

1484.11.2™

IEEE Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication

IEEE Computer Society

Sponsored by the
Learning Technology Standards Committee



Published by
The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

4 March 2004

Print: SH95175
PDF: SS95175

IEEE Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication

Sponsor

Learning Technology Standards Committee
of the
IEEE Computer Society

Approved 11 September 2003

IEEE-SA Standards Board

Abstract: An ECMAScript application programming interface (API) for content-to-runtime-services communication is described in this standard. It is based on a current industry practice called “CMI—computer managed instruction.” This API enables the communication of information between content and a runtime service (RTS) typically provided by a learning management system (LMS) via common API services using the ECMAScript language. The purpose of this standard is to build consensus around, resolve ambiguities, and correct defects in existing specifications for an ECMA-Script API for exchanging data between learning-related content and an LMS.

Keywords: application programming interface, API, computer managed instruction, content object, ECMAScript, ECMAScript API, learning content, learning management system, LMS, runtime service, RTS

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2004 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 4 March 2004. Printed in the United States of America.

IEEE is a registered trademarks in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

Print: ISBN 0-7381-3824-X SH95175
PDF: ISBN 0-7381-3825-8 SS95175

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied “**AS IS**.”

The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

[This introduction is not part of IEEE Std 1484.11.2-2003, IEEE Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication.]

This standard describes a learning content ECMAScript API to support the data transfer needs of content with a runtime service in a Web-browser-based content delivery environment. This standard provides a starting point for specifying a data communication channel and methods to support the data transfer needs of education, training, and human performance support content. The capabilities of this API model may be extended as other communication needs or methods arise.

Acknowledgements

The IEEE LTSC P1484.11.2 CMI Working Group wishes to thank Tom King and Boyd Nielsen for their contributions of initial documents used for the preparation of this standard.

This standard is based on material from Appendix B of the AICC/CMI Guidelines for Interoperability, Revision 3.4.

Participants

At the time this standard was completed, the working group had the following membership:

Tyde Richards, *Chair*
Jack Hyde, *Chair (December 1998–March 2001)*
Scott Lewis, *Technical Editor*

Mitchell Bonnett
Frank Farance
Mike Fore
Leonard Greenberg

Tom King
Rolf Lindner
Kiyoshi Nakabayashi
Boyd Nielsen

Claude Ostyn
Daniel Rehak
Robby Robson
Schawn Thropp

The following members of the balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Mitchell Bonnett
Keith Chow
Hockemeyer Cord
Guru Dutt Dhingra
Kameshwar Eranki
Frank Farance
Harriet Feldman
Mike Fore
Eddy Forte

Ronald Hofer
Jack Hyde
Robert Bruce Kelsey
Tom King
Rolf Lindner
Gregory Luri
William Melton
George Miao

Kiyoshi Nakabayashi
Boyd Nielsen
Claude Ostyn
Daniel Rehak
Tyde Richards
Robby Robson
Larry Stern
Brian Taliesin
Schawn Thropp

When the IEEE-SA Standards Board approved this standard on 11 September 2003, it had the following membership:

Don Wright, *Chair*
Howard M. Frazier, *Vice Chair*
Judith Gorman, *Secretary*

H. Stephen Berger
Joe Bruder
Bob Davis
Richard DeBlasio
Julian Forster*
Toshio Fukuda
Arnold M. Greenspan
Raymond Hapeman

Donald M. Heirman
Laura Hitchcock
Richard H. Hulett
Anant Jain
Lowell G. Johnson
Joseph L. Koepfinger*
Tom McGean
Steve Mills

Daleep C. Mohla
William J. Moylan
Paul Nikolic
Gary Robinson
Malcolm V. Thaden
Geoffrey O. Thompson
Doug Topping
Howard L. Wolfman

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Alan Cookson, *NIST Representative*
Satish K. Aggarwal, *NRC Representative*

Andrew Ickowicz
IEEE Standards Project Editor

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
2. References	2
3. Definitions, abbreviations, and acronyms	2
3.1 Definitions	2
3.2 Abbreviations and acronyms	3
4. Conformance	3
4.1 Behavior	4
4.2 API implementation	4
4.3 Content object use of an API implementation	4
4.4 Outside of scope	5
5. Conceptual model (informative)	5
5.1 Simplified learning content ECMAScript API communication model	5
5.2 Basic scenario	6
5.3 Implementation for Web-browser-based content	7
5.4 ECMAScript API extensibility	9
6. API instantiation and binding	9
6.1 Instantiation of an instance of an API implementation	9
6.2 Multiple instances	10
6.3 Binding a content object to an API instance	10
6.4 Sample implementation to find an API instance (informative)	10
7. Content communication state model	11
7.1 Communication and error states	11
7.2 Events	12
8. ECMAScript API methods and syntax	16
8.1 Session methods	16
8.2 Data-transfer methods	18
8.3 Support methods	21
8.4 API implementation error codes	23
Annex A (informative) Bibliography	26
Annex B (informative) When to call the terminate communication session method	27

IEEE Standard for Learning Technology—ECMAScript Application Programming Interface for Content to Runtime Services Communication

1. Overview

The scope and purpose of this standard are discussed in 1.1 and 1.2.

1.1 Scope

This standard describes an ECMAScript application programming interface (API) for content-to-runtime-services communication. This standard is based on an API defined in AICC/CMI Guidelines for Interoperability, Revision 3.4 [B1]¹, defined by the Aviation Industry CBT Committee (AICC). It defines common API services in the ECMAScript language that enable the communication of information between learning-related content and a runtime service (RTS) used to support learning management. This standard does not address the data structures that may be transmitted, data security, or communication between an RTS and a related management system.

1.2 Purpose

There is widespread acknowledgement that the ECMAScript API for content-to-runtime-services communication defined in the AICC/CMI Guidelines for Interoperability, Revision 3.4, has broad applicability to systems used for learning management. The purpose of this standard is to build consensus around, resolve ambiguities in, and correct defects in this ECMAScript API for exchanging data between learning-related content and a runtime service used to support learning management.

¹The numbers in brackets correspond to those of the bibliography in Annex A.

2. References

This standard shall be used in conjunction with the following publication.

ISO/IEC 16262:1998, Information technology—ECMAScript language specification.²

W3C, Document Object Model (DOM) Level 3 Core Specification, Version 1.0, W3C Working Draft 26 February 2003.³ (See: <http://www.w3.org/DOM/>)

3. Definitions, abbreviations, and acronyms

3.1 Definitions

For purposes of this standard, the following terms and definitions apply. IEEE 100, *The Authoritative Dictionary of IEEE Standards Terms*, Seventh Edition [B2], should be referenced for terms not defined in this clause.

3.1.1 application programming interface (API): A set of standard software interrupts, calls, functions, and data formats that can be used by an application program to access network services, devices, or operating systems.

3.1.2 application programming interface implementation (API implementation): An implementation of this standard that supplies the services of the application programming interface, in contrast to an implementation of this standard that uses the application programming.

3.1.3 application programming interface instance (API instance): An individual execution context and state of an application programming interface implementation.

NOTE—The notion of “execution context” in this standard is the same as in ECMAScript.

3.1.4 communication session: An active connection between a content object and an application programming interface instance.

3.1.5 content object: A collection of digital content that is intended for presentation to a learner by a learning technology system. A content object may include learning material and processing code. *Example:* A content object might be an HTML page with an embedded video clip and an ECMAScript.

3.1.6 Document Object Model (DOM): A platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure, and style of documents. The document can be further processed, and the results of that processing can be incorporated back into the presented page (see W3C, Document Object Model (DOM) Level 3 Core Specification, Version 1.0⁴).

3.1.7 ECMAScript: An object-oriented programming language for performing computations and manipulating computational objects within a host environment as defined by ISO/IEC 16262:1998.

²ISO/IEC publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembé, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iso.ch/>). ISO/IEC publications are also available in the United States from Global Engineering Documents, 15 Inverness Way East, Englewood, Colorado 80112, USA (<http://global.ihs.com/>). Electronic copies are available in the United States from the American National Standards Institute, 25 West 43rd Street, 4th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

³More information may be obtained from the following URL: <http://www.w3.org/DOM/>

⁴Information on references can be found in Clause 2.

3.1.8 implementation behavior: Observable actions or appearance of an implementation. *See also:* **implementation-defined behavior; undefined behavior; unspecified behavior.**

3.1.9 implementation-defined behavior: Unspecified behavior for which each implementation documents how the choice among the available alternatives is made. *Example:* Permitting a maximum size, as measured in octets, of a coding. *See also:* **implementation behavior; undefined behavior; unspecified behavior.**

3.1.10 launch (v.): To cause a content object to be delivered to a learner.

3.1.11 learner: An individual engaged in acquiring knowledge or skills with a learning technology system.

3.1.12 learning content: Digital content intended for use in a learning experience.

3.1.13 learning management system (LMS): A computer system that may include the capabilities to register learners, schedule learning resources, control and guide the learning process, analyze and report learner performance, and schedule and track learners.

NOTE—Some implementations of LMSs also have the ability to launch and deliver content. For this standard, these capabilities are known as a runtime service.

3.1.14 runtime service (RTS): Software that controls the execution and delivery of learning content and that may provide services such as resource allocation, scheduling, input-output control, and data management.

3.1.15 undefined behavior: Implementation behavior for which a standard imposes no requirements. Undefined behavior describes nonconforming implementations of this standard. Possible undefined behaviors include, but are not limited to: ignoring the situation completely, unpredictable results, behaving in a documented manner characteristic of the environment, and terminating processing. *See also:* **implementation behavior; implementation-defined behavior; unspecified behavior.**

3.1.16 unspecified behavior: Implementation behavior for which a standard provides two or more alternatives and imposes no further requirements on what is chosen in any instance. *See also:* **implementation behavior; implementation-defined behavior; undefined behavior.**

3.2 Abbreviations and acronyms

AICC	Aviation Industry CBT Committee
API	application programming interface
CBT	computer-based training
CMI	computer managed instruction
DOM	Document Object Model
LMS	learning management system
RTS	runtime service

4. Conformance

Conformance to this standard is discussed in 4.1–4.4.

4.1 Behavior

In this standard, “shall” is to be interpreted as a requirement on an implementation; “shall not” is to be interpreted as a prohibition. This standard defines undefined behavior in the following ways:

- If a “shall” requirement or “shall not” prohibition is violated;
- By the words “undefined behavior;” and
- By the omission of any explicit definition of behavior.

There is no difference in emphasis among these ways; they all describe “behavior that is undefined.”

NOTES:

1—“Unspecified behavior” and “implementation-defined behavior” both describe conforming behavior. With “unspecified behavior,” several choices may be possible. This standard does not specify which one is chosen, and the implementation *is not required* to “define” (i.e., tell in advance) which set of behaviors it has chosen. Similarly, with “implementation-defined behavior,” several choices may be possible and this standard does not specify which one is chosen. However, the implementation *is required* to “define” (i.e., tell in advance) which set of behaviors it has chosen.

2—“Implementation behavior” describes observable action or appearance, which may be conforming or nonconforming. “Undefined behavior” describes nonconforming implementations of this standard.

4.2 API implementation

A conforming ECMAScript API implementation shall

- Be instantiated as an object within the Document Object Model (DOM) (see W3C, Document Object Model (DOM) Level 3 Core Specification, Version 1.0) environment of the content object as specified in Clause 6 of this standard;
- Conform to the state model in Clause 7 of this standard; and
- Implement all methods and error returns, as specified in Clause 8 of this standard.

4.3 Content object use of an API implementation

A conforming content object shall

- Find an instance of the API implementation as an ECMAScript-compatible object within the DOM environment of the content object as described in Clause 6 of this standard;
- Call `Initialize` after the content object is launched and has located the API instance before calling any other method (see 8.1.1);
- Call `Terminate` when the content object determines it no longer needs to communicate with the API instance or is unloaded, whichever happens first (see 8.1.2);
- Not call any data-transfer method or `Initialize` after `Terminate` has been called; and
- Use conforming syntax when calling any optional data-transfer methods (see 8.2) or when calling any optional support methods (see 8.3).

A content object strictly conforms to this standard if it does not depend on implementation-defined behavior or unspecified behavior.