



IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)— Federate Interface Specification

IEEE Computer Society

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

IEEE Std 1516.1™-2025
(Revision of IEEE Std 1516.1-2010)

STANDARDS

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)— Federate Interface Specification

Developed by the

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

Approved 14 February 2025

IEEE SA Standards Board

Abstract: The High Level Architecture (HLA) has been developed to provide a common architecture for distributed modeling and simulation. The HLA defines an integrated approach that provides a common framework for the interconnection of interacting simulations. This document, the second in a family of three related HLA documents, defines the standard services of and interfaces to the HLA runtime infrastructure (RTI). These services are used by the interacting simulations to achieve a coordinated exchange of information when they participate in a distributed federation. The standards contained in this architecture are interrelated and need to be considered as a product set, when changes are made. They each have value independently.

Keywords: architecture, class attribute, data distribution management, federate, federation, federation execution, federation object model, HLA, IEEE 1516.1™, instance attribute, instance attribute ownership, interaction class, object class, runtime infrastructure, simulation object model, time-constrained, time-regulating

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

Copyright © 2025 by The Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 21 August 2025. Printed in the United States of America.

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

Print: ISBN 979-8-8557-1882-9 STDPD27717
PDF: ISBN 979-8-8557-1881-2 STD27717

IEEE prohibits discrimination, harassment and bullying.

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

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

Important Notices and Disclaimers Concerning IEEE Standards Documents

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

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

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

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

Use of an IEEE standard is wholly voluntary. The existence of an IEEE standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard.

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

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

Translations

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

Use by artificial intelligence systems

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

Official statements

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

Comments on standards

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

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

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

Laws and regulations

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

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

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

Data privacy

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

Copyrights

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

Photocopies

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

Updating of IEEE Standards documents

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

Every IEEE standard is subjected to review at least every 10 years. When a document is more than 10 years old and has not undergone a revision process, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE standard.

In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit [IEEE Xplore](#) or [contact IEEE](#).³ For more information about the IEEE SA or IEEE's standards development process, visit the IEEE SA Website.

Errata

Errata, if any, for all IEEE standards can be accessed on the [IEEE SA Website](#).⁴ Search for standard number and year of approval to access the web page of the published standard. Errata links are located under the Additional Resources Details section. Errata are also available in [IEEE Xplore](#). Users are encouraged to periodically check for errata.

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

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

Patents

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

Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken by the IEEE with respect to the existence or validity of any patent rights in connection therewith. If a patent holder or patent applicant has filed a statement of assurance via an Accepted Letter of Assurance, then the statement is listed on the IEEE SA Website at <https://standards.ieee.org/about/sasb/patcom/patents.html>. Letters of Assurance may indicate whether the Submitter is willing or unwilling to grant licenses under patent rights without compensation or under reasonable rates, with reasonable terms and conditions that are demonstrably free of any unfair discrimination to applicants desiring to obtain such licenses.

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

IMPORTANT NOTICE

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

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

Participants

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

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

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

Douglas Wood and **Felix Rodriguez**, *Interface Specification Editors*
Mikael Karlsson, *API Editor*
David Drake, *XML Editor*

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

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

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

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

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

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

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

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

*Member Emeritus

Introduction

This introduction is not part of IEEE Std 1516.1-2025, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification.

This document has been developed to record an international standard for the High Level Architecture (HLA). It serves as one of three related standards for the HLA. It defines the services and interfaces to be used by federates when participating in a federation execution.

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

Schema, API, and Documentation

Schema and API: The IEEE hereby grants a general, royalty-free license to copy, distribute, display, and make derivative works from this material, for all purposes, provided that any use of the material contains the following attribution: “Reprinted with permission from IEEE Std 1516.1™-2025.” Should a reader require additional information, contact the Manager, Standards Intellectual Property, IEEE Standards Association (stds-ipr@ieee.org).

Documentation: The IEEE hereby grants a general, royalty-free license to copy, distribute, display, and make derivative works from this material, for noncommercial purposes, provided that any use of the material contains the following attribution: “Reprinted with permission from IEEE Std 1516.1™-2025.” The material may not be used for a commercial purpose without express written permission from the IEEE. Should a reader require additional information, contact the Manager, Standards Intellectual Property, IEEE Standards Association (stds-ipr@ieee.org).

Contents

1.	Overview.....	16
1.1	Scope.....	16
1.2	Purpose.....	16
1.3	Word usage.....	16
1.4	Background.....	16
1.5	General nomenclature and conventions.....	18
1.6	Organization of this document.....	19
2.	Normative references.....	21
3.	Definitions, abbreviations and acronyms.....	22
3.1	Definitions.....	22
3.2	Abbreviations and acronyms.....	37
4.	Federation management.....	38
4.1	Overview.....	38
4.2	Connect service.....	51
4.3	Disconnect service.....	52
4.4	Connection Lost † service.....	52
4.5	Create Federation Execution service.....	53
4.6	Destroy Federation Execution service.....	54
4.7	List Federation Executions service.....	55
4.8	Report Federation Executions † service.....	55
4.9	List Federation Execution Members service.....	56
4.10	Report Federation Execution Members † service.....	56
4.11	Join Federation Execution service.....	57
4.12	Resign Federation Execution service.....	58
4.13	Federate Resigned † service.....	60
4.14	Register Federation Synchronization Point service.....	60
4.15	Confirm Synchronization Point Registration † service.....	61
4.16	Announce Synchronization Point † service.....	62
4.17	Synchronization Point Achieved service.....	63
4.18	Federation Synchronized † service.....	64
4.19	Request Federation Save service.....	65
4.20	Initiate Federate Save † service.....	67
4.21	Federate Save Begun service.....	67
4.22	Federate Save Complete service.....	68
4.23	Federation Saved † service.....	69
4.24	Abort Federation Save service.....	70
4.25	Query Federation Save Status service.....	71
4.26	Federation Save Status Response † service.....	72
4.27	Request Federation Restore service.....	72
4.28	Confirm Federation Restoration Request † service.....	73
4.29	Federation Restore Begun † service.....	74
4.30	Initiate Federate Restore † service.....	75
4.31	Federate Restore Complete service.....	76
4.32	Federation Restored † service.....	76
4.33	Abort Federation Restore service.....	77
4.34	Query Federation Restore Status service.....	78

4.35	Federation Restore Status Response † service	79
5.	Declaration management (DM)	81
5.1	Overview	81
5.2	Publish Object Class Attributes service	90
5.3	Unpublish Object Class Attributes service	91
5.4	Publish Interaction Class service	93
5.5	Unpublish Interaction Class service	93
5.6	Publish Object Class Directed Interactions service	94
5.7	Unpublish Object Class Directed Interactions service	95
5.8	Subscribe Object Class Attributes service	96
5.9	Unsubscribe Object Class Attributes service	97
5.10	Subscribe Interaction Class service	99
5.11	Unsubscribe Interaction Class service	100
5.12	Subscribe Object Class Directed Interactions service	101
5.13	Unsubscribe Object Class Directed Interactions service	102
5.14	Start Registration For Object Class † service	103
5.15	Stop Registration For Object Class † service	104
5.16	Turn Interactions On † service	105
5.17	Turn Interactions Off † service	106
6.	Object management	107
6.1	Overview	107
6.2	Reserve Object Instance Name service	113
6.3	Object Instance Name Reserved † service	113
6.4	Release Object Instance Name service	114
6.5	Reserve Multiple Object Instance Names service	115
6.6	Multiple Object Instance Names Reserved † service	116
6.7	Release Multiple Object Instance Names service	117
6.8	Register Object Instance service	118
6.9	Discover Object Instance † service	119
6.10	Update Attribute Values service	120
6.11	Reflect Attribute Values † service	122
6.12	Send Interaction service	123
6.13	Receive Interaction † service	125
6.14	Send Directed Interaction Service	126
6.15	Receive Directed Interaction † service	128
6.16	Delete Object Instance service	129
6.17	Remove Object Instance † service	131
6.18	Local Delete Object Instance service	132
6.19	Attributes In Scope † service	133
6.20	Attributes Out Of Scope † service	134
6.21	Request Attribute Value Update service	135
6.22	Provide Attribute Value Update † service	137
6.23	Turn Updates On For Object Instance † service	137
6.24	Turn Updates Off For Object Instance † service	138
6.25	Request Attribute Transportation Type Change service	139
6.26	Confirm Attribute Transportation Type Change † service	140
6.27	Change Default Attribute Transportation Type service	141
6.28	Query Attribute Transportation Type service	142
6.29	Report Attribute Transportation Type † service	143
6.30	Request Interaction Transportation Type Change service	144

6.31	Confirm Interaction Transportation Type Change † service.....	145
6.32	Query Interaction Transportation Type service	146
6.33	Report Interaction Transportation Type † service	147
7.	Ownership management	148
7.1	Overview	148
7.2	Unconditional Attribute Ownership Divestiture service.....	154
7.3	Negotiated Attribute Ownership Divestiture service	155
7.4	Request Attribute Ownership Assumption † service	157
7.5	Request Divestiture Confirmation † service	158
7.6	Confirm Divestiture service	159
7.7	Attribute Ownership Acquisition Notification † service	160
7.8	Attribute Ownership Acquisition service.....	161
7.9	Attribute Ownership Acquisition If Available service.....	163
7.10	Attribute Ownership Unavailable † service	165
7.11	Request Attribute Ownership Release † service	166
7.12	Attribute Ownership Release Denied service	167
7.13	Attribute Ownership Divestiture If Wanted service.....	168
7.14	Cancel Negotiated Attribute Ownership Divestiture service	169
7.15	Cancel Attribute Ownership Acquisition service.....	170
7.16	Confirm Attribute Ownership Acquisition Cancellation † service	171
7.17	Query Attribute Ownership service	172
7.18	Inform Attribute Ownership † service	173
7.19	Is Attribute Owned By Federate service	174
8.	Time management.....	175
8.1	Overview	175
8.2	Enable Time Regulation service	188
8.3	Time Regulation Enabled † service	189
8.4	Disable Time Regulation service	190
8.5	Enable Time Constrained service.....	191
8.6	Time Constrained Enabled † service.....	192
8.7	Disable Time Constrained service	193
8.8	Time Advance Request service.....	194
8.9	Time Advance Request Available service	195
8.10	Next Message Request service.....	197
8.11	Next Message Request Available service	198
8.12	Flush Queue Request service	200
8.13	Flush Queue Grant † service.....	202
8.14	Time Advance Grant † service.....	203
8.15	Enable Asynchronous Delivery service	204
8.16	Disable Asynchronous Delivery service	205
8.17	Query GALT service.....	206
8.18	Query Logical Time service.....	206
8.19	Query LITS service	207
8.20	Modify Lookahead service.....	208
8.21	Query Lookahead service.....	209
8.22	Retract service.....	210
8.23	Request Retraction † service.....	211
8.24	Change Attribute Order Type service	212
8.25	Change Default Attribute Order Type service	213
8.26	Change Interaction Order Type service	214

9.	Data distribution management (DDM)	216
9.1	Overview	216
9.2	Create Region service	226
9.3	Commit Region Modifications service	226
9.4	Delete Region service	227
9.5	Register Object Instance With Regions service	228
9.6	Associate Regions For Updates service	230
9.7	Unassociate Regions For Updates service	232
9.8	Subscribe Object Class Attributes With Regions service	233
9.9	Unsubscribe Object Class Attributes With Regions service	235
9.10	Subscribe Interaction Class With Regions service	236
9.11	Unsubscribe Interaction Class With Regions service	238
9.12	Send Interaction With Regions service	239
9.13	Request Attribute Value Update With Regions service	241
10.	Support services	243
10.1	Overview	243
10.2	Get Federate Handle service	245
10.3	Get Federate Name service	245
10.4	Get Object Class Handle service	246
10.5	Get Object Class Name service	247
10.6	Get Known Object Class Handle service	247
10.7	Get Object Instance Handle service	248
10.8	Get Object Instance Name service	249
10.9	Get Attribute Handle service	249
10.10	Get Attribute Name service	250
10.11	Get Update Rate Value service	251
10.12	Get Update Rate Value For Attribute service	252
10.13	Get Interaction Class Handle service	252
10.14	Get Interaction Class Name service	253
10.15	Get Parameter Handle service	254
10.16	Get Parameter Name service	254
10.17	Get Order Type service	255
10.18	Get Order Name service	256
10.19	Get Transportation Type Handle service	256
10.20	Get Transportation Type Name service	257
10.21	Get Available Dimensions For Object Class service	258
10.22	Get Available Dimensions For Interaction Class service	259
10.23	Get Dimension Handle service	259
10.24	Get Dimension Name service	260
10.25	Get Dimension Upper Bound service	261
10.26	Get Dimension Handle Set service	261
10.27	Get Range Bounds service	262
10.28	Set Range Bounds service	263
10.29	Normalize Service Group service	264
10.30	Normalize Federate Handle service	265
10.31	Normalize Object Class Handle service	265
10.32	Normalize Interaction Class Handle service	266
10.33	Normalize Object Instance Handle service	267
10.34	Get Object Class Relevance Advisory Switch service	267
10.35	Set Object Class Relevance Advisory Switch service	268
10.36	Get Attribute Relevance Advisory Switch service	269

10.37	Set Attribute Relevance Advisory Switch service	269
10.38	Get Attribute Scope Advisory Switch service	270
10.39	Set Attribute Scope Advisory Switch service	271
10.40	Get Interaction Relevance Advisory Switch service.....	272
10.41	Set Interaction Relevance Advisory Switch service	272
10.42	Get Convey Region Designator Sets Switch service	273
10.43	Set Convey Region Designator Sets Switch service.....	274
10.44	Get Automatic Resign Directive service.....	274
10.45	Set Automatic Resign Directive service	275
10.46	Get Service Reporting Switch service.....	276
10.47	Set Service Reporting Switch service	277
10.48	Get Exception Reporting Switch service	277
10.49	Set Exception Reporting Switch service	278
10.50	Get Send Service Reports To File Switch service	279
10.51	Set Send Service Reports To File Switch service	280
10.52	Get Auto Provide Switch service.....	280
10.53	Get Delay Subscription Evaluation Switch service	281
10.54	Get Advisories Use Known Class Switch service.....	282
10.55	Get Allow Relaxed DDM Switch service	282
10.56	Get Non Regulated Grant Switch service	283
10.57	Evoke Callback service	284
10.58	Evoke Multiple Callbacks service.....	284
10.59	Enable Callbacks service.....	285
10.60	Disable Callbacks service	286
11.	Management object model (MOM)	287
11.1	Overview	287
11.2	MOM object classes.....	287
11.3	MOM interaction classes.....	288
11.4	MOM-related characteristics of the RTI.....	290
11.5	Service reporting	292
11.6	MOM OMT tables	305
12.	Programming language mappings	353
12.1	Overview	353
12.2	Designators.....	353
12.3	Logical time, timestamps, and lookahead	354
12.4	Standardized time types	354
12.5	Authorization	356
12.6	Predefined authorization service	357
12.7	Connect	357
12.8	Concurrency and reentrancy	357
12.9	Dynamic link compatibility	358
12.10	Callback representation of timestamps, order type, retraction handle and regions	358
12.11	Java.....	358
12.12	C++.....	377
12.13	HLA Federate Protocol	395
13.	Conformance.....	413
13.1	Federate conformance	413
13.2	RTI conformance	413

Annex A (informative) API information	414
Annex B (normative) Java API.....	415
Annex C (normative) C++ API.....	420
Annex D (normative) Federate Protocol API	422
Annex E (normative) Rationale	423
Annex F (informative) FOM Document Data (FDD) XML Schema declaration	433
Annex G (normative) MOM Initialization Module (MIM).....	434
Annex H (informative) Figure table	435
Annex I (informative) Bibliography	437

IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)— Federate Interface Specification

1. Overview

1.1 Scope

This document defines the interface between federates (simulations, supporting utilities, or interfaces to live systems) and the underlying software services that support interfederate communication in a distributed simulation domain.

1.2 Purpose

The High Level Architecture (HLA) has been developed to provide a common architecture for distributed modeling and simulation (M&S). To facilitate interfederate communications, HLA federates interact with an underlying software infrastructure. This specification defines the standard services and interfaces to be used by the federates to support efficient information exchange when participating in a distributed federation execution. (A federation execution occurs when sets of federates are brought together to support an objective.)

1.3 Word usage

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

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

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

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

1.4 Background

This document is being developed to define the HLA interface specification. The HLA requires a language-independent specification (LIS) and multiple language bindings to support intersimulation communication in a distributed simulation domain.

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

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

The formal definition of the M&S HLA comprises three main components: the HLA framework and rules specification (found in IEEE Std 1516TM-2025), the HLA federate interface specification (found in this document), and the HLA object model template (OMT) (found in IEEE Std 1516.2TM-2025). This standard, IEEE Std 1516.1-2025, provides a complete description of the essential elements of the second component of the HLA, the interface specification.⁸

The HLA is an integrated architecture that has been developed to provide a common architecture for M&S. The HLA requires that interfederate communications use a standard application programmer's interface (API). This specification defines the standard services and interfaces to be used by the federates in order to support efficient information exchange when participating in a distributed federation execution. Additionally, the capability for reuse of individual federates that adhere to these standard services and interfaces is increased.

This document provides a specification for the HLA functional interfaces between federates and the runtime infrastructure (RTI). The RTI provides services to federates in a way that is analogous to how a distributed operating system provides services to applications. These interfaces are arranged into seven basic service groups, and an eighth group of support services, as follows:

- a) Federation management
- b) Declaration management (DM)
- c) Object management
- d) Ownership management
- e) Time management
- f) Data distribution management (DMM)
- g) Support services

The eight service groups describe the interface between the federates and the RTI and between the federates and the software services provided by the RTI for use by HLA federates. The RTI requires a set of services from the federate that are referred to as “RTI-initiated” and are denoted with a † (printer's dagger) after the service name.

1.4.1 HLA federation object model (FOM) framework

A concise and rigorous description of the object model framework is essential to the specification of the interface between federates and the RTI and of the HLA services. The rules and terminology used to describe a FOM are described in IEEE Std 1516.2-2025. A simulation object model (SOM) describes salient characteristics of a federate to aid in its reuse and other activities focused on the details of its internal operation. As such, a SOM is not the concern of the RTI and its services. A FOM, on the other hand, deals with interfederate issues and is relevant to the use of the RTI. FOMs describe the set of object classes and interaction classes chosen for a planned federation along with the attributes of the object classes and parameters of the interaction classes.

Every HLA object instance is an instance of an object class found in the FOM. Object classes are chosen by the object model designer to facilitate an appropriate organizational scheme. Each object class has a set of attributes associated with it. An attribute is a distinct, identifiable portion of the object's state. In this discussion, “attribute designator” refers to the attribute, and “attribute value” refers to its contents. From the federation perspective, the set of all attribute values for an object instance shall completely define the state of that instance. Federates may associate additional state information with an object instance that is not communicated between federates, but this action is outside the purview of the HLA FOM.

⁸Information on references can be found in Clause 2.

Federates use the state of the object instances as one of the primary means of communication. At any point during a federation execution, at most one federate or the RTI shall have the privilege to update the value of a given object instance attribute. A joined federate that has the privilege to update values for an instance attribute is said to own that instance attribute. There are specific (MOM) instance attributes that are owned by the RTI. In some cases, an instance attribute may be unowned, neither owned by a federate nor the RTI. The Ownership Management services allow joined federates to gain ownership of unowned instance attributes, exchange ownership of instance attributes, or release ownership of instance attributes (possibly to become unowned). The federate provides instance attribute values to the other federates in the federation execution via Object Management services. The federate providing the instance attribute values is said to be updating that instance attribute. Federates receiving those values are said to be reflecting that instance attribute.

Each object instance shall have a designator. The value of an object instance designator shall be unique for each federation execution. Object instance designators shall be dynamically generated by the RTI.

The FOM framework also allows for interaction classes for each object model. The types of interactions possible and their parameters are specified within the FOM.

A **federation** is the combination of a particular FOM, a particular set of federates, and the HLA services. A federation is designed for a specific purpose using a commonly understood FOM and a set of federates that may associate their individual semantics with that object model. A federation execution is an instance of the *Create Federation Execution* service invocation and entails executing the federation with a specific FOM and RTI and using various execution details.

1.5 General nomenclature and conventions

There are various entities (e.g., object classes, attributes, interaction classes, parameters, dimensions, regions, federates, object instances) referenced in this document that may have the following different views:

- Name: textual or for communication between federates.
- Handle: capable of being manipulated by a computer or for communication between a federate and the RTI. A handle is originated by the RTI, federation execution-wide unique, and unpredictable. The handle for a class attribute that is inherited shall be the same as the handle that is assigned to the class attribute in the object class in which it is declared. The handle for a parameter that is inherited shall be the same as the handle that is assigned to the parameter in the interaction class in which it is declared.

The following types of names shall be case-sensitive:

- Federation execution
- Federate
- Object class
- Attribute
- Object instance
- Interaction class
- Parameter
- Order
- Transportation type
- Dimension

The arguments to the services described in this document will use different views of the entities depending on a particular RTI implementation. For clarity, the first 11 clauses of this document refer to only a generic view, known as a “designator,” when referring to these entities, and the API annexes refer to the actual implementation-specific arguments. Clause 12 ties the two concepts together.

Some of the arguments to the services described in this document are in the form of a group that represents some association. These groupings can take several forms (depending on the requirements of the respective service), each of which is described below. The groupings may be implemented slightly differently in each API, but all implementations shall support the same semantics. These groupings are pairs, sets, collections, constrained sets of pairs, and collections of pairs. Collections are sets that permit duplication of elements. They are used in cases in which duplication is unlikely and not really an error. As such, using a collection rather than a set retains the same effects, but without the expensive checks to eliminate duplicates that using a set would require. In these cases, it is preferred not to penalize the majority of users with the less efficient set just to catch errors that are anticipated to be infrequent.

The following data are needed for the implementation of a running RTI and federation executions:

- FOM Document Data (FDD): FOM information (e.g., class, attribute, parameter names) used by the RTI at runtime. Each federation execution needs FDD. FDD is provided to the RTI starting with the Management Object Model (MOM) and Initialization Module (MIM) and FOM modules specified in the *Create Federation Execution* service invocation. FDD will also be provided by the MIM that is automatically provided by the RTI or optionally provided in the *Create Federation Execution* service invocation. Additional FDD may be provided as joining federates specify additional FOM modules.

The following data may be needed for the implementation of a running RTI and federation executions:

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

For all joined federate-initiated services in this specification (except 4.2, 4.5, 4.6, and 4.11), an implied supplied argument is a joined federate’s conjointment with a federation execution. For all RTI-initiated services, an implied supplied argument is also a joined federate’s connection to a federation execution. The manner in which these arguments are actually provided to the services is dependent on the RTI implementation and the particular API programming language; therefore, it is not shown in the service descriptions. Also, for the RTI-initiated services, some implicit preconditions are not stated explicitly because the RTI is assumed to be well behaved. For many RTI-initiated services, the “Federate is joined” precondition presumes the “Federate is connected” precondition.

As mentioned earlier, all RTI-initiated services are denoted with a † (printer’s dagger) after the service name.

If an RTI-invoked service is not the result of a joined federate-invoked service, but the RTI-invoked service has a user-supplied tag as a mandatory argument, the user-supplied tag shall be present in the service invocation but empty. For example, in the Java API, the tag shall be an empty (zero-length) array. Examples of RTI-invoked services with empty user-supplied tags are Provide Attribute Value Update (via Auto Provide switch), Request Attribute Ownership (via Unconditional Divest resign directive), and MOM HLAmanager.HLAfederate.HLAreport interactions.

1.6 Organization of this document

The seven HLA service groups are specified in Clause 4 through Clause 10. Each service is described using several components:

- Name and description: service name and narrative describing the functionality of the service.

- Supplied arguments: arguments supplied to the service
- Returned arguments: arguments returned by the service.
- Preconditions: conditions that shall exist for the service to execute correctly.
- Postconditions: conditions that shall exist once the service has executed correctly.
- Exceptions: notifications of any irregularity that may occur during service invocation. Wherever possible (excepting situations like RTI internal error), exceptions are side-effect free. Exceptions are listed in order of increasing precedence. That is, if the conditions for more than one exception are met simultaneously, the RTI shall throw whichever of those exceptions comes later in the exception list.
- Reference state charts: reference to state chart(s) that are germane to the service.

After the clauses describing each of the service groups are a clause describing the MOM and then a clause that ties the service narratives with the APIs.

Annex A through Annex D contain HLA APIs for the following languages:

- Java
- C++
- Federate Protocol

Annex E contains additional rationale for elements in the main body of this standard.

Annex F contains the FDD XML Schema.

Annex G contains the MIM.

Annex H and Annex I contain a list of figures and bibliography, respectively.

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 1516TM-2025, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules.^{9,10}

IEEE Std 1516.2TM-2025, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification.

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

⁹IEEE publications are available from the Institute of Electrical and Electronics Engineers (<http://standards.ieee.org/>).

¹⁰The IEEE standards or products referred to in Clause 2 are trademarks owned by The Institute of Electrical and Electronics Engineers, Incorporated.

3. Definitions, abbreviations and acronyms

3.1 Definitions

For the purposes of this document, the following terms and definitions apply. *The IEEE Standards Dictionary Online* should be consulted for terms not defined in this clause.¹¹

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**.

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

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

available dimensions:

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

NOTE—See 9.1.¹²

See also **dimension**.

¹¹The *IEEE Standards Dictionary Online* is available at <http://shop.ieee.org/>. An IEEE Account is required for access to the dictionary, and one can be created at no charge on the dictionary sign-in page.

¹²Notes in text, tables, and figures are given for information only and do not contain requirements needed to implement the standard.

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

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

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 and directed interactions only.

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

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

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

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

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

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

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

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

connection fault: In the Federate Protocol, when the connection between the RTI and a federate becomes inviable due to a broken socket connection or either the federate or the RTI exceeding the timeout interval.

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

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

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

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

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

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

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

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

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

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

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

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

NOTE—See 9.1.

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

NOTE—See 6.16.

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.

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

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

discover: To receive an invocation of the *Discover Object Instance* ^{†13} service for a particular object instance.

NOTE—See 6.9.

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.

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

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

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

fault: The occurrence of an unexpected event, at any point during a federation execution, that prevents one or more joined federates in the federation from executing in an 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**.

¹³All RTI-initiated services are denoted with a † (printer's dagger) after the service name.

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

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

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

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

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

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

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

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

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

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

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

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

inherited dimension:

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

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

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).

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

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

- a) The object instance is known to the joined federate.
- b) The instance attribute is owned by another joined federate.
- c) The instance attribute's corresponding class attribute is 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.

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

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

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

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

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

known class:

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

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

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

local delete: To invoke the *Local Delete Object Instance* service to inform the runtime infrastructure (RTI) that it is to treat the specified object instance as if the RTI had never notified the joined federate to discover

the object instance (however, the object instance is not to be eliminated and may be rediscovered). *See also:* **delete**.

NOTE—See 6.18.

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

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

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

lost federate: A previously joined federate that, as a result of a fault, can no longer continue in the federation execution in a High Level Architecture (HLA) compliant manner.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

NOTE—See 6.1.

overlap:

- a) Pertaining to region sets: given two region sets, to have a region in each set such that the two regions overlap.
- b) Pertaining to regions: given two regions, to have at least one dimension in common if and only if, for each dimension the regions have in common, their respective ranges in the common dimension overlap. If two regions do not have any dimensions in common, the regions do not overlap.
- c) Pertaining to ranges: given two ranges ($A = [a_{\text{lower}}, a_{\text{upper}})$ and $B = [b_{\text{lower}}, b_{\text{upper}})$), to overlap if and only if
 - 1) Either $a_{\text{lower}} = b_{\text{lower}}$ or ($a_{\text{lower}} < b_{\text{upper}}$ and $b_{\text{lower}} < a_{\text{upper}}$) or
 - 2) The Allow Relaxed DDM switch is enabled and the RTI implementation makes the determination, based on implementation-specific criteria, that the ranges are sufficiently close to warrant an expanded interpretation of one or both ranges; neither range may ever be reduced.

NOTE—See 9.1.

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* $\dagger$ 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.

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.

- 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.

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.

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.

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

NOTE—See 9.1.

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

NOTE—See 9.1.

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

NOTE—See 5.1.3.

received parameters: The set of parameters received when the *Receive Interaction* † or *Receive Directed 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.

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

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

NOTE—See 6.11.

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.

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.

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.

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

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

NOTE—See 6.8.

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

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

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

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

NOTE—See 6.17.

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

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

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

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

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

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

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

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

NOTE—See 5.1.3.

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

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

NOTE—See 5.1.3.

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.

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

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

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

subscribed:

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

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

NOTE—See 5.1.2 and 5.8.

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.

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

NOTE—See 5.1.3 and 5.10.

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.

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

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

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

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

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

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

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

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

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

NOTE—See 8.1.

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

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

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

NOTE—See 8.1.

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

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

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.

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

NOTE—See 6.10.

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 of 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.

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.

- 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.

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.

- 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.

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.

- 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.

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

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

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

3.2 Abbreviations and acronyms

The following abbreviations and acronyms pertain to IEEE Std 1516-2025, this standard, and IEEE Std 1516.2-2025.

ADT	abstract data type
ABI	application binary interface
API	application programmer's interface
BNF	Backus Naur Form
BOM	base object model
DDM	data distribution management
DIF	data interchange format
DLC	dynamic link compatibility
DM	declaration management
DSEEP	Distributed Simulation Engineering and Execution Process
DTD	document type declaration
FDD	FOM Document Data
FOM	federation object model
GALT	Greatest Available Logical Time
HLA	High Level Architecture
JAR	Java Archive
LIS	language-independent specification
LITS	Least Incoming Timestamp
LRC	Local Runtime Component
LT	Logical Time
LTI	Logical Time Interval
MIM	MOM and Initialization Module
MOM	Management Object Model
M&S	modeling and simulation
NA	not applicable
OL	Optimistic Lookahead
OLT	Optimistic Logical Time
OMT	object model template
OO	object oriented
OOAD	object-oriented analysis and design
POC	point of contact
RID	RTI initialization data
RO	receive order
RTI	runtime infrastructure
SOM	simulation object model
TRADT	time representation abstract data type
TSO	timestamp order
URI	uniform resource identifier
URL	uniform resource locator
XML	extensible markup language

4. Federation management

4.1 Overview

Federation management refers to the creation, dynamic control, modification, and deletion of a federation execution.

Before any interaction can take place between a federate and an RTI, such as creating a federation execution or joining, the unjoined federate shall perform a *Connect* to the RTI. When a federate has resigned and has no further intent to join or create federation executions, it shall perform a *Disconnect*.

The exact manner in which RTI-initiated services (callbacks) are invoked on a federate by the RTI is programming-language specific. However, the HLA defines two callback models that govern when callbacks may be invoked. The two callback models are: IMMEDIATE and EVOKED. In the IMMEDIATE model, the RTI shall invoke callbacks immediately when they are available, unless the callbacks have been disabled using the *Disable Callbacks* service. In the EVOKED model, callbacks are only made when the federate calls the *Evoke Callback/Evoke Multiple Callbacks* services, unless callbacks have been disabled using the *Disable Callbacks* service. RTIs shall implement both callback models. The callback model is specified as an argument to the *Connect* service.

Figure 1 shows the overall state of a federation execution as certain basic federation management services are employed. The figure starts in the Federation Execution Does Not Exist state, representing that the RTI has been started up, but that no services have been invoked relative to the federation execution. After the federation execution is created, it is now in the No Joined Federates state. This state represents the clean, or freshly initialized, federation execution; no federates are joined, no object instances exist, and no messages are queued. From the point that the first federate joins this execution until the point that the last federate resigns from the execution, the federation execution is in the Supporting Joined Federates state. This state represents an operational federation execution in which the services documented in this specification are usable.

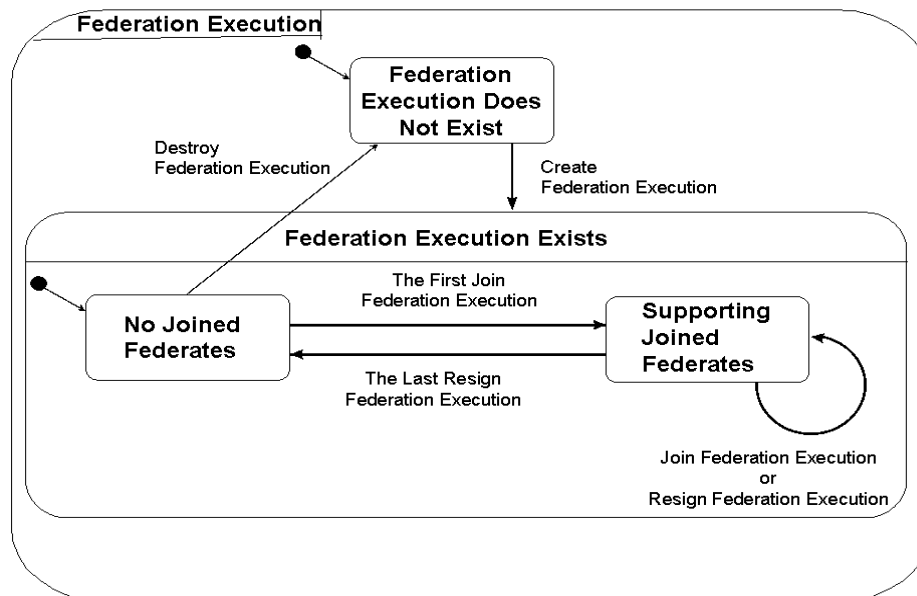


Figure 1—Basic states of the federation execution

Once a federation execution exists, federates may join and resign from it in any sequence that is meaningful to the federation user. Figure 2 presents a generalized view of the basic relationship between a federate and the RTI during the federate participation in a federation execution. The broad arrows in Figure 2 represent the general invocation of HLA service groups and are not intended to demonstrate strict ordering requirements on the use of the services. An RTI shall allow an application to participate in a federation as multiple joined federates, and an RTI shall allow a single application to participate in multiple federations as a joined federate or federates.

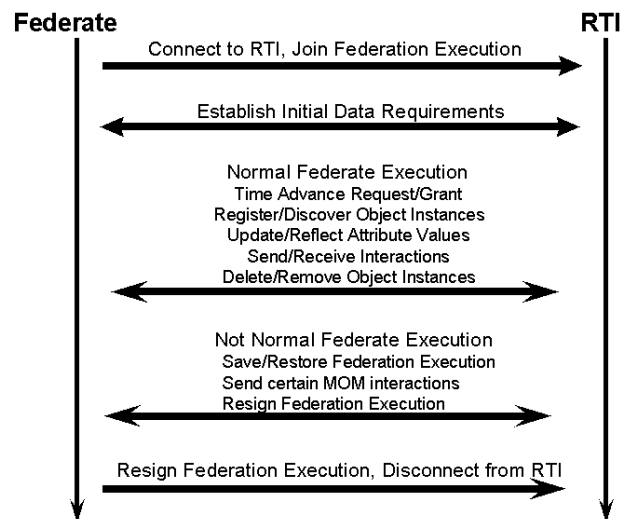


Figure 2—Overall view of federate-to-RTI relationship

The state diagram in Figure 3 is the first of a series of hierarchical state diagrams that formally describe the exchanges between federates and an RTI. These state diagrams are formal, accurate descriptions of federate state information depicted in the highly structured, compact, and expressive state chart notation pioneered by David Harel [B4].¹⁴ This specification contains two independent sets of state charts. The first set, beginning in Figure 3 and including all other state charts except Figure 6 through Figure 9, describes the state of a given federate, from the perspective of that federate, in varying levels of detail. The second set, Figure 6 through Figure 9, describes the state of a federation execution (including the RTI and all joined federates) strictly with respect to the use of synchronization points. Finally, Figure 1 is unrelated to either of these sets of state charts. It is a stand-alone overview of the basic states of a federation execution.

The next few subclauses describe the first two of these state charts in detail as a way of introducing some of Harel's notation and providing an understanding of how the complete collection of state charts in this document are hierarchically interrelated.

¹⁴Numbers in brackets correspond to those in the bibliography in Annex I.

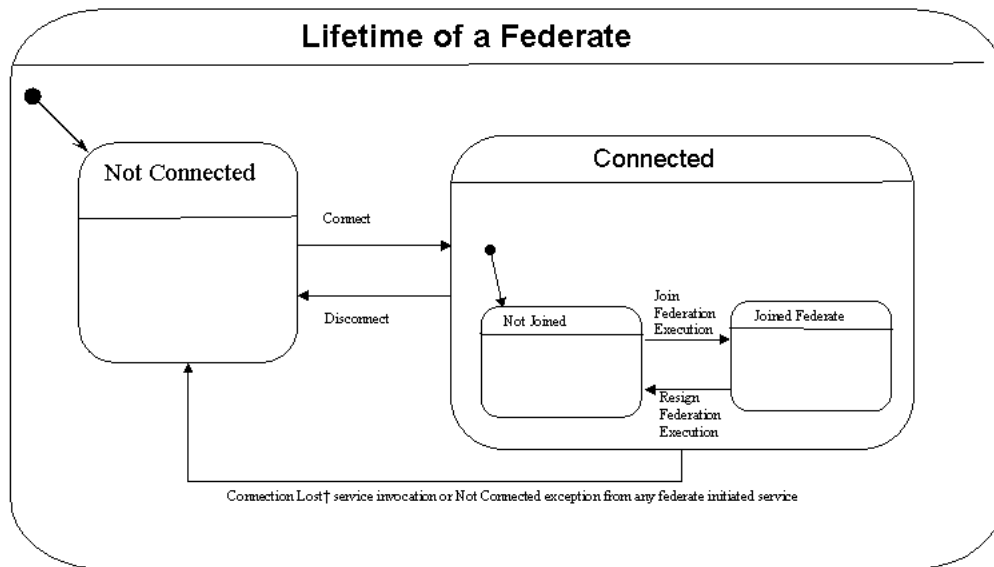


Figure 3—Lifetime of a federate

4.1.1 Federate lifetime

As shown in Figure 3, the initial state of any federate is “Not Connected.” Before a federate can work with an RTI, such as to create a federation execution or join one, the federate shall perform a *Connect* to the RTI. The *Connect* service allows the RTI to set up and initialize its communication infrastructure. After successful invocation of the *Join Federation Execution* service, a federate will be in the Joined Federate state, where distributed simulation activities take place. If an infrastructure problem is detected during these activities, the *Connection Lost* ^{†15} service notifies the federate that a fault has been detected and that it has been forced to the Not Connected state. If previously joined, it shall be now no longer joined to the federation execution. As described in 4.1.5, the effect will be as if the federate had resigned with a resign action specified by the Automatic Resign Directive. Following successful simulation activities, the federate invokes the *Resign Federation Execution* service to leave the Joined Federate state. If a federate is forced to leave the joined state for any reason other than calling the *Resign Federation Execution* service, the *Federate Resigned* [†] service notifies the federate that a fault has been detected and that it has been forced into the Not Joined state. When a federate has resigned and has no further intent to join or create federation executions, it shall perform a *Disconnect*. The *Disconnect* service allows the RTI to break down and clean up its communication infrastructure. Figure 4 shows the federate’s lifetime in the Joined Federate state.

As indicated by the dashed line in Figure 4, the Joined Federate state consists of two parallel state machines: one having to do with whether the joined federate is in the process of saving or restoring the Joined Federate state (depicted to the left of the dashed line), and the other having to do with whether the joined federate is permitted to perform normal activity (depicted to the right of the dashed line). While in the Joined Federate state, the joined federate is simultaneously in both a state depicted in the state machine to the left of the dashed line and a state depicted in the state machine to the right of the dashed line. Initially, upon entering the Joined Federate state, the joined federate will be in the Active and Normal Activity Permitted states, as indicated by the dark-circle start transitions. There are interdependencies between these two parallel state machines and between the state machine on the left and the Temporal state machine, Figure 17 (in 8.1.7).

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

These interdependencies are depicted by the guards (shown within square brackets) that are associated with some state transitions. If a transition has a guard associated with it, then when the assertion within the guard is true, the joined federate may make the associated transition from one state to another. If a transition has only a guard (with no associated service invocation), the joined federate will immediately make the associated transition as soon as the guard is true. As an example of an interdependency between the two parallel state machines depicted in the Joined Federate state, if a joined federate that is in the Active state receives a *Federation Restore Begun* $\dagger$ service invocation, it will transition into the Prepared to Restore state (as indicated by the label on the transition from the Active state to the Prepared to Restore state). Once the joined federate enters the Prepared to Restore state, it also enters the Normal Activity Not Permitted state (as indicated by the guard on the transition from the Normal Activity Permitted state to the Normal Activity Not Permitted state). In other words, the guards impose the following constraints on a joined federate: A joined federate may be in the Normal Activity Permitted state (right side) if and only if it is also in the Active Federate state (left side); and a joined federate may be in the Normal Activity Not Permitted state (right side) if and only if it is also in the Instructed to Save, Saving, Waiting for Federation to Save, Prepared to Restore, Restoring, Waiting for Federation to Restore, or Waiting for Restore to Begin state (left side).

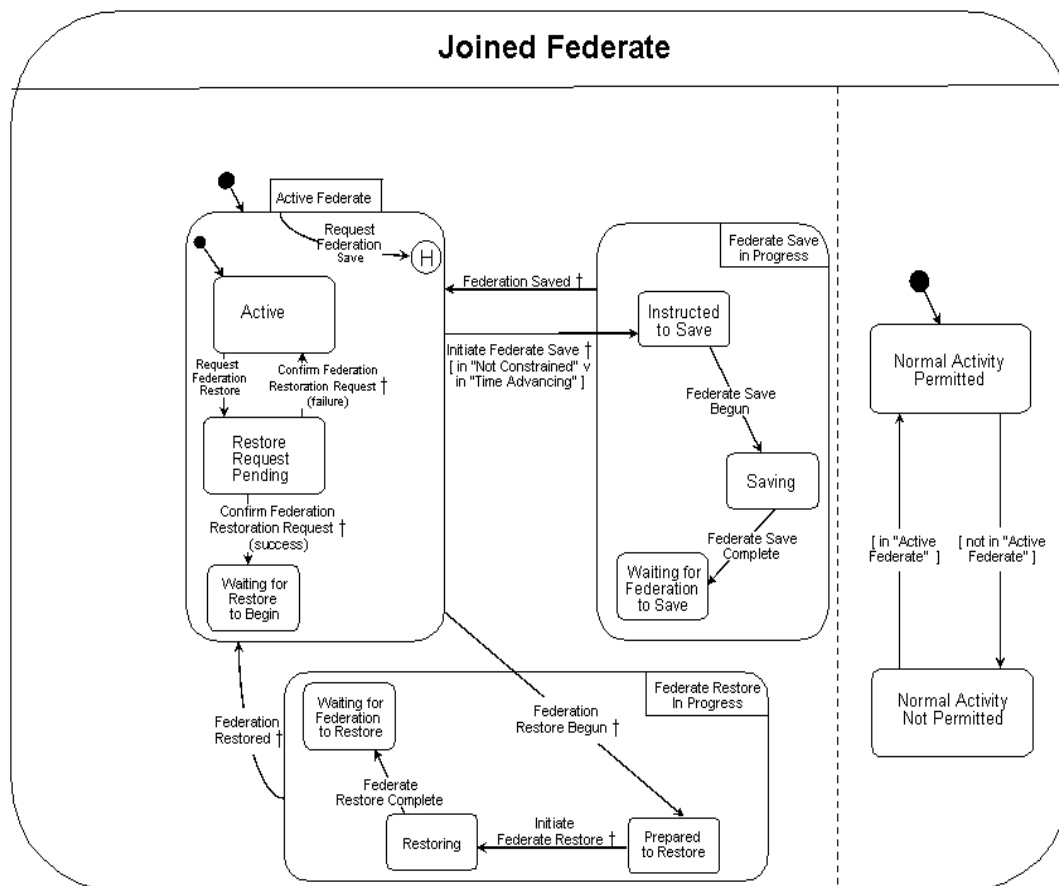


Figure 4—Joined federate state

The interdependency between the state machine on the left and the Temporal state machine, Figure 17 (in 8.1.7), is this: a joined federate that is in the Active state will not receive an invocation of the *Initiate Federate Save* $\dagger$ service unless that joined federate is either in the Not Constrained state or the Time Advancing state. The Not Constrained state and Time Advancing state are depicted in Figure 17. The fact

that these two states related to time management are mentioned in the guard on the transition from the Active state to the Instructed to Save state demonstrates the interdependencies between a joined federate's save/restore state and its Temporal state. Specifically, it indicates that a joined federate shall either not be constrained by time management or be in a position to receive a time advance grant in order for it to receive an invocation of the *Initiate Federate Save* $\dagger$ service.

A state chart notation that needs additional explanation is the shallow history state, depicted by a state labeled **H**. This notation means to enter the most recently visited of all the states at the level of the **H** state or enter the initial state (as indicated by the dark-circle start transition) if this occurrence is the first entry. For example, in Figure 4, there is a shallow history state within the Active Federate state. The transition for the *Request Federation Save* service originates from the outer edge of the Active Federate state, i.e., the transition can occur from any state within the Active Federate state (i.e., Active or Restore Request Pending). The transition terminates at the shallow history state; in other words, on completing the transition, the federate will return to the state within the Active Federate state in which it was previously (i.e., it will return to Active or to Restore Request Pending). Transition from a superstate to a shallow history state is a common idiom denoting that the transition can occur while in any substate of the superstate and will result in the machine's returning to the same state in which it was prior to the transition.

If a joined federate is in the Normal Activity Permitted state, the joined federate may perform normal joined federate activity, such as registering and discovering object instances, publishing and subscribing to object class attributes and interactions, updating and reflecting instance attribute values, sending and receiving interactions, deleting and removing object instances, and requesting or receiving time advance grants. During federation save or restore, a joined federate will enter the Normal Activity Not Permitted state. In this state, the only permitted activities are those performed by the joined federate and with the joined federate (by the RTI) related to the save or restore in progress. Additionally, while in the Normal Activity Not Permitted state, a joined federate may:

- Resign the federation execution or
- Send certain save- or restore-related interactions.

The Normal Activity Permitted state, simple as it may appear in Figure 4, actually contains all of the other states that appear in the first set of state charts that appear subsequently in this document. Together, these state charts formally describe the state of a joined federate from that joined federate's perspective. These state charts are complete in the sense that all transitions shown represent legal operations and transitions that are not shown represent illegal operations. The RTI shall generate exceptions if illegal operations are invoked. The state charts do not include MOM or data distribution management (DDM) activities.

The Normal Activity Permitted state depicted in Figure 4 is enhanced in Figure 5 to identify the major portions of the Joined Federate state. Additionally, the states in Figure 5 are further detailed as follows:

- Temporal state (Figure 17, in 8.1.7)
- States associated with each object class (Figure 10, in 5.1.5)
- States associated with each interaction class (Figure 13, in 5.1.5)

When a federate enters the Joined Federate state, the joined federate will have a Temporal state, object class states, and interaction class states. The joined federate will have an Object Class state for each object class that is defined in the FDD that is associated with the federation execution. Likewise, the joined federate will have an Interaction Class state for each interaction class that is defined in the FDD. A joined federate will be in the Temporal state, in each of these Object Class states, and in each of these Interaction Class states simultaneously (as depicted by the dashed lines separating the state machines within the Normal Activity Permitted state). Time management is detailed in Figure 17. The state of an arbitrary object class is detailed in Figure 10, and the state of an arbitrary interaction class is detailed in Figure 13. Indices, such as “i” in Object Class (i), are used in the names of many state charts to denote that the state chart refers to an arbitrary occurrence of something. These indices are used consistently throughout this specification and are described as follows:

- i Object class designator
- j Attribute designator
- k Object instance designator
- m Interaction class designator
- L Synchronization label
- s sth joined federate; also used as F_s

Another state chart notation that needs additional explanation is the deep history state, depicted by a state labeled H^* . This notation means to enter the most recently visited of all the states at the level of the H^* state or enter the initial state (as indicated by the dark-circle start transition) if this occurrence is the first entry and to do the same for all relevant substates.

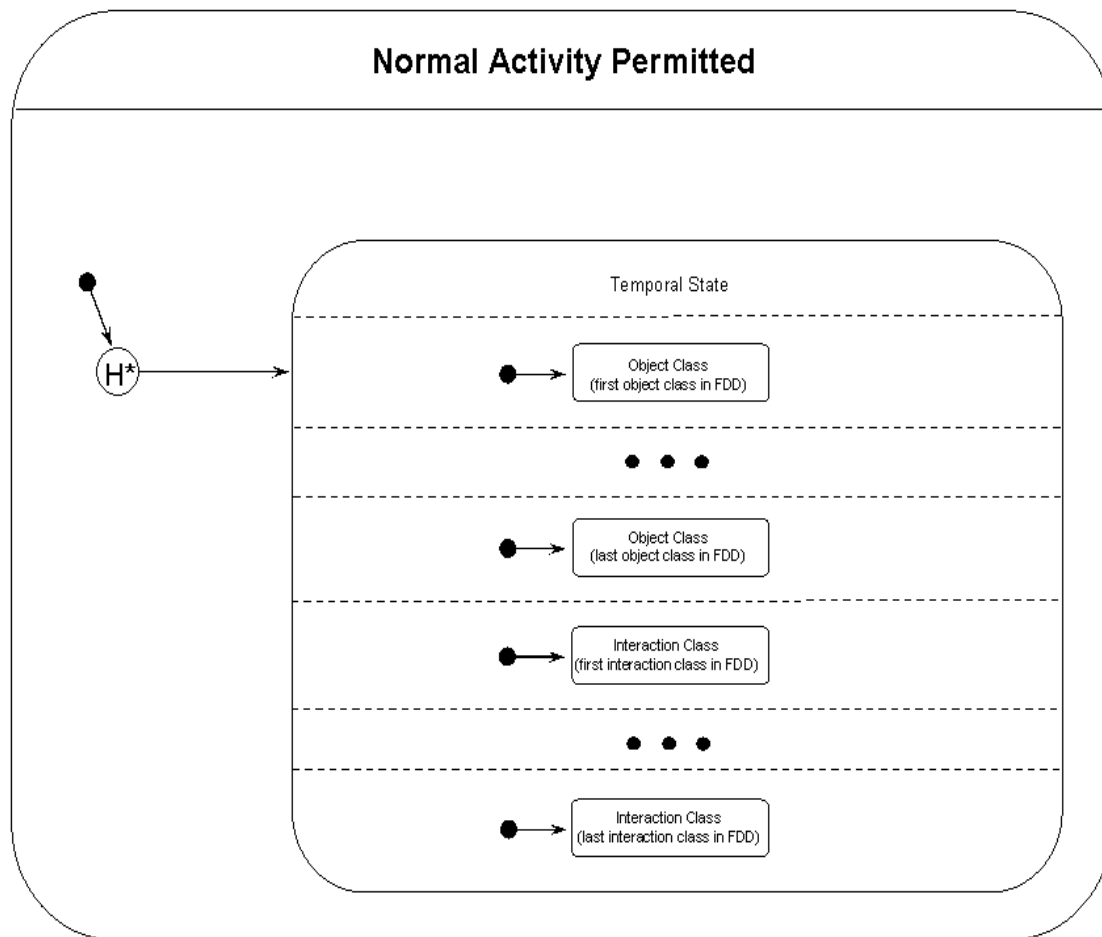


Figure 5—Normal Activity Permitted state

4.1.2 Federation execution save and restore

Any joined federate in the federation execution may initiate a save by invoking the *Request Federation Save* service. If there is no timestamp argument provided with the invocation of this service, the RTI shall instruct all of the joined federates in the federation execution (including the requesting joined federate) to save state by invoking the *Initiate Federate Save* $\dagger$ service at all of these joined federates as soon as possible. If there is a timestamp argument provided, the RTI will invoke the *Initiate Federate Save* $\dagger$ service at each of the time-constrained joined federates when their time position advances to the value provided, and it will invoke the

Initiate Federate Save † service at all nontime-constrained joined federates as soon as all of the time-constrained joined federates are eligible to be instructed to save.

When a joined federate receives an *Initiate Federate Save* † service invocation and subsequently saves its state, it shall use the federation save label (that was specified by the joined federate requesting the save in the *Request Federation Save* service), its federate type (that it specified when it joined the federation execution), and its federate designator (that was returned when it joined the federation execution) to distinguish the saved information.

Once a federation execution save has begun, an RTI shall save all queued messages that have not been delivered to the intended joined federates (e.g., future timestamp order (TSO) messages or receive order (RO) messages that were sent by other joined federates not yet in the Federate Save in Progress state). All saved information (by both the RTI and the joined federates) shall be persistent, i.e., it is stored onto disk or some other persistent medium, and it remains intact even after the federation execution is destroyed.

The saved information can be used at a later date, by some new set of joined federates, to restore all joined federates in the federation execution to the state in which they were when the save was accomplished. The federation can then resume effecting the execution from that saved point. Once a restore has begun, any messages that have not been delivered to the intended joined federates in the restoring federation execution shall be lost. After a restore is complete, the RTI shall deliver all related saved messages as appropriate to the joined federates in the restored federation execution (e.g., TSO messages at the federate's appropriate logical time position or RO messages as soon as possible in wall-clock time).

The set of federates joined to a federation execution when state is restored from a previously saved state need not be the exact set of federates that were joined to the federation execution when the state being restored was saved. The number of federates of each federate type that are joined to the federation execution, however, shall be the same. The federate-type argument supplied in the *Join Federation Execution* service invocation, therefore, is crucial to the save-and-restore process. Declaring a federate to be of a given type shall be equivalent to asserting that the joined federate can be restored using the state information saved by any other joined federate of that type.

There is no requirement that a save taken by one RTI implementation be restorable by another RTI implementation.

4.1.3 Synchronization discussion

Figure 6 depicts the state of a federation execution with respect to synchronization points. In order to fully explain synchronization points, both the state of the RTI and the state of each joined federate are detailed. Because multiple joined federates are involved, each transition within this set of state charts includes an argument denoting to which joined federate a service invocation applies (i.e., an argument of F_s denotes the sth joined federate).

The RTI shall keep track of each synchronization point separately, and the treatment of an arbitrary synchronization point by the RTI is expanded in Figure 7. Not shown in this diagram is how the RTI calculates the synchronization set (shown as *synch_set* within the state chart); this information is detailed in the service descriptions. Use of the synchronization set, however, is depicted in the state chart on the transition from the Synchronizing (s, L) states to the Synch Point Not Registered state. This transition denotes, by use of the universal quantifier over the synchronization set in the guard, that synchronization throughout a federation is achieved only when all joined federates in the synchronization set have concluded their individual synchronizations. A joined federate's individual synchronization, from the RTI's point of view, is detailed in Figure 8.

The final state chart in the set describing synchronization points is Figure 9. This state chart depicts how an arbitrary joined federate keeps track of an arbitrary synchronization point. This diagram includes how a joined federate registers a synchronization point as well as how a joined federate actually synchronizes.

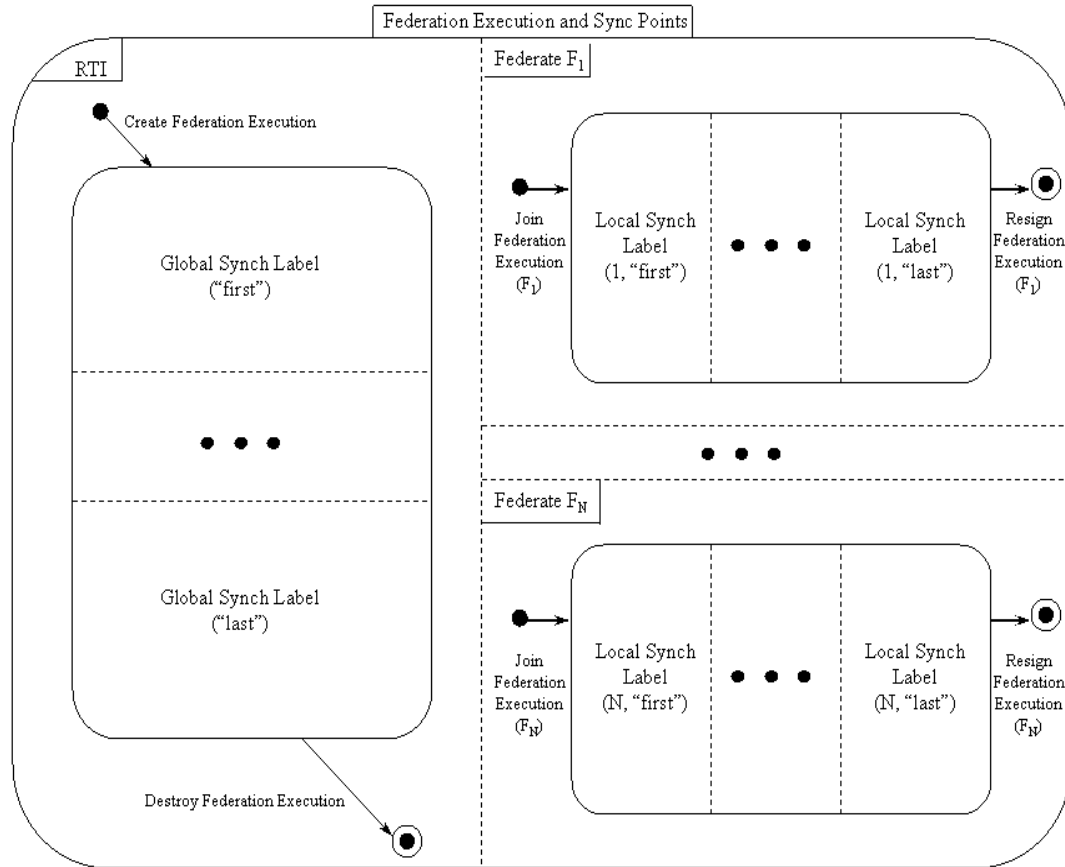


Figure 6—Federation execution and sync points

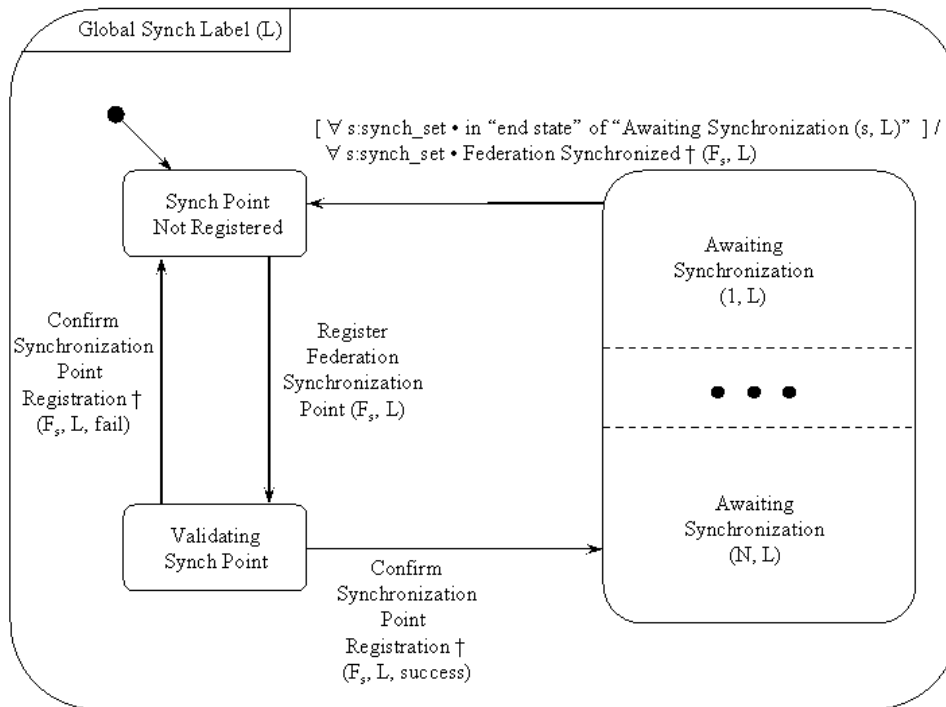


Figure 7—Global synch label (L)

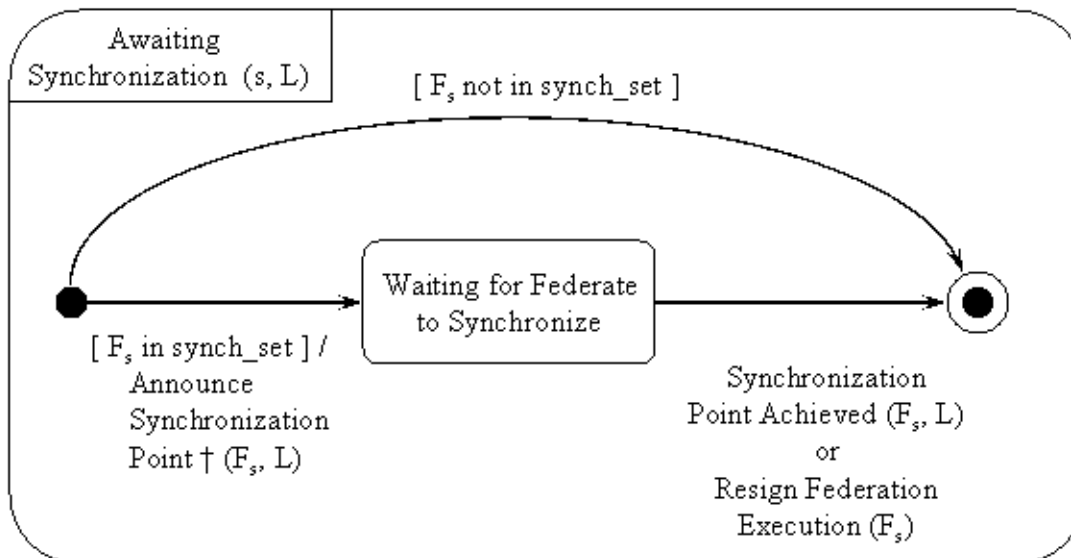


Figure 8—Awaiting synchronization (s, L)

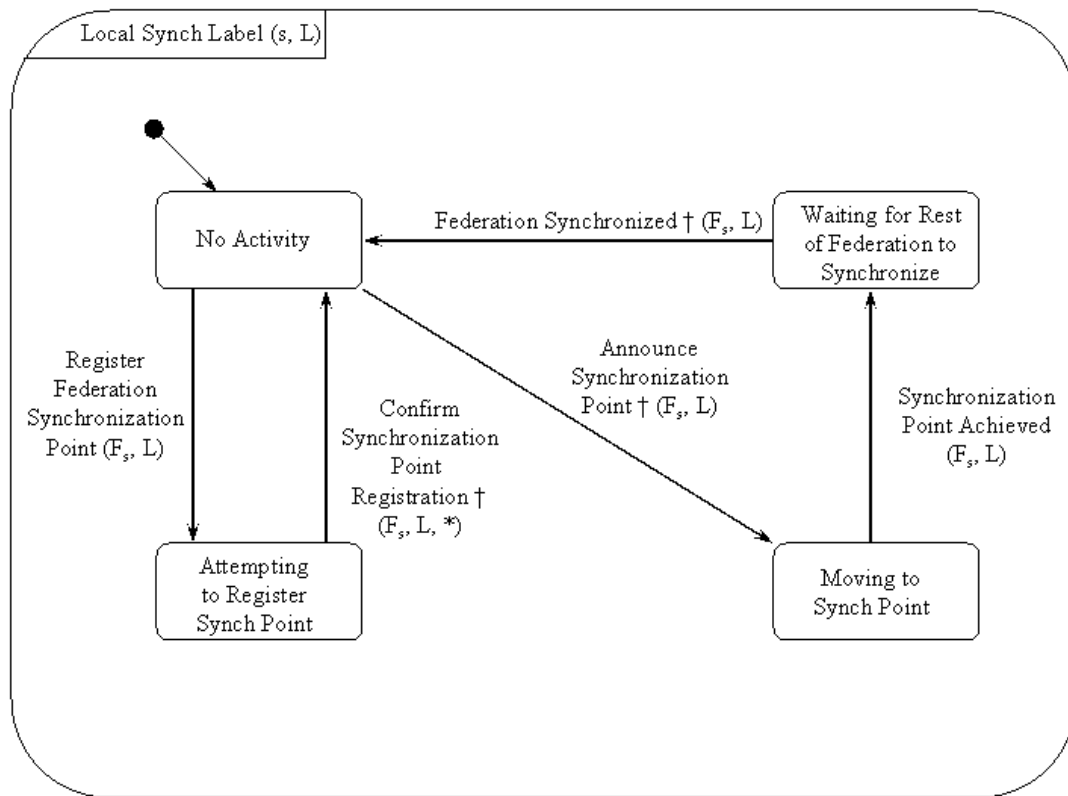


Figure 9—Local synch label (s, L)

4.1.4 FOM Document Data (FDD)

The FDD is data and information extracted and combined from FOM modules in order to support functionality described in this specification. The FOM module format and rules for combination are defined in IEEE Std 1516.2-2025. The FDD that the RTI retains and combines from the FOM modules shall be, at a minimum, the following:

- Object class structure table (complete)
- Interaction class structure table (complete)
- Directed interaction table (complete)
- Attribute table (object, attribute, transportation, and order columns)
- Parameter table (interaction, parameter, transportation, and order columns)
- Dimension table (complete)
- Transportation type table (name and reliability columns)
- Update rate table (name and maximum update rate columns)
- Switches table (complete)

All the names within an FDD shall be case sensitive.

When the *Create Federation Execution* service is invoked, the RTI shall initialize the federation execution using FDD first from the MIM and then from the specified FOM modules. A user-extended MIM may be provided; otherwise, the RTI shall provide the standard MIM automatically.

Additional FOM modules may be provided in the *Join Federation Execution* service invocations. The FOM module designator arguments supplied to the *Create Federation Execution* service and the *Join Federation Execution* service shall designate compliant FOM modules (IEEE Std 1516-2025 and IEEE Std 1516.2-2025). *Create Federation Execution* and *Join Federation Execution* service invocations shall reject, issuing an exception, any entity that is not a compliant FOM module.

4.1.4.1 Rules for combining information from FOM modules

The RTI shall combine the FDD data of the MIM and FOM modules according to the rules for combining FOM data in IEEE Std 1516.2-2025 when the *Create Federation Execution* service is invoked as well as whenever the set of FOM modules has changed due to an invocation of the *Join Federation Execution* service. The RTI shall consider only the FDD when combining FOM modules. After combining switches from all FDD provided during the *Create Federation Execution* service, if a switch remains unspecified, the RTI shall provide a default value for the switch in the current FOM as specified in IEEE Std 1516.2-2025. In case the FDD of the FOM modules provided in such a service call cannot be successfully combined or the invocation fails for any other reason, the FDD provided in the call shall not be included in the current FOM.

The RTI shall reject, issuing an exception, any attempt to load a FOM module for the following cases:

- When trying to define a named object class attribute that has different transportation or order when compared to an already existing named object class attribute with the same name and belonging to the same object class in the current FOM.
- When trying to define a named interaction class that has different transportation or order when compared to an already existing named interaction class with the same qualified name in the current FOM.
- When trying to define a named dimension that has different upper bound or value when unspecified when compared to an already existing named dimension in the current FOM.
- When trying to define a named transportation type that has a different reliability when compared to an already existing named transportation type in the current FOM. (Comment note - description is not part of FDD nor should it be and thus cannot be used for rejection.)
- When trying to define a named update rate that has a different maximum rate when compared to an already existing named update rate in the current FOM.
- When trying to define a named switch that has a different definition when compared to an already existing named switch in the current FOM.
- When the MIM or FOM module is noncompliant.

If a FOM module or MIM contains any definition that is rejected by the RTI, the RTI shall reject the FOM module or MIM in its entirety and shall not add any element to the FDD.

The union of all FOM modules in a given *Create Federation Execution* or *Join Federation Execution* service when added to the previously existing current FOM shall not result in any object class or interaction class not having a full definition. If this is the case, the entire service invocation shall fail. In case any of the FOM modules provided was rejected by the RTI, the entire service invocation shall also fail.

After a *Create Federation Execution* or *Join Federation Execution* service has completed successfully, all the provided FOM modules shall be available in the current FOM, and the corresponding MOM information shall be updated to reflect this.

Furthermore, any FDD provided as per above, once available in the federation execution, shall be available for the remaining life span of the federation execution.

4.1.4.2 Access to Current FOM information

The RTI shall provide access to the current FDD, the content of individual FOM modules, as well as the MIM. The following MOM interactions are provided:

- a) For the federation execution:
 - 1) The attribute HLAFOMmoduleDesignatorList provides a list of designators for the FOM modules that specified the current FOM. Only FOM modules that resulted in the addition or modification of one or more classes, directed interactions, dimensions, transportations, update rates, or switches shall be included. Specifically, if several FOM modules with identical content have been specified, only the first shall be included and then only if it satisfies the preceding requirement.
 - 2) The attribute HLAfederation.HLAMIMDesignator provides the designator for the MIM used in the current FOM.
 - 3) The attribute HLAcurrentFDD provides the contents of the current FDD.
 - 4) The interactions HLArequestFOMmoduleData and HLAreportFOMmoduleData provide the contents of a specified FOM module.
 - 5) The interactions HLArequestMIMData and HLAreportMIMData provides the content of the MIM.
- b) For each federate:
 - 1) The attribute HLAfederate.HLAFOMmoduleDesignatorList provides a list of designators for the FOM modules that were specified by that federate when invoking the *Create Federation Execution* and *Join Federation Execution* service.
 - 2) The interactions HLArequestFOMmoduleData and HLAreportFOMmoduleData provide the contents of a specified FOM module.

A FOM module designator shall be equal to the string (file name or URL) as supplied as part of the *Create Federation Execution* or *Join Federation Execution* service invocation. The FOM module designator for the HLAstandardMIM as supplied automatically by the RTI shall be “HLAstandardMIM.”

4.1.5 Logical time implementation

The RTI and joined federates of a federation execution shall share a common implementation of logical time in order to have meaningful interactions involving logical time. No universal implementation of logical time has been proposed that will satisfy all federation executions. When a federation execution is created, a logical time implementation shall be identified. RTIs shall provide two default implementations of logical time, one integer and one floating point. The federate creating a federation execution may provide its own implementation of logical time for use in the federation execution.

4.1.6 Fault tolerance support

A fault shall have occurred if at any point during a federation execution an unexpected event prevents one or more joined federates in the federation from executing in an HLA-compliant manner. The criteria and detection methods of a fault depend on the technical implementation of the RTI and are not part of the HLA standard. A fault may result in one or more federates being disconnected or lost from the federation. Such a federate shall be referred to as a lost federate.

A fault may divide the federation into multiple independent segments, each having the potential to continue as an HLA-compliant federation execution. In this situation, the RTI shall select exactly one such segment to