

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

Sponsor

**Simulation Interoperability Standards Committee**  
of the  
**IEEE Computer Society**

Approved 21 September 2000

**IEEE-SA Standards Board**

**Abstract:** The High Level Architecture (HLA)—Object Model Template (OMT) specification defines the format and syntax (but not content) of HLA object models. Simulations are abstractions of the real world, and no one simulation can solve all of the functional needs for the modeling and simulation community. It is anticipated that advances in technology will allow for new and different modeling and simulation (M&S) implementations within the framework of the HLA. The standards contained in this architecture are interrelated and need to be considered as a product set, as a change in one is likely to have an impact on the others. As such, the HLA is an integrated approach that has been developed to provide a common architecture for simulation.

**Keywords:** architecture, class attribute, data distribution management, federate, federation, federation execution, federation object model, framework, high level architecture, instance attribute, instance attribute ownership, interaction class, joined federate, object class, object model template, rules, runtime infrastructure, simulation object model, time-constrained, time-regulating

---

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

Copyright © 2001 by the Institute of Electrical and Electronics Engineers, Inc.  
All rights reserved. Published 9 March 2001. Printed in the United States of America.

*Print:* ISBN 0-7381-2623-3 SH94884  
*PDF:* ISBN 0-7381-2624-1 SS94884

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

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

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

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

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

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

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

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

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

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

IEEE is the sole entity that may authorize the use of certification marks, trademarks, or other designations to indicate compliance with the materials set forth herein.

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

# Introduction

(This introduction is not part of IEEE Std 1516.2-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification.)

This document has been developed to record an international standard for the HLA. It serves as one of three related standards for the HLA. It defines the format and syntax for recording information in HLA object models.

## Participants

At the time this standard was completed, the working group on HLA, sponsored by the Simulation Interoperability Standards Committee (SISC) of the IEEE Computer Society, had the following membership:

**Susan F. Symington, *Chair***  
**Katherine L. Morse, *Vice Chair***  
**Mikel Petty, *Secretary***

Stephen Bachinsky  
Joe Batman  
Brian Beebe  
Keith Briggs  
Jim Calvin  
Richard M. Fujimoto  
Deborah Fullford  
Allison Griffin

James H. Hammond  
James Ivers  
Andreas Kemkes  
John F. Kramer  
Fredrick S. Kuhl  
Gary M. Lightner  
Reed Little  
Rodney Long  
Margaret Loper  
Robert R. Lutz

Jeffrey M. Nielsen  
Marnie R. Salisbury  
Randy Saunders  
Roy Scrudder  
David W. Seidel  
Robert C. Turrell  
Richard Weatherly  
Annette Wilson

The working group acknowledges the following individuals who also contributed to the preparation of this standard:

Phillippe Annic  
Tobin Bergen-Hill  
Dominique Canazzi

Judith S. Dahmann  
Peter Hoare  
Mikael Karlsson

Michael Mazurek  
J. Russell Noseworthy  
Graham Shanks

The following members of the balloting committee voted on this standard:

|                      |                     |                         |
|----------------------|---------------------|-------------------------|
| Dale E. Anglin       | Robert J. Howard    | Marnie R. Salisbury     |
| Stephen Bachinsky    | Ken Hunt            | Randy Saunders          |
| Michael R. Bachmann  | Kyle Isakson        | Roy Scrudder            |
| David L. Barton      | James Ivers         | David W. Seidel         |
| Brian Beebe          | Andreas Kemkes      | Richard J. Severinghaus |
| Robert P. Bennett    | John F. Kramer      | Joseph F. Seward        |
| Christina L. Bouwens | Frederick S. Kuhl   | Sean Sharp              |
| Keith Briggs         | Tom Lake            | Steven M. Sheasby       |
| Richard Briggs       | Gary M. Lightner    | John W. Sheppard        |
| Danny Cohen          | Paul R. Little      | Allen H. Skees          |
| Glenn E. Conrad      | Reed Little         | Col. Mark Smith         |
| Mark Crooks          | Bjorn Lofstrand     | Susan D. Solick         |
| Dannie Cutts         | Margaret Loper      | Sandra Swearingen       |
| Judith S. Dahmann    | Robert R. Lutz      | Susan F. Symington      |
| Steven Drake         | Jayne Lyons         | Donald Theune           |
| Laura E. Feinerman   | Theodore F. Marz    | William V. Tucker       |
| Jeff E. Fischer      | Mark McAuliffe      | Grant R. Tudor          |
| Richard M. Fujimoto  | David R. McKeeby    | John A. Tufarolo        |
| Deborah Fullford     | Duncan C. Miller    | Robert C. Turrell       |
| Brian F. Goldiez     | Katherine L. Morse  | Andrew J. Ventre        |
| Victor M. Gonzalez   | Thomas H. Mullins   | William F. Waite        |
| James H. Hammond     | Jeffrey M. Nielsen  | Charles E. Walters      |
| Jack G. Harrington   | Michael J. O'Conner | Richard Weatherly       |
| Robert V. Head Jr.   | John Peng           | Chris C. Wertman        |
| Callie M. Hill       | Paul Perkinson      | Annette Wilson          |
| Ronald C. Hofer      | Hung Q. Phan        | Douglas D. Wood         |
| James W. Hollenbach  | David R. Pratt      | Philomena M. Zimmerman  |
| Margaret M. Horst    | Paul M. Reeves      | William H. Zimmerman    |
|                      | David Roberts       |                         |

When the IEEE-SA Standards Board approved this standard on 21 September 2000, it had the following membership:

**Donald N. Heirman**, *Chair*  
**James T. Carlo**, *Vice Chair*  
**Judith Gorman**, *Secretary*

|                    |                       |                    |
|--------------------|-----------------------|--------------------|
| Satish K. Aggarwal | James H. Gurney       | James W. Moore     |
| Mark D. Bowman     | Richard J. Holleman   | Robert F. Munzner  |
| Gary R. Engmann    | Lowell G. Johnson     | Ronald C. Petersen |
| Harold E. Epstein  | Robert J. Kennelly    | Gerald H. Peterson |
| H. Landis Floyd    | Joseph L. Koepfinger* | John B. Posey      |
| Jay Forster*       | Peter H. Lips         | Gary S. Robinson   |
| Howard M. Frazier  | L. Bruce McClung      | Akio Tojo          |
| Ruben D. Garzon    | Daleep C. Mohla       | Donald W. Zipse    |

\*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Alan Cookson, *NIST Representative*  
Donald R. Volzka, *TAB Representative*

Jennifer McClain Longman  
*IEEE Standards Project Editor*

# Contents

|         |                                                             |     |
|---------|-------------------------------------------------------------|-----|
| 1.      | Overview.....                                               | 1   |
| 1.1     | Scope.....                                                  | 1   |
| 1.2     | Purpose.....                                                | 1   |
| 1.3     | Background.....                                             | 1   |
| 2.      | References.....                                             | 3   |
| 3.      | Definitions, abbreviations, acronyms, and conventions ..... | 3   |
| 3.1     | Definitions .....                                           | 3   |
| 3.2     | Abbreviations and acronyms .....                            | 15  |
| 3.3     | Conventions .....                                           | 16  |
| 4.      | HLA OMT components.....                                     | 17  |
| 4.1     | Object model identification table.....                      | 18  |
| 4.2     | Object class structure table .....                          | 20  |
| 4.3     | Interaction class structure table.....                      | 24  |
| 4.4     | Attribute table .....                                       | 27  |
| 4.5     | Parameter table .....                                       | 31  |
| 4.6     | Dimension table.....                                        | 33  |
| 4.7     | Time representation table.....                              | 36  |
| 4.8     | User-supplied tag table .....                               | 38  |
| 4.9     | Synchronization table .....                                 | 39  |
| 4.10    | Transportation type table .....                             | 40  |
| 4.11    | Switches table .....                                        | 41  |
| 4.12    | Datatype tables.....                                        | 43  |
| 4.13    | Notes table .....                                           | 59  |
| 5.      | FOM/SOM lexicon .....                                       | 60  |
| 5.1     | Purpose/background.....                                     | 60  |
| 5.2     | Object class definition table.....                          | 60  |
| 5.3     | Interaction class definition table .....                    | 61  |
| 5.4     | Attribute definition table.....                             | 61  |
| 5.5     | Parameter definition table.....                             | 62  |
| Annex A | (informative) Table entry notation.....                     | 64  |
| Annex B | (normative) Common normalization functions.....             | 65  |
| Annex C | (normative) OMT data interchange format (DIF).....          | 67  |
| Annex D | (informative) OMT DIF SOM Example .....                     | 79  |
| Annex E | (informative) OMT DIF FOM Example .....                     | 92  |
| Annex F | (informative) Bibliography .....                            | 129 |



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

## 1. Overview

### 1.1 Scope

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

### 1.2 Purpose

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

### 1.3 Background

#### 1.3.1 Object model template rationale

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

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

HLA object models may be used to describe an individual federation member (federate), creating an HLA Simulation Object Model (SOM), or to describe a named set of multiple interacting federates (federation), creating a Federation Object Model (FOM). In either case, the primary objective of the HLA Object Model Template (OMT) is to facilitate interoperability among simulations and reuse of simulation components. All

discussion of HLA object models in this document applies to both SOMs and FOMs unless explicitly stated otherwise.

### 1.3.2 Federation object models

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

### 1.3.3 Simulation object models

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

The HLA OMT formats described in this document are generally applicable to either FOMs or SOMs. Thus, SOMs are also characterized by their objects, attributes, interactions, and parameters. The primary benefit from the common utilization of the OMT formats for FOMs and SOMs is that it provides a common frame of reference for describing object models in the HLA community. In some cases, this commonality may even allow SOM components to be integrated as “piece parts” in a FOM, facilitating FOM construction.

### 1.3.4 Relationship of HLA and object-oriented concepts

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

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

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

HLA object class as a template. Each attribute contained by an HLA object instance is called an instance attribute.

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

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

## 2. References

This standard shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision shall apply:

ANSI Std X3.4-1986 (R1997), Information Systems—Coded Character Sets—7 Bit American National Standard Code Information Interchange (7-Bit ASCII).<sup>1</sup>

IEEE Std 754-1985 (R1990), IEEE Standard for Binary Floating-Point Arithmetic.<sup>2</sup>

IEEE Std 1516-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules.

IEEE Std 1516.1-2000, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification.

REC-XML-19980210, Extensible Markup Language (XML) 1.0, W3C Recommendation, Feb. 1998.<sup>3</sup>

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

## 3. Definitions, abbreviations, acronyms, and conventions

### 3.1 Definitions

For the purposes of this standard, the following terms and definitions apply. The *Authoritative Dictionary of IEEE Standards Terms* [B2]<sup>4</sup> should be referred for terms not defined in this clause.

**3.1.1 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.

<sup>1</sup>ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

<sup>2</sup>IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (<http://standards.ieee.org/>).

<sup>3</sup>This document is available at <http://www.w3.org/>.

<sup>4</sup>The numbers in brackets correspond to those of the bibliography in Annex F.

**3.1.2 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 this 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.

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

**3.1.4 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**.

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

**3.1.6 available dimensions:**

- a) Pertaining to an attribute: the dimensions associated with the class attribute in the Federation Object Model (FOM). The available dimensions of an instance attribute are the available dimensions of the corresponding class attribute.
- b) Pertaining to an interaction class: the dimensions associated with that interaction class in the FOM. The available dimensions of a sent interaction are the available dimensions of the interaction class specified in the *Send Interaction With Regions* service invocation.

NOTE—See 9.1 in IEEE Std 1516.1-2000.

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

**3.1.8 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. The term candidate discovery class pertains to object instances only.

**3.1.9 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. The term candidate received class pertains to interactions only.

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

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

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

**3.1.13 coadunate:** To attach an object instance handle (and possibly an object instance name) to an object instance.

**3.1.14 collection:** A set in which an element may occur multiple times (this corresponds to the mathematical notion of a bag).

**3.1.15 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.

**3.1.16 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).

NOTE—See IEEE Std 1516.2-2000.

**3.1.17 constrained set of pairs:** A group of pairs in which no two pairs have the same first component (this 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.

**3.1.18 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.

**3.1.19 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.

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

**3.1.21 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.

**3.1.22 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.

**3.1.23 default range:** A range lower bound and a range upper bound, defined in the Federation Object Model Document Data (FDD) and specified in terms of [0, the dimension's upper bound), for a dimension.

**3.1.24 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 Document Data (FDD). The bounds of each of these ranges are [0, the range's dimension's upper bound). There is no way for a federate to refer to the default region.

NOTE—See 9.1 in IEEE Std 1516.1-2000.

**3.1.25 designator:** Some arguments (mostly identifiers) to services may have different views (implementations), depending on a particular programming language, application programmer's interface (API). For clarity, service descriptions refer to a generic view of the arguments, known as a *designator*.

**3.1.26 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 upperbound as specified in the Federation Object Model Document Data (FDD)]. This interval provides a single basis for communication of all dimension-related data with an RTI. All normalized intervals communicated to the RTI will be subsets of this interval.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

**3.1.27 discover:** To receive an invocation of the *Discover Object Instance* <sup>†5</sup> service for a particular object instance.

NOTE—See 6.5 of IEEE Std 1516.1-2000.

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

NOTE—See 5.1.2 in IEEE Std 1516.1-2000.

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

**3.1.30 federate:** An application that may be or is currently coupled with other software applications under a Federation Object Model Document Data (FDD) and a runtime infrastructure (RTI). *See also:* **federate application** and **joined federate**.

**3.1.31 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**.

**3.1.32 federation:** A named set of federate applications and a common Federation Object Model that are used as a whole to achieve some specific objective.

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

**3.1.34 Federation Object Model (FOM):** A specification defining the information exchanged at runtime to achieve a given set of federation objectives. This includes object classes, object class attributes, interaction classes, interaction parameters, and other relevant information.

**3.1.35 Federation Object Model (FOM) Document Data (FDD):** The data and information in an FOM document that is used by the *Create Federation Execution* service to initialize a newly created federation execution

NOTE—See IEEE Std 1516.2-2000.

**3.1.36 handle:** An identifier originated/created by the runtime infrastructure (RTI) that is federation execution-wide unique and unpredictable.

<sup>5</sup>All RTI initiated services are denoted with a † (printer's dagger) after the service name.

**3.1.37 High Level Architecture (HLA) time axis:** A totally ordered sequence of values in which each value typically represents an HLA instant of time in the physical system being modeled. For any two points  $T_1$  and  $T_2$  on the time axis, if  $T_1 < T_2$ ,  $T_1$  represents an instant of time that occurs before the instant represented by  $T_2$ .

**3.1.38 in scope:** Of or pertaining to an instance attribute of an object instance for which

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

NOTE—See 6.1 of IEEE Std 1516.1-2000.

**3.1.39 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).

**3.1.40 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).

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

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

**3.1.43 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.

**3.1.44 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.

**3.1.45 joined federate:** A member of a federation execution, actualized by a federate application invoking the *Join Federation Execution* service as prescribed in IEEE Std 1516.1-2000. *See also:* **federate application**.

**3.1.46 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.

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

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

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

**3.1.50 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), and
- c) Control of the functioning of the RTI, the federation execution, and the individual joined federates

**3.1.51 message:** A change of object instance attribute value, an interaction, or a deletion of an existing object instance, often associated with a particular point on the High Level Architecture (HLA) time axis, as denoted by the associated time stamp.

**3.1.52 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. A template for a set of characteristics that is common to a group of object instances. These characteristics correspond to the class attributes that individual federates may publish and to which other federates may subscribe.

**3.1.53 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.

**3.1.54 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.

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

**3.1.56 order type:** A runtime infrastructure (RTI) provided 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.

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

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

region set of the instance attribute's corresponding class attribute at the known class of the instance attribute at the subscribing joined federate.

NOTE—See 6.1 of IEEE Std 1516.1-2000.

### 3.1.58 overlap:

- a) Pertaining to region sets: Two region sets overlap if there is a region in each set, such that the two regions overlap.
- b) Pertaining to regions: If two regions have at least one dimension in common, they overlap if all ranges of dimensions that are contained in both regions overlap pairwise. If two regions do not have any dimensions in common, they do not overlap.
- c) Pertaining to ranges: Two ranges  $A = [a_{\text{lower}}, a_{\text{upper}})$  and  $B = [b_{\text{lower}}, b_{\text{upper}})$  overlap, if and only if either  $a_{\text{lower}} = b_{\text{lower}}$  or  $(a_{\text{lower}} < b_{\text{lower}} < a_{\text{upper}})$ .

NOTE—See 9.1 of IEEE Std 1516.1-2000.

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

**3.1.60 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 Document Data (FDD).

**3.1.61 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.

**3.1.62 parameter:** A named characteristic of an interaction.

**3.1.63 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, time stamp (if present), and set of sent region designators (if present). A passel is a message.

**3.1.64 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, but 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.

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

**3.1.66 published:**

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

NOTE—See 5.1.2 of IEEE Std 1516.1-2000.

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

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

NOTE—See 5.1.2 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1 of IEEE Std 1516.1-2000.

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

NOTE—See IEEE 9.1 of Std 1516.1-2000.

**3.1.71 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 time stamp value will be provided with the message if one was specified when the message was sent, but that time stamp has no bearing on message receipt order.

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

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

NOTE—See 6.7 of IEEE Std 1516.1-2000.

**3.1.75 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.

**3.1.76 region:** A generic term that refers to either a region specification or a region realization. If not using Data Distribution Management (DDM), region arguments may be ignored.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1.1 of IEEE Std 1516.1-2000.

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

NOTE—See 6.4 of IEEE Std 1516.1-2000.

**3.1.80 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.

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

**3.1.82 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.

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

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

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

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

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

NOTE—See 5.1.3 of IEEE Std 1516.1-2000.

**3.1.88 set:** A group of elements in which each element occurs at most once (this 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.

**3.1.89 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 standard format in which SOMs are expressed facilitates determination of the suitability of federates for participation in a federation.

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

NOTE—See 9.1.2 of IEEE Std 1516.1-2000.

**3.1.91 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.

**3.1.92 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.

**3.1.93 subscribed:**

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

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

NOTE—See 5.1.2 and 5.6 of IEEE Std 1516.1-2000.

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

NOTE—See 9.8 of IEEE Std 1516.1-2000.

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

NOTE—See 5.1.3 and 5.8 of IEEE Std 1516.1-2000.

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

NOTE—See 9.10 of IEEE Std 1516.1-2000.

**3.1.98 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** and **used for subscription of an interaction class**.

**3.1.99 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 only inherit from a single class at the next highest level of the class hierarchy).

**3.1.100 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, if they are successful, thereby synchronize their respective processing at that point.

**3.1.101 time advancing state:** A joined federate may advance its logical time only by requesting a time advancement from the runtime infrastructure (RTI) via one of the following services:

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

The joined federate's logical time will not actually be advanced until the RTI responds with a *Time Advance Grant* service invocation at that joined federate. During the interval between a request to advance its logical time and the corresponding grant, the joined federate is in the time advancing state.

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

NOTE—See 8.1 of IEEE Std 1516.1-2000.

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

NOTE—See 8.1 of IEEE Std 1516.1-2000.

**3.1.104 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.

**3.1.105 time stamp (of message or save):** The value of the time stamp argument provided to the relevant service invocation.

**3.1.106 time stamp order (TSO):** An ordering of messages provided by a runtime infrastructure (RTI) for joined federates making use of time management services and messages containing time stamps. Messages having different time stamps are said to be delivered in TSO if for any two messages M1 and M2 (time stamped 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 time stamp will be delivered in an arbitrary order (i.e., no tie-breaking mechanism is provided by an RTI).

**3.1.107 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.

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

NOTE—See 9.1.3 of IEEE Std 1516.1-2000.

**3.1.109 update:** Invoke the *Update Attribute Values* service for one or more instance attributes.

NOTE—See 6.6 of IEEE Std 1516.1-2000.

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

**3.1.111 used for sending:**

- a) Pertaining to a region that, along with the specified interaction class designator, is being used as an argument in the *Send Interaction With Regions* service.
- b) Pertaining to the default region when the specified interaction class designator is being used as an argument in the *Send Interaction* service.

NOTE—See 9.1 of IEEE Std 1516.1-2000.

**3.1.112 used for subscription of a class attribute:**

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

NOTE—See 9.1 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1 of IEEE Std 1516.1-2000.

**3.1.113 used for subscription of an interaction class:**

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

NOTE—See 9.1 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1 of IEEE Std 1516.1-2000.

**3.1.114 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; and the region has not subsequently been used along with the specified object instance designator as an argument in the *Unassociate Regions For Updates* service; nor has the joined federate subsequently lost ownership of the specified instance attribute(s).

NOTE—See 9.1 of IEEE Std 1516.1-2000.

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

NOTE—See 9.1 of IEEE Std 1516.1-2000.

**3.2 Abbreviations and acronyms**

The following abbreviations and acronyms pertain to this standard.

|       |                                                 |
|-------|-------------------------------------------------|
| API   | application programmer's interface              |
| BNF   | Backus Naur Form                                |
| DDM   | Data Distribution Management                    |
| DIF   | data interchange format                         |
| DM    | Declaration Management                          |
| DTD   | Document Type Declaration                       |
| FDD   | FOM Document Data                               |
| FOM   | Federation Object Model                         |
| FQR   | <i>Flush Queue Request</i>                      |
| GALT  | Greatest Available Logical Time                 |
| HLA   | High Level Architecture                         |
| LITS  | Least Incoming Time Stamp                       |
| MOM   | Management Object Model                         |
| M&S   | modeling and simulation                         |
| NA    | not applicable                                  |
| NMR   | <i>Next Message Request</i>                     |
| NMRA  | <i>Next Message Request Available</i>           |
| 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                          |
| SISC  | Simulation Interoperability Standards Committee |
| SOM   | Simulation Object Model                         |
| TAR   | <i>Time Advance Request</i>                     |
| TARA  | <i>Time Advance Request Available</i>           |
| TRADT | time representation abstract datatype           |
| TSO   | time stamp order                                |
| XML   | eXtensible Markup Language                      |

### 3.3 Conventions

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

#### 3.3.1 Names

Names in object models shall adhere to XML naming conventions. XML conventions require that names be constructed from a combination of letters, digits, hyphens, colons, full stops (periods), and underscores with no spaces or other breaking characters (e.g., tabs, carriage returns, etc.). In HLA object models, allowable names shall be further restricted as follows:

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

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

- Object class names
- Interaction class names
- Attribute names
- Parameter names
- Datatype names
- Enumerated datatype enumerators
- Enumerated datatype values
- Fixed record field names
- Variant record alternative names
- Basic data representation names
- Dimension names
- Transportation type names
- Synchronization point labels
- Note identifying labels

## 4. HLA OMT components

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

The OMT consists of the following components:

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

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

The final HLA OMT component, the FOM/SOM lexicon, is essential to ensure that the semantics of the terms used in an HLA object model are understood and documented.

The HLA MOM specifies a designated set of information elements that are associated with federation executions. Implementation of the MOM information elements as specified in IEEE Std 1516.1-2000 provides a mechanism for management of federation executions using existing HLA services. Inclusion of the MOM is required for all FOMs. In addition, federates that have an inherent capability to interact with or extend the MOM shall include the MOM in their SOM. Inclusion of the MOM means the inclusion of all tables in Clause 11 of IEEE Std 1516.1-2000, along with any applicable MOM extensions.

Any FOM or SOM that fully conforms to all of the rules and constraints stated in this specification is a *compliant object model*.

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

## 4.1 Object model identification table

### 4.1.1 Purpose/background

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

### 4.1.2 Table format

The information that identifies HLA object models shall be provided in a simple two-column table. The structure to be used to describe this information is provided in Table 1.

The first column (Category) specifies the categories of data that shall be provided in this table. The second column (Information) shall specify the required information. Entries shall be provided for all rows except References and Other; “NA” shall be entered for these rows if no information is appropriate. Definitions of the categories and required information are as follows:

- a) *Name*: This field shall specify the name assigned to the object model.
- b) *Type*: This field shall specify the type that the object model represents; valid values are
  - *FOM*: the object model describes a federation
  - *SOM*: the object model describes a federate
- c) *Version*: This field shall specify the version identification assigned to the object model.
- d) *Modification Date*: This field shall specify the latest date on which this version of the object model was created or modified. The modification date shall be specified in the format “YYYY-MM-DD” (e.g., 1999-04-15).
- e) *Purpose*: This field shall specify the purpose for which the federate or federation was developed.

- f) *Application Domain*: This field shall specify the type or class of application to which the federate or federation applies. The following values may be used for this field, although other values are also valid:
- Analysis
  - Training
  - Test and Evaluation
  - Engineering
  - Acquisition
- g) *Sponsor*: This field shall specify the organization that sponsored the development of the federate or federation
- h) *POC*: This field shall specify the point of contact for information on the federate or federation and the associated object model, including an honorific (e.g., Dr., Ms., etc.) or rank, first name, and last name where appropriate
- i) *POC Organization*: This field shall specify the organization with which the POC is affiliated
- j) *POC Telephone*: This field shall specify the telephone number for the POC, including the international telephone code for the POC's country
- k) *POC E-mail*: This field shall specify the e-mail address of the POC
- l) *References*: This field shall specify pointers to additional sources of information. For example, documents that describe save/restore semantics, time management strategies, or other relevant federation policy decisions may be identified through this field. "NA" shall be entered when no such references are appropriate.
- m) *Other*: This field shall specify other data deemed relevant by the author of the object model. "NA" shall be entered when no such data are appropriate.

**Table 1—Object model identification table**

| Category           | Information          |
|--------------------|----------------------|
| Name               | <name>               |
| Type               | <type>               |
| Version            | <version>            |
| Modification Date  | <date>               |
| Purpose            | <purpose>            |
| Application Domain | <application domain> |
| Sponsor            | <sponsor>            |
| POC                | <poc>                |
| POC Organization   | <poc organization>   |
| POC Telephone      | <poc telephone>      |
| POC Email          | <poc email>          |
| References         | <references>         |
| Other              | <other>              |

### 4.1.3 Inclusion criteria

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

### 4.1.4 Example

Table 2 shows a simple example of the object model identification table. In this example, the *Purpose* entry is shorter (by design) than entries that would be found in this table for most practical applications. The subclauses that follow provide examples of each OMT table for the notional SOM identified in this table.

**Table 2—Object model identification table example**

| Category           | Information                                                   |
|--------------------|---------------------------------------------------------------|
| Name               | RestaurantExample                                             |
| Type               | SOM                                                           |
| Version            | 1.0 Alpha                                                     |
| Modification Date  | 1998-01-01                                                    |
| Purpose            | Example of an object model for a restaurant federate          |
| Application Domain | Restaurant operations                                         |
| Sponsor            | Federated foods                                               |
| POC                | Mr. Joseph Doe                                                |
| POC Organization   | Joe's Place                                                   |
| POC Telephone      | 1-977-555-1234                                                |
| POC E-mail         | doej@fedfoods.com                                             |
| References         | www.fedfoods.com/restsim.html                                 |
| Other              | See Mobil International Restaurant Guide for more information |

## 4.2 Object class structure table

### 4.2.1 Purpose/background

The object class structure of an HLA object model is defined by a set of relations among classes of objects from the simulation or federation domain. An HLA object class is a collection of objects with certain characteristics or attributes in common. Each of the individual objects that are realizations of a class is said to be an instance of that class.

An HLA object class structure shall be defined by hierarchical relationships among classes of objects. Class relationships shall be represented by the inclusion of the associated class names in the appropriate columns of the object class structure table. Relationships among classes at nonadjacent levels of a class hierarchy can be derived through transitivity: If A is a superclass of B, and B is a superclass of C, A is also a superclass of C. Superclass and subclass play inverse roles in these relations: If A is a superclass of B, B is a subclass of A.

Subclasses can be considered to be specializations, or refinements, of their superclasses. Subclasses shall always inherit the attributes of their superclasses, and they may possess additional attributes to provide the desired specialization. For instance, suppose a "Circle" class is defined as a subclass of a "Figure" class. Attributes defined for the Figure class are those that apply to all types of figures, such as "Color" or "Line Thickness." The Circle subclass automatically inherits the attributes of the Figure class; however, an additional "Radius" attribute might also be defined that is only appropriate for this special type of figure.

A class is a leaf of a class structure if it has no subclasses. The object class “HLAobjectRoot” shall be a superclass of all other object classes in a FOM or SOM. The HLA object model shall support only single inheritance; in this mechanism, each class has at most one immediate superclass.

HLA defines three types of object class attributes:

- **declared attributes:** The set of class attributes of an object class that are associated with that object class in the object class hierarchy tree.
- **inherited attributes:** The set of class attributes of an object class that was declared in a superclass of that object class in the object class hierarchy tree.
- **available attributes:** The set of declared attributes of an object class in union with the set of inherited attributes of that object class.

Federates participating in a federation execution may subscribe to attributes of object classes at any level of the class hierarchy. Conditions and circumstances surrounding discovery and reflection of instance attributes are described in IEEE Std 1516.1-2000.

Object classes provide the means for federation participants to subscribe to information about all individual instances of HLA objects with common characteristics, such as all M1A1 tanks or F117A fighters. Classes are also essential to specifying characteristics (attributes) of simulation objects, because these are defined relative to classes of objects, not unique to individual instances. In addition, because basic HLA services (as described in IEEE Std 1516.1-2000) support subscriptions to object class attributes by federates participating in a federation execution, the RTI requires knowledge of all object classes and their attributes if it is to support distribution of HLA object information by class to the federates of a federation execution.

A class hierarchy expands the capabilities of a flat classification scheme by enabling federates to subscribe to information about broad superclasses of objects, such as all tanks, all attack fighters, or even all ground vehicles, air vehicles, or sea vehicles. IEEE Std 1516.1-2000 supports subscription to attributes of any class in an object class hierarchy so that federates can easily subscribe to attributes of all or only those classes of interest. An object class hierarchy also supports simplification of the specification of attributes by placing common attributes of multiple subclasses in a common superclass. Thus, class hierarchies enable simpler management of the interests of different federates in the objects and attributes involved in a federation execution.

#### 4.2.2 Table format

The object class structure template of Table 3 provides a format for representing the class–subclass hierarchy of object classes. It shall be populated by first entering the root class of all object classes in the left-most column; this object class is named “HLAobjectRoot.” Then, the most general object classes shall be entered in the next column, followed by all of their immediate subclasses in the next column, and then further levels of subclasses, as required. The number of intermediate columns used depends on the needs of the federate or federation. A federate or federation that uses a deeper hierarchy than illustrated by the template of the table shall add columns as needed. Finally, the most specific object classes shall be specified in the right-most column. Object class names shall adhere to HLA naming conventions (see 3.3.1).

Although individual class names need not be unique, all object classes shall be uniquely identifiable in an HLA object model when concatenated (via dot notation) with the names of higher level superclasses.

Each object class in the object class structure table shall be followed by information on publication and subscription capabilities enclosed in parentheses, as designated in the template using the abbreviated variable name *<p/s>*.

For a SOM, valid entries for  $\langle p/s \rangle$  shall be as follows:

- *P (Publish)*: The federate is capable of publishing at least one attribute of the object class.
- *S (Subscribe)*: The federate is capable of subscribing to at least one attribute of the object class.
- *PS (PublishSubscribe)*: The federate is capable of publishing at least one attribute and subscribing to at least one attribute of the object class.
- *N (Neither)*: The federate is incapable of either publishing or subscribing to any attributes of the object class.

For a FOM, the same entries for  $\langle p/s \rangle$  are valid. However, an object class is classified as *Publish* or *Subscribe* in the federation if at least one federate is capable of publishing or subscribing to at least one attribute in the context of the object class.

Classes designated as *Subscribe* or *Neither* can have no registered instances, but they can have subclasses that can be registered.

**Table 3—Object class structure table**

|                                             |                                                      |                                                      |                                                      |     |                                                      |
|---------------------------------------------|------------------------------------------------------|------------------------------------------------------|------------------------------------------------------|-----|------------------------------------------------------|
| HLAobject Root<br>( $\langle p/s \rangle$ ) | $\langle \text{class} \rangle (\langle p/s \rangle)$ | $\langle \text{class} \rangle (\langle p/s \rangle)$ | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ... | $\langle \text{class} \rangle (\langle p/s \rangle)$ |
|                                             |                                                      | $\langle \text{class} \rangle (\langle p/s \rangle)$ | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ... | $\langle \text{class} \rangle (\langle p/s \rangle)$ |
|                                             |                                                      |                                                      | ⋮                                                    | ... | ⋮                                                    |
|                                             |                                                      |                                                      | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ... | $\langle \text{class} \rangle (\langle p/s \rangle)$ |
|                                             |                                                      | $\langle \text{class} \rangle (\langle p/s \rangle)$ | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ... | $\langle \text{class} \rangle (\langle p/s \rangle)$ |
|                                             |                                                      |                                                      | ⋮                                                    | ... | ⋮                                                    |
|                                             | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ⋮                                                    | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ... | $\langle \text{class} \rangle (\langle p/s \rangle)$ |
|                                             |                                                      | $\langle \text{class} \rangle (\langle p/s \rangle)$ | $\langle \text{class} \rangle (\langle p/s \rangle)$ | ... | $\langle \text{class} \rangle (\langle p/s \rangle)$ |
|                                             |                                                      |                                                      | ⋮                                                    | ... | ⋮                                                    |
|                                             |                                                      | ⋮                                                    | ⋮                                                    | ... | ⋮                                                    |
|                                             | ⋮                                                    | ⋮                                                    | ⋮                                                    | ... | ⋮                                                    |

#### 4.2.3 Inclusion criteria

In all HLA object models, any object class that is referenced in any table within the object model shall also be included in the object class hierarchy. The criteria for designing an object class hierarchy for an HLA object model is fundamentally different for individual federates than for federations. The object class structure table of a FOM represents an agreement between the federates in a federation on how to classify object classes for the purposes of federation executions. The object class structure table of a SOM is a type of advertisement of the classes of objects that the federate can support (as *Publish* or *Subscribe*) in potential federations. In neither case does the HLA require specific object classes (other than the required root class) or object class hierarchies to appear in the object class structure table.

For individual federates, it is the intrinsic functionality (expressed as classes of objects) that the federate can offer to future HLA federations—and any object classes whose instances (and associated instance attribute values) may potentially represent useful information if generated by other HLA federates—that defines the content of the object class structure table. The structure of the object class hierarchy is driven by how the federate supports class publication and subscription. Rich, deep SOM class hierarchies can provide federates with a significant degree of flexibility in how they can support and participate in federations in the future.

For federations, it is the subscription requirements of the collective set of simulations participating in the federation that drives the content and structure of the object class hierarchy. Although a set of object classes for the most specific types of entities involved in a federation (e.g., M1 tanks and Bradley fighting vehicles) may completely satisfy the subscription requirements of some types of HLA applications, additional higher level object classes will be needed if federates wish to be able to subscribe to HLA object information at higher levels of abstraction (e.g., tanks, armored vehicles, or ground vehicles). For a federate to be able to subscribe to HLA object information at a desired level of abstraction, an object class at that level of abstraction shall appear in the object class structure table.

For example, suppose a federation involved air, land, and sea forces of many specific types. If a particular federate did not require notification of the specific types of land vehicles, but did require notification of land vehicles in its area of interest, a suitable class (such as *GroundVehicle*) that contained specific vehicle types as subclasses would be needed to make this possible.

Although classes are clearly needed for all objects of interest in a federation, many alternative class hierarchies can be devised to cover any given set of objects. The demarcations and levels of classes selected for an HLA FOM are the result of the federation development process. Object class hierarchies that already exist for individual SOMs may be incorporated into a FOM object class hierarchy if they meet the interests of the federation as a whole.

#### 4.2.4 Example

Table 4 provides an example of how the object class structure table can be used to represent a simple system. In this case, the system being represented is a typical neighborhood restaurant. The simulation of this restaurant's operations can be considered to be a potential federate in a larger scale federation, perhaps representing the combined, coordinated operation of a chain of restaurants. The intent of this example is not to specify a complete SOM for this system, but rather to provide partial illustrations of how the OMT tables can be used to capture relevant information about the system.

In this example, a subset of a complete object class hierarchy is shown as consisting of five object classes at the most general level. For this simulation, no class decomposition was necessary for the first three classes. For the fourth class, a single level of decomposition is shown resulting in five leaf classes. For the fifth class, several levels of decomposition are shown to illustrate a partial representation of the restaurant's menu. Some of the deeper levels in this hierarchy could have been modeled as attributes (e.g., *ClamChowder* could have been a leaf node, with an attribute of *Type* to represent the enumerated values of *Manhattan* or *New-England*). However, the modeler in this example opted to represent the most specific food types as individual classes.

In this example, most nonleaf classes are designated as *Subscribe* only, whereas the leaf nodes (concrete classes) are designated as both *Publish* and *Subscribe*. One exception to this is the *Employee* class which contains several attributes that are only published or subscribed to at the subclass level. Another exception is the *ClamChowder* class, which provides an example of a nonleaf class that can be registered (i.e., has attributes that can be published for that class). Finally, *Manhattan* and *NewEngland* provide examples of publishable leaf classes that do not contain subscribable attributes.

**Table 4—Object class structure table example**

|                              |               |                 |                  |                  |                |
|------------------------------|---------------|-----------------|------------------|------------------|----------------|
| HLA<br>object<br>Root<br>(N) | Customer (PS) |                 |                  |                  |                |
|                              | Bill (PS)     |                 |                  |                  |                |
|                              | Order (PS)    |                 |                  |                  |                |
|                              | Employee (N)  | Greeter (PS)    |                  |                  |                |
|                              |               | Waiter (PS)     |                  |                  |                |
|                              |               | Cashier (PS)    |                  |                  |                |
|                              |               | Dishwasher (PS) |                  |                  |                |
|                              |               | Cook (PS)       |                  |                  |                |
|                              | Food (S)      | MainCourse (PS) |                  |                  |                |
|                              |               | Drink (S)       | Water (PS)       |                  |                |
|                              |               |                 | Coffee (PS)      |                  |                |
|                              |               |                 | Soda (PS)        |                  |                |
|                              |               | Appetizers (S)  | Soup (S)         | ClamChowder (PS) | Manhattan (P)  |
|                              |               |                 |                  | BeefBarley (PS)  | NewEngland (P) |
|                              |               |                 | Nachos (PS)      |                  |                |
|                              |               | Entree (S)      | Beef (PS)        |                  |                |
|                              |               |                 | Chicken (PS)     |                  |                |
|                              |               |                 | Seafood (S)      | Fish (PS)        |                |
|                              |               |                 |                  | Shrimp (PS)      |                |
|                              |               |                 |                  | Lobster (PS)     |                |
|                              |               |                 | Pasta (PS)       |                  |                |
|                              |               | SideDish (S)    | Corn (PS)        |                  |                |
|                              |               |                 | Broccoli (PS)    |                  |                |
|                              |               |                 | BakedPotato (PS) |                  |                |
|                              |               | Dessert (S)     | Cake (PS)        |                  |                |
|                              |               |                 | IceCream (S)     | Chocolate (PS)   |                |
|                              |               |                 |                  | Vanilla (PS)     |                |

### 4.3 Interaction class structure table

#### 4.3.1 Purpose/background

An interaction is defined as an explicit action taken by a federate that may have some effect or impact on another federate within a federation execution. Interactions shall be specified in the interaction class structure table of HLA object models in terms of their class–subclass relationships, in much the same way that objects are described in the object class structure table. The hierarchical structure of interactions supported by this table is composed of relations of generalization (or specialization) between different types of interactions. For example, an engagement interaction might be specialized by air-to-ground engagements, ship-to-air engagements, and others. This engagement interaction, then, would be said to generalize its more specific types.

An interaction hierarchy in an HLA FOM is primarily designed to support inheritance. In SOMs, an interaction hierarchy reflects how the federate supports publication and subscription of interaction classes.

The interaction class “HLAinteractionRoot” shall be a superclass of all other interaction classes in a FOM or SOM. The HLA object model shall support only single inheritance; in this mechanism, each class has at most one immediate superclass. Conditions and circumstances surrounding the processing of interactions are described in IEEE Std 1516.1-2000.

Interactions are one of the principal determinants of interoperability among simulations. Interoperability ordinarily requires some consistency in the treatment of interactions afforded by the different federates in which they appear. In distributed war fighting, for example, some uniformity in treatment of engagement interactions is commonly required to ensure a fair fight between objects simulated by different federates. Thus, all interactions in a FOM shall be identified, and all federates in an HLA federation must treat the specified interactions in a consistent fashion.

In addition, the types of interactions involved in a simulation execution shall be made known to the RTI in order to support publication and subscription to their occurrences. Thus, the HLA object model shall document all of the interactions that may be sent during a federation execution so that the RTI can recognize them.

### 4.3.2 Table format

The template that shall be used for recording HLA interactions for a federation or an individual federate is illustrated in Table 5. The basic format for this table follows the format specified earlier for the object class structure table. Thus, the root interaction class shall be specified in the left-most column and shall be named “HLAinteractionRoot.” Subsequent columns to the right shall contain interaction classes with increasing degrees of class specificity. Like the object class structure table, additional columns may be added to the table as needed to specify the full hierarchy. Although individual class names need not be unique, all interaction classes shall be uniquely identifiable in an HLA object model when concatenated (via dot notation) with the names of higher level superclasses. Interaction class names shall adhere to HLA naming conventions (see 3.3.1).

Table 5—Interaction class structure table

|                            |                 |                 |                 |     |                 |
|----------------------------|-----------------|-----------------|-----------------|-----|-----------------|
| HLAinteractionRoot (<p/s>) | <class> (<p/s>) | <class> (<p/s>) | <class> (<p/s>) | ... | <class> (<p/s>) |
|                            |                 | <class> (<p/s>) | <class> (<p/s>) | ... | <class> (<p/s>) |
|                            |                 |                 | :               | ... | :               |
|                            |                 |                 | <class> (<p/s>) | ... | <class> (<p/s>) |
|                            |                 | <class> (<p/s>) | <class> (<p/s>) | ... | <class> (<p/s>) |
|                            |                 |                 | :               | ... | :               |
|                            |                 | :               | :               | ... | :               |
|                            | <class> (<p/s>) | <class> (<p/s>) | <class> (<p/s>) | ... | <class> (<p/s>) |
|                            |                 |                 | <class> (<p/s>) | ... | <class> (<p/s>) |
|                            | :               | :               | :               | ... | :               |

Each interaction class in the interaction class structure table shall be followed by information on publishing and subscribing capabilities enclosed in parentheses, as designated in the template using the abbreviated variable name  $\langle p/s \rangle$ .

For a SOM, valid entries for  $\langle p/s \rangle$  shall be as follows:

- *P (Publish)*: The federate is capable of publishing the interaction class.
- *S (Subscribe)*: The federate is capable of subscribing to the interaction class.
- *PS (PublishSubscribe)*: The federate is capable of publishing and subscribing to the interaction class.
- *N (Neither)*: The federate is incapable of either publishing or subscribing to the interaction class.

For a FOM, the same entries for  $\langle p/s \rangle$  are valid. An interaction class is classified as *Publish* or *Subscribe* in the federation if at least one federate is capable of publishing or subscribing to the class.

Classes designated as *Subscribe* or *Neither* are never sent, but they can have subclasses that are sent.

#### 4.3.3 Inclusion criteria

A type of interaction shall be included in a FOM whenever it can take place “across” a federation, i.e., when it is published by one federate and subscribed to by another. Common examples of such interactions in warfighting simulations include a variety of engagement interactions between platforms that may be simulated by different federates. In order to document the types of interactions that federation members may need to accommodate, a FOM shall include all cross-federate interactions.

When interactions are expected to occur within an individual federate, they need not appear in a FOM. For example, in an engineering simulation, the interactions involved in the internal dynamics of a vehicle engine should not be part of a FOM if only one federate in the federation will interact directly with the engine component.

Because HLA SOMs are intended to be developed independently of any particular federation application, the relevance of any currently supported interaction class to future federations will generally be unknown. Thus, a simulation that supports either publishing or subscribing for an interaction class should document that support in its SOM if it might be of interest to future federations.

#### 4.3.4 Example

A representation of some illustrative interactions, based on the restaurant example introduced in 4.2.4, is given in Table 6. Here, the operations of the restaurant are described according to a set of interactions between customers and the employees of the restaurant. At the highest level, the restaurant federate can publish generic customer–employee transactions. This would, for instance, allow federates that simulate management operations across the restaurant chain to subscribe to and monitor general activity levels for individual restaurants. This single high-level interaction class is then decomposed into the basic types of customer–employee interactions that occur in the restaurant. Although all classes at this level can be published by the federate, *OrderTaken*, *FoodServed*, and *CustomerPays* are all further decomposed into more specialized classes that can also be published by the restaurant federate depending on the needs of the federation. In addition, this federate also shows the ability to subscribe to (*S* designation) interactions of classes *CustomerSeated* and *CustomerLeaves*, possibly to monitor customer arrival activity in other restaurants within the chain, and to monitor the rate at which other restaurants can service their customers.

Table 6—Interaction class structure table example

|                        |                         |                     |                      |
|------------------------|-------------------------|---------------------|----------------------|
| HLAinteractionRoot (N) | CustomerTransaction (P) | CustomerSeated (PS) |                      |
|                        |                         | OrderTaken (P)      | FromKidsMenu (P)     |
|                        |                         |                     | FromAdultMenu (P)    |
|                        |                         | FoodServed (P)      | DrinkServed (P)      |
|                        |                         |                     | AppetizerServed (P)  |
|                        |                         |                     | MainCourseServed (P) |
|                        |                         |                     | DessertServed (P)    |
|                        |                         | CustomerPays (P)    | ByCreditCard (P)     |
|                        |                         |                     | ByCash (P)           |
|                        |                         | CustomerLeaves (PS) |                      |

4.4 Attribute table

4.4.1 Purpose/background

Each class of simulation domain objects shall be characterized by a fixed set of attribute types. These attributes are named portions of their object’s state whose values can change over time (such as location or velocity of a platform). The values of HLA instance attributes are updated through the RTI and provided to other federates in a federation. Both federates and federations shall document all such object attributes in the attribute table of their SOM or FOM. The object class “HLAobjectRoot” shall be a superclass of all other object classes in a FOM or SOM; attributes may be assigned to it like any other object class.

Attributes of HLA object classes are specified in order to support subscription to their values by other interested members of a federation. Thus, the names of attributes and associated object classes are essential information when initializing a federation execution. Knowledge of object attributes is commonly required for effective communication between federates in a federation. In addition, although the datatypes and update policies of attributes represent characteristics that are not directly utilized by the RTI, they are important to ensuring compatibility among federates in a federation. A federate operating with very low update rates for an instance attribute could create problems for interacting federates that are operating at high update rates. The specification of datatypes and update rates in an HLA FOM is a part of the FOM “contract” among federates to interoperate at the specified levels. These specifications help to ensure a common perception of the simulation space across federates in a federation, lowering the potential for inconsistency.

4.4.2 Table format

The attribute table of a FOM shall describe all object attributes represented in a federation. The attribute table of a SOM shall describe all class attributes that are published or subscribed to by the federate. The template that shall be used for the attribute table is provided by Table 7.

**Table 7—Attribute table**

| Object          | Attribute                     | Datatype   | Update type   | Update condition   | D/A   | P/S   | Available dimensions | Transportation | Order   |
|-----------------|-------------------------------|------------|---------------|--------------------|-------|-------|----------------------|----------------|---------|
| HLA object Root | HLA privilege ToDelete Object | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |
| <object class>  | <attribute>                   | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |
|                 | <attribute>                   | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |
|                 | <attribute>                   | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |
| <object class>  | <attribute>                   | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |
|                 | <attribute>                   | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |
|                 | <attribute>                   | <datatype> | <update type> | <update condition> | <d/a> | <p/s> | <dimensions>         | <transport>    | <order> |

The first column (Object) shall contain names from the object class structure table for object classes that are associated with attributes. Object class names shall include the parentage (superclasses) of the class to the depth necessary to uniquely identify the class in this table and may include the full parentage of the class. In general, to reduce redundancy, attributes should be specified for classes at the highest point in the hierarchy to which they apply. For example, if all air vehicles have an attribute of *MinimumTurnRadiusAtMaximumSpeed*, some redundancy can be avoided if this attribute is specified just once for the entire class of *AirVehicle*. Given that all object classes inherit the attribute types of their superclasses, the subclasses of *AirVehicle*, such as *FixedWing* and *RotaryWing*, also have this attribute with its specified characteristics.

The second column (Attribute) shall list the attributes of the specified object class. Attribute names shall adhere to HLA naming conventions (see 3.3.1). The names assigned to attributes of any particular object class shall not duplicate (overload) the names of attributes of this class or any higher level superclass. There may be many attributes for a single object class.

The third column (Datatype) shall identify the datatype of the attribute. This datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in subclauses of 4.12. A value of “NA” may also be provided if the attribute never contains a value; an example of such an attribute is a token that is owned but never updated.

Although an attribute’s datatype can be specified “NA,” that attribute must still have valid (non-“NA”) transportation and order types specified. When a datatype of “NA” is used, the update type, update condition, and available dimensions shall also be “NA.”

The fourth column (Update type) shall record the policy for updating an instance of the class attribute. The unique designations for this column shall be as follows:

- *Static*: The value of the attribute is static; the federate updates it initially and when requested.
- *Periodic*: The federate updates the attribute at regular time intervals.
- *Conditional*: The federate updates the attribute when unique conditions dictate.
- *NA*: The federate never provides a value for this attribute.

The fifth column (Update condition) shall expand and explain the policies for updating an instance of the class attribute. When the update type is *periodic*, a rate of the number of updates per time unit shall be specified in the Update Condition column. Attributes with a *conditional* update type shall have the conditions for update specified in the “Update condition” column. The “Update condition” column may also be used to specify initial conditions for instance attribute updates. If a federate is capable of varying the rate and circumstances of the update of an instance object attribute, this information shall also be recorded in this column or through the OMT notes feature. If the update type is *Static* or *NA*, “NA” shall be entered in this column.

The sixth column (D/A) shall indicate whether ownership of an instance of the class attribute can be divested or acquired.

In a FOM, if an instance attribute can be divested by a federate, it should be acquirable by some other federate in the federation. Therefore, the unique designations for this column shall be as follows:

- *N (NoTransfer)*: Ownership of an instance of this class attribute is not transferable in this federation.
- *DA (DivestAcquire)*: Some federate is capable of divesting ownership of instances of this class attribute to another federate and another federate is capable of acquiring ownership of instances of this class attribute.

In a SOM, the federate may be able to divest ownership of an instance attribute, acquire ownership, both, or neither. Moreover, the corresponding class attribute shall be published for the federate to divest or acquire ownership. The unique designations for this column shall be as follows:

- *D (Divest)*: The federate is capable of publishing the class attribute and can divest ownership of an instance of this attribute to another federate using the HLA ownership management services.
- *A (Acquire)*: The federate is capable of publishing the class attribute and can acquire ownership of an instance of this attribute from another federate.
- *N (NoTransfer)*: The federate is neither capable of divesting ownership of instances of this class attribute nor acquiring ownership of instances of this attribute.
- *DA (DivestAcquire)*: The federate is capable of both divesting ownership of instances of this class attribute and acquiring ownership of instances of this class attribute.

The seventh column (P/S) shall identify the capabilities of a federate or federation with respect to class attribute publishing and subscribing.

In a SOM, the entry for this column shall take one of the following values:

- *P (Publish)*: The federate is capable of publishing this class attribute, either in the context of the class where it is declared or in at least one subclass of that class (i.e., capable of invoking appropriate HLA publishing services with the class as a supplied argument).
- *S (Subscribe)*: The federate is capable of subscribing to this class attribute, either in the context of the class where it is declared or in at least one subclass of that class (i.e., capable of invoking HLA appropriate subscribing services with the class as a supplied argument).
- *PS (PublishSubscribe)*: The federate is capable of both publishing and subscribing to this class attribute.
- *N (Neither)*: The federate neither publishes nor subscribes to this class attribute.

In a FOM, the same basic designations shall apply.

The eighth column (Available dimensions) shall record the association of the class attribute with a set of dimensions if a federate or federation is using DDM services for this attribute. The column shall contain a

comma-separated list of names of rows in the dimension table described in 4.6. Class attributes can be individually associated with dimensions, regardless of the dimensions associated with other attributes of the same class. For SOMs and FOMs of federates and federations, respectively, that are not using DDM services or for class attributes that are not associated with dimensions, “NA” shall be entered in this column.

The ninth column (Transportation) shall specify the type of transportation to be used with this attribute. These values shall be chosen from the name of any row in the transportation type table as described in 4.10.

The tenth column (Order) shall specify the order of delivery to be used with instances of this class attribute. Valid values for entries in this column are as follows:

- *Receive*: Instances of the class attribute are delivered to a receiving federate in an undetermined order.
- *TimeStamp*: Instances of the class attribute are delivered to a receiving federate in an order determined by time stamps assigned when the instance attributes were updated.

NOTE—One entry is required for Table 7. For this entry, the first column shall contain “HLAobjectRoot” and the second shall contain “HLAprivilegeToDeleteObject;” subsequent columns shall be completed as appropriate for the federate or federation. Uses and restrictions on this attribute are described in IEEE Std 1516.1-2000.

#### 4.4.3 Inclusion criteria

All instance attributes whose values can be exchanged during the course of an HLA federation execution shall be documented in the attribute table of a FOM. All class attributes that can be either published or subscribed to by an individual federate shall be documented in the attribute table of its SOM.

#### 4.4.4 Example

Table 8 shows examples of attributes from the restaurant application described in 4.2.4. First, the *Employee* object class is characterized according to the four attributes shown in the table. The datatypes specified are defined in the examples provided in 4.12. Each of these four attributes is updated conditionally except for the *YearsOfService* attribute, which is updated periodically (yearly) on the employee’s start date anniversary. As with most of the domain-specific attributes shown in this example, the attributes of *Employee* are assumed divestable, acquirable, publishable, and subscribable.

The *Waiter* subclass of *Employee* is shown with three attributes. These are in addition to the four inherited attributes from its superclass. Each of the first two attributes, *Efficiency* and *Cheerfulness*, is intended to represent a performance measure that is assigned to the employee at performance reviews. The third attribute is intended to represent the state of the employee (the task he/she is performing) at any point in time during restaurant operations.

In the final two entries, the *Drink* and *Soda* classes are each shown with a single attribute. The *NumberCups* attribute of the *Drink* class has been associated with the *BarQuantity* dimension, and the *Flavor* attribute of the *Soda* class has been associated with the *SodaFlavor* and *BarQuantity* dimensions; these dimensions are shown in the dimension table example in 4.6.4.

The HLAprivilegeToDeleteObject attribute of class HLAobjectRoot is a different kind of attribute than the others in the table. This attribute has a datatype of “NA,” meaning that an instance of this attribute does not have a value to be updated. Instead, it is more of a token attribute, in which information is conveyed simply by determining which, if any, federate owns the attribute at any time. In this case, HLAprivilegeToDeleteObject is a token attribute known by the RTI and used to determine whether or not a federate is permitted to delete an object instance.

Two examples of notes have also been included in Table 8; this feature is fully explained in 4.13.

**Table 8—Attribute table example**

| Object           | Attribute                     | Datatype        | Update type | Update condition      | D/A | P/S | Available dimensions    | Transportation | Order      |
|------------------|-------------------------------|-----------------|-------------|-----------------------|-----|-----|-------------------------|----------------|------------|
| HLAobject Root   | HLA privilege ToDelete Object | NA              | NA          | NA                    | N   | N   | NA                      | HLA reliable   | Time Stamp |
| Employee         | PayRate                       | DollarRate      | Conditional | Merit increase *[1,2] | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
|                  | YearsOf Service               | Years           | Periodic    | 1/year *[2]           | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
|                  | Home Number                   | HLAASCII string | Conditional | Employee request      | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
|                  | Home Address                  | Address Type    | Conditional | Employee request      | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
| Employee. Waiter | Efficiency                    | Waiter Value    | Conditional | Performance review    | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
|                  | Cheerfulness                  | Waiter Value    | Conditional | Performance review    | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
|                  | State                         | Waiter Tasks    | Conditional | Work flow             | DA  | PS  | NA                      | HLA reliable   | Time Stamp |
| Food.Drink       | Number Cups                   | DrinkCount      | Conditional | Customer request      | N   | PS  | BarQuantity             | HLA reliable   | Time Stamp |
| Food.Drink. Soda | Flavor                        | FlavorType      | Conditional | Customer request      | N   | PS  | SodaFlavor, BarQuantity | HLA reliable   | Time Stamp |

## 4.5 Parameter table

### 4.5.1 Purpose/background

Most interaction classes are characterized according to a list of one or more interaction parameters. Interaction parameters are used to associate relevant and useful information with classes of interactions. Examples of interaction parameters include object class names, object instance attribute values, strings, and numerical constants. Although some circumstances may require that a subset of the associated parameters be sent with an interaction, other circumstances will require that the full set of parameters be sent. However, for every interaction class identified in the interaction class structure table, the full set of parameters associated with that interaction class shall be described in the parameter table. The interaction class “HLAinteractionRoot” shall be a superclass of all other interaction classes in a FOM or SOM: Parameters may be assigned to it like any other interaction class.

Unlike class attributes, interaction parameters may not be subscribed to on an individual basis. Thus, dimension, transportation, and delivery order information is specified at the interaction class level rather than at the parameter level.

Interaction parameters may be associated with an interaction class at any level of an interaction class hierarchy. An interaction class inherits the parameters defined for its superclasses. In fact, the mechanisms and rules for inheritance of interaction parameters are identical to those of attributes. Thus, the specific placement of parameters throughout an interaction class hierarchy for a given federation is driven by the same needs that drive the placement of attributes in an object class hierarchy.

#### 4.5.2 Table format

The parameter table of an HLA object model is designed to provide descriptive information about all parameters of all interactions represented in a federation. The template that shall be used for the parameter table is provided in Table 9.

**Table 9—Parameter table**

| Interaction         | Parameter   | Datatype   | Available dimensions | Transportation   | Order   |
|---------------------|-------------|------------|----------------------|------------------|---------|
| <interaction class> | <parameter> | <datatype> | <dimensions>         | <transportation> | <order> |
|                     | <parameter> | <datatype> |                      |                  |         |
|                     | <parameter> | <datatype> |                      |                  |         |
| <interaction class> | <parameter> | <datatype> | <dimensions>         | <transportation> | <order> |
|                     | <parameter> | <datatype> |                      |                  |         |
|                     | <parameter> | <datatype> |                      |                  |         |

The first column (Interaction) shall contain names of interaction classes from the interaction class structure table. Interaction class names shall include the parentage (superclasses) of the class to the depth necessary to uniquely identify the class in this table and may include the full parentage of the class. In general, to reduce redundancy, parameters should be specified for classes at the highest point in the hierarchy in which they represent useful information, although this is not required. For example, if all weapon firings include a parameter that defines the infrared signature of the platform at the time the firing occurs, some redundancy will be avoided if this parameter is specified just once at the uppermost level of a *WeaponFires* class. Given that all interaction subclasses inherit the parameter types of their superclasses, the subclasses of *WeaponFires*, such as *TankFires* and *TELFires*, also have this parameter with its specified characteristics. Note that an interaction class need not contain any parameters; however, all interaction classes shall be included in this table so that transportation and order can be specified.

The second column (Parameter) shall list the parameters of each interaction. Parameter names shall adhere to HLA naming conventions (see 3.3.1). The names assigned to parameters of any particular interaction class shall not duplicate (overload) the names of parameters of this class or any higher level superclass. There may be many parameters for a single interaction class. If no parameters are specified for an interaction class, “NA” shall be entered in this column.

The third column (Datatype) shall identify the datatype of the parameter. This datatype shall be chosen from the name of any simple, enumerated, array, fixed record, or variant record datatype described in 4.12. If the interaction class has no parameters, “NA” shall be entered for this column.

The fourth column (Available dimensions) shall record the association of an interaction class with a set of dimensions if the federate or federation is using DDM services for this interaction. The column shall contain a comma-separated list of names of rows from the dimension table described in 4.6. This indicates that whole interactions of this class will be subject to filtering through regions containing a subset of these dimensions when the interaction is sent. The available dimensions of an interaction class shall be a superset of the available dimensions of its immediate superclass. For SOMs and FOMs in which DDM services are not used or for interactions that are not associated with dimensions, “NA” shall be entered in this column.

The fifth column (Transportation) shall specify the type of transportation to be used with this interaction. These values shall be chosen from the name of any row in the transportation type table as described in 4.10.

The sixth column (Order) shall specify the order of delivery to be used with this interaction. Valid values for entry in this column are as follows:

- *Receive*: The interaction is delivered to a receiving federate in an undetermined order.
- *TimeStamp*: The interaction is delivered to a receiving federate in an order determined by a time stamp assigned when the interaction was initiated.

#### 4.5.3 Inclusion criteria

In federations, any information elements that can be provided and associated with a given interaction class (by the class publishers) that are deemed to be useful to subscribers to that interaction class shall be included and documented as interaction parameters in the parameter table. For individual federates, SOM developers should associate with their publishable interaction classes whatever information they feel will be needed by subscribers to their interactions to calculate changes to the values of affected instance attributes. In addition, for interaction classes that the federate can subscribe to, SOM developers should determine what types of information need to be included with those interactions so that the federate can calculate associated effects.

#### 4.5.4 Example

Table 10 shows a partial example of interactions and parameters chosen from the restaurant application described in 4.2.4. Here, the *FoodServed.MainCourseServed* interaction has three parameters associated with it. The third parameter uses a Boolean datatype to reflect whether the meal was served in a reasonable amount of time. In addition, this interaction has been associated with the *WaiterId* dimension from the dimension table example in 4.6.4 as a means of filtering on waiter identification numbers. *CustomerSeated* is an example of an interaction with no parameters; the sending and the receiving of the interaction is sufficient to convey the necessary information.

**Table 10—Parameter table example**

| Interaction                  | Parameter     | Datatype    | Available dimensions | Transportation | Order     |
|------------------------------|---------------|-------------|----------------------|----------------|-----------|
| CustomerSeated               | NA            | NA          | NA                   | HLAreliable    | TimeStamp |
| FoodServed.MainCourse Served | TemperatureOk | ServiceStat | WaiterId             | HLAreliable    | TimeStamp |
|                              | AccuracyOk    | ServiceStat |                      |                |           |
|                              | TimelinessOk  | HLAboolean  |                      |                |           |

### 4.6 Dimension table

#### 4.6.1 Purpose/background

Federations use DM services to enable the flow of instance attribute and interaction data between federates and to limit the delivery of some data on the basis of object class, interaction class, and attribute name. This reduction of data may be insufficient to meet the needs of federations with large numbers of federates, large numbers of objects or interactions in classes, or large numbers of instance attribute updates per object. Such federations can use DDM services to further reduce the volume of data delivered to federates. When DDM services are used by a federation, a common framework for specifying the data distribution model for the federation is essential for the following reasons:

- It provides a commonly understood mechanism for specifying the exchange of public data and DDM coordination among members of a federation.
- It facilitates the design and application of common tool sets for specification of federation DDM needs.

Dimensions are the most fundamental DDM concept. Each attribute or interaction with which DDM services can be used will have a set of available dimensions, as indicated in the attribute table or parameter table, respectively. Each set of available dimensions is a subset of all dimensions that are defined in the dimension table, and defines a multidimensional coordinate system through which federates either express an interest in receiving data or declare their intention to send data. These intentions are specified by creating regions within the coordinate system that indicate particular areas of interest. The regions associated with an instance attribute or interaction shall contain only dimensions that are a subset of the attribute's or interaction's available dimensions, but need not contain all of the available dimensions. These regions are then used in one of the two following ways:

- *Subscription regions*: Sets of ranges that narrow the scope of interest of the subscribing federate.
- *Update regions*: Sets of ranges that are guaranteed to enclose an object's location in the coordinate system.

During development of an HLA federation that uses DDM, it is critical that all federation members achieve a common understanding of DDM dimensions and their semantics, and that they agree to a common set of dimension specifications. These agreements are necessary for federates to filter instance attribute updates and interactions in a correct and it consistent manner. This shall include the names of dimensions, the association of attributes and interactions with dimensions, the federate view of each dimension, and the default ranges the RTI should supply for available dimensions of attributes and interactions when a federate does not specify ranges at region creation time.

The federate view of each dimension is different from the RTI view of each dimension. The federate view gives each dimension user-defined semantics and an associated datatype in which values are expressed. The federate shall provide a normalization function for each dimension, that maps values from the federate view of a dimension to values in the RTI view of a dimension. Note that the federate view of a dimension is the same for each federate in a federation.

The RTI view of a dimension is an interval of nonnegative integers. The lower bound is fixed for all dimensions, but the upper bound varies by dimension as indicated in the dimension table. The resolution of the RTI view of the dimension does not have to be the same as the resolution of the federate view of the same dimension.

The dimensions form the basis by which update and subscription regions are specified by the federates to the RTI. By having this agreement about the meaning of dimensions enforced at the federate level, the RTI can calculate the overlaps of update and subscription regions efficiently without having to understand the semantics of the dimensions. The dimension table shall record all of the elements necessary to specify this agreement in a standard format.

#### 4.6.2 Table format

The template that shall be used for recording dimensions is illustrated in Table 11.

**Table 11—Dimension table**

| Name        | Datatype | Dimension upper bound | Normalization function   | Value when unspecified   |
|-------------|----------|-----------------------|--------------------------|--------------------------|
| <dimension> | <type>   | <bound>               | <normalization function> | <default range/excluded> |
| <dimension> | <type>   | <bound>               | <normalization function> | <default range/excluded> |
| <dimension> | <type>   | <bound>               | <normalization function> | <default range/excluded> |

The first column (Name) shall specify the name of the dimension. Dimension names shall adhere to HLA naming conventions (see 3.3.1). The names of dimensions in this column shall be unique. Each dimension represents a specific characteristic that may be used for filtering. For example, a dimension named *AltitudeLimits* might indicate the desire to filter information based on altitude.

The second column (Datatype) shall identify the datatype for the federate view of the dimension. This datatype shall be chosen from the name of a simple datatype or an enumerated datatype, as described in 4.12.

The third column (Dimension upper bound) shall specify the upper bound for the dimension that meets the federation's requirement for dimension subrange resolution. The value for this column shall be a positive integer. This value limits the resolution the RTI must provide to meet the federation's filtering requirements. It is also the maximum value a federate may use when communicating a range in this dimension to the RTI. For example, if a dimension represents radio frequencies in the range 88.1 MHz to 107.3 MHz in increments of .2, a dimension upper bound of 97 is sufficient.

The fourth column (Normalization function) shall specify the map from a subscription/update region's bounding coordinates to nonnegative integer subranges in the range [0, dimension upper bound). Because regions communicated with an RTI are expressed by bounding coordinates expressed in values from an RTI's view of dimensions, normalization functions shall be used by federates to construct regions. For each dimension in a region, that dimension's normalization function shall be used to map the dimension bounds as expressed in the federate view of the dimension to subranges of [0, dimension upper bound). To ensure that the federation exhibits correct semantics, federation participants should agree on the normalization functions that they will use for each dimension. This agreement shall include the specification of the normalization function and all values that are to be used by the function. If an attribute or parameter is used in the normalization function, its class hierarchy shall be specified in dot notation to the extent necessary to identify the attribute or parameter uniquely. The normalization function shall specify whether it uses a subset of the entire range or a subset of the datatype that is the basis of the dimension type. Example normalization functions are listed in Annex B.

The fifth column (Value when unspecified) shall specify a default range for the dimension that the RTI is to use in overlap calculations if the dimension is an available dimension of an attribute or interaction and has been left unspecified when a federate creates a region that is subsequently used (either for update or for subscription) with the attribute or interaction. The default range is specified as a nonnegative integer subrange of [0, dimension upper bound). Ranges shall be specified either by two values separated by two periods or by a single value. All ranges are closed on the lower bound (indicated by a left bracket), and open on the upper bound (indicated by a right parenthesis). The use of a single value shall specify a "point" range with the specified point as the lower bound and specified point plus one (1) for the upper bound, so that no other valid value lies between these points. An entry of "Excluded" indicates that the dimension shall not be used in overlap calculations unless a specific range is provided in the region specification.

#### 4.6.3 Inclusion criteria

Federations that require more data reduction than the class-based filtering provided by DM services should use the dimension table to document federation-wide agreements on the semantics and use of DDM dimensions. The type and normalization function information is guidance to federates on the consistent use of dimensions to maintain correct DDM semantics across the federation. Similarly, federates that normally operate in such federations should document the mechanisms that they use to take advantage of DDM.

#### 4.6.4 Example

Table 12 illustrates an example of the use of the dimension table. The first dimension, *SodaFlavor*, is an enumerated dimension, and its *linearEnumerated* normalization function (see Annex B) would map *Cola* to [0 .. 1), *Orange* to [1 .. 2), and *RootBeer* to [2 .. 3). The next two dimensions, *BarQuantity* and *WaiterId*, are inte-