



IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)— Object Model Template (OMT) Specification

IEEE Computer Society

Developed by the
Simulation Interoperability Standards Organization/
Standards Activities Committee (SISO/SAC)

IEEE Std 1516.2™-2025
(Revision of IEEE Std 1516.2-2010)

STANDARDS

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)— Object Model Template (OMT) Specification

Developed by the

**Simulation Interoperability Standards Organization/
Standards Activities Committee (SISO/SAC)**
of the
IEEE Computer Society

Approved 14 February 2025

IEEE SA Standards Board

Abstract: The High Level Architecture (HLA) has been developed to provide a common architecture for distributed modeling and simulation. The HLA defines an integrated approach that provides a common framework for the interconnection of interacting simulations. This document, the third in a family of three related HLA documents, defines the format and syntax (but not content) of HLA object models. The standards contained in this architecture are interrelated and need to be considered as a product set, when changes are made. They each have value independently.

Keywords: architecture, class attribute, data distribution management, federate, federation, federation execution, federation object model, framework, High Level Architecture, IEEE 1516.2™, instance attribute, instance attribute ownership, interaction class, joined federate, object class, object model template, rules, runtime infrastructure, simulation object model, time-constrained, time-regulating.

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

Copyright © 2025 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 1 August 2025. Printed in the United States of America.

IEEE and 802 are registered trademarks in the U.S. Patent & Trademark Office, owned by The Institute of Electrical and Electronics Engineers, Incorporated.

Print: ISBN 979-8-8557-2020-4 STDPD27823
PDF: ISBN 979-8-8557-2019-8 STD27823

IEEE prohibits discrimination, harassment and bullying.

For more information, visit <http://www.ieee.org/web/aboutus/whatis/policies/p9-26.html>.

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.

Important Notices and Disclaimers Concerning IEEE Standards Documents

IEEE Standards documents are made available for use subject to important notices and legal disclaimers. These notices and disclaimers, or a reference to this page (<https://standards.ieee.org/ipr/disclaimers.html>), appear in all standards and may be found under the heading “Important Notices and Disclaimers Concerning IEEE Standards Documents.”

Notice and Disclaimer of Liability Concerning the Use of IEEE Standards Documents

IEEE Standards documents are developed within IEEE Societies and subcommittees of IEEE Standards Association (IEEE SA) Board of Governors. IEEE develops its standards through an accredited consensus development process, which brings together volunteers representing varied viewpoints and interests to achieve the final product. IEEE standards are documents developed by volunteers with scientific, academic, and industry-based expertise in technical working groups. Volunteers are not necessarily members of IEEE or IEEE SA and participate without compensation from IEEE. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

IEEE makes no warranties or representations concerning its standards, and expressly disclaims all warranties, express or implied, concerning this standard, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. IEEE Standards documents do not guarantee safety, security, health, or environmental protection, or guarantee against interference with or from other devices or networks. In addition, IEEE does not warrant or represent that the use of the material contained in its standards is free from patent infringement. IEEE Standards documents are supplied “AS IS” and “WITH ALL FAULTS.”

Use of an IEEE standard is wholly voluntary. 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.

In publishing and making its standards available, IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity, nor is IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing any IEEE Standards document, should rely upon their own independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

IN NO EVENT SHALL IEEE BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO: THE NEED TO PROCURE SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE PUBLICATION, USE OF, OR RELIANCE UPON ANY STANDARD, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE AND REGARDLESS OF WHETHER SUCH DAMAGE WAS FORESEEABLE.

Translations

The IEEE consensus balloting process involves the review of documents in English only. In the event that an IEEE standard is translated, only the English version published by IEEE is the approved IEEE standard.

Use by artificial intelligence systems

In no event shall material in any IEEE Standards documents be used for the purpose of creating, training, enhancing, developing, maintaining, or contributing to any artificial intelligence systems without the express, written consent of IEEE SA in advance. “Artificial intelligence” refers to any software, application, or other system that uses artificial intelligence, machine learning, or similar technologies, to analyze, train, process, or generate content. Requests for consent can be submitted using the Contact Us form.

Official statements

A statement, written or oral, that is not processed in accordance with the IEEE SA Standards Board Operations Manual shall not be considered or inferred to be the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal position of IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that the presenter’s views should be considered the personal views of that individual rather than the formal position of IEEE, IEEE SA, the Standards Committee, or the Working Group. Statements made by volunteers may not represent the formal position of their employer(s) or affiliation(s).

Comments on standards

Comments for revision of IEEE Standards documents are welcome from any interested party, regardless of membership affiliation with IEEE or IEEE SA. However, **IEEE does not provide interpretations, consulting information, or advice pertaining to IEEE Standards documents.**

Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Since IEEE standards represent a consensus of concerned interests, it is important that any responses to comments and questions also receive the concurrence of a balance of interests. For this reason, IEEE and the members of its Societies and subcommittees of the IEEE SA Board of Governors are not able to provide an instant response to comments, or questions except in those cases where the matter has previously been addressed. For the same reason, IEEE does not respond to interpretation requests. Any person who would like to participate in evaluating comments or in revisions to an IEEE standard is welcome to join the relevant IEEE working group. You can indicate interest in a working group using the Interests tab in the Manage Profile & Interests area of the [IEEE SA myProject system](#).¹ An IEEE Account is needed to access the application.

Comments on standards should be submitted using the [Contact Us](#) form.²

Laws and regulations

Users of IEEE Standards documents should consult all applicable laws and regulations. Compliance with the provisions of any IEEE Standards document does not constitute compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable

¹ Available at: <https://development.standards.ieee.org/myproject-web/public/view.html#landing>.

² Available at: <https://standards.ieee.org/content/ieee-standards/en/about/contact/index.html>.

regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Data privacy

Users of IEEE Standards documents should evaluate the standards for considerations of data privacy and data ownership in the context of assessing and using the standards in compliance with applicable laws and regulations.

Copyrights

IEEE draft and approved standards are copyrighted by IEEE under U.S. and international copyright laws. They are made available by IEEE and are adopted for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making these documents available for use and adoption by public authorities and private users, neither IEEE nor its licensors waive any rights in copyright to the documents.

Photocopies

Subject to payment of the appropriate licensing fees, IEEE will grant users a limited, non-exclusive license to photocopy portions of any individual standard for company or organizational internal use or individual, non-commercial use only. To arrange for payment of licensing fees, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; +1 978 750 8400; <https://www.copyright.com/>. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Updating of IEEE Standards documents

Users of IEEE Standards documents should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect.

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, 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 order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#).³ For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#).⁴ Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the

³ Available at: <https://ieeexplore.ieee.org/browse/standards/collection/ieee>.

⁴ Available at: <https://standards.ieee.org/standard/index.html>.

Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

Patents

IEEE standards are developed in compliance with the [IEEE SA Patent Policy](#).⁵

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 by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

Essential Patent Claims may exist for which a Letter of Assurance has not been received. The IEEE is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims, or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

IMPORTANT NOTICE

Technologies, application of technologies, and recommended procedures in various industries evolve over time. The IEEE standards development process allows participants to review developments in industries, technologies, and practices, and to determine what, if any, updates should be made to the IEEE standard. During this evolution, the technologies and recommendations in IEEE standards may be implemented in ways not foreseen during the standard's development. IEEE standards development activities consider research and information presented to the standards development group in developing any safety recommendations. Other information about safety practices, changes in technology or technology implementation, or impact by peripheral systems also may be pertinent to safety considerations during implementation of the standard. Implementers and users of IEEE Standards documents are responsible for determining and complying with all appropriate safety, security, environmental, health, and interference protection practices and all applicable laws and regulations.

⁵ Available at: <https://standards.ieee.org/about/sasb/patcom/materials.html>.

Participants

At the time this revision was sent to SA ballot, the HLA Product Development Group/Working Group had the following membership:

Randy Saunders, *Chair*
Björn Möller, *Vice Chair*
Katherine L. Morse, *Secretary*

The working group acknowledges the following Object Model Template Specification Editors and Drafting Group members who contributed to the preparation of this standard:

Robert Lutz, *Object Model Template Editor*
David Drake, *XML Editor*

Fredrik Antelius	Paul Houldsworth	Jose Ruiz
Ryan Bernhold	Kyle Isakson	Heath Rush
Curtis Blais	Mikael Karlsson	Kevin Seavey
Ryan Brunton	Kyle Isakson	Graham Shanks
Robert Butterfield	Mikael Karlsson	Pierre Siron
Andrew Ceranowicz	Patrice Le Leydour	Keith Snively
Anthony Cramp	Jean Philippe Lebel	Steffen Strassburger
Aaron Dubois	Björn Löfstrand	Tom van den Berg
Matt Figueroa	Mark McCall	Leon van der Heijden
Rick Gaylor	Matthew McGlawn	Alexander Vankov
Cristian Gheorghiu	Spencer Ochsner	René Verhage
Peggy Gravitz	Tim Pokorny	Douglas Wood
Paul Gustavson	David Ronnfeldt	Simone Youngblood

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

Fredrik Antelius	Mikael Karlsson	Manfred Roza
Martin Asprusten	James Kogler	Andrey Sadovykh
Tom Berg	Murray Little	Randy Saunders
Curtis Blais	Zeeger Lubsen	Stephen Schwarm
Shelly Blake-Plock	Robert Lutz	Graham Shanks
Anthony Cramp	Björn Möller	Keith Snively
Dannie Cutts	Katherine L. Morse	Eugene Stoudenmire
Uwe Dobrindt	Thomas Mullins	Steffen Strassburger
David Drake	Rajesh Murthy	Douglas Wood
David Fuschi	Nayeem Ninad	Simone Youngblood
Werner Hoelzl	Bansi Patel	Yu Yuan
	Venkatesha Prasad	

When the IEEE SA Standards Board approved this recommended practice on 14 February 2025, it had the following membership:

David J. Law, *Chair*
Jon Walter Rosdahl, *Vice Chair*
Gary Hoffman, *Past Chair*
Alpesh Shah, *Secretary*

Sara R. Biyabani
Ted Burse
Stephen Dukes
Doug Edwards
J. Travis Griffith
Guido R. Hiertz
Ronald W. Hotchkiss
Hao Hu

Yousef Kimiagar
Joseph L. Koepfinger*
Howard Li
Xiaohui Liu
John Haiying Lu
Kevin W. Lu
Hiroshi Mano
Paul Nikolich

Robby Robson
Jon W. Rosdahl
Mark Siira
Lei Wang
F. Keith Waters
Sha Wei
Philip B. Winston
Don Wright

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1516.2-2025, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification.

This document has been developed to record an international standard for the High Level Architecture (HLA). It serves as one of three related standards for the HLA.

This standard, describing the HLA Object Model Template (OMT), defines the format and syntax for recording information in HLA object models.

This new version of the IEEE 1516 HLA was produced in 2016 through 2024 by the HLA Product Development Group of the Simulation Interoperability Standards Organization (SISO). It incorporates a number of updates based on practical application of earlier versions of the standard. The purpose of the new version is to better support development, deployment, and net-centricity of distributed simulations.

Schema, API, and Documentation

Schema and API: The IEEE hereby grants a general, royalty-free license to copy, distribute, display, and make derivative works from this material, for all purposes, provided that any use of the material contains the following attribution: “Reprinted with permission from IEEE Std 1516™-2025.” Should a reader require additional information, contact the Manager, Standards Intellectual Property, IEEE Standards Association (stds-ipr@ieee.org).

Documentation: The IEEE hereby grants a general, royalty-free license to copy, distribute, display, and make derivative works from this material, for noncommercial purposes, provided that any use of the material contains the following attribution: “Reprinted with permission from IEEE Std 1516™-2025.” The material may not be used for a commercial purpose without express written permission from the IEEE. Should a reader require additional information, contact the Manager, Standards Intellectual Property, IEEE Standards Association (stds-ipr@ieee.org).

CONTENTS

1.	Overview.....	12
1.1	Scope.....	12
1.2	Purpose.....	12
1.3	Word Usage.....	12
1.4	Background	13
2.	Normative references	15
3.	Definitions, acronyms, and abbreviations.....	15
3.1	Definitions.....	15
3.2	Abbreviations and acronyms.....	30
3.3	Conventions	31
4.	HLA OMT components	32
4.1	Object model identification table.....	33
4.2	Object class structure table.....	38
4.3	Interaction class structure table.....	42
4.4	Directed interaction table	45
4.5	Attribute table	46
4.6	Parameter table.....	51
4.7	Dimension table	53
4.8	Time representation table.....	56
4.9	User-supplied tag table.....	58
4.10	Synchronization table.....	60
4.11	Transportation type table.....	61
4.12	Update rate table	63
4.13	Switches table	64
4.14	Data type tables	67
4.15	Notes table.....	85
5.	FOM/SOM lexicon	87
5.1	Purpose/background.....	87
5.2	Object class definition table.....	87
5.3	Interaction class definition table	87
5.4	Attribute definition table	88
5.5	Parameter definition table	89
6.	Conformance.....	89
6.1	Conformance labels.....	90
6.2	Conformance verification.....	90
6.3	Interface specification services usage table	94
7.	FOM module/SOM module merging rules	100
	Annex A (informative) Table entry notation	105

Annex B (normative) Common normalization functions	106
Annex C (normative) FOM module/SOM module merging principles.....	108
Annex D (normative) OMT data interchange format	113
Annex E (normative) OMT conformance XML Schema	115
Annex F (informative) OMT DIF SOM example.....	116
Annex G (informative) OMT DIF FOM example	117
Annex H (informative) Bibliography	122

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)— Object Model Template (OMT) Specification

1. Overview

1.1 Scope

This document defines the format and syntax for recording information in High Level Architecture (HLA) object models, to include objects, attributes, interactions, and parameters. It does not define the specific data (e.g., vehicles, unit types) that will appear in the object models.

1.2 Purpose

The HLA has been developed to provide a common architecture for distributed modeling and simulation. The HLA requires that federations (sets of federates) and individual federates (simulations, supporting utilities, or interfaces to live systems) be described by an object model that identifies the data exchanged at runtime to achieve federation objectives. This standard defines the documentation of the object model.

1.3 Word Usage

The word *shall* indicates mandatory requirements strictly to be followed in order to conform to the standard and from which no deviation is permitted (shall equals is required to).^{6,7}

The word *should* indicates that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others; or that a certain course of action is preferred but not necessarily required (should equals is recommended that).

The word *may* is used to indicate a course of action permissible within the limits of the standard (may equals is permitted to).

The word *can* is used for statements of possibility and capability, whether material, physical, or causal (can equals is able to).

⁶The use of the word *must* is deprecated and cannot be used when stating mandatory requirements, *must* is used only to describe unavoidable situations.

⁷The use of the word *will* is deprecated and cannot be used when stating mandatory requirements, *will* is only used in statements of fact.

1.4 Background

1.4.1 Object model template rationale

A standardized structural framework, or template, for specifying HLA object models is an essential component of the HLA for the following reasons:

- It provides a commonly understood mechanism for specifying the exchange of data and general coordination among members of a federation.
- It provides a common, standardized mechanism for describing the capabilities of potential federation members.
- It facilitates the design and application of common tool sets for development of HLA object models.

HLA object models may be used to describe an individual federation member (federate), creating an HLA simulation object model (SOM), or to describe a named set of multiple interacting federates (federation), creating a federation object model (FOM). HLA object models may also be used to define subsets of a FOM or SOM, which are identified as FOM modules and SOM modules respectively. In whichever case, the primary objective of the HLA object model template (OMT) is to facilitate interoperability among simulations and reuse of simulation components. All discussion of HLA object models in this document applies to FOMs, SOMs, FOM modules, and SOM modules, unless explicitly stated otherwise.

1.4.2 Federation object models

During development of an HLA federation, it is critical that all federation members achieve a common understanding as to the nature or character of all required communications among participating federates. The primary purpose of an HLA FOM is to provide a specification for data exchange among federates in a common, standardized format. The content of this data includes an enumeration of all object and interaction classes pertinent to the federation and a specification of the attributes or parameters that characterize these classes. Taken together, the individual components of an HLA FOM establish the “information model contract” that is necessary (but not sufficient) to achieve interoperability among the federates.

A FOM can be constructed as a single self-contained entity, or can be composed from multiple FOM modules. In this first case, the FOM is said to consist of exactly one FOM module.

1.4.3 Simulation object models

A critical step in the formation of a federation is the process of determining the composition of individual federates to best meet the sponsor’s overall objectives. An HLA SOM is a specification of the types of information that an individual federate can provide to HLA federations and the information that an individual federate can receive from other federates in HLA federations. The standard format in which SOMs are expressed facilitates determination of the suitability of federates for participation in a federation.

The HLA OMT formats described in this document are generally applicable to either FOMs or SOMs, and also the FOM modules and SOM modules that can be used to formulate a FOM or SOM. Thus, SOMs and SOM modules are also characterized by their objects, attributes, interactions, and parameters. A SOM is specified using one or more SOM modules and one optional Management Object Model (MOM) and Initialization Module (MIM). The primary benefit from the common utilization of the OMT formats for FOMs, FOM modules, SOMs, and SOM modules is that it provides a common frame of reference for describing object models in the HLA community. In some cases, this commonality may even allow SOM modules to be integrated as piece parts in a FOM, facilitating FOM construction.

1.4.4 Relationship of HLA and object-oriented concepts

Although the HLA OMT is the standardized documentation structure for HLA object models, FOMs, FOM modules, SOMs, and SOM modules do not completely correspond to common definitions of object models in object-oriented analysis and design (OOAD) techniques. In the OOAD literature, an object model is described as an abstraction of a system developed for the purpose of fully understanding the system. To achieve this understanding, most object-oriented (OO) techniques recommend defining several views of the system. For HLA object models, the intended scope of the system description is much narrower, focusing mainly on requirements and capabilities for federate information exchange. For SOMs and SOM modules, the intent is to describe the public interface of the federate in terms of an identified set of supported HLA object classes and interaction classes. A more complete description of how a federate is designed and functions internally (for example, a traditional OO object model) should be provided via documentation resources other than the SOM. For FOMs and FOM modules, the intent is to describe information exchange that happens during a federation execution.

Differences between HLA and OOAD principles and concepts also appear at the individual object level. In the OOAD literature, objects are defined as software encapsulations of data and operations (methods). In the HLA, objects are defined entirely by the identifying characteristics (attributes), values of which are exchanged between federates during a federation execution. Any OO-related behaviors and operations that affect the values of HLA object attributes are kept resident in the federates. Although HLA class structures are driven by subscription requirements and FOM growth concerns, class structures in OO systems are typically driven by efficiency and maintainability concerns.

As discussed above, HLA object classes are described by the attributes that are defined for them. These are, in OO parlance, data members of the class. These attributes, which are abstract properties of HLA object classes, are referred to as class attributes. HLA object instances are spawned via an HLA service using an HLA object class as a template. Each attribute contained by an HLA object instance is called an *instance attribute*.

OO objects interact via message passing, in which one OO object invokes an operation provided by another OO object. HLA objects do not directly interact. It is the federates that interact, via HLA services, by updating instance attribute values or sending interactions. Also, responsibility for updating the instance attributes associated with an HLA object instance can be distributed among different federates in a federation (effectively distributing responsibility for maintaining the HLA object instance's state across the federation), whereas OO objects encapsulate state locally and associate update responsibilities with operations that are closely tied to the object's implementation in an OO programming language.

Specialization of HLA object and interaction classes are discussed in terms of data tree structures. A data tree structure is defined as, “an abstract hierarchical structure consisting of nodes [classes] connected by branches, in which; each branch connects one node [class] to a directly subsidiary node; there is a unique node called the root [root class] which is not subsidiary to any other node; and every node [class] besides the root is directly subsidiary to exactly one other node [class]” (*The IEEE Standards Dictionary Online*).⁸ Furthermore, a node without any subsidiary node is called a leaf node [leaf class].

In addition to the stated semantic variations in shared terminology, other differences may also exist. Precise definitions of all HLA terms can be found in Clause 3.

⁸IEEE Standards Dictionary Online is available at <http://dictionary.ieee.org/>. An IEEE account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

ANSI X3.4-1986 (R1997), Information Systems—Coded Character Sets—7 Bit American National Standard Code Information Interchange (7-Bit ASCII).⁹

IEEE Std 754™-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.^{10, 11}

IEEE Std 1516-2025, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules.

IEEE Std 1516.1-2025, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification.

REC-XML-20060816, Extensible Markup Language (XML) 1.0 (Fourth Edition), W3C Recommendation, Aug. 2006.¹²

The Unicode Consortium, Ed. *The Unicode Standard*, Version 3.0. Reading, MA: Addison-Wesley Developers Press, 2000.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary Online* should be referenced for terms not defined in this clause.¹³

accuracy: The measure of the maximum deviation of an attribute or a parameter value in the simulation or federation from reality or some other chosen standard or referent.

active subscription: A request to the runtime infrastructure (RTI) for the kinds of data (class attributes as well as interactions) that the joined federate is currently interested in receiving. The RTI uses these data along with publication data received from other joined federates to support services, such as

- a) *Start/Stop Registration*
- b) *Turn On/Off Updates*
- c) *Turn On/Off Interactions*

The RTI also uses the subscription (and publication) data to determine how to route data to the joined federates that require that data.

⁹ANSI publications are available from (<http://www.ansi.org/>).

¹⁰IEEE publications are available (<http://standards.ieee.org/>).

¹¹The IEEE standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

¹²This document is available at <http://www.w3.org/>.

¹³*IEEE Standards Dictionary Online* is available at <http://dictionary.ieee.org/>. An IEEE account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

attribute: A named characteristic of an object class or object instance. *See also:* **class attribute**; **instance attribute**.

attribute ownership: The property of an instance attribute that gives a joined federate the capability to supply values for that instance attribute to its federation execution. *See also:* **instance attribute**.

authorizer: An interface used to determine if a specific set of service invocations (*Connect*, *Create Federation Execution*, *Destroy Federation Execution*, and *Join Federation Execution*) are allowed for the federate, given its credentials.

available attributes: The set of declared attributes of an object class in union with the set of inherited attributes of that object class.

available dimensions:

- a) Pertaining to an object class: the dimensions associated with that object class in union with the set of inherited dimensions in the federation object model (FOM).
- b) Pertaining to an attribute: The available dimensions of an instance attribute are the available dimensions of the known class. When registering, subscribing, unsubscribing, requesting updates with regions or getting available dimensions, the available dimensions are the available dimensions of the object class specified in the service invocation.
- c) Pertaining to an interaction class: the dimensions associated with that interaction class in union with the set of inherited dimensions the FOM. The available dimensions of a sent interaction are the available dimensions of the interaction class specified in the *Send Interaction with Regions* service invocation.

NOTE—See 9.1 in IEEE Std 1516.1-2025.¹⁴

available directed interactions: The set of directed interaction classes of an object class in union with the set of inherited directed interactions of that object class.

available parameters: The set of declared parameters of an interaction class in union with the set of inherited parameters of that interaction class.

best effort: A transportation type that does not provide any guarantee regarding delivery (messages may be out of order, duplicate, or dropped).

candidate discovery class: The registered class of an object instance, if subscribed. If the registered class of an object instance is not subscribed, the closest superclass of the registered class of the object instance to which the joined federate is subscribed. Candidate discovery class pertains to object instances only.

candidate received class: The sent class of an interaction, if subscribed. If the sent class of an interaction is not subscribed, the closest superclass of the sent class of the interaction to which the joined federate is subscribed. Candidate received class pertains to interactions and directed interactions only.

class: A description of a group of items with similar properties, common behavior, common relationships, and common semantics.

class attribute: A named characteristic of an object class denoted by a pair composed of an object class designator and an attribute designator.

¹⁴Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

class directed interaction: A named characteristic of an object class denoted by a pair composed of an object class designator and an interaction class designator. The characteristic is the capability to have interactions directed to instances of the object class or its subclasses.

class hierarchy: A specification of a class-subclass or “is-a” relationship between classes in a given domain.

coadunate: To attach an object instance handle and an object instance name to an object instance. The object instance name can be provided specifically by the federate or implicitly by the runtime infrastructure (RTI).

collection: A group of elements in which an element may occur multiple times (this definition corresponds to the mathematical notion of a bag).

collection of pairs: A group of pairs in which multiple pairs may have the same first component and a given pair may occur multiple times.

compliant object model: A High Level Architecture (HLA) federation object model (FOM) or simulation object model (SOM) that fully conforms with all of the rules and constraints specified in object model template (OMT).

constrained set of pairs: A group of pairs in which no two pairs have the same first component (this definition corresponds to the mathematical notion of a function). An example would be a group of instance attribute and value pairs; each instance attribute may have at most one value associated with it.

corresponding class attribute of an instance attribute: The class attribute that, from the perspective of a given joined federate, is the class attribute of the joined federate’s known class for the object instance containing the instance attribute that has the same attribute designator as the instance attribute.

corresponding instance attributes of a class attribute: The instance attributes that, from the perspective of a given joined federate, are

- a) Unowned instance attributes of object instances that have a known class at the joined federate equal to the object class of the class attribute and that have the same attribute designator as the class attribute, or
- b) Instance attributes owned by the joined federate that belong to object instances that have a known class at the owning federate equal to the object class of the class attribute and that have the same attribute designator as the class attribute.

current federation object model (FOM): The union of the FOM modules and one Management Object Model (MOM) and Initialization Module (MIM) that have been specified in the creation of the federation execution or by any federate that has joined the federation execution. The sum operation is carried out according to the rules as prescribed in IEEE Std 1516.2-2025. When all FOM modules have been provided, the current FOM will be equal to the FOM; and before this step has happened, the current FOM will be a true subset of the FOM.

current federation object model (FOM) Document Data (FDD): The data and information used to configure the federation execution that is the union of the FOM modules and one Management Object Model (MOM) and Initialization Module (MIM) that have been specified in the creation of the federation execution or by any federate that has joined the federation execution. The sum operation is carried out according to the rules as prescribed in IEEE Std 1516.2-2025. When all FOM modules have been provided, the current FDD will be equal to the FDD; and before this step has happened, the current FDD will be a true subset of the FDD.

data type: A representation convention for a data element establishing its format, resolution, cardinality, and ordinality.

declared attributes: The set of class attributes of a particular object class that are listed in the federation object model (FOM) as being associated with that object class in the object class hierarchy tree.

declared directed interaction: The set of class directed interactions of a particular object class that are listed in the federation object model (FOM) as being associated with that object class in the object class hierarchy tree.

declared parameters: The set of parameters of a particular interaction class that are listed in the federation object model (FOM) as being associated with that interaction class in the interaction class hierarchy tree.

default range: A range lower bound and a range upper bound, defined in the federation object model (FOM) Document Data (FDD) and specified in terms of [0, the dimension's upper bound), for a dimension.

default region: A multidimensional region provided by the runtime infrastructure (RTI) that is composed of one range for each dimension found in the federation object model (FOM) Document Data (FDD). The bounds of each of these ranges are [0, the range's dimension's upper bound). There is no way for a federate to refer to the default region.

NOTE—See 9.1 in IEEE Std 1516.1-2025.

delete: To invoke the *Delete Object Instance* service to eliminate a particular object instance. *See also:* **remove**.

NOTE—See 6.16 in IEEE Std 1516.1-2025.

designator: A generic view of arguments referenced in service descriptions. This view improves clarity in situations when arguments (mostly identifiers) to services have different views (implementations) due to a particular programming language or application programmer's interface (API).

dimension: A named interval. The interval is defined by an ordered pair of values, the first being the dimension lower bound and the second being the dimension upper bound. A runtime infrastructure (RTI) provides a predefined interval, whose lower and upper bounds are fixed as [0, the dimension's upper bound) as specified in the federation object model (FOM) Document Data (FDD). This interval provides a single basis for communication of all dimension-related data with an RTI. All normalized intervals communicated to the RTI will be subsets of this interval.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

directed interaction: An explicit action taken by a federate directed to a specific object instance that may have some effect or impact on another federate within a federation execution that owns attributes of that object instance.

directed interaction class: An association between an interaction class and an object class where instances of the interaction class may be directed to be received by interested federate that owns attribute instances of the object class.

discover: To receive an invocation of the *Discover Object Instance* ^{†15} service for a particular object instance.

NOTE—See 6.9 of IEEE Std 1516-2025.

¹⁵All RTI-initiated services are denoted with a † (printer's dagger) after the service name.

discovered class: The class that was an object instance's candidate discovery class at a joined federate when that object instance was discovered by that joined federate. *See also:* **candidate discovery class**.

NOTE—See 5.1.2 in IEEE Std 1516.1-2025.

dynamic link compatibility (DLC): The ability to interchange the implementation of an API with another without having to recompile or relink the application software that uses the API.

epsilon time interval: The distance between any two consecutive logical times T1 and T2 (i.e. the set of logical times between T1 and T2 is empty). For some representations of local time (e.g. floating point), the distance may not be the same for all such instances of T1 and T2.

exception: Notification of any irregularity that may occur during a service invocation.

fault: An event that prevents the entire federation from executing in a High Level Architecture (HLA) compliant manner.

federate: An application that may be or is currently coupled with other software applications under a federation object model (FOM) Document Data (FDD) and a runtime infrastructure (RTI). *See also:* **federate application; joined federate**.

federate application: An application that supports the High Level Architecture (HLA) interface to a runtime infrastructure (RTI) and that is capable of joining a federation execution. A federate application may join the same federation execution multiple times or may join multiple federation executions. However, each time a federate application joins a federation execution, it is creating a new joined federate. *See also:* **joined federate**.

federation: A named set of federate applications and a common federation object model (FOM) that are used as a whole to achieve some specific objective.

federation execution: The actual operation, over time, of a set of joined federates that are interconnected by a runtime infrastructure (RTI).

federation objectives: The statement of the problem that is to be addressed by the establishment and execution of a federation.

federation object model (FOM): A specification defining the information exchanged at runtime to achieve a given set of federation objectives. This information includes object classes, object class attributes, interaction classes, interaction parameters, and other relevant information. The FOM is specified to the runtime infrastructure (RTI) using one or more FOM modules. The RTI assembles a FOM using these FOM modules and one Management Object Model (MOM) and Initialization Module (MIM), which is provided automatically by the RTI or, optionally, provided to the RTI when the federation execution is created.

federation object model (FOM) Document Data (FDD): The data and information in a FOM that are used by the *Create Federation Execution* service and successive *Join Federation Execution* service invocations to configure the federation execution.

federation object model (FOM) module: A partial FOM (containing some or all object model template (OMT) tables) that specifies a modular component of a FOM. A FOM module may contain classes not inherent to it but upon which the FOM module depends, i.e., superclasses to the modular components. These superclasses will be included in the FOM module as either complete or scaffolding definitions.

federation requirements: A statement that identifies a federation characteristic, constraint, process, or product that is unambiguous and testable and that is necessary for a federation to be acceptable for its intended use.

federation scenario: A set of initial conditions and timeline of significant events used within a federation execution to achieve federation objectives.

final time value: The point on the time axis for which no other points occurs after it.

handle: An identifier originated/created by the runtime infrastructure (RTI) that is unique to a federation execution.

inherited attribute: A class attribute of an object class that was declared in a superclass of that object class in the object class hierarchy tree defined in the federation object model (FOM).

inherited dimension:

- a) Pertaining to an object class: a dimension of an object class that was declared in a superclass of that object class in the object class hierarchy tree defined in the federation object model (FOM) or simulation object model (SOM).
- b) Pertaining to an interaction class: a dimension of an interaction class that was declared in a superclass of that interaction class in the interaction class hierarchy tree defined in the federation object model (FOM) or simulation object model (SOM).

inherited directed interaction class: A directed interaction class of an object class that was declared in a superclass of that object class in the object class hierarchy tree defined in the federation object model (FOM).

initial time value: The point on the time axis for which no other point occurs before it.

in scope: Of or pertaining to an instance attribute of an object instance for which all of the following apply:

- a) The object instance is known to the joined federate.
- b) The instance attribute is owned by another joined federate.
- c) The instance attribute's corresponding class attribute is a one of the following:
 - 1) A subscribed attribute of the known class of the object instance, or
 - 2) A subscribed attribute of the known class of the object instance with regions, and the update region set of the instance attribute at the owning joined federate overlaps the subscription region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate.

NOTE—See 6.1 of IEEE Std 1516.1-2025.

instance attribute: A named characteristic of an object instance denoted by a pair composed of the object instance designator and the attribute designator.

interaction: An explicit action taken by a federate that may have some effect or impact on another federate within a federation execution.

interaction class: A template for a set of characteristics that is common to a group of interactions. These characteristics correspond to the parameters that individual federates may associate with interactions.

interaction parameters: The information associated with an interaction that a federate potentially affected by the interaction may receive to calculate the effects of that interaction on its current state.

joined federate: A member of a federation execution, actualized by a federate application invoking the *Join Federation Execution* service as prescribed in IEEE Std 1516.1-2025. *See also:* **federate application**.

known class:

- a) An object instance's registered class if the joined federate knows about the object instance as a result of having registered it, or
- b) An object instance's discovered class if the joined federate knows about the object instance as a result of having discovered it.

known object instance: An object instance that a given joined federate has either registered or discovered and that the joined federate has not subsequently deleted (globally or locally) or been notified to remove. *See also:* **register**; **discover**; **delete**; **local delete**; **remove**.

last known time position: The last time position to which a lost joined federate was successfully granted, as seen from the remaining federation execution.

local delete: To invoke the *Local Delete Object Instance* service to inform the runtime infrastructure (RTI) that it is to treat the specified object instance as if the RTI had never notified the joined federate to discover the object instance (however, the object instance is not to be eliminated and may be rediscovered). *See also:* **delete**.

NOTE—See 6.18 of IEEE Std 1516.1-2025.

logical time: The representation of discrete points on the time axis. Logical time is used to represent a federate's time position and the timestamps of messages.

logical time interval: The representation of the distance between two points on the time axis. Logical time interval is used to represent a time-regulating federate's lookahead and by the RTI to perform arithmetic on logical time values.

lookahead: A logical time interval provided by joined federates that become time-regulating. When added to the time-regulating joined federate's time position, the resulting timestamp is used to calculate the bound on the timestamps that the time-regulating joined federate can use to send timestamp order (TSO) messages.

lost joined federate: A previously joined federate that has been disconnected from the runtime infrastructure (RTI) as a result of a fault so that the joined federate can no longer continue in the federation execution in a High Level Architecture (HLA) compliant manner.

Management Object Model (MOM): A group of predefined High Level Architecture (HLA) constructs (object and interaction classes) that provide the following:

- a) Access to federation execution operating information
- b) Insight into the operations of joined federates and the runtime infrastructure (RTI)
- c) Control of the functioning of the RTI, the federation execution, and the individual joined federates

Management Object Model (MOM) and Initialization Module (MIM): A subset of the federation object model (FOM) that contains object model template (OMT) tables that describe the High Level Architecture (HLA) MOM. The MIM also contains additional predefined HLA constructs such as object and interaction roots, data types, transportation types, and dimensions. HLA specifies a standard MIM that is incorporated into all FOM Document Data (FDD) automatically by the runtime infrastructure (RTI). The standard MIM can be replaced with a user-supplied MIM containing the standard MIM plus extensions.

message: A change of object instance attribute value(s), an interaction, a directed interaction, or a deletion of an existing object instance, often associated with a particular point on the time axis, as denoted by the associated timestamp.

null designator: A designator reserved to indicate an empty value that is distinguishable from all other designators. A null designator is guaranteed to compare unequal to any other designator with a nonempty value.

object class: A fundamental element of a conceptual representation for a federate that reflects the real world at levels of abstraction and resolution appropriate for federate interoperability. An object class is a template for a set of characteristics that is common to a group of object instances. These characteristics correspond to the class attributes and class directed interactions that individual federates may publish and to which other federates may subscribe.

object instance: A unique instantiation of an object class that is independent of all other instances of that object class. At any point during a federation execution, the state of a High Level Architecture (HLA) object instance is defined as the collection of the values of all its instance attributes.

object model: A system specification defined primarily by class characteristics and relationships. The High Level Architecture (HLA) idea of an object model is similar in many ways, but not identical, to the common idea of an object model in object-oriented literature.

object model framework: The rules and terminology used to describe High Level Architecture (HLA) object models.

optimistic logical time (OLT): A supplementary logical time that facilitates optimistic time advancement. The optimistic logical time is provided, along with the update to the federate's logical time, in a *Flush Queue Grant* service call as a response to a *Flush Queue Request* service call. It is the minimum of the time supplied in the *Flush Queue Request* and the timestamps of any TSO messages delivered in response to that *Flush Queue Request*. The optimistic logical time indicates a lower bound on the next logical time the federate could be advanced to if no TSO messages with a timestamp less than it are received in the future.

optimistic lookahead (OL): The lookahead the federate would have if actually granted to the optimistic logical time.

order type: A runtime infrastructure (RTI) means of ordering messages originating from multiple joined federates that are delivered to a single joined federate. Different categories of service are defined with different characteristics regarding whether and how an RTI orders messages that are to be delivered to a joined federate.

out of scope: Of or pertaining to an instance attribute of an object instance for which one or more of the following are not true:

- a) The object instance is known to the joined federate.
- b) The instance attribute is owned by another joined federate.
- c) The instance attribute's corresponding class attribute is one of the following:
 - 1) A subscribed attribute of the known class of the object instance, or
 - 2) A subscribed attribute of the known class of the object instance with regions, and the update region set of the instance attribute at the owning joined federate overlaps the subscription region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate.

NOTE—See 6.1 of IEEE Std 1516.1-2025.

overlap:

- a) Pertaining to region sets: given two region sets, to have a region in each set such that the two regions overlap.
- b) Pertaining to regions: given two regions, to have at least one dimension in common if and only if, for each dimension the regions have in common, their respective ranges in the common dimension overlap. If two regions do not have any dimensions in common, the regions do not overlap.

- c) Pertaining to ranges: given two ranges ($A = [a_{\text{lower}}, a_{\text{upper}})$ and $B = [b_{\text{lower}}, b_{\text{upper}})$), to overlap if and only if
- 1) Either $a_{\text{lower}} = b_{\text{lower}}$ or ($a_{\text{lower}} < b_{\text{upper}}$ and $b_{\text{lower}} < a_{\text{upper}}$) or
 - 2) The Allow Relaxed DDM switch is enabled and the RTI implementation makes the determination, based on implementation-specific criteria, that the ranges are sufficiently close to warrant an expanded interpretation of one or both ranges; neither range may ever be reduced.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

owned: Pertaining to the relationship between an instance attribute and a joined federate: when the joined federate has the unique right to update the instance attribute's value.

owned instance attribute: An instance attribute that is explicitly modeled by the owning joined federate. A joined federate that owns an instance attribute has the unique responsibility to provide values for that instance attribute to the federation, through the runtime infrastructure (RTI), as documented in the federation object model (FOM) Document Data (FDD).

pair: A grouping of two related elements (a first component and a second component), the combination of which is treated as an entity. An example of a pair would be an instance attribute grouped with its current value.

parameter: A named characteristic of an interaction.

passel: A group of attribute handle/value pairs from an *Update Attribute Values* service invocation that are delivered together via a *Reflect Attribute Values* † service invocation. All pairs within the passel have the same user-supplied tag, sent message order type, transportation type, receive message order type, timestamp (if present), and set of sent region designators (if present). A passel is a message.

passive subscription: A request to the runtime infrastructure (RTI) for the kinds of data (object classes and attributes as well as interactions) that the joined federate is currently interested in receiving. However, unlike an active subscription, this information is not used by the RTI to arrange for data to be delivered, nor is it used to tell publishing joined federates that another joined federate is subscribing to that data (by way of *Start/Stop Registration*, *Turn On/Off Updates*, or *Turn On/Off Interactions* service invocations). This form of subscription is provided to support certain types of logger joined federates.

promoted: Pertaining to an object instance, as known by a particular joined federate: having a discovered class that is a superclass of its registered class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2025.

publish: To announce to a federation the information a federate may provide to the federation.

published:

- a) Pertaining to an object class from the perspective of a given joined federate: having at least one available attribute of the object class that was an argument, along with the object class, to a *Publish Object Class Attributes* service invocation that was not subsequently unpublished via the *Unpublish Object Class Attributes* service.

NOTE—See 5.1.2 of IEEE Std 1516.1-2025.

- b) Pertaining to an interaction class from the perspective of a given joined federate: being an argument to a *Publish Interaction Class* service invocation that was not subsequently followed by an *Unpublish Interaction Class* service invocation for that interaction class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2025.

published attributes of an object class: The class attributes that have been arguments to *Publish Object Class Attributes* service invocations by a given joined federate for that object class that have not subsequently been unpublished (either individually or by unpublishing the whole object class), and possibly the **HLAprivilegeToDeleteObject** attribute for that object class.

NOTE—See 5.1.2 of IEEE Std 1516.1-2025.

range: A subset of a dimension, defined by an ordered pair of values, the first being the range lower bound and the second being the range upper bound. This pair of values defines a semi-open interval [range lower bound, range upper bound) (i.e., the range lower bound is the smallest member of the interval, and the range upper bound is just greater than any member of the interval).

NOTE—See 9.1.1 of IEEE Std 1516.1-2025.

range lower bound: The first component of the ordered pair of values that is part of a range.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

range upper bound: The second component of the ordered pair of values that is part of a range.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

received class: The class that was an interaction's candidate received class at the joined federate when that interaction was received at that joined federate via an invocation of the *Receive Interaction* † or *Receive Directed Interaction* † service.

NOTE—See 5.1.3 of IEEE Std 1516.1-2025.

received parameters: The set of parameters received when the *Receive Interaction* † service is invoked. These parameters consist of the subset of the sent parameters of an interaction that are available parameters of the interaction's received class.

NOTE—See 5.1.3 of IEEE Std 1516.1-2025.

receive order (RO): A characteristic of no ordering guarantee for messages. Messages that are received as RO messages will be received in an arbitrary order by the respective joined federate. A timestamp value will be provided with the message if one was specified when the message was sent, but that timestamp has no bearing on message receipt order.

reflect: To receive new values for one or more instance attributes via invocation of the *Reflect Attribute Values* † service.

NOTE—See 6.11 of IEEE Std 1516.1-2025.

reflected instance attribute: An instance attribute that is represented but not explicitly modeled in a joined federate. The reflecting joined federate accepts new values of the reflected instance attribute as they are produced by some other federation member and provided to it by the runtime infrastructure (RTI), via the *Reflect Attribute Values* † service.

region: A generic term that refers to either a region specification or a region realization. If not using data distribution management (DDM), region arguments may be ignored.

NOTE— See 9.1 of IEEE Std 1516.1-2025.

region realization: A region specification (set of ranges) that is associated with an instance attribute for update, with a sent interaction, or with a class attribute or interaction class for subscription. Region realizations are created from region specifications via the *Commit Region Modifications* (only when modifying a region specification from which region realizations are already derived), *Register Object Instance With Regions*, *Associate Regions for Updates*, *Subscribe Object Class Attributes With Regions*, *Subscribe Interaction Class With Regions*, *Send Interaction With Regions*, or *Request Attribute Value Update With Regions* services.

NOTE—See 9.1.1 of IEEE Std 1516.1-2025.

region specification: A set of ranges. Region specifications are created using the *Create Region* service, and a runtime infrastructure (RTI) is notified of changes to a region specification using the *Commit Region Modifications* service.

NOTE—See 9.1.1 of IEEE Std 1516.1-2025.

region template: A region template is an incomplete region specification where one or more dimensions have not been assigned ranges.

register: To invoke the *Register Object Instance* or the *Register Object Instance With Regions* service to create a unique object instance designator.

NOTE—See 6.8 of IEEE Std 1516.1-2025.

registered class: The object class that was an argument to the *Register Object Instance* or the *Register Object Instance With Regions* service invocation that resulted in the creation of the object instance designator for a given object instance.

regular interaction class description: A description of an interaction class that follows IEEE Std 1516.2-2025 and contains at a minimum the interaction class name and publish/subscribe indicator.

regular object class description: A description of an object class that follows IEEE Std 1516.2-2025 and contains at a minimum the object class name and publish/subscribe indicator.

remove: To receive an invocation of the *Remove Object Instance* $\dagger$ service for a particular object instance.

NOTE—See 6.15 of IEEE Std 1516.1-2025.

repeated description: A description of a construct (e.g., object class, interaction class, data type, dimension) in a federation object model (FOM) module with a name and additional properties that are identical to a description that is already available in the current FOM.

resolution: The smallest resolvable value separating attribute or parameter values that can be discriminated. Resolution may vary with magnitude for certain data types.

retraction: An action performed by a federate to unschedule a previously scheduled message. Message retraction may be visible to the federate to whom the scheduled message was to be delivered. Retraction is widely used in classic event-oriented discrete event simulations to model behaviors such as preemption and interrupts.

runtime infrastructure (RTI): The software that provides common interface services during a High Level Architecture (HLA) federation execution for synchronization and data exchange.

runtime infrastructure (RTI) initialization data (RID): RTI vendor-specific information needed to run an RTI. If required, a RID is supplied when an RTI is initialized.

scaffolding interaction class description: A description of an interaction class in a federation object model (FOM) module or simulation object model (SOM) module that follows IEEE Std 1516.2-2025. However, it will contain only the name. The name of the scaffolding interaction class description is identical to the name of a regular interaction class description provided by another FOM/SOM module. Scaffolding interaction class descriptions act as placeholders in order to represent the class hierarchy from the other FOM/SOM module(s) from which a new regular interaction class description can be specified.

scaffolding object class description: A description of an object class in a federation object model (FOM) module or simulation object model (SOM) module that follows IEEE Std 1516.2-2025. However, it will contain only the name. The name of the scaffolding object class description is identical to the name of a regular object class description provided by another FOM/SOM module. Scaffolding object class descriptions act as placeholders in order to represent the class hierarchy from the other FOM/SOM module(s) from which a new regular object class description can be specified.

sent class: The interaction class that was an argument to the *Send Interaction*, *Send Interaction With Regions*, or *Send Directed Interaction* service invocation that initiated the sending of a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2025.

sent interaction: A specific interaction that is transmitted by a joined federate via the *Send Interaction*, *Send Interaction With Regions*, or *Send Directed Interaction* service and received by other joined federates in the federation execution via the *Receive Interaction* † or *Receive Directed Interaction* † service.

sent parameters: The parameters that were arguments to the *Send Interaction*, *Send Interaction With Regions*, or *Send Directed Interaction* service invocation for a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2025.

set: A group of elements in which each element occurs at most once (this definition corresponds to the mathematical notion of sets). An example of a set would be a group of class attributes, each of which belongs to the same object class.

simulation object model (SOM): A specification of the types of information that an individual federate could provide to High Level Architecture (HLA) federations as well as the information that an individual federate can receive from other federates in HLA federations. The SOM is specified using one or more SOM modules. The standard format in which SOMs are expressed facilitates determination of the suitability of federates for participation in a federation.

simulation object model (SOM) module: A subset of the SOM that contains some or all object model template (OMT) tables. SOM modules and one optional Management Object Model (MOM) and Initialization Module (MIM) are used to specify the SOM of a federate. A SOM module contains complete or scaffolding definitions for all object classes and interaction classes that are superclasses of object classes and interaction classes in the same SOM module.

specified dimensions: The dimensions that are explicitly provided when the region specification is created or modified.

NOTE—See 9.1.2 of IEEE Std 1516.1-2025.

stop publishing: To take action that results in a class attribute that had been a published attribute of a class no longer being a published attribute of that class.

subclass: A class that adds additional detail to (specializes) another more generic class (superclass). A subclass, by inheriting the properties from its parent class (closest superclass), also inherits the properties of all superclasses of its parent as well.

subscribe: To announce to a federation the information a federate wants from the federation.

subscribed:

- a) Pertaining to an object class from the perspective of a given joined federate: having subscribed attributes of that class or subscribed attributes of that class with regions, for some region. *See also:* **subscribed attributes of an object class; subscribed attributes of an object class with regions.**
- b) Pertaining to an interaction class: being a subscribed interaction class or a subscribed interaction class with regions, for some region. *See also:* **subscribed interaction class; subscribed interaction class with regions.**

subscribed attributes of an object class: The class attributes that have been arguments to *Subscribe Object Class Attributes* service invocations by a given joined federate for a given object class that have not subsequently been unsubscribed, either individually or by unsubscribing the whole object class.

NOTE—See 5.1.2 and 5.8 of IEEE Std 1516.1-2025.

subscribed attributes of an object class with regions: The class attributes that have been arguments to *Subscribe Object Class Attributes With Regions* service invocations by a given joined federate for a given object class and a given region, assuming the joined federate did not subsequently invoke the *Unsubscribe Object Class Attributes With Regions* service for that object class and region.

NOTE—See 9.8 of IEEE Std 1516.1-2025.

subscribed interaction class: An interaction class that, from the perspective of a given joined federate, was an argument to a *Subscribe Interaction Class* or *Subscribe Interaction Class With Regions* service invocation that was not subsequently followed by an *Unsubscribe Interaction Class* or *Unsubscribe Interaction Class With Regions* service invocation for that interaction class.

NOTE—See 5.1.3 and 5.10 of IEEE Std 1516.1-2025.

subscribed interaction class with regions: An interaction class and a region that, from the perspective of a given joined federate, were arguments to a *Subscribe Interaction Class With Regions* service invocation that was not subsequently followed by an *Unsubscribe Interaction Class With Regions* service invocation for that interaction class and that region.

NOTE—See 9.10 of IEEE Std 1516.1-2025.

subscription region set: A set of regions used for subscription of a class attribute or used for subscription of an interaction class. *See also:* **used for subscription of a class attribute; used for subscription of an interaction class.**

superclass: A class that generalizes a set of properties that may be inherited by more refined (i.e., detailed) versions of the class. In High Level Architecture (HLA) applications, a class may have at most one immediate superclass (i.e., can inherit only from a single class at the next higher level of the class hierarchy).

synchronization point: A logical point in the sequence of a federation execution that all joined federates forming a synchronization set for that point attempt to reach and by which, if they are successful, they synchronize their respective processing.

target object instance: An object instance for which a directed interaction is intended.

time advancing state: The interval between a joined federate's request to advance its time position and the corresponding grant by the runtime infrastructure (RTI). A joined federate may advance its time position only by requesting a time advancement from the RTI via one of the following services:

- a) *Time Advance Request*
- b) *Time Advance Request Available*
- c) *Next Message Request*
- d) *Next Message Request Available*
- e) *Flush Queue Request*

The joined federate's time position will not actually be advanced until the RTI responds with a *Time Advance Grant* or *Flush Queue Grant* service invocation at that joined federate.

time axis: A totally ordered sequence of discrete values in which each value typically represents an HLA instant of time in the physical system being modeled. For any two points, T1 and T2, on the time axis, if $T1 < T2$, T1 represents an instant of time that occurs before the instant represented by T2. The sequence begins with an initial value, for which no point is less, and ends with a final value, for which no point is greater.

time-constrained federate: A joined federate that may receive timestamp order (TSO) messages and whose time advances are constrained by other joined federates within a federation execution.

NOTE—See 8.1 of IEEE Std 1516.1-2025

time management: A collection of High Level Architecture (HLA) services that support controlled message ordering and delivery to the cooperating joined federates within a federation execution in a way that is consistent with federation requirements.

time position: A joined federate's association with a point on the time axis.

time-regulating federate: A joined federate that may send timestamp order (TSO) messages and that constrains the time advances of other joined federates within a federation execution.

NOTE—See 8.1 of IEEE Std 1516.1-2025.

timestamp (of message or save): The association of a logical time with an activity of a federate (e.g., updating instance attribute values, sending interactions and federation save).

timestamp order (TSO): An ordering of messages provided by a runtime infrastructure (RTI) for joined federates making use of time management services and messages containing timestamps. Messages having different timestamps are said to be delivered in TSO if for any two messages, M1 and M2 (timestamped with T1 and T2, respectively), that are delivered to a single joined federate where $T1 < T2$, then M1 is delivered before M2. Messages having the same timestamp will be delivered in an arbitrary order (i.e., no tie-breaking mechanism is provided by an RTI).

token: An attribute without a value, which conveys the information of which federate, if any, owns the attribute.

transportation type: A runtime infrastructure (RTI) provided means of transmitting messages between joined federates. Different categories of service are defined with different characteristics such as reliability of delivery and message latency.

unspecified dimensions: The available dimensions of a class attribute, instance attribute, interaction class, or sent interaction less the specified dimensions of the region specification from which the region realization is derived.

NOTE—See 9.1.3 of IEEE Std 1516.1-2025.

update: To invoke the *Update Attribute Values* service for one or more instance attributes.

NOTE—See 6.10 of IEEE Std 1516.1-2025.

update rate: The rate at which instance attribute values are provided, either by an owning joined federate to the runtime infrastructure (RTI) or by the RTI to a subscribing joined federate.

update region set: A set of regions used for sending interactions or used for updating instance attributes. *See also:* **used for sending**; **used for update**.

used for sending:

- a) Pertaining to a region and the specified interaction class designator: being an argument in the *Send Interaction With Regions* service.
- b) Pertaining to the default region: when the specified interaction class designator is an argument in the *Send Interaction* service.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

used for subscription of a class attribute:

- a) Pertaining to a region, an object class, and a class attribute: when the class attribute is a subscribed attribute of the object class with that region. *See also:* **subscribed attributes of an object class with regions**.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

- b) Pertaining to the default region: when the specified class attribute is a subscribed attribute of the specified class. *See also:* **subscribed attributes of an object class with regions**.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

used for subscription of an interaction class:

- a) Pertaining to a region and an interaction class: when the interaction class is a subscribed interaction class with regions. *See also:* **subscribed interaction class**.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

- b) Pertaining to the default region: when the specified interaction class is a subscribed interaction class. *See also:* **subscribed interaction class**.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

used for update:

- a) Pertaining to a region that, along with the specified object instance and attribute designators, has been used as an argument in either the *Register Object Instance With Regions* service or the *Associate Regions For Updates* service: when the region has not subsequently been used along with the specified object instance designator as an argument in the *Unassociate Regions For Updates* service, nor has the joined federate subsequently lost ownership of the specified instance attribute(s).

NOTE—See 9.1 of IEEE Std 1516.1-2025.

- b) Pertaining to the default region: when the specified instance attribute(s) is not currently used for update with any other region.

NOTE—See 9.1 of IEEE Std 1516.1-2025.

valid federate designator: A federate designator which, during the federation execution, was returned by the *Join Federation Execution* service to some joined federate. The federate does not have to remain a joined federate for its federate designator to remain a valid federate designator.

wall-clock time: Time as determined by a chronometer such as a wristwatch or wall clock.

zero time interval: The distance between two logical time values that are equal. This time interval, when added to a logical time T_1 , results in a logical time $T_2 == T_1$.

3.2 Abbreviations and acronyms

The following abbreviations and acronyms pertain to this standard, IEEE Std 1516-2025, and IEEE Std 1516.1-2025:

ABI	application binary interface
ADT	abstract data type
API	application programmer's interface
BNF	Backus Naur Form
DDM	data distribution management
DIF	data interchange format
DLC	dynamic link compatibility
DM	declaration management
DSEEP	Distributed Simulation Engineering and Execution Process
DTD	document type declaration
FDD	FOM Document Data
FOM	federation object model
GALT	Greatest Available Logical Time
HLA	High Level Architecture
JAR	Java Archive
LIS	language-independent specification
LITS	Least Incoming Timestamp
LRC	Local Runtime Component
LT	Logical Time
LTI	Logical Time Interval
MIM	MOM and Initialization Module
MOM	Management Object Model
M&S	modeling and simulation
NA	not applicable
OL	Optimistic Lookahead
OLT	Optimistic Logical Time

OMT	object model template
OO	object oriented
OOAD	object-oriented analysis and design
POC	point of contact
RID	RTI initialization data
RO	receive order
RTI	runtime infrastructure
SOM	simulation object model
TRADT	time representation abstract data type
TSO	timestamp order
URI	uniform resource identifier
URL	uniform resource locator
WSPRC	Web Service Provider RTI Component
XML	extensible markup language

3.3 Conventions

Conventions in this standard specify how entries are to be formulated when completing an HLA object model. The following conventions pertain to this standard.

3.3.1 Names

Names in object models shall adhere to eXtensible Markup Language (XML) naming conventions. XML conventions state that names cannot begin with a number or punctuation character, and require that names be constructed from a combination of letters, digits, hyphens, colons, full stops (periods), and underscores with no spaces or other breaking characters (e.g., tabs, carriage returns). In HLA object models, allowable names shall be further restricted as follows:

- a) The period (full stop) is reserved for qualifying class names and shall not be included in a user-defined name within an object model. When fully qualified, the class name includes all predecessor class names (separated by periods) beginning at the root and ending at the class name in question.
- b) As recommended in the XML specification, the colon character shall not be used.
- c) Names beginning with the string “hla,” or any string that would match (('H'|'h') ('L'|'l') ('A'|'a')), are reserved and shall not be included in user-defined names.
- d) The case of all textual data within HLA object models (including names) shall be significant; all textual data defined in an object model document shall be case-sensitive.
- e) A name consisting of the string “na,” or any string that would match (('N'|'n') ('A'|'a')), is reserved to indicate that a name is not applicable in this circumstance and shall not be included as a user-defined name.

These rules apply to the following names in HLA object models:

- Object class names
- Interaction class names
- Attribute names
- Parameter names
- Data type names

- Enumerated data type enumerators
- Fixed record field names
- Variant record alternative names
- Basic data representation names
- Dimension names
- Transportation type names
- Synchronization point labels
- Note identifying labels

4. HLA OMT components

HLA object models are composed of a group of interrelated components specifying information about classes of objects and their attributes and interactions and their parameters. The information content of these components can be represented in many different ways or presentations. A presentation is the formatting of the information contained in the object model in a particular manner for a particular purpose. For example, the OMT tabular format is designed for presentation on a printed page, whereas the OMT data interchange format (DIF) is a presentation designed for passing an object model between tools. All HLA object models shall be capable of being presented in both the OMT tabular format and the OMT DIF (the primary and authoritative format). This standard defines the OMT content and presents it in OMT tabular format throughout.

The OMT consists of the following components:

- *Object model identification table*: To associate important identifying information with the HLA object model
- *Object class structure table*: To record the namespace of all federate or federation object classes and to describe their class-subclass relationships
- *Interaction class structure table*: To record the namespace of all federate or federation interaction classes and to describe their class-subclass relationships
- *Attribute table*: To specify features of object attributes in a federate or federation
- *Directed interaction class table*: To specify features of directed interactions for objects in a federate or federation
- *Parameter table*: To specify features of interaction parameters in a federate or federation
- *Dimension table*: To specify dimensions for filtering instance attributes and interactions
- *Time representation table*: To specify the representation of time values
- *User-supplied tag table*: To specify the representation of tags used in HLA services
- *Synchronization table*: To specify representation and data types used in HLA synchronization services
- *Transportation type table*: To describe the transportation mechanisms used
- *Update rate table*: To specify update rate information
- *Switches table*: To specify initial settings for parameters used by the RTI
- *Data type tables*: To specify details of data representation in the object model
- *Notes table*: To expand explanations of any OMT table item
- *Interface specification services usage table*: To specify the HLA services used by a federate or in a federation
- *FOM/SOM lexicon*: To define all of the objects, attributes, interactions, and parameters used in the HLA object model

All of the OMT components shall be completed when specifying an HLA object model for both federations and individual federates. However, certain tables may be empty or devoid of domain-specific content. For instance, although federations typically support interactions among their federates, some federates (such as a stealth viewer) might not be involved in interactions. In this situation, the interaction class structure table would contain only the single interaction class required by the HLA and the parameter table would be empty in that federate's SOM or supporting SOM module. It is also expected that federates commonly have objects with attributes of interest across the federation; in such cases, these objects and attributes shall be documented. However, a federate or an entire federation may exchange information solely via interactions; in which case, its object class structure table and attribute table would contain only HLA-required data. The specific rules for the applicability of each OMT table are provided in the table descriptions.

The final HLA OMT component, the FOM/SOM lexicon, is essential to help users understand the semantics of the terms used in an HLA object model.

The HLA MOM specifies a designated set of information elements that are associated with federation executions. Implementation of the MOM information elements as specified in IEEE Std 1516.1-2025 provides a mechanism for management of federation executions using existing HLA services. Inclusion of the MOM is required for all FOMs and is achieved by incorporating the MOM data using a MOM and MIM. In addition, federates that have an inherent capability to interact with or extend the MOM shall include the MOM in their SOM. Inclusion of the MOM means the inclusion of a MIM that contains all tables in Clause 11 of IEEE Std 1516.1-2025, along with any applicable MOM extensions.

The last row in any OMT table is labeled "Note." This allows the object model developer to associate one or more notes with the entire table. If no note is being provided for that table, a value of "NA" should be entered. Although the "Note" row is not explicitly identified or discussed in the table descriptions, it is shown in the table examples for illustration purposes. The notes feature is fully explained in 4.15.

A FOM or SOM is specified using one or more FOM or SOM modules. For a FOM, a MIM is also required whereas for a SOM, inclusion of a MIM depends on federate capabilities with respect to the MOM. Any FOM, SOM, FOM module, or SOM module that fully conforms to all of the rules and constraints stated in this specification is a *compliant object model*. A FOM or SOM module is compliant if it conforms to all rules and constraints for a compliant object model as well as all rules and constraints that apply to FOM or SOM modules. A MIM is compliant if it conforms to all rules and constraints of a compliant object model and contains at least the information provided in the standard MIM (see IEEE Std 1516.1-2025).

The basics of each OMT component are presented in the following separate subclauses. The template format for each component is provided and described, and criteria are suggested to help guide decisions on when to include specific federate or federation features within each of these components for a specific HLA object model. Examples of the usage of each OMT component are also provided; these examples are for illustrative purposes only, and they are not meant to imply any additional requirements. Clause 7 specifies the rules and principles for merging FOM modules and SOM modules. Annex A specifies the notations used in describing OMT tables.

4.1 Object model identification table

4.1.1 Purpose/background

A key design goal for all HLA object models is to facilitate reuse. HLA object models provide information that enables inferences to be drawn regarding the reuse potential of individual federates for new applications. Reuse can also occur at the level of the object model. An existing FOM or FOM module may provide a foundation for the development of new FOMs, or SOMs or SOM modules may be merged to form new FOMs. In either case, to expedite reuse, it is important to include a minimum but sufficient degree of descriptive information in the object model. For instance, when federation developers wish to pose detailed questions about how a federate or federation was constructed, point of contact (POC) information within an HLA object model is extremely important. The purpose of this table is to document certain key identifying information within the object model description.

4.1.2 Table format

The information that identifies HLA object models shall be provided in a simple two-column table. The first column (Category) specifies the categories of data that shall be provided in this table. The second column (Information) shall specify the required information. Entries shall be provided for all rows. “NA” shall be entered in any row for which no information is appropriate. Definitions of the categories, descriptions, and occurrences are shown in Table 1.

The concept of FOM modules and SOM modules is supported via the “References” field of this table. “Dependent” modules in this context are those that contain references to data in other module(s), such as object class definitions or data type definitions. “Standalone” modules do not contain such references. Table 2 provides the structure to be used to describe the Table 1 metadata elements.

Table 1—OMT metadata descriptions

Category	Description	Occurs
<i>Name</i>	This field shall specify the name assigned to the object model.	1
<i>Type</i>	<i>Type</i> : This field shall specify the type that the object model represents; valid values are as follows: — <i>FOM</i> : the object model describes a federation — <i>SOM</i> : the object model describes a federate	1
<i>Version</i>	This field shall specify the version identification assigned to the object model.	1
<i>Modification date</i>	This field shall specify the latest date on which this version of the object model was created or modified. The modification date shall be specified in the format “YYYY-MM-DD” (e.g., 1999-04-15).	1
<i>Security classification</i>	This field shall specify the security classification of the object model. The following values may be used for this field, although other values are also valid: — Unclassified — Confidential — Secret — Top Secret	1
<i>Copyright</i>	This field shall provide a copyright notice for the object model.	0..1
<i>Release Restriction</i>	This field shall contain any restrictions on the release of the object model to specific organizations or individuals.	0..many
<i>Purpose</i>	This field shall specify the purpose for which the object model was developed.	0..1
<i>Application domain</i>	This field shall specify the type or class of application to which the object model applies. The following values may be used for this field, although other values are also valid: — Analysis — Training — Test and Evaluation — Engineering — Acquisition	0..1
<i>Description</i>	This field shall provide an account of the content of the object model.	1
<i>Use limitation</i>	This field shall identify any known constraints on the application of this object model.	0..1
<i>Use history</i>	This field shall identify any past applications of the object model.	0..many
<i>Keyword</i>	The two subfields of this field shall collectively identify the keywords that are associated with the object model.	0..many
<i>Taxonomy</i>	This field shall specify the source of the keyword vocabulary.	0..1
<i>Keyword value</i>	This field shall specify the word or concept that is addressed by the object model.	1
<i>POC</i>	The subfields of this field shall collectively identify the necessary point of contact (POC) information for this object model.	1..many

Table 1—OMT metadata descriptions (continued)

Category	Description	Occurs
<i>POC type</i>	This subfield shall specify the role that the POC has with respect to the object model. The following values may be used for this field, although other values are also valid: — Primary author — Contributor — Proponent — Sponsor — Release authority — Technical POC	1
<i>POC name</i>	This subfield shall specify the name of the object model POC, including an honorific (e.g., Dr., Ms) or rank, first name, and last name where appropriate.	0..1
<i>POC organization</i>	This subfield shall specify the organization with which the POC is affiliated.	0..1
<i>POC telephone</i>	This subfield shall specify the telephone number for the POC, including the international telephone code for the POC's country.	0..many
<i>POC e-mail</i>	This subfield shall specify the e-mail address of the POC.	0..many
<i>References</i>	This field shall specify pointers to additional sources of information. If the object model is intended to represent a FOM module or SOM module, then this field shall characterize the module as “standalone” or as having a “dependency” upon another module. If the object model represents a composition of existing modules, then this field shall identify references to each FOM module or SOM module component.	0..many
<i>Type</i>	This subfield shall specify the form of the reference material. For FOM or SOM modules, the following predefined values may be used for this field: — Standalone — Dependency — Composed From Otherwise, the general type of reference source should be indicated, such as “Text Document,” “Spreadsheet,” or “Powerpoint File.”	1
<i>Identification</i>	The following conditions shall be met: — If the Reference Type subfield = “Standalone,” this subfield shall contain the value “NA”. — If the Reference Type subfield = “Dependency,” this subfield shall identify the name of all dependent FOM modules or SOM modules. — If the Reference Type subfield = “Composed From,” this subfield shall identify the names of all FOM modules or SOM modules that have been merged to form the current object model. If the Reference type subfield is not equal to any of the three predefined values, this subfield shall specify the location of the reference source [e.g., uniform resource identifier (URI), ISBN].	1
<i>Other</i>	This field shall specify other data deemed relevant by the author of the object model.	0..1
<i>Glyph</i>	This field specifies the image that can be used to visually represent a FOM or SOM. The image shall be specified using a URI. For images that are embedded in the FOM file an RFC-2397 Data URI shall be used. The image data shall be represented using one of the following formats: image/jpeg, image/gif, image/png, image/svg+xml.	0..1
<i>Alt</i>	This subfield provides alternative text if the image represented in the <i>Glyph</i> field cannot be displayed.	0..1

Table 2—Object model identification table

Category	Information
Name	<name>
Type	<type>
Version	<version>
Name	<date>
Security Classification	<security classification>
Copyright	[<copyright>]
Release Restriction	[<release restriction>]
Purpose	[<purpose>]
Application Domain	[<application domain>]
Description	<description>
Use Limitation	[<limitation>]
Use History	[<history>]
Keyword	
Taxonomy	[<taxonomy>]
Keyword Value	[<keyword value>]
POC	
POC Type	<poc type>
POC Name	[<poc name>]
POC Organization	[<poc organization>]
POC Telephone	<poc telephone>
POC E-mail	<poc email>
References	
Type	[<ref type>]
Identification	[<identification>]
Other	<other>
Glyph	[<glyph>]
Alt	[<alt>]

4.1.3 Inclusion criteria

The categories of information specified in this table shall be included for all HLA FOMs, FOM modules, SOMs, and SOM modules.

4.1.4 Example

Table 3 shows a simple example of the object model identification table. In this example, the *Purpose* entry is shorter (by design) than entries that would be found in this table for most practical applications. The subclauses that follow provide examples of each OMT table for the notional SOM identified in this table.

Table 3—Object model identification table example

Category	Information
Name	RestaurantExample
Type	SOM
Version	2.0
Modification Date	20022024-1102-1804
Security Classification	Unclassified
Copyright	©2022 2024 IEEE
Release Restriction	NA
Purpose	To define an HLA object model for a restaurant federate
Application Domain	Restaurant operations
Description	This object model is intended to define the external HLA interface of the “Joe’s Place” restaurant simulation.
Use Limitation	NA
Use History	Federated Foods federation
Keyword	
Taxonomy	Food Service Industry Taxonomy
Keyword	Restaurant, Menu, Server
POC	
POC Type	Sponsor
POC Name	Mr. Joseph Doe
POC Organization	Joe’s Place
POC Telephone	1-977-555-1234
POC Email	doej@joesplace.com
References	
Type	HTML document
Identification	www.fedfoods.com/restsim.html
Other	See Mobil International Restaurant Guide for more information
Glyph	
Alt	Restaurant
Note	NA