INTERNATIONAL STANDARD

ISO 21757-1

First edition

Document management — ECMAScript for PDF —

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

PROOF/ÉPREUVE

Reference number
ISO 21757-1:2020(E)

COPYRIGHT PROTECTED DOCUMENT

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

Contents

Page

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.

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.

Document management — ECMAScript for PDF —

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

1 Scope

This document defines a set of ECMAScript objectives 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*.

Token characters used to delimit objects and describe the structure of PDF files, as defined in ISO 32000-2:—, 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:—, 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.

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:—, "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
```