvent properties ... 243
12.36 StateEventHandler ... 244
12.36.1 General ... 244
12.36.2 StateEventHandler methods ... 244
12.37 Texture ... 244
12.37.1 General ... 244
12.37.2 Texture properties ... 244
12.38 TimeEvent ... 245
12.38.1 General ... 245
12.38.2 TimeEvent properties ... 245
12.39 TimeEventHandler ... 245
12.39.1 General ... 245
12.39.2 TimeEventHandler methods ... 245
12.40 ToolEvent ... 246
12.40.1 General ... 246
12.40.2 ToolEventHandler properties ... 246
12.41 ToolEventHandler ... 246
12.41.1 General ... 246
12.41.2 ToolEventHandler methods ... 246
12.42 Vector3 ... 247
12.42.1 General ... 247
12.42.2 Vector3 methods ... 247
12.43 View ... 252
12.43.1 General ... 252
12.43.2 View properties ... 252

Bibliography ... 253

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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*.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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
```

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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`

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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`**.

© ISO 2020 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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;
}
```

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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");
```


© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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`.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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).

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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*.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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).

© ISO 2020 – All rights reserved

AVA Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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)`.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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

© ISO 2020 – All rights reserved

AVA Documents Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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 ]`

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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***.

© ISO 2020 – All rights reserved

Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020

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.

© ISO 2020 – All rights reserved
Documents
Document Sharing Hub
Full document details: BS ISO 21757-1:2020