IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules

IEEE Computer Society

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

1516™

IEEE
3 Park Avenue
New York, NY 10016-5997, USA

18 August 2010

IEEE Std 1516™-2010
(Revision of
IEEE Std 1516-2000)

IEEE Std 1516™-2010
(Revision of
IEEE Std 1516-2000)

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules

Sponsor

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

Approved 25 March 2010

IEEE SA-Standards Board

Abstract: This standard, describing the framework and rules of the High Level Architecture (HLA), is the capstone document for a family of related HLA standards. It defines the HLA, its components, and the rules that outline the responsibilities of HLA federates and federations to ensure a consistent implementation. Simulations are abstractions of the real world, and no one simulation can solve all of the functional needs for the modeling and simulation community. It is anticipated that technology advances will allow for new and different modeling and simulation (M&S) implementations within the framework of the HLA. The standards contained in this architecture are interrelated and need to be considred as a product set, as a change in one is likely to have an impact on the others. As such, the HLA is an integrated approach that has been developed to provide a common architecture for simulation.
Keywords: architecture, class attribute, federate, federation, federation execution, federation object model, framework, High Level Architecture, instance attribute, interaction class, joined federate, object class, object model template, rules, runtime infrastructure, simulation object model

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

PDF: ISBN 978-0-7381-6251-5 STD96061
Print: ISBN 978-0-7381-6252-2 STDPD96061

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

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

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

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

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon his or her 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.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments. Recommendations to change the status of a stabilized standard should include a rationale as to why a revision or withdrawal is required. Comments and recommendations on standards, and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

Introduction

This introduction is not part of IEEE Std 1516-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules.

This standard is the capstone document for a family of related standards that together describe a unified approach to constructing interoperable simulation systems.

The High Level Architecture (HLA) provides a general framework within which simulation developers can structure and describe their simulation applications. Flexibility is the aim of the HLA. In particular, the HLA addresses two key issues: promoting interoperability between simulations and aiding the reuse of models in different contexts. Three main components are described within the set of products forming the HLA. The first component, the HLA Framework and Rules Specification (i.e., this standard), provides a set of ten rules that together ensure the proper interaction of federates in a federation and define the responsibilities of federates and federations. The second component, the object model template (OMT), is a necessary basis for reuse and forms a documentation standard describing the data used by a particular model. The third component, the federate interface specification, addresses interoperability and describes a generic communications interface that allows simulation models to be connected and coordinated. Although the HLA is an architecture, not software, use of runtime infrastructure (RTI) software is required to support operations of a federation execution. The RTI software provides a set of services, as defined by the federate interface specification, used by federates to coordinate operations and data exchange during a runtime execution.

Simulations are necessarily abstractions of the real world, and no one simulation design can meet the functional needs of the entire modeling and simulation community. However, in defining an overriding architecture, generic issues can be addressed. When doing so, it is essential that such an architecture encompass both differing computing environments and differing classes of simulations.

This standard, describing the framework and rules, is intended to provide some of the general philosophy behind the HLA, including guidance about how to design, use, and adhere to the HLA vision.

Notice to users

Laws and regulations

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

Copyrights

Updating of IEEE documents

Users of IEEE standards 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. 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 the IEEE Standards Association web site at http://ieeexplore.ieee.org/xpl/standards.jsp, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA web site at http://standards.ieee.org.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: http://standards.ieee.org/reading/ieee/updates/errata/index.html. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: http://standards.ieee.org/reading/ieee/interp/index.html.

Patents

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE 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.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the HLA Evolved Working Group had the following membership:

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

Martin Adelantado
Bill Andrews
Fredrik Antelius
Joanne Atherton
Trevor Bakker
Shelby Barrett
Tolga Basturk
William Beavin
Emmet Beeker
Alan Berry
Mark Biwer
David Bodoh
Jake Borah
Steve Boswell
Derrick Briscoe
Dominique Canazzi
Andy Ceranowicz
Tram Chase
Scott Clarke
David Coppler
Anthony Cramp
Dannie Cutts
Timothy Daigle
Bradford Dillman
Steven Dix
Hoang Doan
Uwe Dobrindt
David Drake
David Edmondson
Craig Eidman
Gary Eiserman
Gary England
Jason Esteve
James Evans
Robert Farraher
John Fay

Reginald Ford
Masakazu Furuichi
Michael Gagliano
Ralph Gibson
Edward Gordon
Len Granowetter
Jean-Baptiste Guillerit
Paul Gustavson
Per Gustavsson
Steve Hall
Fawzi Hassaine
Mark Hazen
William Helfinstine
Amy Henninger
Frank Hill
Jim Hollenbach
Torbjörn Hultén
Jean-Louis Igarza
Mark S. Johnson
Stephen Jones
Stephen Jones
Gunnar Karlsson
Mikael Karlsson
Rosemarie Keener
James Kogler
Jonathan Labin
Jennifer Lewis
Mike Lightner
Reed Little
Laurie Litwin
Staffan Löf
Björn Löfstrand
Paul Lowe
Franklin Lue
Robert Lutz
Farid Mamaghani
Lee Marden
Kim Marshall
Regis Mauget
James McCall
Michael McGarity
Sandy McPherson
Rob Minson
Mike Montgomery
Neil Morris
William Oates
Gunnar Ohlund
Mike Papay
Trevor Pearce
Mikel Petty
Tim Pokorny
Edward Powell
Laurent Prignac
Guillaume Radde
Peter Ross
Chris Rouget
Joseph Sardella
Geoff Sauerborn
John Schloman
Kevin Seavey
Graham Shanks
Petr Shlyaev
John Shockley
Pierre Siron
Keith Snively
Susan Solick
Joseph Steel
Steffen Strassburger
Marcy Stutzman
Jerry Szulinski
Martin Tapp
Gary Thomas
Andreas Tolk
Cam Tran
Ben Watrous
Marc Williams
Annette Wilson
Douglas Wood
Roger Wuerfel
Troy Yee
William Zimmerman

The working group acknowledges the following Framework and Rules Specification Drafting Group members who also contributed to the preparation of this standard:

Mike Lightner, *Editor*

Doug Flournoy
Reed Little
Robert Lutz
Chris Turrell

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

Ali Al Awazi
Bakul Banerjee
Steven Bezner
Jack Borah
Juan Carreon
Bertram Chase
Keith Chow
Tommy Cooper
Paul Croll
Dannie Cutts
Uwe Dobrindt
Sourav Dutta
Andre Fournier
Masakazu Furuichi
Len Granowetter
Randall Groves
Paul Gustavson
M. Hashmi
William Helfinstine
Rutger A. Heunks
Frank Hill
Werner Hoelzl
James Ivers
Mikael Karlsson
Mark Knight
James E. Kogler
Jonathan Labin
Susan Land
Reed Little
Björn Löfstrand
William Lumpkins
G. Luri
Robert Lutz
Edward McCall
James McCall
Gary Michel
William Milam
Björn Möller
Katherine Morse
Thomas Mullins
Ronald G. Murias
Michael S. Newman
Miroslav Pavlovic
T. Pearce
Ulrich Pohl
Jonathan Prescott
Jose Puthenkulam
Robert Robinson
Michael Rush
Peter Ryan
Randall Safier
Randy Saunders
Bartien Sayogo
John Schloman
Roy Scrudder
Steven Smith
Keith Snively
Susan Solick
Joseph Stanco
Thomas Starai
Adrian P. Stephens
Gerald Stueve
Marcy Stutzman
Thomas Tullia
Cam Van Tran
Annette Wilson
Douglas Wood
Paul Work
Roger Wuerfel
Oren Yuen
Janusz Zalewski

When the IEEE-SA Standards Board approved this standard on 25 March 2010, it had the following membership:

Robert M. Grow, *Chair*
Richard H. Hulett, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Karen Bartleson
Victor Berman
Ted Burse
Clint Chaplin
Andy Drozd
Alexander Gelman
Jim Hughes
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Hung Ling
Oleg Logvinov
Ted Olsen
Ronald C. Petersen
Thomas Prevost
Jon Walter Rosdahl
Sam Sciacca
Mike Seavey
Curtis Siller
Don Wright

*Member Emeritus

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

Satish Aggarwal, *NRC Representative*
Richard DeBlasio, *DOE Representative*
Michael Janezic, *NIST Representative*

Lisa Perry
IEEE Standards Program Manager, Document Development

Michael D. Kipness
IEEE Standards Program Manager, Technical Program Development

Contents

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules

1. Overview

1.1 Scope

This document provides an overview of the High Level Architecture (HLA), defines a family of related HLA documents, and defines the principles of HLA in terms of responsibilities that federates (simulations, supporting utilities, or interfaces to live systems) and federations (sets of federates working together) must uphold.

1.2 Purpose

This document describes the general principles defining the HLA and delineates the set of rules that apply to HLA federations and federates. Each rule is then described, and the rationale for its inclusion is provided.

Many different classes of simulations exist. Each class has changing application characteristics and needs to be flexibly supported to allow for interoperability and reuse across classes and to limit the need to maintain multiple interoperability approaches. The HLA is a common integrated architecture, which has been

developed to provide a flexible approach for addressing these interoperability and reuse needs. The related standards need to be considered as a set of products because changes in one are likely to have an impact on the others.

1.3 High Level Architecture (HLA)

The HLA is a technical architecture developed to facilitate the reuse and interoperation of simulation systems and assets (e.g., federation managers, data collectors, passive viewers, real world ("live") systems, and other utilities). It was developed in response to the clear recognition of the value of modeling and simulation (M&S) in supporting a wide variety of applications as well as the need to manage M&S assets as cost effective tools. The selected definition of an architecture as intended in HLA—"major functional elements, interfaces, and design rules, pertaining as feasible to all simulation applications, and providing a common framework within which specific system architectures can be defined"—is one that is commonly accepted and is consistent with the definition of architecture for computer simulations provided by the IEEE.

1.3.1 Components of the HLA standard

The HLA provides a general framework within which developers can structure and describe their simulation systems and/or assets and interoperate with other simulation systems and assets. The HLA consists of three main components. The first is the HLA Framework and Rules Specification (i.e., this standard), which provides a set of ten rules, five applying to *federates* (see 3.1) and five applying to *federations* (see 3.1), that together ensure the proper interaction of federates in a federation and define the responsibilities of federates and federations. The second component is the HLA object model template (OMT) Specification (found in IEEE Std 1516.1™-2010).[1] It provides a specification for describing *object models* (see 3.1) that define the information produced or required by a simulation application and for reconciling definitions among simulations to produce a common data model for mutual interoperation. The third component is a specification of services that allow simulations to connect to one another, exchange data, and coordinate activities during a distributed runtime execution. This component is the HLA federate interface specification (found in IEEE Std 1516.2™-2010), which describes the capabilities and principles of operation of an underlying software infrastructure, called the *runtime infrastructure (RTI)*, that will be required for runtime operation.

1.3.2 Terminology

Some terminology is necessary to give an overview of the HLA framework. A federate application is an application that supports the HLA interface to a RTI and that is capable of joining a federation execution. Note that, in addition to a simulation, a federate application may represent an interface to a live system or a support utility such as an event logger or performance monitor. A federate application may be coupled with other software applications and a RTI. A joined federate is a member of a federation execution, actualized by a federate application invoking the *Join Federation Execution* service. A federation is a named set of federate applications and a common federation object model (FOM) that are used as a whole to achieve some specific objective. A federation execution is the actual operation, over time, of a set of joined federates that are interconnected by a RTI.

NOTE—Formal definitions for each of these terms can be found in the definitions clause of all three HLA specifications.[2]

[1]Information on references can be found in Clause 2.

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

1.3.3 Object models

There are three types of object models in the HLA framework. First is a simulation object model (SOM). The SOM is a specification of the types of information that an individual federate application could provide to HLA federations as well as the information that an individual joined federate can receive from other joined federates in HLA federation executions. The SOM may be viewed as a federate application's native expression of its capabilities and data interfaces. The SOM facilitates simulation reuse and provides a starting point for combining the simulation with other participating simulations for a common purpose. Second is a federation object model (FOM). The FOM is 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, Management Object Model (MOM) contents, and other relevant information. Third is the MOM, which is a group of predefined constructs that provide support for monitoring and controlling a federation execution. FOMs can be developed from scratch by using relevant elements of the SOMs of the federate applications that will make up the federation or by using existing FOMs or reference FOMs and any combination of these approaches. If existing SOMs or FOMs are used, it is important to understand that there may be parts of these SOMs/FOMs that may not be relevant to other federate applications in the federation and thus will not be incorporated in the FOM. For example, a ship simulation may model boiler pressure, but if a federation needs only to know ship location, boiler pressure may optionally be dropped from the FOM. Also, it may be necessary to reconcile differing units and differing representations among simulations. The OMT provides a standard structure and a set of representational conventions for specifying both SOMs and FOMs.

1.3.4 Service groups

The federate interface specification defines the functional interface between federates and the RTI. The RTI provides services to joined federates in a way that is analogous to how a distributed operating system provides services to applications. The services are arranged into seven basic groups:

a) Federation Management. These services allow for the coordination of federation-wide activities throughout the life of a federation execution. Such services include federation execution creation and destruction, federate application joining and resigning, federation synchronization points, and save and restore operations.
b) Declaration Management. These services allow joined federates to specify the types of data they will supply to, or receive from, the federation execution. This process is done via a set of publication and subscription services along with some related services.
c) Object Management. These services support the life-cycle activities of the objects and interactions used by the joined federates of a federation execution. These services provide for registering and discovering object instances, updating and reflecting the instance attributes associated with these object instances, deleting or removing object instances as well as sending and receiving interactions and other related services. (Note: Formal definitions for each of these terms can be found in the definitions clause of all three HLA specifications.)
d) Ownership Management. These services are used to establish a specific joined federate's privilege to provide values for an object instance attribute as well as to facilitate dynamic transfer of this privilege (ownership) to other joined federates during a federation execution.
e) Time Management. These services allow joined federates to operate with a logical concept of time and to jointly maintain a distributed virtual clock. These services support discrete event simulations and assurance of causal ordering among events.
f) Data Distribution Management. These services allow joined federates to further specify the distribution conditions (beyond those provided via Declaration Management services) for the specific data they send or ask to receive during a federation execution. The RTI uses this information to route data from producers to consumers in a more tailored manner.
g) Support Services. This group includes miscellaneous services utilized by joined federates for performing such actions as name-to-handle and handle-to-name transformations, the setting of advisory switches, region manipulations, and RTI startup and shutdown.

The federate interface specification includes programming language-independent definitions of the syntax and semantics of all the services in the interface. It also provides language bindings for Java, C++, and Web Services Definition Language (WSDL).

1.3.5 Rules

The HLA rules complete the framework. They require the use of the OMT and Federate Interface specifications as well as sound development principles and practices to ensure the proper operation of a federation execution.

1.3.6 Implementation independence

The HLA allows distributed simulation systems to be developed and executed for a range of purposes in a standardized manner. The HLA does not prescribe a specific implementation, nor does it mandate the use of any particular programming language or software. Over time, as technology advances become available, new and different implementations will be possible within the framework of the HLA.

1.4 Relationship of HLA and object-oriented (OO) concepts

Although the HLA OMT is the standardized documentation structure for HLA object models, FOMs and SOMs do not completely correspond to common definitions of object models in OO 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 OO techniques recommend defining several views of the system. For HLA object models, the intended scope of the system description is much narrower and focuses specifically on requirements and capabilities for federate information exchange. For SOMs, 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, 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 attributes are, in OO parlance, data members of the class. The 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.

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.

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.

IEEE Std 1516.1-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification.[3]

IEEE Std 1516.2-2010, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification.

3. Definitions, acronyms, and abbreviations

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary: Glossary of Terms & Definitions* should be referenced for terms not defined in this clause.[4]

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.

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**.

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

[3]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).
[4]*The IEEE Standards Dictionary: Glossary of Terms & Definitions* is available at http://shop.ieee.org/.

available dimensions:

a) Pertaining to an attribute: the dimensions associated with the class attribute in the federation object model (FOM). The available dimensions of an instance attribute are the available dimensions of the corresponding class attribute.
b) Pertaining to an interaction class: the dimensions associated with that interaction class in 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 1516.1-2010.

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

base object model (BOM): A piece-part of a conceptual model, simulation object model (SOM), or federation object model (FOM) that can be used as a building block in the development and/or extension of a simulation or federation.

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 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.

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 1516.2-2010. 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 1516.2-2010. 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.

datatype: 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 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 1516.1-2010.

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

NOTE—See 6.10 in IEEE Std 1516.1-2010.

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 1516.1-2010.

discover: To receive an invocation of the *Discover Object Instance †*[5] service for a particular object instance.

NOTE—See 6.5 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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.

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

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.

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

High Level Architecture (HLA) time axis: A totally ordered sequence of 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.

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 parameter: An interaction parameter that was declared in a superclass of that interaction class in the interaction class hierarchy tree defined in the federation object model (FOM).

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 1516.1-2010.

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 1516.1-2010. *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 good logical timestamp: The last timestamp to which a lost joined federate was successfully granted, as seen from the remaining High Level Architecture (HLA) compliant federation.

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 6.12 of IEEE Std 1516.1-2010.

logical time: A federate's current point on the High Level Architecture (HLA) time axis. Federates making use of the time management services follow restrictions on what timestamps can be sent in timestamp order (TSO) messages (relative to their logical time) to ensure that federates receiving those messages receive them in TSO.

lookahead: A nonnegative value that establishes a lower value on the timestamps that can be sent in timestamp order (TSO) messages by time-regulating joined federates. Once established, a joined federate's lookahead value may be changed only by using the *Modify Lookahead* service. Each time-regulating joined federate must provide a lookahead value when becoming time-regulating.

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, datatypes, 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, or a deletion of an existing object instance, often associated with a particular point on the High Level Architecture (HLA) 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 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.

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 1516.1-2010.

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_{lower}, a_{upper}) and B = [b_{lower}, b_{upper})), to overlap if and only if either $a_{lower} = b_{lower}$ or ($a_{lower} < b_{upper}$ and $b_{lower} < a_{upper}$).

NOTE—See 9.1 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 †* service.

NOTE—See 5.1.3 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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 6.7 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 6.4 of IEEE Std 1516.1-2010.

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 1516.2-2010 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 1516.2-2010 and contains at a minimum the object class name and publish/subscribe indicator.

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

NOTE—See 6.11 of IEEE Std 1516.1-2010.

repeated description: A description of a construct (e.g., object class, interaction class, datatype, 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 datatypes.

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, an 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 1516.2-2010. 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 1516.2-2010. 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* or *Send Interaction With Regions* service invocation that initiated the sending of a given interaction.

NOTE—See 5.1.3 of IEEE Std 1516.1-2010.

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

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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 5.6 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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 5.8 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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.

Time Advancing state: The interval between a joined federate's request to advance its logical time and the corresponding grant by the runtime infrastructure (RTI). A joined federate may advance its logical time 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 logical time will not actually be advanced until the RTI responds with a *Time Advance Grant †* service invocation at that joined federate.

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 1516.1-2010.

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-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 1516.1-2010.

timestamp (of message or save): The value of the timestamp argument provided to the relevant service invocation.

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).

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 1516.1-2010.

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

NOTE—See 6.6 of IEEE Std 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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 1516.1-2010.

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.

3.2 Abbreviations and acronyms

The following abbreviations and acronyms pertain to this standard, IEEE Std 1516.1-2010, and IEEE Std 1516.2-2010:

ADT	abstract datatype
API	application programmer's interface
BNF	Backus Naur Form
BOM	base object model
DDM	data distribution management
DIF	data interchange format
DM	declaration management
DTD	document type declaration
FDD	FOM Document Data
FEDEP	Federation Development and Execution Process
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
MIM	MOM and Initialization Module
MOM	Management Object Model
M&S	modeling and simulation
NA	not applicable
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 datatype
TSO	timestamp order
WSDL	Web Service Definition Language
WSPRC	Web Service Provider RTI Component
XML	extensible markup language

4. Summary of the HLA rules

The rules for federations are as follows:

1) Federations shall have an HLA FOM, documented in accordance with the HLA OMT.
2) In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.
3) During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.
4) During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.
5) During a federation execution, an instance attribute shall be owned by at most one joined federate at any given time.

The rules for federates are as follows:

6) Federates shall have an HLA SOM, documented in accordance with the HLA OMT.
7) Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.
8) Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.
9) Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.
10) Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

5. Federation rules

This clause describes the five rules that apply to HLA federations. Rationale for the inclusion of each rule is located in Annex A.

5.1 Rule 1

Federations shall have an HLA FOM, documented in accordance with the HLA OMT.

All data to be exchanged in accordance with the HLA shall be documented in a FOM. The FOM is specified using one or more FOM modules and one MOM and Initialization Module (MIM). A FOM shall document the agreement among federates on data to be exchanged using the HLA services during federation execution and the minimal set of conditions of the data exchange (e.g., updates to be sent when changes exceed a certain value). As such, a FOM is an essential element in defining a federation. The HLA does not prescribe which data are included in a FOM (this task is the responsibility of the federation user and developer). FOMs shall be documented in the format prescribed in IEEE Std 1516.2-2010 to support reuse of a FOM by new users for their purposes.

5.2 Rule 2

In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.

In the HLA, the responsibility for maintaining the values of HLA object instance attributes shall take place in the joined federate. In an HLA federation, all joined federate-associated instance attributes shall be owned

by federates, not by the RTI. However, the RTI may own instance attributes associated with the federation MOM. The RTI may use data about instance attributes and interactions to support RTI services (e.g., Declaration Management), but these data are merely used by the RTI, not changed.

5.3 Rule 3

During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.

The HLA federate interface specification (see IEEE Std 1516.1-2010) specifies a set of interfaces to services in the RTI to support coordinated exchange of instance attribute values and interactions in accordance with a federation's FOM. Under the HLA, intercommunication of FOM data among joined federates participating in a given federation execution shall be executed by the exchange of data via the RTI services. Based on the FOM, joined federates shall identify to the RTI what information they will provide and require, along with instance attribute and interaction data corresponding to the changing state of object instances in the joined federate. The RTI shall then provide the coordination, synchronization, and data exchange among the joined federates to permit a coherent execution of the federation.

Ensuring that the right data are provided at the right times and that the data are used in a substantively correct way shall be the responsibility of the joined federates. The RTI shall ensure that the data are delivered using joined federates in accordance with their declared requirements (e.g., which data, reliability of transport, event ordering) to provide a common view of shared data across a given federation execution, as specified in the FOM.

RTI services shall be used to ensure that the coordination needs of the distributed applications (federations) are met in a consistent way across all participants in a federation and over the life of a federation execution.

5.4 Rule 4

During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.

The HLA provides a specification for a standard interface between the federate application and the RTI. Joined federates shall use this standard interface to access RTI services (see IEEE Std 1516.1-2010). The specification shall define how federate applications interact with the infrastructure. However, because the interface and the RTI will be used for a wide variety of applications requiring data exchange of diverse characteristics, the interface specification says nothing about the specific federate data to be exchanged over the interface.

5.5 Rule 5

During a federation execution, an instance attribute shall be owned by, at most, one joined federate at any given time.

The HLA allows for different joined federates to own different attributes of the same object instance (e.g., a simulation of an aircraft might own the location of the airborne sensor, whereas a sensor system model might own other instance attributes of the sensor). To ensure data coherency across the federation, at most, one joined federate may own any given instance attribute of an object instance at any given time. Joined federates may request that the ownership of instance attributes be acquired or divested, dynamically, during federation execution. Thus, ownership can be transferred, dynamically during execution, from one joined federate to another.

6. Federate rules

This clause describes the five rules that apply to HLA federates. Rationale for the inclusion of each rule is located in Annex A.

6.1 Rule 6

Federates shall have an HLA SOM, documented in accordance with the HLA OMT.

The HLA SOM shall include the object classes, class attributes, and interaction classes of the federate that can be made public in a federation. The SOM is specified using one or more SOM modules and one optional MIM. The HLA does not prescribe which data are included in the SOM; this task shall be the responsibility of the federate developer. SOMs shall be documented in the format prescribed in IEEE Std 1516.2-2010.

6.2 Rule 7

Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.

The HLA allows for joined federates to make internal object representations and interactions available for external use as part of federation executions. These capabilities for external interaction shall be documented in the SOM for the federate. If documented in the SOM, these federate capabilities shall include the obligation to export updated values of instance attributes that are calculated internally in the federate and the obligation to be able to exercise interactions represented externally (i.e., by other federates in a federation).

6.3 Rule 8

Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.

The HLA allows ownership of instance attributes of an object instance to be transferred dynamically during a federation execution. The instance attributes of a federate that can be either owned or reflected, and whose ownership can be dynamically acquired or divested during execution, shall be documented in the SOM for that federate.

6.4 Rule 9

Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.

The HLA permits federates to own (i.e., provides the privilege to produce updated values for) instance attributes of object instances represented in the federate and to then make those values available to other federates through the RTI. Different federations may specify different conditions under which instance attributes will be updated [e.g., at some specified rate, or when the amount of change in value exceeds a specified threshold (such as altitude changes of more than 1000 ft)]. The conditions applicable to the update of specific instance attributes owned by a federate shall be documented in the SOM for that federate.

6.5 Rule 10

Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

Federation designers will identify their time management approach as part of their implementation design. Federates shall adhere to the time management approach of the federation.

Annex A

(informative)

Rationale

A.1 Federation rules

A.1.1 Rationale for Rule 1

Federations shall have an HLA FOM, documented in accordance with the HLA OMT.

The HLA is domain-independent and can be used to support federations for a wide variety of uses. The formalization of agreements for information exchange is an important element of the HLA. By formalizing the development of these agreements and requiring that the results be documented in a common format, the HLA provides the means for understanding the key elements of a federation and for assisting in the reuse of the federation, in whole or in part. Such reuse is a goal of the HLA. In addition, the FOM provides the basis for some of the data used to initialize the RTI for the federation.

A.1.2 Rationale for Rule 2

In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.

One basic idea behind the HLA is to separate federate-specific functionality from general-purpose supporting infrastructure. A necessary part of this feature is that the supporting infrastructure have no information about the objects being simulated. If the opposite were allowed, and information about a given application were stored in the supporting infrastructure, that infrastructure would by definition be tailored for that specific application and no longer be general purpose. Federate functionality was separated from federation support services for several reasons. First, the RTI services are intended to be the basic set of broadly reusable capabilities needed to support federations across the widest range of users. These are essentially coordination and management services supporting federation operations, time coordination, data distribution, etc. Because they apply across a range of HLA applications, these services can be provided most cost effectively as services to the applications rather than as components of the applications. Second, separation of the RTI services has the added advantage of freeing the federates to focus on their primary objective of representing object instances to meet the needs of a user or application domain. This approach frees the developers of federates from investing time and resources in these basic common services.

A.1.3 Rationale for Rule 3

During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.

The reason for providing common RTI services to federations is to provide in common the needed basic functionality to permit coherency in data exchange among the joined federates. If a federation were to exchange data representing state changes of shared object instances or interactions outside of the RTI service suite, the coherency of the distributed application would be violated.

A.1.4 Rationale for Rule 4

During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.

By requiring a standardized, common interface between federates and the RTI, along with a common application programmer's interface (API), the HLA allows for independent development and implementation of federate applications. Federate developers can work independently and develop interfaces to the RTI without regard to RTI implementation, and RTI developments can proceed without explicit consideration of federate development. The separation of the interfaces from the requirements for federate data exchange allows for the reuse of a common interface specification across the broad spectrum of distributed M&S applications, with specific application needs tailored through the FOM mechanism.

A.1.5 Rationale for Rule 5

During a federation execution, an instance attribute shall be owned by at most one joined federate at any given time.

The HLA also provides the mechanism during federation execution to dynamically transfer ownership of an instance attribute from one joined federate to another or for a joined federate to unconditionally divest its ownership of an instance attribute. In this second case, the instance attribute becomes "unowned" by all joined federates, hence the rule explicitly states that "an instance attribute shall be owned by *at most one* joined federate at any given time." Joined federates can request ownership of "unowned" instance attributes. By defining ownership at the instance attribute level and providing the tools to hand off ownership during federation execution, the HLA provides a flexible tool set for using various combinations of joined federates to meet user needs.

A.2 Federate rules

A.2.1 Rationale for Rule 6

Federates shall have an HLA SOM, documented in accordance with the HLA OMT.

A major goal of the HLA is to support interoperability and reuse of simulations. The HLA provides this support by providing for reuse at the level of simulations (or, more generally, federates) and allowing access to the representations in those simulations.

Lack of cost-effective access to information about the object representations available in federates inhibits reuse. The requirement for a SOM addresses information access by requiring federates to document the minimum essential, salient aspects of their capabilities to allow for easy identification of the federates' potential application in new federations. Although the full set of information required by a potential user will go well beyond the SOM contents, providing easy access to characterizations of simulation based on reuse potential will enable users to decide more effectively whether to invest in further assessment of the simulations to their applications. There are certain circumstances, such as dynamically configured passive subscribers, where a federate's SOM is based on the current FOM.

A.2.2 Rationale for Rule 7

Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.

By designing federates from the outset with the ability to present internal objects/attributes/interactions as public, the mechanisms for reuse of the federate will be in place from the start.

A.2.3 Rationale for Rule 8

Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.

By building in the functionality to transfer and accept ownership of instance attributes, federates designed in accordance with the HLA provide some of the basic structural tools to become federates in the widest possible range of future federations. With this capability, it is possible to allow a federate designed for one purpose to be coupled with one designed for another purpose to meet a new requirement.

A.2.4 Rationale for Rule 9

Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.

By designing and building in the capability to vary the conditions under which they can export instance attributes, federates designed in accordance with the HLA provide some of the basic structural tools to become federates in the widest possibly range of future federations.

A.2.5 Rationale for Rule 10

Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.

The HLA time management structure is intended to support interoperability among federates that may rely on different internal concepts of time. For example, some federates might operate without any explicit notion of time or passage of time; other federates may need to run faster than real time; and some federates, as is the case with some discrete event simulations, might rely upon a scrupulously coordinated causal order of events throughout the federation.

The HLA provides a single, unifying approach to time management, in that it offers a “toolbox” of services that support (not guarantee) federations in managing their event synchronization in a way that will meet both the needs of the federation and the range of internal time management strategies of the member federates.

Different categories of federates typically use a subset of an RTI’s services to implement their mechanisms. Federates do not explicitly indicate to the RTI how time flow is being managed (e.g., time stepped, event driven, external time synchronization, independent advance). Federates for which passage of time is implicit may not need to use the services described in Clause 8 of IEEE Std 1516.1-2010. However, a federate must ensure that its internal mechanisms permit coherent operation of the time management paradigm (or paradigms) used in the federation. Some of the considerations that occur when a federation combines multiple, differing time management paradigms are discussed in 8.1 of IEEE Std 1516.1-2010.

Annex B

(informative)

Bibliography

[B1] IEEE Std 1516.3™, IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP).[6,7]

[B2] IEEE Std 1516.4™-2007, IEEE Recommended Practice for Verification, Validation, and Accreditation of a Federation — An Overlay to the High Level Architecture Federation Development and Execution Process.

[B3] SISO-STD-003-2006, Base Object Model (BOM) Template Specification, Simulation Interoperability Standards Organization (SISO), March 2006.[8]

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

[7]IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854, USA (http://standards.ieee.org/).

[8]This document is available from the Simulation Interoperability Standards Organization (http://www.sisostds.org/).