SISO-STD-001-2015

Standard for Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Reference Federation Object Model

Version 2.0

10 August 2015

**Prepared by
Real-time Platform Reference
Federation Object Model
Product Development Group**

P.O. Box 781238
Orlando, FL 32878-1238, USA

Revision History

Version	Section	Date (MM/DD/YYYY)	Description
2.0		08/10/2015	Support for IEEE Std 1278.1™-1998
1.0		08/24/1999	Support for IEEE Std 1278.1™-1995

Participants

SISO-STD-001-2015, Standard for Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Reference Federation Object Model (GRIM 2.0) and SISO-STD-001.1-2015, Standard for Real-time Platform Reference Federation Object Model (RPR FOM 2.0) were together created as a community effort by the Real-time Platform Reference Federation Object Model 2.0 Product Development Group (PDG). The RPR FOM 2.0 development consisted of two separate efforts separated by a period of years. The *initial effort* occurred between 2000 and 2008 and the *final effort* between 2012 and 2015.

RPR FOM 2.0 PDG Initial Effort (2000 - 2008)

The initial RPR FOM 2.0 effort occurred between 2000 and 2008 and included a round of balloting without being approved. The RPR FOM PDG became inactive in 2005, although an additional draft was later produced in 2007, which was followed by another attempt to restart the group in 2008. In 2009 the formal process to dissolve the RPR FOM PDG was approved by the SISO Standards Activity Committee (SAC) and Executive Committee. However, the dissolution never became official due to an administrative technicality.

The hard work of those who participated in the initial effort is greatly appreciated as they produced good drafts that were able to support the balloting process.

Product Development Group Officers

Former Chairpersons: Graham Shanks, Richard Schaffer, Jim Gregg, Jim Kogler
Former Vice-Chair: *vacant*
Former Secretary: Douglas Wood

Drafting Group

Former GRIM Editors: Steve Dix, Mark Rybka, Sean Reilly, Keith Briggs
Former Associate GRIM Editors: Jeff Fisher, Ron Bertin

Former FOM Editors: Graham Shanks, Michael O'Connor
Former Associate FOM Editor: Mark Rybka

Technical Area Director: Paul Lowe

RPR FOM PDG Members

Wayne Belanger
Ron Bertin
Keith Briggs
Andy Cox
Steve Dix
Adam Faier
Jeff Fischer
Sibylle Gonzales
Len Granowetter
Jim Gregg
Carl Ito
Jim Kogler
Reed Little
Robert Lutz
Paul Metzger
Steve Monson
Michael O'Connor
Beth Pettit
Sean Reilly
Ed Roberts
Peter Ryan
Mark Rybka
Jerry Sanders
Randy Saunders
Richard Schaffer
Steve Seidensticker
Graham Shanks
Steven Sheasby
Jack Sheehan
Chris Turrell
Grant Tudor
Jeff Wicks
Earl Williamson
Chris Winters
Douglas Wood

RPR FOM 2.0 PDG Final Effort (2012 - 2015)

The final effort occurred from 2012 to 2013. This effort was initiated by Björn Möller who asked the SAC to reactivate the RPR FOM PDG to complete the effort to produce a SISO Standard for RPR FOM 2.0. The Product Nomination (PN) was updated and approved by the SAC and active work resumed on RPR FOM 2.0 in 2012.

At the time this product was submitted to the SAC for approval, the RPR FOM 2 PDG had the following membership and was assigned the following SAC Technical Area Director:

Product Development Group Officers

Chairperson: Björn Möller
Vice-Chair: Paul E. Murtha, Stephen Chappell
Secretary: Michael Heffernan
Technical Area Director: Thom McLean

Drafting Group

GRIM Editors: Aaron Dubois, Steven Sheasby
FOM Editors: René Verhage, Patrice Le Leydour
DG Recording Secretary: Aaron Dubois

RPR FOM PDG Members

Fredrik Antelius
Andy Bowers*
Andy Ceranowicz*
Tony Darlington
Aaron Dubois*
Åsa Falkenjack*
Michael Gagliano
Frank Hill*
Kyle Isakson*
Roger Jansen*
Stephen Jones*
Patrice Le Leydour*
Farid Mamaghani
Lance Marrou
Björn Möller*
Mike Montgomery*
Robert Murray
Shagoto Nandi
Lennart Olsson*
Peter Ross
Chris Rouget
Peter Ryan*
Graham Shanks*
Steven Sheasby*
Brett Terry*
Tom van den Berg*
René Verhage*

**denotes a Drafting Group member*

NATO Modelling and Simulation Group Task Group 068, NATO Education and Training Network, and Task Group 106, Enhanced CAX Architecture, Design and Methodology – SPHINX also made valuable contributions to the RPR FOM PDG.

The following individuals comprised the ballot group for this product.

Ballot Group

Fredrik Antelius
Curtis Blais
Andy Bowers
Veronica Charlton
Ann Clark
Mark Crnarich
Uwe Dobrindt
Aaron Dubois
Michael Gagliano
Michael Heffernan
Frank Hill
Kyle Isakson
Roger Jansen
Patrice Le Leydour
Paul Lowe
Lance Marrou
Mark McCall
Björn Möller
David Murray
Shagoto Nandi
Michael O'Connor
Lennart Olsson
Tim Pokorny
Félix Rodríguez
Peter Ross
Peter Ryan
Graham Shanks
Steven Sheasby
Tom van den Berg
René Verhage

When the SAC approved this product on 21 July 2015, it had the following membership:

Standards Activity Committee

Jeff Abbott (Chair)
Marcy Stutzman (Vice Chair / Secretary)

Grant Bailey
Curt Blais
Peggy Gravitz
Kevin Gupton
Jean-Louis Igarza
Bob Lutz
Lance Marrou
Lana McGlynn
Thom McLean
William Oates
Simone Youngblood

When the Executive Committee approved this product on 10 August 2015, it had the following membership:

Executive Committee

Michael O'Connor (Chair)
James Coolahan (Vice Chair)
Jane Bachman (Secretary)

Jeff Abbott
John Daly
John Diem
David Graham
Paul Gustavson
Shel Ocasio
Roy Scrudder
Robert Siegfried
Eric Whittington

Introduction

The Real-time Platform Reference Federation Object Model 2.0 (RPR FOM 2.0) defines a hierarchy of object and interaction classes for the High Level Architecture (HLA) that provides the capabilities defined in IEEE Std 1278.1™-1995, IEEE Standard for Distributed Interactive Simulation — Application Protocols, and its supplement, IEEE Std 1278.1a™-1998, IEEE Standard for Distributed Interactive Simulation — Application Protocols. RPR FOM 2.0 is designed to link simulations of discrete physical entities into complex virtual worlds. Its capabilities include representations of:

- Physical entities such as vehicles, lifeforms, cultural features, munitions, and collisions between them.
- Collections of individual entities collected as a single aggregate entity.
- Environmental objects and processes.
- Minefields.
- Communications between entities.
- Emissions generated by entities.
- Underwater acoustics.
- Weapon fire and detonations.
- Logistics, including repair and resupply.

SISO-STD-001-2015, Standard for Guidance, Rationale, and Interoperability Modalities for the Real-time Platform Reference Federation Object Model encapsulates guidance in the use of RPR FOM 2.0. It provides descriptions of FOM classes and datatypes and the relationship between the Distributive Interactive Simulation and the HLA-based RPR FOM, as well as rules for accomplishing specific distributed simulation tasks.

Changes from RPR FOM 1.0 made in RPR FOM 2.0 fall into one of the following categories, depending on the reason for the change:

- Support of IEEE Std 1278.1a™-1998 extensions – this resulted in new object and interaction classes, added attributes and parameters, new complex datatypes and enumerations.
- Representation of Spatial entity information was changed from separate attributes to a single attribute consisting of a variant-record.
- Changes to radio-related object and interaction classes were made due to community comments. The changes were made to support improved performance.
- The ModulationStruct complex datatype was removed because the functionality was moved to the SpreadSpectrumStruct complex datatype.
- Padding fields were added to complex datatypes to comply with the IEEE Std 1516.2™-2010, IEEE Standard for Modeling and Simulation High Level Architecture – Object Model Template Specification default encoding.
- Updated enumerated datatypes based on SISO-REF-010, Reference for Enumerations for Simulation Interoperability, version 00v20-0.

Appendix A lists all of the new, changed, and deleted structures for RPR FOM 2.0 versus RPR FOM 1.0.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Overview

This SISO Standard for Guidance, Rationale, and Interoperability Modalities (GRIM) for the Real-time Platform Reference Federation Object Model (RPR FOM) encapsulates guidance in the use of the RPR FOM. It provides descriptions of FOM classes and datatypes and the relationship between Distributed Interactive Simulation (DIS) and the High Level Architecture (HLA)-based RPR FOM, as well as rules for accomplishing specific distributed simulation tasks.

1.1 Scope

A FOM is 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. The GRIM accompanies the RPR FOM. It provides the usage rules for the RPR FOM, and the definitions, descriptions and rationale not otherwise specified within the FOM format.

As an example, this document, via reference to the IEEE standards for DIS application protocols [6, 7], defines the responsibility of federates that fire a weapon and of federates that are targeted and hit by a weapon. While the FOM definition provides for the message definition, it does not directly define federate responsibilities in generating, transmitting, or responding to message content.

1.2 Purpose

Prior to the development of HLA[1], the IEEE Standard for DIS [6] defined a standard protocol which permitted networked simulations to interact through the common communication of simulation data. These two architectures create significantly different environments for distributed simulation applications. DIS combines a simple data delivery architecture with strictly controlled message format standards (known as protocol data units (PDUs)) to create a system that maximizes interoperability between simulation partners. In contrast, the HLA defines a robust data delivery architecture but leaves the definition of content standards to individual simulator federations. The services of an HLA Runtime Infrastructure (RTI) include a delivery mechanism with capabilities to optimize performance by allowing individual federates to filter data at many different levels in the delivery process.

In HLA, data content standards are defined in an object oriented interchange format called the Object Model Template (OMT). A Simulation Object Model (SOM) is the specification of the types of information that an individual federate could provide to 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. The set of attributes that federates agree to share during a particular execution (an HLA federation) is documented in the FOM. By allowing federates to specify data content standards on an execution-by-execution basis, the HLA allows rapid adaptation to changes in simulation requirements and objectives.

The RPR FOM (pronounced “reaper fom”) was designed to organize the PDUs of DIS into a robust HLA object class and interaction class hierarchy. The priorities for developing this design are, in order:

1. Support transition of legacy DIS systems to the HLA.
2. Enhance *a priori* interoperability among RPR FOM users.
3. Support newly developed federates with similar requirements.

[1] The fundamental core of the High Level Architecture (HLA) is defined by three IEEE standards: the Framework and Rules [14], the Federate Interface Specification [15], and the Object Model Template (OMT) Specification [16]. IEEE standards are available through IEEE (http://www.ieee.org).

Like DIS, the RPR FOM is designed to support real time simulations of discrete physical entities such as planes, ships, soldiers, and munitions. These simulations are considered "real-time" because each second of elapsed execution time is equivalent to one second of time in the virtual world. Real-time, platform simulations are often used to support man-in-the-loop or hardware-in-the-loop systems.[2]

The RPR FOM is an instance of a Common Foundation Reference FOM (CF-RFOM) as defined by the SISO Reference FOM Study Group [1]. A CF-RFOM differs from a normal FOM because it refers to a notional rather than an actual collection of federates. The goal of a CF-RFOM is to enhance *a priori* interoperability by specifying content standards for common capabilities. Building upon the Reference FOM to meet the needs of a given execution creates the FOM for a particular federation. Because each federation's changes only extend this core functionality, simulations that do not require interoperability beyond the "starter" level of the CF-RFOM can participate, without software modification, in more specialized federations.

Version 1.0 of the RPR FOM provided an HLA conversion path for the DIS capabilities defined in IEEE Std 1278.1™-1995 [6]. Version 2.0 of the RPR FOM adds the functionality of the IEEE Std 1278.1a™-1998 [7]. The convention used within this document for referring to these two DIS standards are: DIS 1995 and DIS 1998. Future versions of the RPR FOM may accommodate new IEEE DIS standards and/or include new real-time platform-level simulation capabilities beyond those included in the DIS standards.[3]

Three different HLA versions have been developed over the years. The original version remains in use, though its use is waning and is no longer support by the U.S. Department of Defense. IEEE Std 1516™-2000 was intended to replace DoD HLA 1.3. IEEE Std 1516™-2000 OMT provided substantial improvements over DoD HLA 1.3 OMT. IEEE Std 1516™-2000 was superseded by IEEE Std 1516™-2010. IEEE Std 1516™-2010 is the latest HLA standard. Again, IEEE Std 1516™-2010 provides improvements in the OMT format. The concept of FOM Modules is also introduced. The convention used within this document for referring to these three versions of HLA are: HLA 1.3, HLA 2000, and HLA 2010.

In 2015, HLA 1.3, HLA 2000, and HLA 2010 coexist. The RPR FOM version described in this standard is provided in the OMT formats of each of these versions. There is a single HLA 2010 OMT format of the RPR FOM as well as a format separated into FOM Modules.

The three formats of the RPR FOM are "buffer compatible". Data encoded using the HLA 1.3 OMT representation of the RPR FOM and data encoded using the HLA 2000 or HLA 2010 representations of the FOM will result in identical encodings. This will provide an actual layout of the data that is the same in the HLA 1.3, HLA 2000, and HLA 2010 formats. The goal is to avoid the rewriting of data marshalling routines when migrating from older HLA versions to newer HLA versions. For example, HLA 1.3 does not provide support for variant records. The HLA 1.3 RPR FOM places the discriminator (and required padding) for variant records in the same position as specified by the HLA 2010 default encodings.

Because the RPR FOM is a reference FOM, users are free to make non-compliant changes to FOM elements or practices to meet their own development needs. However, simulations based on these kinds of modified FOMs might not have *a priori* interoperability with other systems based on the RPR FOM.

The GRIM is intended to foster interoperability by providing rules and rationale for common distributed simulation operations using the RPR FOM.

---

[2] Usage of the RPR-FOM is not limited to real time simulations only. Non-real time simulations using the RPR FOM are also possible.
[3] At the time of publication of this standard, the latest DIS version is IEEE Std 1278.1™-2012.

1.3 Objectives

The HLA infrastructure is a general-purpose mechanism allowing differing approaches to distributed simulation (i.e., there are many ways to accomplish the same thing). The RPR FOM was developed as a common approach to distributed simulations, with the end goal of fostering interoperability. Its objectives are to support the transition of DIS systems to the HLA, enhance *a priori* interoperability among RPR FOM users, and support newly developed federates with similar requirements.

1.4 Intended Audience

This document is intended to be used by individuals or groups involved with RPF FOM federations, or those interfacing between RPR FOM federations and DIS.

2 References (Normative)

The following references contain material that must be understood and used to implement the product.

2.1 SISO References

#	**Document Number**	**Title**
1	SISO-REF-001-1998	Reference FOM Study Group Final Report, Version 1.0, March 9, 1998
2	SISO-REF-010-00v20-0	Enumerations for Simulation Interoperability, November 19, 2013

2.2 Other References

#	**Document Number**	**Title**
3		High-Level Architecture Rules, Version 1.3, U.S. Department of Defense, 5 February 1998.
4		High Level Architecture Interface Specification, Version 1.3, U.S. Department of Defense, 2 April 1998.
5		High-Level Architecture Object Model Template Specification, Version 1.3, U.S. Department of Defense, 5 February 1998.
6	IEEE Std. 1278.1™-1995	IEEE Standard for Distributed Interactive Simulation - Application Protocols, 1995, Institute of Electrical and Electronics Engineers
7	IEEE Std. 1278.1a™-1998	IEEE Standard for Distributed Interactive Simulation - Application Protocols, 1998, Institute of Electrical and Electronics Engineers
8	IEEE Std. 1278.1™-2012	IEEE Standard for Distributed Interactive Simulation - Application Protocols, 2012, Institute of Electrical and Electronics Engineers
9		J. Towers, J. Hines, "Highly Dynamic Vehicles in a Real/Simulated Virtual Environment (HyDy), Equations of Motion of the DIS 2.0.3 Dead Reckoning Algorithms," Advanced Research Projects Agency, February 7, 1994.
10		IEEE Standards Definition Database, Institute of Electrical and Electronics Engineers, http://dictionary.ieee.org/
11	IEEE Std. 1516™-2000	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, 2000, Institute of Electrical and Electronics Engineers, NY, NY
12	IEEE Std. 1516.1™-2000	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Interface Specification, 2000, Institute of Electrical and Electronics Engineers, NY, NY
13	IEEE Std. 1516.2™-2000	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template, 2000, Institute of Electrical and Electronics Engineers, NY, NY
14	IEEE Std. 1516™-2010	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules, 2010, Institute of Electrical and Electronics Engineers

15	IEEE Std. 1516.1™-2010	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Interface Specification, 2010, Institute of Electrical and Electronics Engineers
16	IEEE Std. 1516.2™-2010	IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template, 2010, Institute of Electrical and Electronics Engineers
17		DoD Modeling and Simulation Glossary, Defense Modeling and Simulation Coordination Office, http://msco.mil/MSGlossary.html

3 Definitions, Acronyms, and Abbreviations

3.1 Definitions

The current version of the DoD Modeling and Simulation Glossary [17] is applicable for most terms in this standard. Any definitions included in this section are specific to the RPR FOM standard. IEEE Standards Definition Databas*e* [10] should be consulted for terms not defined in this section.

Term	Definition
a priori interoperability	Federated operation requiring no special adjustment when changing federations.
buffer compatible	Producing identical ‘on the wire’ data encoding.
endian	Referring to the byte storage order of multibyte data.
object instance ID	The string used to uniquely identify an object, implemented using the HLA object instance name.
optional attribute	An attribute that does not need to be provided to the federation.
reflection	Update of attributes at an HLA federate.
required attribute	An attribute that has to be provided to the federation for each instance of the Object Class that has this attribute. Typically, the attribute is provided right after the object instance is registered.

3.2 Acronyms and Abbreviations

Acronym or Abbreviation	Meaning
ALVB	Armored Vehicle Launch Bridge
ANDB	Additional Narrowband Database
APA	Additional Passive Activities
API	Application Programmer’s Interface
ASCII	American Standard Code for Information Interchange
ATaS	Acoustic Training and Simulation Database
ATC	Air Traffic Control
ATCRB	Air Traffic Control Radar Beacon
BE	Big Endian
BRG	Bearing
CSDB	Common Sensor Database
CF-RFOM	Common Foundation Reference Federation Object Model
CPU	Central Processing Unit
D/E	Depression/Elevation
DDM	Data Distribution Management
DER	Distributed Emission Regeneration

DG	Drafting Group
DIS	Distributed Interactive Simulation
DM	Declaration Management
DoD	Department of Defense
EE	Electromagnetic Emission
ERP	Effective Radiated Power
ESU	Entity State Update
EW	Electronic Warfare
FED	Federation Execution Data
FOM	Federation Object Model
GED	Group Entity Descriptor
GMT	Greenwich Mean Time
GRIM	Guidance, Rationale and Interoperability Modalities
HLA	High Level Architecture
ID	Identifier
IDM	Improved Data Modem
IEEE	Institute of Electrical and Electronics Engineers
IFF	Identification, Friend or Foe
LE	Little Endian
MIM	MOM and Initialization Module
MOM	Management Object Model
MSG	Modeling and Simulation Group
NATO	North Atlantic Treaty Organization
NAVAID	Navigation Aid
NETN	NATO Education and Training Network
NTP	Network Time Protocol
OMT	Object Model Template
PDF	Portable Document Format
PDG	Product Development Group
PDU	Protocol Data Unit
PN	Product Nomination
PRF	Pulse Repetition Frequency
PSDB	Platform Sensor Database
QRP	Query Response Protocol
RNG	Range
RPM	Revolutions Per Minute
RPR FOM	Real-time Platform Reference Federation Object Model
RRB	Reply Receiver "B"
RTI	Runtime Infrastructure
SAC	Standards Activity Committee
SAE	Site, Application, Entity
SAER	Site, Application, Entity and Radio
SEES	Supplemental Emissions/Entity State
SI	International System of Units
SIMAN	Simulation Management
SISO	Simulation Interoperability Standards Organization
SOM	Simulation Object Model
TADIL-J	Tactical Digital Information Link-J
TCAS	Traffic Alert/Collision Avoidance System
TDL	Tactical Data Link
TSPI	Time, Space, Position Information
UA	Underwater Acoustics
UDP/IP	User Datagram Protocol/Internet Protocol
UTC	Universal Time Constant
VP	Variable Parameter
XML	Extensible Markup Language

4 General Overview

The SISO Standard for RPR FOM 2.0 consists of two parts as described in the following paragraphs. RPR FOM 2.0, as documented by SISO-STD-001 (GRIM 2.0) and SISO-STD-001.1 (RPR FOM 2.0), supersedes SISO-STD-001-1999 (GRIM) and SISO-STD-001.1-1999 (RPR FOM) and the associated data files.

Note that changes have been made in RPR FOM 2.0 to some of the original RPR FOM 1.0 requirements based on experience in implementing RPR FOM 1.0 in HLA federations. In addition, some features of the DIS 1998 standard design were not fully functional and therefore the RPR FOM 2.0 implementation is different. These differences are so noted in the GRIM descriptions of those requirements.

The RPR FOM 2.0 also incorporates feedback from users of RPR FOM 2.0 drafts which have been available for a number of years. It was also influenced by the NATO Education and Training Network FOM. The changes are identified in the GRIM.

4.1 SISO-STD-001 - GRIM 2.0

This document (or part) is commonly referred to as GRIM 2.0. It provides functional requirements for HLA federations that use the RPR FOM. It includes definitions and guidance for object classes, interactions and other aspects of the FOM in order to support implementation by HLA federations. Reference is often made to specific paragraphs in the DIS 1995 and DIS 1998 standards for such things as issuance and receipt rules related to the data. It also includes background information on the relationship between the various HLA specifications and DIS standards.

This document is available at no charge. It can be downloaded from the Approved Standards section of the SISO website: http://www.sisostds.org/ProductsPublications/Standards/SISOStandards.aspx

4.2 SISO-STD-001.1 - RPR FOM 2.0

This part is commonly referred to as RPR FOM 2.0. It defines the RPR FOM in different formats and contains a document describing the RPR FOM. The data files support the development of RPR FOM-based federations and help ensure a basic level of interoperability between federates. It is expected that the RPR FOM will be used as is or modified to meet the needs of a specific federation.

Four sets of data files are provided to specifically support users of HLA 1.3, HLA 2000 and HLA 2010.

- A set of XML FOM modules for HLA 2010, representing the individual FOM modules (normative).
- A single XML file for HLA 2010, representing a single FOM where all FOM modules are combined.
- A single XML file for HLA 2000.
- A Federation Execution Data (FED) file and OMT file for HLA 1.3.

A document describing the RPR FOM is generated from the data files listed above.

The HLA 2010 FOM modules shall be considered normative. The human-readable document and all other data files shall be considered informative annexes. If any statement in the GRIM is interpreted to be in conflict with the HLA 2010 FOM modules, the FOM modules shall take precedence.

The computer-interpretable files comprising this part are available at no charge. They can be downloaded from the SISO XML Schemas section of the SISO website: http://www.sisostds.org/Schemas.aspx

5 Overview of HLA Functionality

The RPR FOM has been designed to be compliant with all applicable HLA rules and standards as described in the following sections.

5.1 HLA Rules

RPR FOM compliant simulations shall comply with the HLA Rules [3, 11, 14].

5.2 HLA Object Model Template (OMT)

RPR FOM compliant simulations shall comply with the HLA OMT [5, 13, 16].

5.3 HLA Services

The HLA Interface Specifications [4, 12, 15] define a set of services that permit distributed federates to participate in a common federation and transfer object and interaction data between the participating federates. Federates implementing the RPR FOM shall utilize the RTI to communicate with other federates participating in the federation.

Additional guidance on the use of HLA services is presented in the following subsections.

5.3.1 Time Management

Support for HLA Time Management services by RPR-FOM compliant federates is optional, and should be negotiated on an exercise by exercise basis. As a default and at a minimum, RPR FOM federates shall operate with time stepped, clock driven, independent time advance (see [3], rule 8.5). Correct operation of the RPR FOM in modes other than this time-flow mechanism is not guaranteed.

Clock driven simulations are considered "real-time" because each second of elapsed execution time is equivalent to one second of time in the virtual world. Time synchronization, if it is used at all, is performed outside of the simulation itself. For example, Network Time Protocol (NTP) is often used to synchronize "wall clock" times across a federation.

5.3.2 Ownership Management

DIS does not support the concept of distributed ownership of an entity. While HLA services and the design of the RPR-FOM will allow this, the RPR-FOM was developed with the assumption that all attributes of an object instance would be owned by the same federate, so there could be anomalies or unforeseen repercussions in cases where that does not hold true. For more details on ownership management when using the RPR FOM, please refer to section 8.1.

5.3.3 Data Distribution Management (DDM)

Data distribution management services can be utilized to more specifically control the flow of object and interaction data. These services primarily benefit large exercises where the total exercise traffic could require that the more advanced DDM data distribution services be implemented.

However, the RPR FOM currently makes no effort to standardize the use of DDM as the requirements for DDM can vary widely across the RPR FOM user domain. RPR FOM federate support for the DDM services is optional. Testing of the RPR FOM functionality for any specific set of defined routing spaces is the responsibility of the participating federates.

6 General FOM Guidance and Rationale

6.1 Structural Changes from DIS

The RPR FOM maps DIS PDUs into appropriate HLA object and interaction classes. In general, individual PDU fields are mapped into corresponding class attributes or parameters. An individual PDU could be mapped across one or more HLA object or interaction class. This change in structure is designed to take advantage of the HLA's Declaration Management and Object Management. This capability can be used to limit network traffic in two ways:

1. reducing the transmission of unchanged data, and
2. providing delivery only to federates which have expressed interest.

DIS provided developers with a set of fixed data structures that communicate directly with Internet Protocol sockets. In an effort to separate the requirements of information content from those of delivery, the RTI hides the mechanisms for its data distribution scheme behind a generic application programmer's interface (API).

Object classes in the RPR FOM are organized into a four level hierarchy. Object classes are separated by logical distinctions between groups of attributes. An effort was made during the design to minimize the repeating of attributes between object classes. This creates a hierarchy in which each object class represents fundamental characteristics (e.g., dead reckoning capability) instead of the behaviors of complete units (e.g., M1A1 tanks). This results in some attributes that might have no meaning for a particular subclass. Whenever this type of conflict occurs, it will be noted in the description of the attribute. For example, SubmersibleVessel(s) have AfterBurnerOn as an unrealistic attribute inherited from Platform. However, this form of organization was chosen to provide implicit guidance for Reference FOM specialization; new attributes and sub-classes are to be added to the RPR FOM based on attribute commonality. The PDG has chosen a similar, but shallower hierarchy for interaction classes (two levels). A mapping from the DIS PDU structures into the RPR FOM is provided in Section 9.

The new structure has a direct effect on the Boolean and enumeration values previously provided as DIS bit-structured fields. In the RPR FOM, all attributes and parameters previously represented as bit values have been expanded into independent object class attributes or interaction class parameters.

The new structure also affects many of the fields used to express array size in DIS. Since RPR FOM arrays are transmitted as separate attributes, the number of elements can usually be derived directly from the array's length in bytes and the size of each element. In such cases, the RPR FOM generally excludes the array length as a separate attribute. Exceptions to this approach occur when the element size is variable or when the number of elements is commonly used for other purposes.

The DIS PDU structures were intentionally defined to limit the size of each packet to the minimum supported for UDP/IP broadcast. Because the HLA removes these UDP/IP restrictions, these limitations are not part of the RPR FOM. DIS gateway developers and the users of these gateway tools need to characterize the performance of their DIS to RPR FOM translation in the case of large data updates.

6.2 Attribute Update Types

HLA provides a means for specifying the update type of each object attribute. The update type indicates when an update of the attribute is expected to be provided by the publishing federate. For some update types an additional update condition is specified to further identify specific conditions that results in an update. The RPR FOM specifies an update type for all attributes and an update condition where appropriate. Publishing federates shall update all required attributes and all provided optional attributes at a minimum as specified by the update type and update condition. Publishing federates may provide updates more frequently, but this is not necessary.

6.3 Default Instance Attribute and Parameter Values

In many cases, federates may choose not to update attributes or send interaction parameters that have no meaning for that federate. The object and interaction class definitions provided in Section 7 of this document specify each attribute and parameter that can be treated in this manner. Subscribing federates shall assume default values for any attribute or parameter not provided by the updating/sending federate. Unless otherwise specified in the object or interaction class definition, default values shall be treated in the following manner:

- All integer and floating point numeric attributes and parameters default to zero.
- All Boolean attributes and parameters default to false.
- All enumerated attributes and parameters default to an enumerator indicating the information to remain unspecified, typically using the value 0, such as Other or No_Statement.
- All arrays and strings default to the empty string.

Attributes and parameters with datatypes other than those listed above do not have standard defaults. Modifications to the RPR FOM should attempt to use the above default values whenever practical.

There are some attributes and parameters that are conditionally required based upon the value or presence/absence of other parameters. The condition upon which these attributes and parameters are required is defined in the class description.

In general an attribute or parameter is not sent if it conveys no additional information to the recipient. In particular, there are instances where the absence of an attribute or parameter conveys information. For example, the TargetObjectIdentifier parameter of the WeaponFireInteraction shall not be sent if the intended target is unknown.

In Section 7, some attributes and parameters have default values of "No Specific Entity". The absence of such an attribute in an update or interaction should be interpreted as indicating that no specific entity is being referenced. This is similar to the DIS entity ID 0.0.0, which is not representable as an HLA object instance handle.

The absence of an attribute or parameter with a default value of “No Information Available” should be interpreted as indicating that the sender does not model that aspect of the object.

6.4 Filter Support

Unlike DIS, the HLA supports multiple levels of data filtering:

- Application Level filtering refers to the ability of federates to accept or reject received FOM data based on content. This level of filtering is accomplished at the application layer instead of by the RTI. Application Level filtering was the only form of filtering supported by DIS. Although this type of filtering is usually simple for information senders, it can require extensive processing on the receiver side when many simulations are involved.
- Declaration Management (DM) Filtering refers to the ability of the RTI to deliver information based on each federate's expression of interest in (subscription to) object and interaction classes. Applications never receive information for object or interaction classes to which they have not subscribed. This type of filtering can also decrease network bandwidth usage.
- Data Distribution Management (DDM) Filtering allows the RTI to route data based on factors other than class type, which is handled by DM. DDM services provide a runtime capability to control the delivery of data within an exercise; however, the specific performance benefits can vary depending upon RTI and Federation design. In some cases, DDM usage can add processing overhead or data delivery latency.

In order to support common applications of DM filtering, several attributeless sub-classes are included in the RPR FOM Object Class Hierarchy. For example, in order to support DM filtering on the equivalent of the DIS Entity Type's "domain" field, the FOM includes seven attributeless subclasses of the Platform object class. Subscribing to the Aircraft object class, for instance, is the equivalent of passing entities with the domain of "Air" in a DIS Application Level filter.

To fully support this form of DM filtering, federates shall only register object instances that are leaf nodes of the RPR FOM. If a federation extends the RPR FOM so that a leaf node is subclassed, then the federate may register object instances at the newly created leaf nodes, while all object classes that are leaf nodes in the RPR FOM standard remain eligible for registering instances. A leaf node is defined as the lowest level available in the object class hierarchy table (no subclasses). In contrast, object class subscription should be used at the highest level (farthest from the leaf nodes) that supports all of the attributes and DM filtering required by the receiving federate.

6.5 DIS Entity Identifiers

The publishing federate shall be responsible for generating a unique Entity Identifier for each new object instance registered with an object class derived from the BaseEntity object class (see Section 7).

DIS uses the Entity Identifier triplet of Site ID–Application ID–Entity ID to uniquely identify entities in a distributed simulation. Each federate is responsible for establishing a unique Site ID–Application ID pair and then generating locally unique entity numbers. Additional identifiers are used to uniquely specify systems that are attached to an entity (e.g., radios and emitters). Since the HLA RTI provides an object instance naming mechanism to uniquely identify object instances, the use of the DIS Entity Identifiers for object class attributes in the RPR FOM appears redundant and unnecessary. The fact that attribute and parameter references to federate object instances use the RTI object instance name further supports this view. However, there are two purposes for the existence of these attributes. The first purpose is to support legacy applications that are migrating from DIS to HLA; the primary motivation for the RPR FOM design. The second purpose is the use of group addressing in simulation management interaction classes.

6.5.1 Entity Identifiers for DIS Legacy Applications

Many DIS legacy applications use the DIS Entity Identifier and system identifier internally for entity and system lookups. To ease the transition from DIS, the Entity Identifier and system identifier attributes were maintained in the associated RPR FOM object classes. It could be that, over time, the lookup requirement for these identifier attributes becomes less prevalent, and ultimately they could be removed. In order for this to happen though, an alternative solution for simulation management addresses is required.

6.5.2 Simulation Management

Although the RTI provides many simulation management (SIMAN) functions, the DIS group addressing scheme is not supported directly by the API. In DIS, certain values are reserved for use in addressing multiple sites, applications, or entities from a single SIMAN PDU. These are defined in IEEE Std 1278.1™ [6] with the symbolic names ALL_SITES, ALL_APPLIC, and ALL_ENTITIES. Each has a value of FFFF (H). The RPR FOM implements DIS compatible SIMAN services as HLA interaction classes (see Section 7.15.1.1). No clear way to duplicate this group addressing scheme using the RTI object instance names is readily apparent. The addition of the "EntityIdentifier" attribute was required in appropriate object classes (see Section 7) to facilitate the use of group addressing for these services in an HLA environment.

6.6 Dead Reckoning

The basic architecture of DIS specified the use of a dead reckoning mechanism for reducing communication processing (section 1.3.1.f of IEEE Std 1278.1™-1995 [6]). The RPR FOM has adopted this mechanism for the same purpose. For each registered object instance, the use of dead reckoning requires that a federate maintain a dead reckoning model in addition to its own internal model. The dead reckoning model shall follow one of the prescribed dead reckoning algorithms defined by DIS 1995 and enumerated in the RPR FOM. Dead Reckoning shall be applied to all object instances that are derived from the BaseEntity object class.

A federate shall issue a Spatial attribute update whenever the differences in position or orientation between its internal model and its dead reckoning model have exceeded established thresholds. The default thresholds for this Spatial attribute update condition are defined by the DIS 1995 standard as DRA_ORIENT_THRSH_DFLT = 3 degrees and DRA_POS_THRSH_DFLT = 1 meter [6]. The default values for the attribute thresholds are given in Table 1. The Spatial attribute is a variant record. The DeadReckoningAlgorithm provides the discriminant, which specifies the alternative. Each alternative contains only the information required to meet the requirements of the dead reckoning algorithm, such as SpatialStaticStruct for static entities.

Table 1 Default Dead Reckoning Thresholds

Threshold	Value
Position	1 m
Orientation	3 degrees

As in DIS, it is the receiving federate's responsibility to maintain a dead reckoning model for each external entity of interest. By applying the specified dead reckoning algorithm, the dead reckoning model provides a close approximation of the reflected object instance's actual Spatial attribute's value. Reflected Spatial attributes shall be used to correct the dead reckoning model so that future approximations are based on the most recent Time, Space, Position Information (TSPI) data.

6.7 Time Stamps

Dead reckoning and other simulation requirements supported by DIS required the transmittal of time stamp information. For example, this functionality can be used to account for network transport delays in exercises where federates are synchronized to a common external clock. In contrast, HLA's Time Management is concerned with the mechanisms for controlling the advancement of each federate to deliver information in a causally correct and ordered fashion. In the 1.3 version of the RTI, time stamp information is not passed between federates which are not using Time Management services. As a result, the RPR FOM encodes the time stamp within the RTI's user defined tag API parameter.

Federates shall send the time at which the data is valid in the user defined tag with every Update Attribute Values or Send Interaction call. The time shall be in the first 8 bytes (octets) of the user defined tag, using the DIS time stamp field format (see section 5.2.31 of IEEE Std 1278.1™-1995) converted into hexadecimal American Standard Code for Information Interchange (ASCII) character representation (0-9 and A-F), with leading zeros included. The ordering of the characters shall be in accordance with section 5.1.1 of IEEE Std 1278.1™-1995, that is most significant octet first, with the most significant bits first (i.e., the character for bits 4-7 precedes the character for bits 0-3). This encoding is equivalent to the result of the "C" statement "sprintf(UserTag, "%08X", DIStimestamp)", where "DIStimestamp" is represented in native format. All federates shall transmit this field, even if they do not use it themselves, so that other federates can use its value.

6.8 Datatypes and Encoding

6.8.1 Datatype Naming Conventions

The RPR FOM uses a naming convention when defining new datatypes, with the goal of allowing users to understand the class parameter and attribute definitions without the need to look up the exact datatype definitions.

6.8.1.1 Basic Data Representations

Basic data representations are the fundamental representations on which all other datatypes are defined. Basic data representation names follow the template "RPR[unsigned]<type><size><endianness>", where:

- [unsigned] – This is included for all unsigned datatypes and omitted for signed datatypes.

- <type> – Indicates the type of data represented. Valid types include “Float”, “Integer”, and “Octet”.
- <size> – Included for all datatypes and specifies the size of the datatype in bits.
- <endianness> – The endian representation of the basic data representation, with “BE” indicating big endian, and “LE” indicating little endian.

For example, the RPR FOM defines a basic data representation named “RPRunsignedInteger32BE”. This specifies an unsigned 32 bit integer with a big endian representation.

In the HLA 2010 format, all basic data representations are defined in the Foundation module.

6.8.1.2 Simple Datatypes

Simple datatypes are used to describe simple, scalar data items and are based on basic data representations. Simple datatype names follow the template “<property><unit><type><size>”, where:

- <property> – Indicates the meaning of the data. Can be omitted for dimensionless quantities (e.g., counters) or when the datatype can represent different properties.
- <unit> – Represents the unit of the property. Can be omitted when no unit is applicable. The unit is not abbreviated and is always singular.
- <type> – Indicates the basic data representation’s type (e.g., “Float”, “Integer”, or “Octet”). The type is not abbreviated.
- <size> – Indicates the number of bits in the basic data representation.

For example, the RPR FOM defines a simple datatype named “AccelerationMeterPerSecondSquaredFloat32”. This represents the acceleration of an object in meters per second squared. Its basic data representation’s type is a 32 bit floating point number.

6.8.1.3 Enumerated Datatypes

Enumerated datatypes are used to describe data that can take on a finite discrete set of possible values. Enumerated datatype names follow the template “<property>Enum<size>”, where:

- <property> – Indicates the meaning of the data.
- <size> – Indicates the number of bits in the basic data representation.

For example, the RPR FOM defines an enumerated datatype named “AntennaPatternTypeEnum32”. It describes the type of antenna pattern in use and is represented by a 32 bit integer.

6.8.1.4 Array Datatypes

Array datatypes are used to describe an indexed collection of data values of the same datatype. Array datatype names follow the template <datatype>[encoding]Array[cardinality]", where:

- <datatype> – Indicates the datatype of an element in the array.
- [encoding] – Indicates how the array is encoded. This portion of the name can have several different values:
  - Omitted when the encoding is of a type pre-defined by the HLA standard (e.g., HLAfixedArray or HLAvariableArray).
  - "Lengthless" when the encoding is RPRlengthlessArray.
  - "NullTerminated" when the encoding is RPRnullTerminatedArray.
  - "Padding32" when the encoding is RPRpaddingTo32Array.
  - "Padding64" when the encoding is RPRpaddingTo64Array.
- [cardinality] – Indicates the size of the array. For dynamic arrays that could be of any size, no cardinality is specified. For dynamic arrays that need to have at least one element, "1Plus" is used. For fixed arrays, the numerical size is specified.

For example, the RPR FOM defines an array datatype named "EntityTypeStructLengthlessArray". This datatype describes a dynamic array of elements, each represented by the EntityTypeStruct datatype. It is encoded as an RPRlengthlessArray. Another example is the datatype named "OctetArray8". This is a fixed array of eight elements with the Octet datatype.

An exception to this naming convention is made in the case where an array is composed of elements of a datatype defined by the HLA standard. These datatype names begin with the string "HLA". The HLA standards do not permit user-defined names to begin with "HLA". Therefore, if an array of HLA standard datatypes is defined, the string "HLA" shall be omitted from the beginning of the name.

In some cases, exceptions were made to the array datatype naming convention in order to provide a better description as to how the datatype is used. The following datatypes do not comply with the naming convention:

- RTIobjectId
- RPRUserDefinedTag
- MarkingArray11
- MarkingArray31
- MissingRecordNumbersLengthlessArray1Plus
- CoefficientsLengthlessArray1Plus
- SignalDataLengthlessArray1Plus

In the HLA 2010 format, all array datatypes are defined in the same module as the datatype of their elements.

6.8.1.5 Fixed Record Datatypes

Fixed record datatypes are used to describe a collection of data elements of heterogeneous datatypes. Fixed record datatype names follow the template "<property>Struct", where:

- <property> – Indicates the meaning of the data.

For example, the RPR FOM defines an enumerated datatype named "EntityTypeStruct". It describes a structure containing a number of fields that together define an entity's type.

6.8.1.6 Variant Record Datatypes

Variant record datatypes are used to describe data that contains a discriminant followed by one of a number alternative representations. Variant record datatype names follow the template "<property>VariantStruct", where:

- <property> – Indicates the meaning of the data.

For example, the RPR FOM defines an enumerated datatype named "SpatialVariantStruct". It describes a structure containing a number of fields that together define an object's location. Its representation differs based on the discriminant of the dead reckoning algorithm.

6.8.2 Endian Representation

To ensure interoperability among federates, federations have to agree on byte ordering conventions. As in DIS (in [6], Section 5.1.1), the big-endian network byte order convention shall be used in the RPR FOM. Federations can choose to use the little-endian convention, but they might not have *a priori* interoperability with other systems based on the RPR FOM.

6.8.3 Word Alignment

Some computer systems have alignment rules that have to be taken into consideration when constructing complex types. The guidance for developing RPR FOM complex data types has been derived from the equivalent DIS guidance. Complex types shall be organized such that all base types (integers and floating point numbers) start on an offset which is a multiple of their own size. For example, the offset of a 32 bit float, within a complex type could be zero, 32, 64 or any other multiple of 32. Padding shall be added to the complex type if this internal alignment cannot be achieved through simple re-arrangement. All padding fields shall be set to zero.

The following example illustrates this guidance: Using C syntax, we show two versions of a complex datatype below:

```
struct BadType {
   char  aChar ;  /* 8 bits */
   short aShort ; /* 16 bits */
   long  aLong ;  /* 32 bits */
} ;
```

```
struct GoodType {
   long  aLong ;  /* 32 bits */
   short aShort ; /* 16 bits */
   char  aChar ;  /* 8 bits */
} ;
```

The "BadType" on the left is improperly aligned. The attribute "aShort" starts on an 8-bit boundary that is not a multiple of the size of a short (i.e., a multiple of 16). The attribute "aLong" starts on a 24-bit boundary, which is not a multiple of the size of a long (i.e., 32). The "GoodType" on the right is properly aligned. Even though the attribute "aChar" does not fill up the second 32 bit word, terminal padding is not required by these rules. Padding at the end of the datatype is not required unless that form of alignment is needed for structures-within-structures or other forms of aggregation. For example, if the "GoodType" above were to be used as an array element, 8 bits of terminal padding would be required at the end to maintain proper alignment.

The HLA 1.3 Version of the RPR FOM explicitly specifies all padding required to ensure correct word alignment. Therefore, when encoding an attribute for transmission, the federate shall not add any additional padding. For example, if an attribute in the FOM consists of three shorts, the size of attribute values that a federate passes to the RTI for this attribute has to be exactly 6 bytes; there cannot be any padding between the three shorts, or following the last one.

In contrast, much of the padding in the HLA 2000 and HLA 2010 formats of the RPR FOM is implicit, as specified by the predefined encoding. The IEEE Std 1516.2™ predefined data encoding standards specify the conditions under which padding is to be added when encoding FOM data. So, in the HLA

2000 and HLA 2010 formats of the FOM, explicit padding is added only when required in addition to the predefined data encodings standards.

In some cases the amount of padding required is not known ahead of time. This occurs when the previous item in a structure is a datatype of variable length. To account for this, the RPR FOM defined the new encoding types RPRpaddingTo32Array and RPRpaddingTo64Array. These encodings are defined in section 6.8.5.

6.8.4 Basic Data Representations

The RPR FOM utilizes a number of basic data representations that are defined by the HLA standards. The names of these basic data representations changed between HLA 1.3 and HLA 2000. Further details on these representations are contained in the DoD HLA 1.3 [5], IEEE Std 1516™-2000 [13], and IEEE Std 1516™-2010 [16] OMT specifications.

While HLA 1.3 defined basic data representations for signed as well as unsigned integers, in HLA 2000 and HLA 2010, no basic data representations were defined to represent unsigned integers. To represent these values, the RPR FOM has defined the following new representations:

Datatype Name	Size	Description
RPRunsignedInteger8BE	8 bits	An unsigned integer in the range of $[0, 2^{8}-1]$.
RPRunsignedInteger16BE	16 bits	An unsigned integer in the range of $[0, 2^{16}-1]$.
RPRunsignedInteger32BE	32 bits	An unsigned integer in the range of $[0, 2^{32}-1]$.
RPRunsignedInteger64BE	64 bits	An unsigned integer in the range of $[0, 2^{64}-1]$.

6.8.5 Encoding Types

In HLA 2000 and HLA 2010, a datatype can be defined with an encoding. The encoding specifies how the datatype shall be encoded when provided to or received from the RTI. In addition to utilizing the encodings defined by the HLA 2010 standard, the RPR FOM adds the encodings RPRlengthlessArray, RPRnullTerminatedArray, RPRpaddingTo32Array, RPRpaddingTo64Array, and RPRextendedVariantRecord.

HLA 1.3 does not provide a means of specifying the encoding in the FOM. However, because of the naming convention described in section 6.8.1.4, the encoding can be determined based on the name of the array datatype. For example, an array datatype that uses the RPRlengthlessArray encoding will specify “Lengthless” in the name.

6.8.5.1 RPRlengthlessArray

The RPRlengthlessArray encoding is intended for arrays with variable cardinality and shall consist of the encoding of each element in sequence. In contrast to the HLAvariableArray encoding, the number of elements in the array is not included in the encoding. The number of elements is determined by the total number of bytes in the array divided by the size of a single element, including its padding if necessary. If the elements can be of variable size, use of HLAvariableArray is recommended instead.

The number of padding bytes after each element in the array is calculated in the same way as for HLAfixedArray as described in section 4.13.9.3 of the IEEE Std 1516.2™-2010 OMT specification [16]. The size of the RPRlengthlessArray shall include any padding bytes.

6.8.5.2 RPRnullTerminatedArray

The RPRnullTerminatedArray encoding is intended for arrays with variable cardinality and shall consist of the encoding of each element in sequence followed by a single NULL element. An empty array is specified by a single NULL element. As with RPRlengthlessArray, the number of elements in the array is not included in the encoding. The number of elements is determined by iterating over each element in the array until the NULL element is encountered. The RPRnullTerminatedArray encoding is intended to be used only for ASCII strings, where the NULL element is well defined as a single byte with a value of 0. If a federation specific extension to the RPR FOM uses this encoding for datatypes other than ASCII strings, then the representation of the NULL element has to be defined in the federation agreement.

The number of padding bytes after each element in the array is calculated in the same way as for HLAfixedArray as described in section 4.13.9.3 of the IEEE Std 1516.2™-2010 OMT specification [16]. The size of the RPRnullTerminatedArray shall include any padding bytes.

6.8.5.3 RPRpaddingTo32Array

The RPRpaddingTo32Array encoding is intended for use as a padding field of variable size. It is used as a field within complex datatypes following another field of variable length. It ensures that the next contiguous field in memory is aligned on a 32-bit boundary. All bytes in an RPRpaddingTo32Array shall have a value of 0.

For example, the fixed record datatype GridValueType0Struct includes the fields NumberOfBytes-A-Values followed by PaddingTo32. NumberOfBytes-A-Values is an array of octets with a cardinality of at least one. The PaddingTo32 field has a datatype of OctetPadding32Array, which has an encoding of RPRpaddingTo32Array. This ensures that a GridValueType0Struct always ends on a 32-bit boundary. If NumberOfBytes-A-Values has one element, PaddingTo32 will be an array of three bytes, each with a value of 0. If NumberOfBytes-A-Values has four elements, PaddingTo32 will be empty because no padding is necessary.

6.8.5.4 RPRpaddingTo64Array

The RPRpaddingTo64Array encoding is intended for use as a padding field of variable size. It is used as a field within complex datatypes following another field of variable length. It ensures that the next contiguous field in memory is aligned on a 64-bit boundary. All bytes in an RPRpaddingTo64Array shall have a value of 0.

For example, the fixed record datatype IrregularGridAxisStruct includes the fields NumberOfGridLocations-A-GridLocations followed by PaddingTo64. NumberOfGridLocations-A-GridLocations is an array of unsigned 16-bit integers with a cardinality of at least one. The PaddingTo64 field has a datatype of OctetPadding64Array, which has an encoding of RPRpaddingTo64Array. This ensures that an IrregularGridAxisStruct always ends on a 64-bit boundary. If NumberOfGridLocations-A-GridLocations has one element, PaddingTo64 will be an array of six bytes, each with a value of 0. If NumberOfGridLocations-A-GridLocations has eight elements, PaddingTo64 will be empty because no padding is necessary.

6.8.5.5 RPRextendedVariantRecord

The RPRextendedVariantRecord encoding type is a specialized encoding for variant records that adds a 32-bit, big-endian, unsigned integer field immediately after the discriminant to indicate the size, in bytes, of the variant portion of the structure.

The additional size field provides an easy way to determine the size of the variant portion of the structure without having to immediately parse the discriminant. It also allows a federation to add new discriminant values and alternative types to the variant record datatype without necessarily requiring changes to existing federates that do not need to use the new alternative. Even if the federate does not recognize the discriminant value, it will know the size of the record, and therefore will be able to determine where the variant record ends to begin decoding the next field in the attribute.

For example, an EnvironmentRecVariantStruct has many alternative types. One alternative is Point1GeometryData, which has a datatype of WorldLocationStruct. A WorldLocationStruct is 24 bytes long. Another alternative is Line1GeometryData, which has a datatype of Line1GeomRecStruct. A Line1GeomRecStruct is 48 bytes long. Therefore, if an EnvironmentRecVariantStruct specified the Point1GeometryData alternative, immediately following the discriminant would be a 32-bit integer with a value of 24. If another EnvironmentRecVariantStruct specified the Line1GeomRecStruct, this same 32-bit integer would have a value of 48. The alternative data would then be encoded following this integer.

6.8.6 Empty Strings

A number of attributes and parameters within the RPR FOM are represented by strings. One potential issue with using strings is that a federate may supply an empty string in an attribute update or interaction. An empty string can have different meanings. It could mean that the default value is to be used; it could mean that no value is specified at all; or it could actually mean that a string with no elements is to be used. This decision has to be made on a case-by-case basis and a general rule cannot be applied for all cases.

6.8.6.1 Empty RTIobjectId Strings

One common RPR FOM datatype that is represented by a string is RTIobjectId. An RTIobjectId attribute or parameter contains a string to identify another object that is in some way associated with this object or interaction. When an empty string is specified, this indicates that there is no association with another object. Since an empty string is the default value when string attributes are never updated, the most common way to indicate that such an association does not exist is to simply never send an update for the attribute. However, if an object was associated with another object, but that association has been removed, sending an attribute update with an empty string explicitly specified is the correct way to indicate that the association has been removed.

6.8.7 Variant Records in HLA 1.3

HLA 2000 introduced the capability to define variant record datatypes that was unavailable in HLA 1.3. The RPR FOM defines several variant record types. In order to preserve buffer compatibility between HLA 1.3, HLA 2000, and HLA 2010, complex datatypes have been defined in the HLA 1.3 OMT file that replicate the structure of HLA 2000 and HLA 2010 variant records. The first field in such an HLA 1.3 datatype is the discriminant. This is used to determine the type of data to follow, just us in an HLA 2000 or HLA 2010 variant record. If necessary, following the discriminant is an explicit padding field to replicate the implicit padding rules of HLA 2000 and HLA 2010. In many cases such padding is not necessary. Following the padding, if any, are fields defining each of the variant types of the record. Each field is defined with a cardinality of "0-1". Only one of these fields will ever be included in a variant record. The included field is identified by the value of the discriminant field.

Below is an example taken from the HLA 1.3 version of the RPR FOM. In this case, the discriminant is a 16 bit enumeration. As a result, a padding field of two bytes was required to ensure proper alignment of the following data.

```
(ComplexDataType (Name "StationNameVariantStruct")
    (ComplexComponent (FieldName "StationName")
        (DataType "ConstituentPartStationNameEnum16")
        (Cardinality "1")
        (Accuracy "perfect")
        (AccuracyCondition "always")
    )
    (ComplexComponent (FieldName "Padding")
        (DataType "octet")
        (Cardinality "2")
        (Units "N/A")
        (Resolution "N/A")
        (Accuracy "perfect")
        (AccuracyCondition "always")
    )
    (ComplexComponent (FieldName "RelativeLocation")
        (DataType "RelativePositionStruct")
```

```
            (Cardinality "0-1 (StationName = OnStationXYZ)")
            (Accuracy "perfect")
            (AccuracyCondition "always")
        )
        (ComplexComponent (FieldName "RelativeRangeAndBearing")
            (DataType "RelativeRangeBearingStruct")
            (Cardinality "0-1 (StationName = OnStationRangeBearing)")
            (Accuracy "perfect")
            (AccuracyCondition "always")
        )
    )
```

6.9 Delivery Category

The HLA supports two different delivery categories – reliable and best effort. Federation developers have to specify in their FED file a delivery category to be used for each interaction class, and for each object class attribute. Either delivery category may be used with any of the elements of the RPR FOM, however, the following guidance is recommended:

On LANs, or other networks where reliability is not known to be a problem, best effort should be used for all interactions and attributes. This will usually allow optimal real-time performance of a federation execution. The RPR FOM convention of sending many attributes only upon change might necessitate the use of HLA's reliable transport to maintain at least the level of reliability provided by the DIS "heartbeat" mechanism. If the HLA best effort category does not provide an acceptable level of reliability for a federation execution, the reliable category should be used for attributes that are unlikely to be updated periodically. For example, when entities are moving, the Spatial attribute is typically updated relatively frequently, so reliable delivery is typically not required for this attribute.

Atomic reflection of all attributes updated in a single RTI update is not guaranteed when using different delivery categories for different attributes in the update. That is, a set of attributes sent together might not be received together unless they use the same delivery category.

6.10 Civilian Versus Military Applications

Many simulation exercises contain both military and civilian entities, as well as require more than two-sided exercise play. The RPR FOM mappings specifically omit any distinction between civilian and military data at the object class level. Any such distinction can be derived from the attribute or parameter values (e.g., EntityType and/or ForceIdentifier). Full support of this distinction might actually require extension of the enumerated types. It is the intent that any such extension would be reciprocated by changes to SISO-REF-010.

This attribute-based distinction between military and civilian data requires Data Distribution Management to support interest management between the two. If declaration management support is required, existing RPR FOM leaf object classes can be extended with subclasses that make this distinction (e.g., MilitaryAircraft, CivilianAircraft). These distinctions should be reflected in the values of the existing inherited attributes so that interoperability is maintained with federates that rely on them.

6.11 Live Entity Mapping

Live entities are inherently real time platform level representations such as planes, ships, soldiers, and munitions which are represented in the synthetic environment for integration, interoperability and data rendering purposes. Under DIS 1998 developed standards, live entities were normally, post-instantiation, represented through Entity State Update (ESU) or TSPI data protocols. Under the HLA, data protocols are not used, rather object instance attributes are exchanged based on federate publications and subscriptions. Within this HLA scope, the DIS-based use of data field updates maps neatly into the HLA/RTI attribute update process.

Live entities are inherently represented in the HLA federation through the instrumentation system which acts as the owning federate. These Live federate entities inherently exist within a data communication latency environment close to, but not within, simulation real time. Live entity representations exist within the federation subject to the same rules as all other RPR FOM entities, but with certain additional requirements or restrictions which have to be considered when constructing the FOM. Live federates should have a process established for communications interruptions with an instrumented entity and the resulting severe variation between actual TSPI and dead reckoned TSPI. Live federate entities generally should not support entity ownership transfer functionality.

6.12 SpatialStruct Attributes

RPR FOM 2.0 has changed the way that Time, Space and Position Information (TSPI) is transmitted for performance reasons. This does cause a compatibility issue with the initial release of RPR FOM 1.0, but it was felt that the performance gains warranted the transition and RPR FOM 1.0 may be modified to use the new system as well.

The new system uses one variant record to store all of the TSPI and dead reckoning information into a single attribute instead of several attributes. The attributes that have been incorporated into this record are: AccelerationVector, AngularVelocityVector, DeadReckoningAlgorithm, IsFrozen, Orientation, VelocityVector and WorldLocation. The DeadReckoningAlgorithm provides the discriminant for the variant record, selecting which structure is used, determining how much information needs to be sent.

7 RPR FOM Classes and Modules

7.1 Class Hierarchy and Module Structure

The following sections provide details on the classes that make up the RPR FOM. Table 2 shows the object class hierarchy of the FOM and Table 3 shows the interaction class hierarchy. HLA 2010 introduced the ability to divide a single FOM into a number of different FOM modules. As a result, the HLA 2010 version of the RPR FOM has been divided into modules based on functional areas. The color coding in these tables indicates in which functional area and FOM module each class can be found. A single FED/FDD file containing all of the RPR FOM classes is also being provided for each of the three supported versions of HLA.

Color	*Module / Functional Area*
	Base
	Physical
	Aggregate
	Synthetic Environment
	Minefield
	Communication
	Distributed Emission Regeneration
	Underwater Acoustics
	Warfare
	Logistics
	Simulation Management

Table 2 Object Class Hierarchy

BaseEntity	PhysicalEntity	Platform	Aircraft
			AmphibiousVehicle
			GroundVehicle
			MultiDomainPlatform
			Spacecraft
			SubmersibleVessel
			SurfaceVessel
		Lifeform	Human
			NonHuman
		CulturalFeature	
		Munition	
		Expendables	
		Radio	
		Sensor	
		Supplies	
	AggregateEntity		
	EnvironmentalEntity		
Minefield			
EmbeddedSystem	MinefieldData		
	RadioTransmitter		
	RadioReceiver		
	Designator		
	EmitterSystem		
	IFF	NatoIFF	NatoIFFInterrogator
			NatoIFFTransponder

		SovietIFF	SovietIFFInterrogator
			SovietIFFTransponder
		RRB	
	UnderwaterAcousticsEmission	ActiveSonar	
		PropulsionNoise	
		AdditionalPassiveActivities	
EnvironmentObject	AreaObject	OtherArealObject	
		MinefieldObject	
	LinearObject	BreachableLinearObject	
		BreachObject	
		ExhaustSmokeObject	
		OtherLinearObject	
		MinefieldLaneMarkerObject	
	PointObject	BreachablePointObject	
		BurstPointObject	
		CraterObject	
		OtherPointObject	
		RibbonBridgeObject	
		StructureObject	
EnvironmentProcess			
GriddedData			
EmitterBeam	RadarBeam		
	JammerBeam		
ActiveSonarBeam			

In the object class attribute tables in Section 7, required attributes will be indicated using "**bold**" typesetting. Attributes that can be conditionally required based on the value of other settings will be indicated using "*italic*" typesetting. The condition upon which these attributes are required is defined in the Default Values column. Optional attributes will be indicated using "normal" typesetting.

Table 3 Interaction Class Hierarchy

Collision	CollisionElastic	
EnvironmentObjectTransaction	ArealObjectTransaction	OtherArealObjectTransaction
		MinefieldObjectTransaction
	LinearObjectTransaction	BreachableLinearObjectTransaction
		BreachObjectTransaction
		ExhaustSmokeObjectTransaction
		OtherLinearObjectTransaction
		MinefieldLaneMarkerObjectTransaction
	PointObjectTransaction	BreachablePointObjectTransaction
		BurstPointObjectTransaction
		CraterObjectTransaction
		OtherPointObjectTransaction
		RibbonBridgeObjectTransaction
		StructureObjectTransaction
MinefieldQuery		
MinefieldData		
MinefieldResponseNACK		
RadioSignal	EncodedAudioRadioSignal	
	RawBinaryRadioSignal	
	DatabaseIndexRadioSignal	
	ApplicationSpecificRadioSignal	

AcousticTransient	
WeaponFire	
MunitionDetonation	
RepairComplete	
RepairResponse	
ResupplyCancel	
ResupplyOffer	
ResupplyReceived	
ServiceRequest	
Acknowledge	AcknowledgeR
ActionRequest	ActionRequestR
ActionResponse	ActionResponseR
Comment	
CreateEntity	CreateEntityR
Data	DataR
DataQuery	DataQueryR
EventReport	
RemoveEntity	RemoveEntityR
SetData	SetDataR
StartResume	StartResumeR
StopFreeze	StopFreezeR
RecordR	
RecordQueryR	
SetRecordR	
TransferControl	
AttributeChangeRequest	AttributeChangeRequestR
AttributeChangeResult	AttributeChangeResultR
CreateObjectRequest	CreateObjectRequestR
CreateObjectResult	CreateObjectResultR
RemoveObjectRequest	RemoveObjectRequestR
RemoveObjectResult	RemoveObjectResultR
ActionRequestToObject	ActionRequestToObjectR
ActionResponseFromObject	ActionResponseFromObjectR

In the interaction parameter tables that follow, required parameters are indicated using "**bold**" typesetting. Parameters that cab be conditionally required based on the value of other settings will be indicated using "*italic*" typesetting. The condition upon which these parameters are required is defined in the Default Values column. Optional parameters will be indicated using "normal" typesetting.

Figure 1 shows the FOM module structure of the HLA 2010 format of the RPR FOM. If a module is shown above another module, this indicates that it is dependent on the lower module. For instance, all modules are dependent on the MIM, and each of the functional area modules is dependent on the Base module.

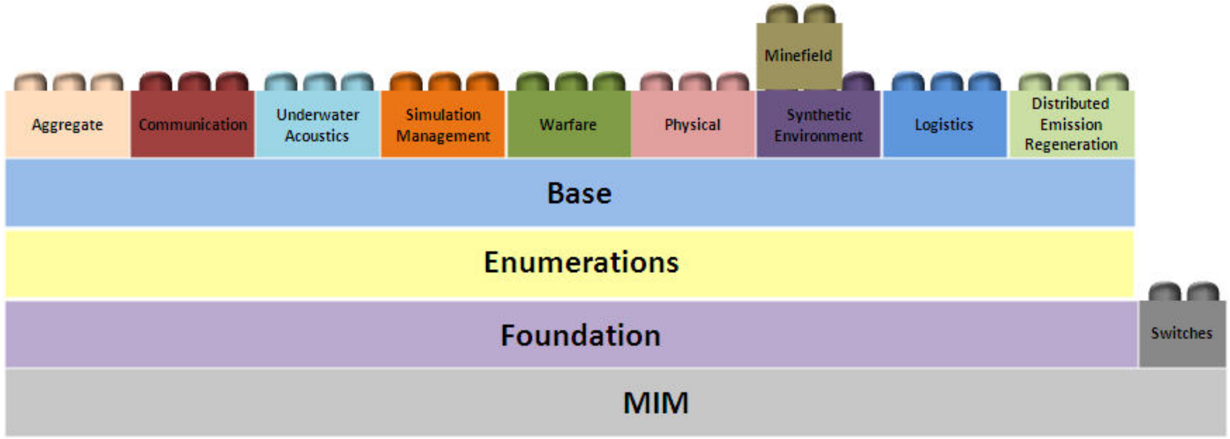

Figure 1 RPR FOM Module Structure

The following sections are organized by FOM module with subsections for the module's object and interaction classes. At the beginning of each subsection a diagram is shown to indicate the class hierarchy of the classes within this module. Since a FOM module might depend on another FOM module for certain base class definitions, the module could only include a scaffolding class description. A scaffolding class description contains only the name of the class and acts as a placeholder in order to represent the class hierarchy. The full description of the class can be found in another FOM module. In such cases the class structure diagram will show scaffolding classes in a box outlined with a dotted line, as in the example in Figure 2. Also visible in the example in Figure 2, unreferenced classes from dependent modules are shown grayed-out.

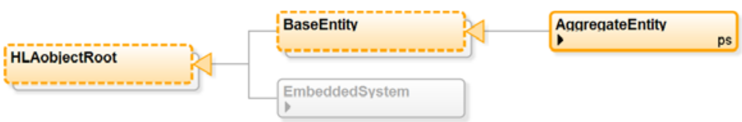

Figure 2 Example Object Class Structure with Scaffolding Classes

7.2 Foundation Module

The Foundation module contains definitions for common datatypes that are used in many of the other FOM modules and can be regarded as independent from the intended RPR FOM application domain[4]. This module contains no class definitions. This module contains the definitions for the datatypes RPRunsignedInteger16BE, RPRunsignedInteger32BE, RPRunsignedInteger64BE, RPRunsignedInteger8BE, RPRboolean, RTIobjectId, and RTIobjectIdArray.

7.3 Switches Module

HLA 2000 and HLA 2010 defined a set of switches that shall be set in the FOM. These switches regulate the behavior of some of the optional actions the RTI can perform on behalf of the federate, such as automatically requesting updates of an instance attribute when an object instance is discovered or advising the federates when certain events occur. In HLA 1.3, these switches were not available. HLA 2010 also added new switches that did not exist in HLA 2000. As a result, not all switches are available in all formats of the RPR FOM. The default values in the RPR FOM were chosen to preserve equivalent behavior across all three HLA versions whenever possible. To facilitate easy replacement of these

---

[4] The datatypes defined in the Foundation module might be perceived as fundamental datatypes missing from IEEE Std 1516™. As such, their preferred prefix is 'HLA' instead of 'RPR'. User-defined names starting with 'HLA' are however not allowed according to IEEE Std 1516.2™.

settings, for the modular version of the HLA 2010 RPR FOM the switches have been confined to a single FOM module. It is expected that federations might choose to update this module based on their federation agreement.

7.4 Enumerations Module

The RPR FOM contains definitions for many enumerated datatypes that are defined by the SISO-REF-010 reference document [2]. While the standard provides definitions for these enumerated datatypes, as part of their federation agreement many federations choose to update these standard definitions based on a chosen version of SISO-REF-010 or customization thereof. To facilitate easy replacement of these definitions, for the modular version of the HLA 2010 RPR FOM these enumerated datatypes have been confined to a single FOM module. This module contains no class definitions and no other datatype definitions. Many of the enumerated datatypes in this module are represented by basic datatypes defined in the Foundation module. As a result, this module is dependent upon the Foundation module.

7.5 Base Module

The Base module provides a common base for RPR based FOM Modules. It contains common datatypes and the BaseEntity and EmbeddedSystem object class definitions. Publication of object instances of these classes is not allowed. Instead they form the basis for specialized classes, such as the physical entity representations found in the Physical module and the different kinds of systems found in the Distributed Emission Regeneration module. Hence, although technically this module does not depend on any of the functional area modules, the Base module is of no use within a federation without modules that depend on it.

7.5.1 Object Classes

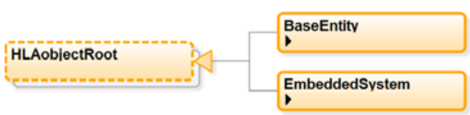

Figure 3 Base Module Object Class Structure

7.5.1.1 BaseEntity Object Class

The BaseEntity object class is designed to provide a basis for the individual entities that are the principal participants in RPR FOM federations. The core attributes shared by all entities include the entity's position and orientation in the virtual world, as well as velocity, acceleration, and angular velocity. These last three characteristics allow reflecting applications to "dead-reckon" the entity – that is, to approximate its position and orientation during the period of time between state updates.

The dead reckoning algorithm field of the spatial attribute allows the publishing federate to dictate whether and how subscribing federates perform dead-reckoning. When all reflecting federates perform dead-reckoning in the same way, they are able to share a more consistent view of the state of the virtual world.

In order to provide for a consistent interpretation for all participants, all federates shall apply a consistent version of the dead reckoning algorithms [9]. The coordinate system and dead reckoning models used shall follow the same form described in Sections 1.3.2, 4.5.2.1.2, 5.2.2, 5.2.17, 5.2.33, 5.2.34, Annex B of IEEE Std 1278.1™-1995 [6].

By combining position/maneuver data with classification information, the BaseEntity object class provides the minimum set of attributes needed to visualize an entity in the virtual world. The EntityType structure shall use DIS Entity Type enumerations to provide each entity classification with a unique identifier. In addition to supporting discrete physical entities, the BaseEntity object class also forms the basis for aggregations (like platoons and battle groups) and other object classes which require basic position/maneuver data. An overview of the BaseEntity attributes is provided in Table 4.

All federates updating instance attributes of this object class or its subclasses shall provide the EntityType, EntityIdentifier, and Spatial attributes. Updates for these attributes shall be provided at a minimum as required by the update types and update conditions specified in the FOM. The IsPartOf and RelativeSpatial attributes shall be treated as optional fields.

Table 4 BaseEntity Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
EntityType	Entity State	Entity Type	1995: 5.3.3.1.e	**(Not Optional)**	Kind, Country, Domain, Category, Subcategory, Specific, and Extra fields of the DIS Entity Type.
EntityIdentifier	Entity State	Entity ID	1995: 5.3.3.1.b	**(Not Optional)**	Identifies the site, application, and entity number of this object instance. It is used for group addressing in the SIMAN interactions. (See section 6.5.2)
IsPartOf	IsPartOf	Originating Entity, Relationship, Named Location	1998: 5.3.9.4.b, 5.3.9.4.d, 5.3.9.4.f,	All zeros	Used to indicate that there is a spatial relationship between this entity and a host entity, i.e., one entity is “part of” another
Spatial	Entity State	Dead reckoning Alg., Position, Velocity, Acceleration, Orientation, Angular Velocity, Is Frozen	1995: 5.3.3.1.g, 5.3.3.1.h, 5.3.3.1.i , 5.3.3.1.k.1, 5.3.3.1.k.4, 5.3.3.1.k.3	**(Not Optional)**	Used to express the spatial relationship between the entity and the center of the Earth.
RelativeSpatial	Entity State, Is Part Of	Dead reckoning Alg., Position, Velocity, Acceleration, Orientation, Angular Velocity	1995: 5.3.3.1.g, 5.3.3.1.h, 5.3.3.1.i , 5.3.3.1.k.1, 5.3.3.1.k.4, 5.3.3.1.k.3	All zeros	Used to express the spatial relationship between the entity and a host entity. Used in addition to the normal spatial attribute that describes absolute location.

As shown in this table, the attributes for the BaseEntity object class were all derived from corresponding values in the DIS Entity State PDU. Although object classes are not registered using the BaseEntity object class directly, BaseEntity can act as a useful subscription level for applications where basic data is required for nearly all entities in the simulation (e.g., visualization tools).

The RPR FOM does not support one of the Dead Reckoning fields included in the DIS Entity State PDU. The DIS dead reckoning parameters include 120 bits to represent “user-defined” dead reckoning attributes not covered in the basic protocol. The RPR FOM mechanism for supporting this functionality is for individual federations to specify a new discriminant for the SpatialVariantStruct, using either one of the existing fixed records or a newly defined one. As an alternative the BaseEntity object class may be extended to support additional attributes.

The BaseEntity class has two optional attributes that allow one to provide information about an entity's physical relationship with another entity, e.g., an aircraft on the deck of a carrier, a missile attached to a missile launcher, an individual combatant inside of a truck, or a plane flying in formation with another. These optional attributes are included in the IsPartOf function described in 9.7.4.

7.5.1.2 EmbeddedSystem Object Class

The EmbeddedSystem object class provides a mechanism for associating system capabilities with a host entity. The association between an EmbeddedSystem and its host provides a means of aggregating many capabilities into a single platform without resorting to multiple inheritance. The attributes listed in Table 5 establish the host identity and describe the positional relationship between this entity and the host. This object class has no optional attributes; federates updating instance attributes of this object class shall provide all attributes specified in Table 5.

Table 5 EmbeddedSystem Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EntityIdentifier	Electro magnetic Emissions or Designator or Transmitter or Receiver	Emitting Entity ID Designating Entity ID Entity ID Entity ID	1995: 5.3.7.1.b, 5.3.7.2.b, 5.3.8.1.b, 5.3.8.3.b	**(Not Optional)**	Identifies the site, application, and entity number of the host to which the object instance is attached.
HostObjectIdentifier	Electro magnetic Emissions or Designator or Transmitter or Receiver	Emitting Entity ID Designating Entity ID Entity ID Entity ID	1995: 5.3.7.1.b, 5.3.7.2.b, 5.3.8.1.b, 5.3.8.3.b	**(Not Optional)**	Object instance ID of the host to which this object instance is attached.
RelativePosition	Electro magnetic Emissions or Transmitter	Location Relative Antenna Location	1995: 5.3.7.1.e.4, 5.3.8.1	**(Not Optional)**	Location of the embedded system with respect to the host's coordinate system.

Embedded systems differ from articulated parts in that the non-visual elements of the embedded system (emissions/detections) are generally their primary simulation feature. The EmbeddedSystem object class is used to specify the association information found in DIS PDUs such as the Electromagnetic Emissions, Designator, Radio Transmitter, and Radio Receiver. Attributes specific to the object instance's emission/detection role are provided by the sub-classes. Defining embedded systems as separate object instances closely matches their use in DIS.

7.6 Physical Module

The Physical module provides object class definitions for representing physical entities including aircraft, ground vehicles, ships, life forms, ammunition, etc. In addition it provides interaction classes to signal collisions between physical entities.

The object classes derive from the BaseEntity class defined in the Base module. The majority of the datatypes used in the object and interaction classes are defined in the Base module, with the enumerations being defined in the Enumerations module.

7.6.1 Object Classes

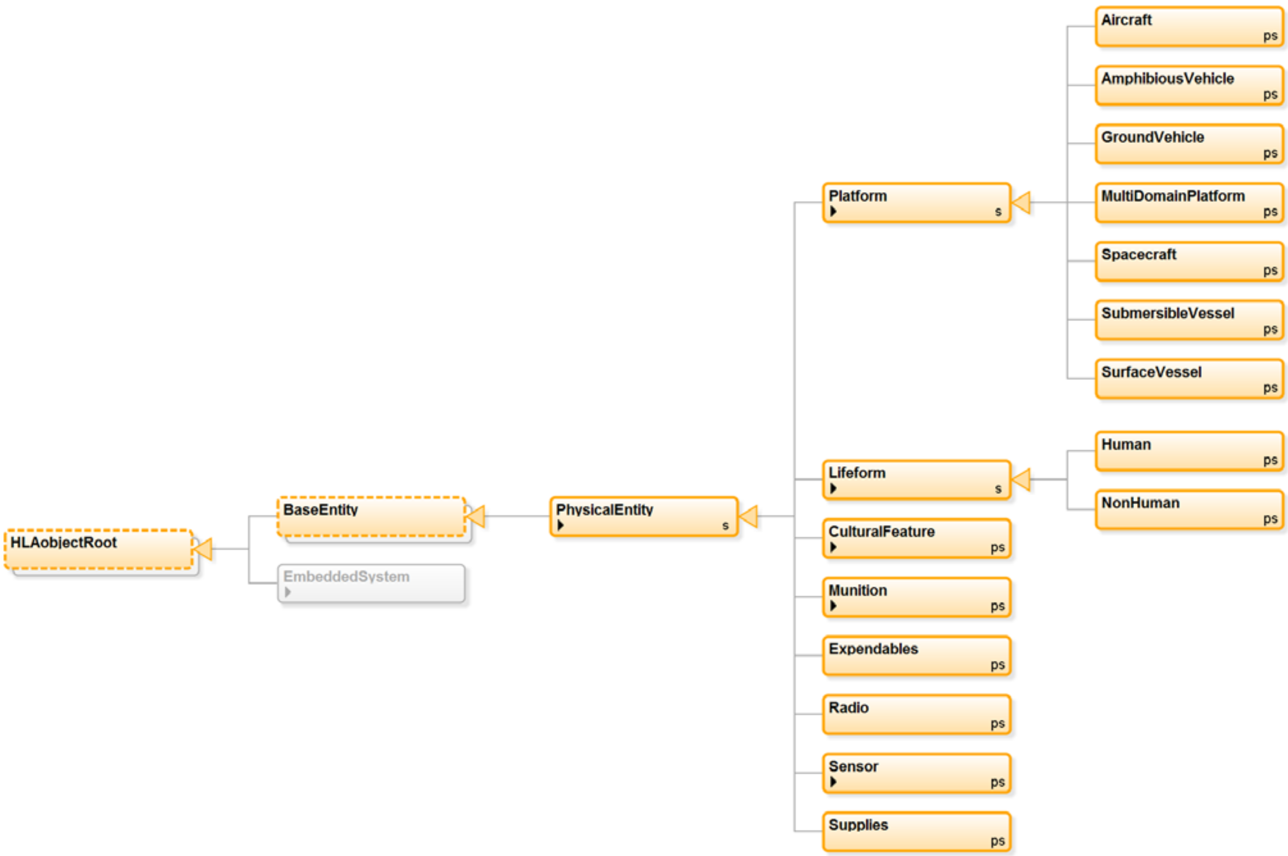

Figure 4 Physical Module Object Class Structure

7.6.1.1 PhysicalEntity Object Class

Entities that can be treated as discrete simulation participants are derived from the PhysicalEntity object class. This object class is a subclass of the BaseEntity class and tailors the behavior of the BaseEntity to include both articulated parts and several status attributes. The status attributes describe the current condition for those capabilities and states generally available to a large variety of physical entities. Articulated parts are attached components of the entity that can exhibit independent motion (such as landing gear or gun turrets). All of the attributes shown in Table 6 shall be treated as optional fields for federates updating instance attributes of this object class or its subclasses.

Table 6 PhysicalEntity Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
AcousticSignatureIndex	SEES	Acoustic Signature Representation Index	1998: 5.3.7.5	0	Index pointer to a data table to define the acoustic signature state for the entity.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
AlternateEntityType	Entity State	Alternate Entity Type	1995: 5.3.3.1.f	BaseEntity. EntityType	Guise function. Allows both sides of an engagement to see their own team members as "friendly force" and their opponents as hostile. The force ID field is used to determine team membership.
ArticulatedParametersArray	Entity State	Articulation Parameters	1995: 5.2.5, 5.3.3.1.n, Annex A	Empty	The specification of articulation parameters for moveable parts and attached parts of an entity.
CamouflageType	Entity State	Entity Appearance	1995: 5.3.3.1.j	Uniform Paint Scheme	Describes the type of camouflage used on the entity.
DamageState	Entity State	Entity Appearance	1995: 5.3.3.1.j	No Damage	Describes the damaged appearance of an entity.
EngineSmokeOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity is creating engine smoke.
FirePowerDisabled	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if an entity's fire power has been disabled.
FlamesPresent	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity is aflame.
ForceIdentifier	Entity State	Force ID	1995: 5.3.3.1.c	Other	Enumeration distinguishing the different teams or sides in an exercise.
HasAmmunitionSupplyCap	Entity State	Capabilities	1995: 5.2.13, 5.3.3.1.m	False	The Entity is able to supply some type of ammunition.
HasFuelSupplyCap	Entity State	Capabilities	1995: 5.2.13, 5.3.3.1.m	False	The Entity is able to supply some type of fuel.
HasRecoveryCap	Entity State	Capabilities	1995: 5.2.13, 5.3.3.1.m	False	The Entity is able to provide recovery (e.g., towing).
HasRepairCap	Entity State	Capabilities	1995: 5.2.13, 5.3.3.1.m	False	The Entity is able to supply repair services.
Immobilized	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if the entity has been immobilized (mobility kill).
InfraredSignatureIndex	SEES	Infrared Signature Representation Index	1998: 5.3.7.5	0	Index pointer to a data table to define the infrared signature state for the entity.
IsConcealed	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity is concealed.
LiveEntityMeasuredSpeed	TSPI	Measured Speed	1998: 7.3.4	0	Entity's own measurement of speed.
Marking	Entity State	Entity Marking	1995: 5.2.15, 5.3.3.1.l	Empty	Character set and the string of characters used to provide display identification for this entity.
PowerPlantOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity's power plant is on.
PropulsionSystemsData	SEES	Propulsion Systems Data	1998: 5.3.7.5	Empty	Basic operational data for the propulsion systems onboard the entity

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
RadarCrossSectionSignatureIndex	SEES	Radar Cross-Section Signature Representation Index	1998: 5.3.7.5	0	Index pointer to a data table to define the radar cross-section signature state for the entity.
SmokePlume Present	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity is creating a smoke plume.
TentDeployed	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity's tent is deployed.
TrailingEffects Code	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity is creating a smoke trail.
VectoringNozzleSystemData	SEES	Vectoring Nozzle System Data	1998: 5.3.7.5	Empty	Basic operational data for the nozzle systems aboard the entity

The PhysicalEntity object class was designed to incorporate most of the DIS Entity State PDU attributes not associated with BaseEntity (see Section 5.3.3.1 and Annex A of IEEE Std 1278.1™-1995 [6]). This combination of features provides PhysicalEntity with the minimum number of attributes needed to represent discrete entities.

As indicated in Table 6, some attributes have been converted from the Entity Appearance in Section 4.4 of SISO-REF-010 [2]. DIS applied many of these only to a particular domain. To maximize interoperability, federates updating instance attributes of one of these object classes shall limit their use to those object classes indicated by a "Yes" in Table 7, when indicated restricted to the enumerators listed. This table makes this guidance easier to follow than incorporating the domain use into Table 6.

Table 7 Domain Appropriateness for PhysicalEntity Attributes

Attribute Name	Aircraft	Amphibious-Vehicle	Ground-Vehicle	Spacecraft	Surface-Vessel	Submersible-Vessel	MultiDomain-Platform	Lifeform	Cultural-Feature	Munition	Expendable	Radio	Sensor	Supplies
CamouflageType	Yes*1	Yes	Yes	Yes*1	Yes*1	Yes*1	Yes	Yes					Yes	
DamageState	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes				Yes	
EngineSmokeOn	Yes	Yes	Yes		Yes		Yes						Yes	
FirePowerDisabled		Yes	Yes				Yes							
FlamesPresent	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes	Yes			Yes	
Immobilized	Yes	Yes	Yes	Yes	Yes	Yes	Yes						Yes	
IsConcealed		Yes	Yes				Yes	Yes					Yes	
PowerPlantOn	Yes	Yes	Yes	Yes	Yes	Yes	Yes			Yes			Yes	
SmokePlumePresent	Yes	Yes	Yes	Yes	Yes	Yes	Yes		Yes				Yes	
TentDeployed		Yes	Yes				Yes						Yes	
TrailingEffectsCode	Yes	Yes	Yes		Yes		Yes			Yes			Yes	

*1) Only the enumerators UniformPaintScheme and GenericCamouflage shall be used.

7.6.1.2 Platform Object Class

The Platform object class is a specialization of PhysicalEntity used to describe status information for vehicles, ships, and aircraft. All of its attributes are derived from the DIS EntityStatePDU (see Section 5.3.3.1 of IEEE Std 1278.1™-1995 [6]). All of the attributes shown in Table 8 shall be treated as optional fields for federates updating instance attributes of this object class or its subclasses.

Table 8 Platform Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AfterburnerOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity's afterburner is on.
AntiCollisionLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if anti-collision lights are on.
BlackOutBrakeLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if blackout brake lights are on.
BlackOutLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if blackout lights are on.
BrakeLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if brake lights are on.
FormationLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if formation lights are on.
HatchState	Entity State	Entity Appearance	1995: 5.3.3.1.j	NotApplicable	Describes the state of the primary hatch.
HeadLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if headlights are on.
InteriorLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if interior lights are on.
LandingLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if landing lights are on.
LauncherRaised	Entity State	Entity Appearance	1995: 5.3.3.1 j	False	True if entity's launcher is raised.
NavigationLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if navigation lights are on.
RampDeployed	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if entity's ramp is deployed.
RunningLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if running lights are on.
SpotLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if spot lights are on.
TailLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if tail lights are on.

The attributes for this object class have been converted from the Entity Appearance in Section 4.4 of SISO-REF-010 [2]. DIS applied many of these only to a particular domain. To maximize interoperability, federates updating instance attributes of one of these object classes shall limit their use to those object classes indicated by a "yes" in Table 9, when indicated restricted to the enumerators listed. This table makes this guidance easier to follow than incorporating the domain use into Table 8.

Table 9 Domain Appropriateness for Platform Attributes

Attribute Name	Aircraft	Amphibious-Vehicle	Ground-Vehicle	Spacecraft	Surface-Vessel	Submersible-Vessel	MultiDomain-Platform
AfterburnerOn	Yes						Yes
AntiCollisionLightsOn	Yes						Yes
BlackOutBrakeLightsOn		Yes	Yes				Yes
BlackOutLightsOn		Yes	Yes				Yes

Attribute Name	Aircraft	Amphibious-Vehicle	Ground-Vehicle	Spacecraft	Surface-Vessel	Submersible-Vessel	MultiDomain-Platform
BrakeLightsOn		Yes	Yes				Yes
FormationLightsOn	Yes						Yes
HatchState		Yes	Yes			Yes*1	Yes
HeadLightsOn		Yes	Yes				Yes
InteriorLightsOn	Yes	Yes	Yes		Yes		Yes
LandingLightsOn	Yes						Yes
LauncherRaised		Yes	Yes				Yes
NavigationLightsOn	Yes						Yes
RampDeployed		Yes	Yes				Yes
RunningLightsOn		Yes			Yes	Yes	Yes
SpotLightsOn	Yes	Yes	Yes		Yes		Yes
TailLightsOn		Yes	Yes				Yes

*1) Only the enumerators NotApplicable, PrimaryHatchIsClosed, and PrimaryHatchIsOpen shall be used.

7.6.1.3 Aircraft Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to the DIS Air domain in that it represents platform entities such as airplanes, balloons, etc. that operate mainly in the air, but that include some limited land operations. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.4 AmphibiousVehicle Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to a cross between DIS Land and DIS Surface domains. It represents platforms that can operate both on the land and the sea. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.5 GroundVehicle Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to the DIS Land domain in that it represents platforms that operate wholly on the surface of the earth. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.6 MultiDomainPlatform Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to the DIS Other domain in that it represents platforms that operate in more than one domain (excluding those combinations explicitly defined as other subclasses of the Platform object class). This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.7 Spacecraft Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to the DIS Space domain in that it represents platforms that operate mainly in space. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.8 SubmersibleVessel Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to the DIS Subsurface domain in that it represents platforms that operate either on the surface of the sea, or beneath it. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.9 SurfaceVessel Object Class

This object class provides an attributeless subclass of Platform used to support DM filtering. It is equivalent to the DIS Surface domain in that it represents platforms that operate wholly on the surface of the sea. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.10 Lifeform Object Class

The Lifeform object class is a specialization of PhysicalEntity used to describe individuals. This object class represents entities of the life form entity kind in DIS. All Lifeform attributes are derived from the DIS Entity State PDU. All attributes shown in Table 10 are optional for federates updating instance attributes of this object class or its subclasses.

Table 10 Lifeform Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
FlashLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if flash lights are on.
StanceCode	Entity State	Entity Appearance	1995: 5.3.3.1.j	NotApplicable	Human behaviors (i.e., running, jumping, etc.).
PrimaryWeaponState	Entity State	Capabilities	1995: 5.2.13, 5.3.3.1.m	NoWeapon	Describes the state of the lifeform’s primary weapon.
SecondaryWeaponState	Entity State	Capabilities	1995: 5.2.13, 5.3.3.1.m	NoWeapon	Describes the state of the lifeform’s secondary weapon.
ComplianceState	Entity State	Entity Appearance	1995: 5.5.5.1.j	Other	The compliance of the lifeform.

7.6.1.11 Human Object Class

This object class provides an attributeless subclass of the Lifeform used to support DM filtering. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.12 NonHuman Object Class

This object class provides an attributeless subclass of the Lifeform used to support DM filtering. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.13 CulturalFeature Object Class

The CulturalFeature object class is a subclass of PhysicalEntity used to describe the physical characteristics of engineering and natural effects such as buildings, craters, bridges, and vehicle tracks. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

Table 11 CulturalFeature Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ExternalLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if exterior lights are on.
InternalHeatSourceOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if interior heat source is on (for infrared viewing).
InternalLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if interior lights are on.

7.6.1.14 Munition Object Class

The DIS protocol allows for two types of munitions. In general, small munitions are tracked at just the launch and impact points using the Fire PDU and Detonation PDU. Simulation developers also have the option of tracking weapons (torpedoes, missiles, etc.) throughout their transit by treating them as independent entities. This latter approach to munition representation is utilized if the representation of its travel between firing and detonation could affect the outcome of the simulation.

The Munition object class is a specialization of PhysicalEntity used to describe the attributes of munitions that act as independent entities. Capabilities equivalent to the DIS Fire PDU and Detonation PDU are now provided by the WeaponFire and MunitionDetonation interaction classes (see Section 7.13.1). All of the attributes shown in Table 12 shall be treated as optional fields for federates updating instance attributes of this object class or its subclasses.

Table 12 Munition Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
LauncherFlashPresent	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if launcher flash is present when munition is fired.

This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.15 Expendables Object Class

The Expendables object class is a subclass of PhysicalEntity used to describe the physical characteristics of countermeasures devices that are dispensed from another entity. Although those devices can be active emitters or passive reflectors of energy, emissions are handled separately. It has no attributes. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.16 Radio Object Class

The Radio object class is an attributeless subclass of PhysicalEntity used to describe the physical characteristics (location, appearance, etc.) of radio installations. It has no attributes. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.1.17 Sensor Object Class

The Sensor object class is a subclass of PhysicalEntity used to describe the physical characteristics (location, appearance, etc.) of sensor installations such as radars. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

Table 13 Sensor Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AntennaRaised	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if the antenna has been raised.
BlackoutLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if blackout lights are on.
LightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if other lights are on.
InteriorLightsOn	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if interior lights are on.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
MissionKill	Entity State	Entity Appearance	1995: 5.3.3.1.j	False	True if mission capability is disabled (e.g., damaged antenna).

7.6.1.18 Supplies Object Class

The Supplies object class is a subclass of PhysicalEntity used to describe the physical characteristics of supplies other than munitions, such as fuel, food and personnel. It has no attributes. This object class is publishable because it qualifies as a leaf node of the RPR FOM.

7.6.2 Interaction Classes

Figure 5 Physical Module Interaction Class Structure

7.6.2.1 Collision Interaction Class

This interaction class provides information on collisions between entities. It includes not only identification for the two object instances involved, but also data required for damage assessment modeling. There shall be a collision interaction issued for each object instance involved in a collision. If a simulation detects that one of its entities has struck another entity, it shall issue a collision interaction. If a simulation receives a collision interaction indicating that one of its entities has been struck, it shall issue a response collision interaction, as long as it has not already issued one for the same collision event.

The form of this interaction class closely follows the layout of the CollisionPDU specified in Section 5.3.3.2 of IEEE Std 1278.1™-1995 [6]. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 14 Collision Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
CollidingObjectIdentifier	Collision	Colliding Entity ID	1995: 5.3.3.2.c	No Specific Entity	The object instance with which the issuing object instance has collided.
IssuingObjectMass	Collision	Mass	1995: 5.3.3.2.g	**(Not Optional)**	Mass in kilograms of the issuing entity.
IssuingObjectVelocityVector	Collision	Velocity	1995: 5.3.3.2.f	**(Not Optional)**	Velocity of the issuing entity at the time the collision is detected.
CollisionType	Collision	Collision Type	1995: 5.3.3.2.e	**(Not Optional)**	Enumeration for collision type.
CollisionLocation	Collision	Location	1995: 5.3.3.2.h	**(Not Optional)**	Relative location with respect to the remote entity with which the issuing entity has collided.
EventIdentifier	Collision	Event ID	1995: 5.3.3.2.d	**(Not Optional)**	ID assigned by the issuing federate to associate related collision events.
IssuingObjectIdentifier	Collision	Issuing Entity ID	1995: 5.3.3.2.b	**(Not Optional)**	The entity that had detected the collision and issued the collision interaction.

7.6.2.2 CollisionElastic Interaction Class

This interaction class is a specialization of the Collision interaction class and should be used instead of the Collision interaction class when it is necessary to provide additional information about elastic collisions. The rules for generating this interaction are the same as for the Collision interaction class. This interaction class closely follows the layout of the CollisionElastic PDU specified in section 5.3.3.3 of IEEE Std 1278.1a™-1998 [7]. When this interaction class is used instead of the Collision interaction class, all fields are mandatory.

Table 15 CollisionElastic Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
CoefficientOfRestitution	Collision Elastic	Coefficient Of Restitution	1998: 5.3.3.3j	**(Not Optional)**	The degree to which energy is conserved (Between 0 and 1).
IntermediateResultXX	Collision Elastic	Intermediate Result XX	1998: 5.3.3.3h	**(Not Optional)**	XX component of the intermediate result matrix.
IntermediateResultXY	Collision Elastic	Intermediate Result XY	1998: 5.3.3.3h	**(Not Optional)**	XY component of the intermediate result matrix.
IntermediateResultXZ	Collision Elastic	Intermediate Result XZ	1998: 5.3.3.3h	**(Not Optional)**	XZ component of the intermediate result matrix.
IntermediateResultYY	Collision Elastic	Intermediate Result YY	1998: 5.3.3.3h	**(Not Optional)**	YY component of the intermediate result matrix.
IntermediateResultYZ	Collision Elastic	Intermediate Result YZ	1998: 5.3.3.3h	**(Not Optional)**	YZ component of the intermediate result matrix.
IntermediateResultZZ	Collision Elastic	Intermediate Result ZZ	1998: 5.3.3.3h	**(Not Optional)**	ZZ component of the intermediate result matrix.
UnitSurfaceNormal	Collision Elastic	Unit Surface Normal	1998: 5.3.3.3i	**(Not Optional)**	Normal to the surface at point of collision (world coordinates).

7.7 Aggregate Module

The Aggregate module provides the object class definition for representing aggregates of entities.

The AggregateEntity class derives from the BaseEntity class defined in the Base module. The Aggregate module also depends on the Base module for datatypes.

7.7.1 Object Classes

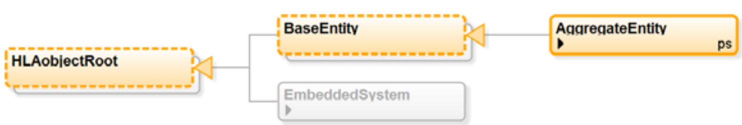

Figure 6 Aggregate Module Object Class Structure

7.7.1.1 AggregateEntity Object Class

An aggregation is a conceptual collection of entities that can be treated as a single unit. An aggregation can save network bandwidth by sending fewer attributes for the aggregate than for individually represented units, or save computational effort by providing information on the group of entities that can be used by a receiver to sort entities. Aggregation can also be used to put hierarchical information about military or functional units or groups on the network. Since aggregation was not part of the original DIS 1995, the attributes for this object class are derived from the Aggregate State PDU introduced in Section 5.3.9.1 of IEEE Std 1278.1a™-1998 [7]. The AggregateEntity object class is a subclass of the BaseEntity object class. All federates updating instance attributes of this object class or its subclasses shall provide the "AggregateState" and "Dimensions" fields. All other attributes in Table 16 shall be treated as optional fields.

Table 16 AggregateEntity Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AggregateMarking	Aggregate State	Aggregate Marking	1998: 5.3.9.1.g	Empty	String representing a unique marking for the aggregate.
AggregateState	Aggregate State	Aggregate State	1998: 5.3.9.1.d	**(Not Optional)**	Enumeration of aggregate state.
Dimensions	Aggregate State	Dimensions	1998: 5.3.9.1.h	**(Not Optional)**	Bounding space, in meters, occupied by the aggregate.
EntityIdentifiers	Aggregate State	Entity ID List, Number of DIS Entities	1998: 5.3.9.1.m, 5.3.9.1.q	Empty	List of ID's for those constituent entities that are also represented by individual object instances.
ForceIdentifier	Aggregate State	Force ID	1998: 5.3.9.1.c	Other	Common force to which the aggregate belongs. Aggregates shall not group opposing forces together.
Formation	Aggregate State	Formation	1998: 5.3.9.1.f	Other	Enumeration reflecting the formation of the aggregate.
NumberOfSilentEntities	Aggregate State	Number of DIS Entities	1998: 5.3.9.1.o	Zero	Number of units in this aggregate. Used to provide size information to those federates who might not be interested in the details of unit composition.
NumberOfVariableDatums	Aggregate State	Number of Variable Datum Records	1998: 5.3.9.1.t	Zero	Number of VariableDatums records.
SilentAggregates	Aggregate State	Silent Aggregate System List	1998: 5.3.9.1.r	Empty	Numbers and types for constituent sub-aggregates that are NOT represented by individual object instances.
SilentEntities	Aggregate State	Silent Entity System List	1998: 5.3.9.1.s	Empty	Numbers, types, and appearances for constituent entities that are NOT represented by individual object instances.
SubAggregateIdentifiers	Aggregate State	Aggregate ID List	1998: 5.3.9.1.p	Empty	List of ID's for those constituent sub-aggregates that are also represented by individual object instances.
VariableDatums	Aggregate State	Variable Datum Records	1998: 5.3.9.1.u	Empty	Data used by entity-level and aggregate-level simulations to transfer control and correlate simulation entities in an aggregate.

Because the AggregateEntity object class is derived from BaseEntity, aggregates behave in some ways as if they were discrete entities. However, instantiations of the AggregateEntity object class represent the conceptual collection itself and not the component entities. The attributes of AggregateEntity identify the aggregation and maintain the organization of the entities incorporated into each unit. This object class is only intended to represent military aggregates. This object class is publishable because it qualifies as a leaf node of the RPR-FOM.

7.8 Synthetic Environment Module

The Synthetic Environment FOM module relates to the simulation of environmental information both under the form of (point, linear, areal) objects and processes.

The EnvironmentalEntity derives from the BaseEntity class of the Base module. The Synthetic Environment module is also dependent on datatypes defined in the Base module.

7.8.1 Object Classes

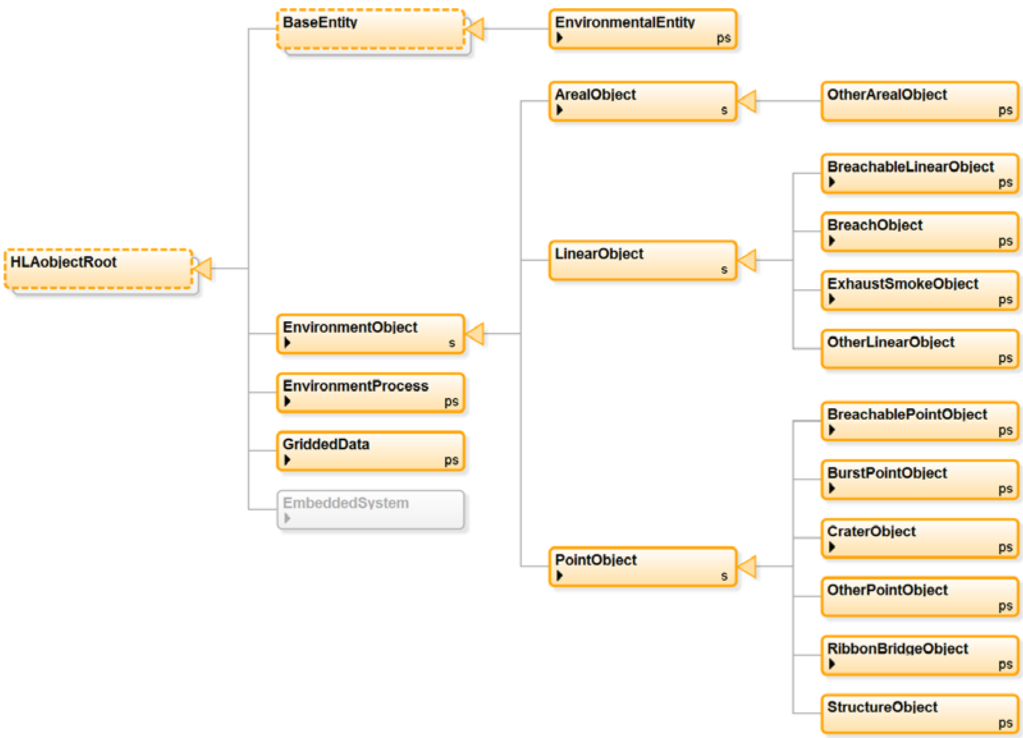

Figure 7 Synthetic Environment Module Object Class Structure

7.8.1.1 EnvironmentalEntity Object Class

The EnvironmentalEntity object class provides a compatible method for environmental representation with the DIS 1278.1 1995 [6]. For simulations that use the simplistic environmental entity representation found in 1278.1, the EnvironmentalEntity class represents the basic type and positional information for an environmental entity. The more advanced environmental features found in DIS 1278.1a-1998 [7] are represented by the EnvironmentalObject object class (see section 7.8.1.2).

The EnvironmentalEntity is a subclass of BaseEntity and as such inherits the Spatial attribute from BaseEntity. This attribute is utilized to locate the environmental entity representation. The inherited EntityType is used to identify the type of EnvironmentalEntity. This subclass provides an additional attribute, OpacityCode, that can be utilized to vary the density of an EnvironmentalEntity that represents, for example, smoke. This object class has no required attributes.

Table 17 EnvironmentalEntity Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OpacityCode	Entity State	Entity Appearance	1995: 5.3.3.1.j	Clear	Specifies the density of an environmental entity.

7.8.1.2 EnvironmentObject Object Class

The EnvironmentObject object class provides for the definition of the states of point, linear, and areal object instances in the simulated environment. This object class supports the simulation of things such as vehicle defilades, trenches, log cribs, abatis, craters, ribbon bridges, rubble, armored vehicle launched bridges, stationary bridges, destructible buildings, anti-tank ditches, wire obstacles, minefield lane markers, minefields, and smoke. In the RPR FOM, the EnvironmentObject object classes and the Synthetic Environment set of transaction interaction classes (see section 7.8.2) support the 1278.1a client/server approach for representing and distributing environmental entity information. The EnvironmentObject object classes support the distribution of environmental entity information by an "environment manager". The transaction interaction classes support sending requests to the environment manager to create, modify, or delete an environment object instance.

The definition of appearance contained in the bits of the DIS Object Appearance fields has been defined in the RPR FOM by separate attributes.

Table 18 EnvironmentObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectIdentifier	Point Object State, Linear Object State, Areal Object State	Object ID	1998: 5.3.11.3.b, 5.3.11.4.b, 5.3.11.5.b	**(Not Optional)**	Identifies the site, application, and object identifier of this object instance. It is used for group addressing in the SIMAN interaction classes.
ReferencedObjectIdentifier	Point Object State, Linear Object State, Areal Object StatePoint Object State	Referenced Object ID	1998: 5.3.11.3.c, 5.3.11.4.c, 5.3.11.5.c	No Referenced Object	Identifies the Synthetic Environment object instance to which this point EnvironmentObject instance is associated.
ForceIdentifier	Point Object State, Linear Object State, Areal Object State	Force ID	1998: 5.3.11.3.e, 5.3.11.4.e, 5.3.11.5.e	**(Not Optional)**	Identifies the force that created or modified the object instance.
ObjectType	Point Object State, Linear Object State, Areal Object StatePoint Object State	Object Type	1998: 5.3.11.3.g, 5.3.11.4.i, 5.3.11.5.g	**(Not Optional)**	Domain, Kind, Category, and Subcategory fields of the DIS Object Type.

7.8.1.3 ArealObject Object Class

The ArealObject object class is a subclass of the EnironmentObject class that provides for the definition of the states of areal objects in the simulated environment. This object class supports the simulation of minefields. The MinefieldObject subclass can be found in the Minefield FOM module.

Table 19 ArealObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
PointsData	Areal Object State	Object Location	1998: 5.3.11.5.l	**(Not Optional)**	Location Physical location (a collection of points) for a point of the object instance relative to the DIS world coordinate system.
PercentComplete	Areal Object State	Object Appearance	1998: 5.3.11.5.h	100% complete	Specifies Describes the percent completion of the areal object.
DamagedAppearance	Areal Object State	Object Appearance	1998: 5.3.11.5.h	No damage	Specifies the damaged appearance of the object instance.
ObjectPreDistributed	Areal Object State	Object Appearance	1998: 5.3.11.5.h	False (created during the exercise)	Specifies whether the object instance was created prior to exercise start or during the exercise.
Deactivated	Areal Object State	Object Appearance	1998: 5.3.11.5.h	False (Active)	Specifies whether the object is deactivated (it has ceased to exist in the synthetic environment).
Smoking	Areal Object State	Object Appearance	1998: 5.3.11.5.h	False	True if object is creating a smoke plume.
Flaming	Areal Object State	Object Appearance	1998: 5.3.11.5.h	False	True if object entity is aflame.

7.8.1.4 OtherArealObject Object Class

This class is a subclass of ArealObject that designates an areal object other than those already defined. This subclass has no independent attributes.

7.8.1.5 LinearObject Object Class

The LinearObject object class is a subclass of EnvironmentObject that provides for the definition of the states of linear objects in the simulated environment. This object class supports the simulation of objects such as anti-tank ditches, wire obstacles, exhaust smoke, and minefield lane markers. This class has no independent attributes. The MinefieldLaneMarkerObject subclass can be found in the Minefield FOM module.

7.8.1.6 BreachableLinearObject Object Class

This object class is a subclass of LinearObject that describes the characteristics of a breachable linear object to include dimensions, location, orientation, and appearance.

Table 20 BreachableLinearObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of a breachable linear object.

7.8.1.7 BreachObject Object Class

This object class is a subclass of LinearObject that describes the characteristics of a breach linear object.

Table 21 BreachObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of a breach linear object.

7.8.1.8 ExhaustSmokeObject Object Class

This object class is a subclass of LinearObject that describes the linear characteristics of exhaust smoke.

Table 22 ExhaustSmokeObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of an exhaust smoke linear object.

7.8.1.9 OtherLinearObject Object Class

This object class is a subclass of LinearObject that designates a linear object other than those already defined. This subclass has no independent attributes.

7.8.1.10 PointObject Object Class

The PointObject object class is a subclass of EnvironmentObject that provides for the definition of the states of point objects in the simulated environment. This object class supports the simulation of objects such as vehicle defilades, trenches, log cribs, abatis, craters, ribbon bridges, rubble, armored vehicle launched bridges, stationary bridges, destructible buildings, and smoke.

Table 23 PointObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Location	Point Object State	Object Location	1998: 5.3.11.3.h	**(Not Optional)**	Location of the object instance relative to the DIS world coordinate system.
Orientation	Point Object State	Object Orientation	1998: 5.3.11.3.ij	**(Not Optional)**	Orientation relative to the world coordinate system, specified by Euler angles.
PercentComplete	Point Object State	Object Appearance	1998: 5.3.11.3.j	100% complete	Specifies the percent completion of the point object.
DamagedAppearance	Point Object State	Object Appearance	1998: 5.3.11.3.j	No damage	Specifies the damaged appearance of the object instance.
ObjectPreDistributed	Point Object State	Object Appearance	1998: 5.3.11.3.j	False (created during the exercise)	Specifies whether the object instance was created prior to exercise start or during the exercise.
Deactivated	Point Object State	Object Appearance	1998: 5.3.11.3.j	False	Specifies whether the object is deactivated (it has ceased to exist in the synthetic environment).

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Smoking	Point Object State	Object Appearance	1998: 5.3.11.3.j	False	True if object is creating a smoke plume.
Flaming	Point Object State	Object Appearance	1998: 5.3.11.3.j	False	True if object is aflame.

7.8.1.11 BreachablePointObject Object Class

This object class is a subclass of PointObject that describes the characteristics of a point object that can be breached such as a log crib, abatis, vehicle defilade, or infantry fighting position.

Table 24 BreachablePointObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BreachedStatus	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the breached appearance of the object instance.

7.8.1.12 BurstPointObject Object Class

This object class is a subclass of PointObject that describes the characteristics of the column(s) of smoke generated from air or ground burst(s).

Table 25 BurstPointObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
PercentOpacity	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the opacity of the smoke; 0% opaque to 100% opaque.
CylinderSize	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the radius of the cylinder approximating an individual smoke burst. For multiple bursts, the center bottom of each cylinder is calculated based on the model used to represent the multiple bursts.
CylinderHeight	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the height of the cylinder approximating an individual smoke burst. For multiple bursts, the center bottom of each cylinder is calculated based on the model used to represent the multiple bursts.
NumberOfBursts	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the number of bursts in the instance of tactical smoke.
ChemicalContent	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the chemical content of the smoke (e.g., white phosphorous).

7.8.1.13 CraterObject Object Class

This object class is a subclass of PointObject that describes the size of a crater.

Table 26 CraterObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
CraterSize	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the diameter of the crater.

7.8.1.14 RibbonBridgeObject Object Class

This object class is a subclass of PointObject that describes the size of a ribbon bridge in terms of bridge segments.

Table 27 RibbonBridgeObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
NumberOfSegments	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the number of segments composing the ribbon bridge.

7.8.1.15 StructureObject Object Class

This object class is a subclass of PointObject that designates a point object for a building /structure, building rubble, stationary bridge, or Armored Vehicle Launch Bridge (ALVB). This subclass has no independent attributes.

7.8.1.16 OtherPointObject Object Class

This object class is a subclass of PointObject that designates a point object other than those already defined. This subclass has no independent attributes.

7.8.1.17 EnvironmentProcess Object Class

This object class is used to distribute environmental information. This can include natural and man-made environments. The type of process is defined by the *Type* attribute using the values defined in SISO-REF-010 [2]. An environmental process is composed of a series of EnvironmentRecStruct. EnvironmentRecStruct can represent geometries or states. Each environmental process has a bounding volume EnvironmentRecStruct that encompasses all of the geometries that make up the environmental process. Each component of the process has a geometry record and a state record.

Table 28 EnvironmentProcess Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
ProcessIdentifier	Environmental Process	Environmental Process ID	1998: 5.3.11.1.b	**(Not Optional)**	Identifier of the environmental process.
Type	Environmental Process	Environment Type	1998: 5.3.11.1.c	**(Not Optional)**	Environmental Process Type as defined in SISO-REF-010 [2].
ModelType	Environmental Process	Model Type	1998: 5.3.11.1.d	**(Not Optional)**	Model used for generating this Environmental Process. Defined in 1278.1a as being exercise specific.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
EnvironmentProcessActive	Environmental Process	Environment Status	1998: 5.3.11.1.e	**(Not Optional)**	Specifies the status of the Environmental Process. Generally used to indicate that the process is active or inactive.
SequenceNumber	Environmental Process	Sequence Number	1998: 5.3.11.1.g	*EP_NO_SEQUENCE*	Defines an optional field used to support update sequencing.
EnvironmentRecData	Environmental Process	Environment Record	1998: 5.3.11.1.f, 5.3.11.1.h	**(Not Optional)**	Defines a collection of EnvironmentRecVariantStruct. It can represent a geometry or state record.

7.8.1.18 GriddedData Object Class

This object class is used to distribute gridded information. The GriddedData object class can be used to provide information on natural and man-made environmental effects. The dimensions of the grid are defined by the GridAxisInfo. Each axis has a GridAxisInfo record. The grid can contain one or more data values at each grid point.

Table 29 GriddedData Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
GridIdentifier	Gridded Data	Environmental Simulation Application ID	1998: 5.3.11.2.b	**(Not Optional)**	Identifier of the environmental simulation application.
CoordinateSystem	Gridded Data	Coordinate System	1998: 5.3.11.2.f	**(Not Optional)**	Specifies the coordinate system used to locate the data grid.
NumberOfGridAxes	Gridded Data	Number of Grid Axes	1998: 5.3.11.2.g	**(Not Optional)**	Specifies the number of grid axes used to define the data grid (e.g., three grid axes for an x, y, z coordinate system).
ConstantGrid	Gridded Data	Constant Grid	1998: 5.3.11.2.h	**(Not Optional)**	Specifies whether the grid axes remain constant for the life of the data grid.
EnvironmentType	Gridded Data	Environment Type	1998: 5.3.11.2.i	**(Not Optional)**	Identifies the type of environmental entity being described.
Orientation	Gridded Data	Orientation	1998: 5.3.11.2.j	**(Not Optional)**	Specifies the orientation of the data grid, with Euler angles.
SampleTime	Gridded Data	Sample Time	1998: 5.3.11.2.k	**(Not Optional)**	Specifies the valid time of the environmental data sample.
TotalValues	Gridded Data	Total Values	1998: 5.3.11.2.l	**(Not Optional)**	Specifies the number of data values that make up this grid.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (If Optional)	Definition
VectorDimension	Gridded Data	Vector Dimension	1998: 5.3.11.2.m	**(Not Optional)**	Specifies the number of data values at each grid point. VectorDimension shall be one for scalar data, and shall be greater than one when multiple enumerated environmental data values are sent for each grid point (e.g., u, v, w wind components have VectorDimension = 3).
GridAxisInfo	Gridded Data	Grid Axis Descriptor	1998: 5.3.11.2.n	**(Not Optional)**	Defines detailed information on the axes of the data grid including start point and spacing. There may be more than one axis.
GridDataInfo	Gridded Data	Grid Data	1998: 5.3.11.2.o	**(Not Optional)**	Defines the actual "data" in the data grid.

7.8.2 Interaction Classes

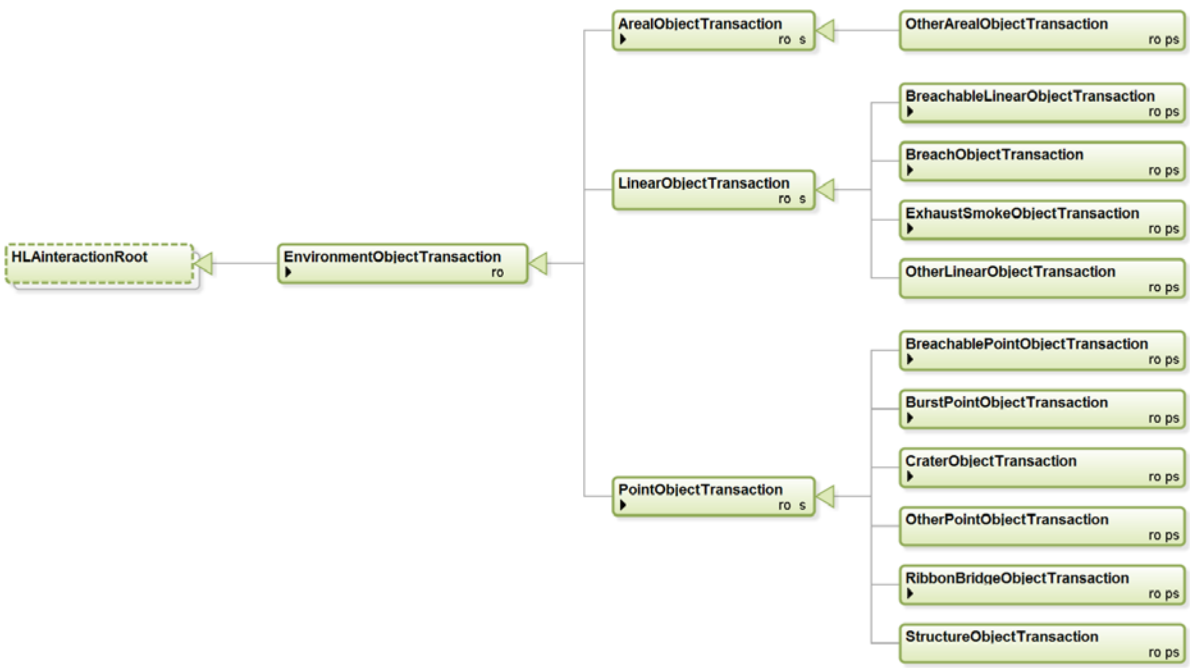

Figure 8 Synthetic Environment Module Interaction Class Structure

The Synthetic Environment interaction classes, along with the EnvironmentObject object class and its subclasses (see section 7.8.1.2) support the DIS object approach for representing and distributing environmental information. The point/linear/areal object classes support the distribution of environmental information by the "environment manager". The transaction interaction classes are sent by federates to request that environment manager federate(s) create, modify, or delete an environment object.

7.8.2.1 EnvironmentObjectTransaction Interaction Class

This interaction class supports the conveyance of common parameters associated with requests published to environmental manager federate(s) to create, modify, or delete an environmental object instance. This interaction class incorporates the parameters RequestingIdentifier and ReceivingIdentifier that specify the DIS simulation application providing the create/modify/delete request of the synthetic environment object and the intended recipient DIS simulation application. Within an HLA RPR-FOM federation the associated object instance, created/modified/deleted in response to the interaction, does not include equivalent parameters. Where a federate, such as an HLA to DIS gateway, requires a knowledge of the RequestingIdentifier/ReceivingIdentifier, that federate will need to subscribe to the interaction class in addition to the associated object class.

Table 30 EnvironmentObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectIdentifier	Object State (Areal, Linear, Point)	Object ID	1998: 5.3.11.3.b, 5.3.11.4.b, 5.3.11.5.b	**(Not Optional)**	Identifies the unique identifier of the object instance.
ReferencedObjectIdentifier	Object State (Areal, Linear, Point)	Referenced Object ID	1998: 5.3.11.3.c, 5.3.11.4.c, 5.3.11.5.c	**(Not Optional)**	Identifies the synthetic environment object instance to which this EnvironmentObject instance is associated.
ForceIdentifier	Object State (Areal, Linear, Point)	Force ID	1998: 5.3.11.3.e, 5.3.11.4.e, 5.3.11.5.e	**(Not Optional)**	Identifies the force that created or modified the environment object instance.
ObjectType	Object State (Areal, Linear, Point)	Object Type	1998: 5.3.11.3.g, 5.3.11.4.i, 5.3.11.5.g	**(Not Optional)**	Domain, Kind, Category and Subcategory fields of the DIS Object Type.
RequestingIdentifier	Object State (Areal, Linear, Point)	Requestor ID / Requester ID	1998: 5.3.11.3.k, 5.3.11.4 g, 5.3.11.5.j	**(Not Optional)**	Identifies the simulation application sending the EnvironmentObjectTransaction interaction.
ReceivingIdentifier	Object State (Areal, Linear, Point)	Receiving ID	1998: 5.3.11.3.l, 5.3.11.4 h, 5.3.11.5.k	**(Not Optional)**	Identifies the simulation application receiving the EnvironmentObjectTransaction interaction.

7.8.2.2 ArealObjectTransaction Interaction Class

This interaction class is a subclass of EnvironmentObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete an ArealObject instance. This interaction class supports the simulation of minefields. The MinefieldObjectTransaction subclass can be found in the Minefield FOM module.

Table 31 ArealObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
PointsData	Areal Object State	Number of Points and Object Location(s)	1998: 5.3.11.5.i	**(Not Optional)**	Physical location (a collection of points) of the object instance.
PercentComplete	Areal Object State	Object Appearance Bits 0-7	1998: 5.3.11.5.h SISO-REF-010: 16.11.2.1	100% complete	Specifies the percent completion of the areal object (0...100%).
DamagedAppearance	Areal Object State	Object Appearance Bits 8-9	1998: 5.3.11.5.h SISO-REF-010: 16.11.2.1	No damage	Specifies the damaged appearance of the areal object.
ObjectPreDistributed	Areal Object State	Object Appearance Bit 10	1998: 5.3.11.5.h SISO-REF-010: 16.11.2.1	False (created during the exercise)	Specifies whether the areal object was created prior to exercise start or during the exercise.
Deactivated	Areal Object State	Object Appearance Bit 11	1998: 5.3.11.5.h SISO-REF-010: 16.11.2.1	False (Active)	Specifies whether the areal object is currently active or deactivated (it has ceased to exist in the synthetic environment).
Smoking	Areal Object State	Object Appearance Bit 12	1998: 5.3.11.5.h SISO-REF-010: 16.11.2.1	False	Specifies whether smoke is rising from the areal object.
Flaming	Areal Object State	Object Appearance Bit 13	1998: 5.3.11.5.h SISO-REF-010: 16.11.2.1	False	Specifies whether flames are rising from the areal object.

7.8.2.3 OtherArealObjectTransaction Interaction Class

This interaction class is a subclass of ArealObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete areal objects other than minefields. This subclass has no independent parameters.

7.8.2.4 LinearObjectTransaction Interaction Class

This interaction class is a subclass of EnvironmentObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete linear objects such as anti-tank ditches, wire obstacles, exhaust smoke, and minefield lane markers in the simulated environment. The MinefieldLaneMarkerObjectTransaction subclass can be found in the Minefield FOM module.

7.8.2.5 BreachableLinearObjectTransaction Interaction Class

This interaction class is a subclass of LinearObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a BreachableLinearObject instance.

Table 32 BreachableLinearObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of a breachable linear object.

7.8.2.6 BreachObjectTransaction Interaction Class

This interaction class is a subclass of LinearObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a BreachObject instance.

Table 33 BreachObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of a breach linear object.

7.8.2.7 ExhaustSmokeObjectTransaction Interaction Class

This interaction class is a subclass of LinearObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete an ExhaustSmokeObject instance.

Table 34 ExhaustSmokeObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Describes the opacity, chemical content, and an indication of attachment to a vehicle for exhaust smoke.

7.8.2.8 OtherLinearObjectTransaction Interaction Class

This interaction class is a subclass of LinearObjectTransaction class that supports the conveyance of common parameters associated with requests sent to environmental manager federates(s) to create, modify, or delete LinearObject instances not represented by other sub-classes of LinearObject. This subclass has no independent parameters.

7.8.2.9 PointObjectTransaction Interaction Class

This interaction class is a subclass of EnvironmentObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete PointObject instances in the simulated environment.

Table 35 PointObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Location	Point Object State	Object Location	1998: 5.3.11.3.h	**(Not Optional)**	Location of the point object instance in the simulated world.
Orientation	Point Object State	Object Orientation	1998: 5.3.11.3.i	**(Not Optional)**	Orientation of the point object instance, specified by Euler Angles.
PercentComplete	Point Object State	Object Appearance Bits 0-7	1998: 5.3.11.3.j SISO-REF-010: 16.11.2.1	100% complete	Specifies the percent completion of the point object (0...100%).
DamagedAppearance	Point Object State	Object Appearance Bits 8-9	1998: 5.3.11.3.j SISO-REF-010: 16.11.2.1	No damage	Specifies the damaged appearance of the point object.
ObjectPreDistributed	Point Object State	Object Appearance Bit 10	1998: 5.3.11.3.j SISO-REF-010: 16.11.2.1	False (created during the exercise)	Specifies whether the point object was created prior to exercise start or during the exercise.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Deactivated	Point Object State	Object Appearance Bit 11	1998: 5.3.11.3.j SISO-REF-010: 16.11.2.1	False (Active)	Specifies whether the point object is currently active or deactivated (it has ceased to exist in the synthetic environment).
Smoking	Point Object State	Object Appearance Bit 12	1998: 5.3.11.3.j SISO-REF-010: 16.11.2.1	False	Specifies whether smoke is rising from the point object.
Flaming	Point Object State	Object Appearance Bit 13	1998: 5.3.11.3.j SISO-REF-010: 16.11.2.1	False	Specifies whether flames are rising from the point object.

7.8.2.10 BreachablePointObjectTransaction Interaction Class

This interaction class is a subclass of PointObjectTransaction class that supports the conveyance of parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a BreachablePointObject instance.

Table 36 BreachablePointObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BreachedStatus	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Description of a breachable point object.

7.8.2.11 BurstPointObjectTransaction Interaction Class

This interaction class is a subclass of PointObjectTransaction class that supports the conveyance of parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a BurstPointObject instance.

Table 37 BurstPointObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
PercentOpacity	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the opacity of smoke; 0% opaque to 100% opaque.
CylinderSize	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the radius of the cylinder approximating an individual smoke burst. For multiple bursts, the center bottom of each cylinder is calculated based on the model used to represent the multiple bursts.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
CylinderHeight	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the height of the cylinder approximating an individual smoke burst. For multiple bursts, the center bottom of each cylinder is calculated based on the model used to represent the multiple bursts.
NumberOfBursts	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the number of bursts in the instance of tactical smoke. Describes the number of bursts in the instance of tactical smoke.
ChemicalContent	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the chemical content of the smoke (e.g., white phosphorous).

7.8.2.12 CraterObjectTransaction Interaction Class

This interaction class is a subclass of PointObjectTransaction class that supports the conveyance of parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a CraterObject instance.

Table 38 CraterObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
CraterSize	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the diameter of the crater.

7.8.2.13 OtherPointObjectTransaction Interaction Class

This interaction class is a subclass of PointObjectTransaction class that supports the conveyance of parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete point objects whose class is not defined by other subclasses of the PointObject class. This subclass has no independent parameters.

7.8.2.14 RibbonBridgeObjectTransaction Interaction Class

This interaction class is a subclass of PointObjectTransaction class that supports the conveyance of parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a RibbonBridgeObject instance.

Table 39 RibbonBridgeObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
NumberOfSegments	Point Object State	Object Appearance	1998: 5.3.11.3.j	**(Not Optional)**	Specifies the number of segments composing the ribbon bridge.

7.8.2.15 StructureObjectTransaction Interaction Class

This interaction class is a subclass of PointObjectTransaction class that supports the conveyance of parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a StructureObject instance. This subclass has no independent parameters.

7.9 Minefield Module

Minefields are described at both an aggregate and individual level simultaneously. In addition, depending on the number of mines within a federation execution, there are two modes for exchanging the data of individual mines within minefields. For federation executions that have a relatively small number of mines, the minefield data is represented using Minefield object class instances only. For federation executions with a large number of mines, a combination of Minefield object class instances and a set of interaction classes are used to specify a Query Response Protocol (QRP). For both the object class mode and the QRP mode, the aggregate data is defined via the Minefield object class.

The MinefieldData class derives from the EmbeddedSystem class of the Base module. The MinefieldObject and MinefieldLaneMarkerObject classes derive from the ArealObject and LinearObject classes of the Synthetic Environment module, respectively. The Minefield module also depends on the Base module for datatypes.

7.9.1 Object Classes

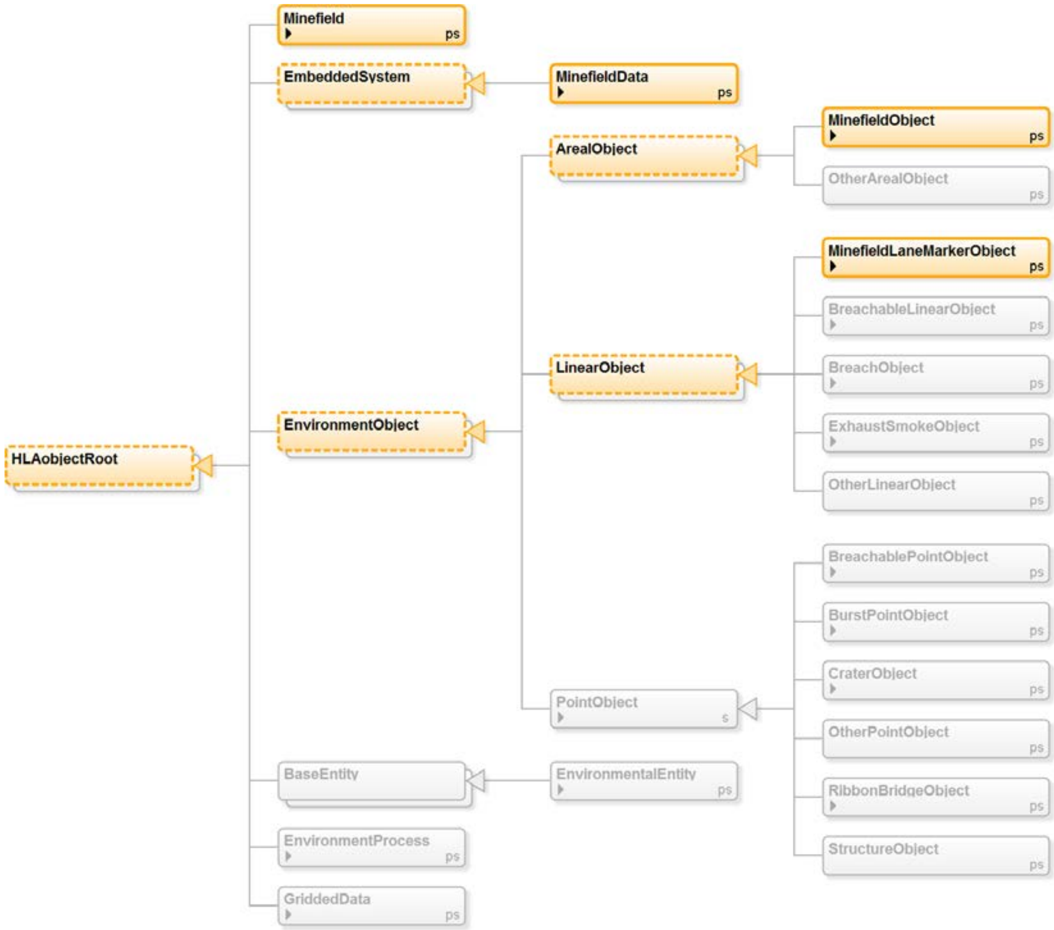

Figure 9 Minefield Module Object Class Structure

7.9.1.1 Minefield Object Class

The Minefield object class represents an aggregate minefield. It describes the general characteristics of a minefield such as its location, orientation, perimeter, and the types of mines contained within it.

All federates updating instance attributes of this object class or its subclasses shall provide the following attributes: MinefieldIdentifier, MinefieldLocation, MinefieldOrientation, MinefieldSequenceNumber, MinefieldType, and MineTypes. The remaining attributes are optional.

Table 40 Minefield Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ActiveStatus	Minefield State	Appearance	1998: 5.3.10.1k	True (Inactive)	Whether the minefield is switched on or off. A minefield that has been switched off is still present, but will not detonate.
ForceIdentifier	Minefield State	Force ID	1998: 5.3.10.1e	Other	Identifies the force to which the minefield belongs
Lane	Minefield State	Appearance	1998: 5.3.10.1k	Inactive	Specifies whether the minefield has an active lane

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
MinefieldAppearanceType	Minefield State	Appearance	1998: 5.3.10.1k	Mixed AntiPersonal AntiTank	Specifies the appearance information needed for displaying the symbology of the minefield as a doctrinal minefield graphic.
MinefieldIdentifier	Minefield State	Minefield ID	1998: 5.3.10.1b	**(Not Optional)**	Uniquely identifies this minefield instance in association with the federate's site and application.
MinefieldLocation	Minefield State	Minefield Location	1998: 5.3.10.1i	**(Not Optional)**	Specifies the location of the center of the minefield
MinefieldOrientation	Minefield State	Minefield Orientation	1998: 5.3.10.1j	**(Not Optional)**	Specifies the orientation of the minefield, with Euler angles.
MinefieldType	Minefield State	Minefield Type	1998: 5.3.10.1g	**(Not Optional)**	Specifies the minefield type
MineTypes	Minefield State	Mine Type	1998: 5.3.10.1o	**(Not Optional)**	Specifies the type of each mine contained within the minefield
PerimeterPointCoordinates	Minefield State	Perimeter Point Coord.	1998: 5.3.10.1n	*Minefield Location (not needed for single mines)*	Specifies the location (X, Y) of each perimeter point, relative to the minefield location
ProtocolMode	Minefield State	Protocol Mode	1998: 5.3.10.1l	Heartbeat Mode	Specifies which mode is being used to communicate the minefield data.
State	Minefield State	Appearance	1998: 5.3.10.1k	**(Not Optional)**	Specifies whether the minefield has ceased to exist in the synthetic environment.

7.9.1.2 MinefieldData Object Class

The MinefieldData object class provides data about collections of mines within a minefield on an individual mine basis. The EmbeddedSystem base class is represented as a scaffolding class in the Minefield FOM module for HLA 2010. The full description of the EmbeddedSystem object class can be found in the Base Module (see section 7.5.1.2). Each MinefieldData object describes a single type of mine. Therefore, a minefield with multiple mine types requires at least one MinefieldData object for each mine type. The relative location/orientation attributes in this object are assumed to be at the location of the minefield itself unless specifically set via an Update Attribute Values call.

Table 41 MinefieldData Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
GroundBurialDepthOffset	Minefield Data	Ground Burial Depth Offset	1998: 5.3.10.3o	{0,0,0} (No fuse, No fuse, No anti-handling device)	Specifies the offset of the origin of the mine coordinate system with respect to the ground surface. Ground burial depth offset is specified as a positive measurement in meters below the ground surface along the up vector. If any of the three burial depth offsets is published, ground burial depth offset and mine orientation shall also be published. If a terrain database does not include the ground surface (such as under a water feature), an arbitrary ground burial depth offset shall be specified.
Fusing	Minefield Data	Fusing	1998: 5.3.10.3w	0	Specifies the primary and secondary fuse and anti-handling device for each mine.
MineEmplacementTime	Minefield Data	Mine Emplacement Time	1998: 5.3.10.3u	0	Specifies the real-world Coordinated Universal Time Constant (UTC) emplacement time of the mine.
MineEntityIdentifier	Minefield Data	Mine Entity ID	1998: 5.3.10.3v	**(Not Optional)**	Identifies the mine entity identifier. The MineEntityIdentifier in conjunction with the MinefieldIdentifier form the unique identifier for each mine.
MinefieldIdentifier	Minefield Data	Minefield ID	1998: 5.3.10.3b	**(Not Optional)**	Identifies the minefield to which the mines belong
MineLocation	Minefield Data	Mine Location	1998: 5.3.10.3n	{0,0,0} (X, Y, Z)	Specifies the location of the mine relative to the minefield location.
MineOrientation	Minefield Data	Mine Orientation	1998: 5.3.10.3r	{0,0,0} (Psi, Theta, Phi)	Specifies the orientation of the center axis direction of fire of the mine, relative to the minefield Coordinate System. If any of the three burial depth offsets is published, mine orientation shall also be published.
MineType	Minefield Data	Mine Type	1998: 5.3.10.3l	**(Not Optional)**	Specifies the type of mine for the collection of mines described by the MinefieldData object.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
NumberTripDetonationWires	Minefield Data	Number of Wires	1998: 5.3.10.3z	*Required if tripwires are used, otherwise the default value is Empty.*	Specifies the number of trip detonation wires that exist for each mine. I sub n designates the trip wire count for mine n of N. Default is that no mine has a trip wire (all counts are equal to 0), but if any one mine has a trip wire, all counts shall be published.
NumberWireVertices	Minefield Data	Number of Vertices	1998: 5.3.10.3aa	*Required if the sum of values in NumberTrip Detonation Wires is greater than 0, otherwise the default value is Empty*	Specifies the number of vertices for each trip wire. A set of I sub n vertex counts is given for each mine n of N, if I sub n > 0. If I sub n is 0, the vertex count set is empty. Each non empty set of vertex counts (I sub n > 0) follows the previous non empty set. J sub i designates the vertex count of tripwire i of I sub n. The size of this attribute is the sum of the values in the NumberTripDetonationWires attribute.
PaintScheme	Minefield Data	Paint Scheme	1998: 5.3.10.3y	Other	Specifies the camouflage scheme/color of the mine.
Reflectance	Minefield Data	Reflectance	1998: 5.3.10.3t	0	Specifies the local dielectric difference between the mine and the surrounding soil.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ScalarDetectionCoefficient	Minefield Data	Scalar Detection Coefficient	1998: 5.3.10.3x	0	Specifies the coefficient to be utilized to insure proper correlation between detectors located on different simulation platforms. In statistically based detection system applications, the detection system simulation will generally compare a random number against an internally calculated probability of detection. M (number of sensors) coefficients are specified for each of the N mines; MxN values formatted with M values for mine 1 followed by M values for mine 2, etc. If not published, the default value of zero indicates that the mines are always detected. If any one value is published all MxN values shall be published.
SensorTypes	Minefield Data	Sensor Types	1998: 5.3.10.3m	Other	In QRP mode, specifies the requesting sensor types which were specified in the minefield query whereas in Heartbeat mode, specifies the sensor types that are being served by the minefield.
SnowBurialDepthOffset	Minefield Data	Snow Burial Depth Offset	1998: 5.3.10.3q	0	Specifies the offset of the origin of the mine coordinate system with respect to the snow surface. Snow burial depth offset is specified as a positive measurement in meters below the snow surface along the up vector. If any of the three burial depth offsets is published, ground burial depth offset and mine orientation shall also be published. The value of the snow burial depth offset shall be set to the value of the ground burial depth offset to indicate there is no snow for the mine to be buried in.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ThermalContrast	Minefield Data	Thermal Contrast	1998: 5.3.10.3s	0	Specifies the temperature difference between the mine and the surround soil in degrees Centigrade. In the case of a buried mine, the delta temperature shall be measured between the ground surface above the mine and the surrounding ground surface temperature.
WaterBurialDepthOffset	Minefield Data	Water Burial Depth Offset	1998: 5.3.10.3p	0	Specifies the offset of the origin of the mine coordinate system with respect to the water surface. Water burial depth offset is specified as a positive measurement in meters below the water's surface along the up vector. If any of the three burial depth offsets is published, ground burial depth offset and mine orientation shall also be published. The value of the water burial depth offset shall be set to the value of the ground burial depth offset to indicate there is no water for the mine to be buried in.
WireVertices	Minefield Data	Vertex	1998: 5.3.10.3ab	*Required if the sum of values in NumberWireVertices is greater than 0, otherwise the default values is Empty.*	Specifies the locations of vertices in a trip wire. A set of J sub i locations of vertices is given for each trip wire i of I sub n, if I sub n > 0 and J sub i > 0. If J sub i is 0, the set is empty. Each non empty set of locations of vertices (I sub n > 0 and J sub i > 0) follows the previous non empty set. The size of this attribute is the sum of the values in the NumberWireVertices attribute.

7.9.1.3 MinefieldLaneMarkerObject Object Class

This object class is a subclass of LinearObject that describes the characteristics of the minefield segments. The LinearObject base class is represented as a scaffolding class in the Minefield FOM module for HLA 2010. The full description of the LinearObject object class can be found in the Synthetic Environment Module (see section 7.8.1.5).

Table 42 MinefieldLaneMarker Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	LinearObject State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of a lane marker linear object.

7.9.1.4 MinefieldObject Object Class

This object class is a subclass of the ArealObject class that describes the quantity of mines in a minefield and the extent of breaching of a minefield. The ArealObject base class is represented as a scaffolding class in the Minefield FOM module for HLA 2010. The full description of the ArealObject object class can be found in the Synthetic Environment Module (see section 7.8.1.3).

Table 43 MinefieldObject Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BreachedStatus	Areal Object State	Object Appearance	1998: 5.3.11.5.h	**(Not Optional)**	Specifies the breached appearance of the minefield.
MineCount	Areal Object State	Object Appearance	1998: 5.3.11.5.h	**(Not Optional)**	Specifies the quantity of mines in the minefield.

7.9.2 Interaction Classes

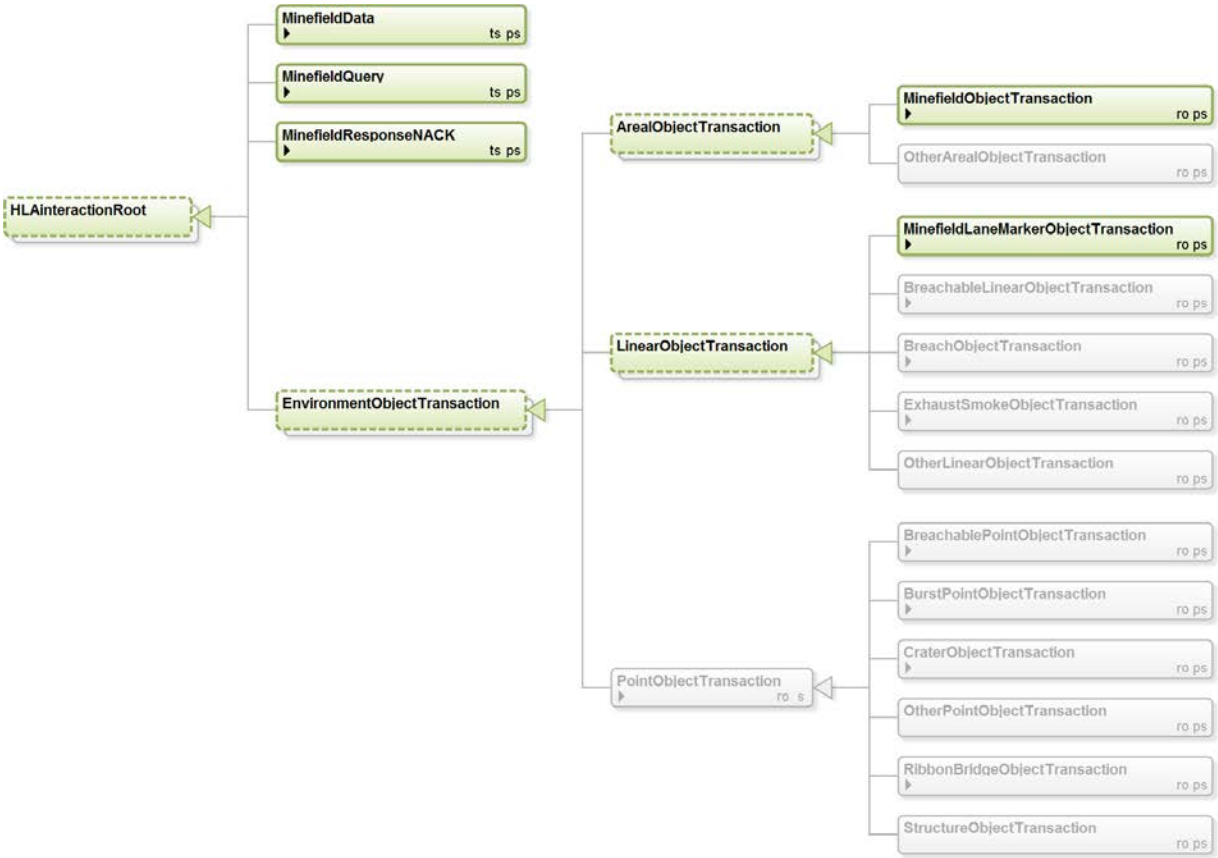

Figure 10 Minefield Module Interaction Class Structure

The Minefield interaction classes support the simulation of a large number of mines within a federation execution. The Minefield object class is still used to describe the aggregate state information for a minefield. However, in place of the MinefieldData object class, a Query Response Protocol (QRP) is defined using the MinefieldQuery, MinefieldData, and MinefieldResponseNACK interaction classes.

7.9.2.1 MinefieldQuery Interaction Class

The MinefieldQuery interaction class is used to request specific data about a minefield. It specifies the types of mines, the type of information about the mines (e.g., reflectance and thermal contrast) and a spatial region that selects the mines.

Table 44 MinefieldQuery Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
MinefieldIdentifier	Minefield Query	Minefield ID	1998: 5.3.10.2.b	**(Not Optional)**	Identifies the minefield to which this query is addressed.
PerimeterPoints	Minefield Query	Requested Perimeter Points	1998: 5.3.10.2.i	**(Not Optional)**	Specifies the location of each perimeter point in the requested area, relative to the minefield location.
QueryFusing	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether fusing is requested.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
QueryMineOrientation	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether orientation is requested.
QueryGroundBurialDepthOffset	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether ground burial depth offset is requested.
QueryMineEmplacementAge	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether emplacement age is requested.
QueryPaintScheme	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether paint scheme is requested.
QueryReflectance	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether reflectance is requested.
QueryScalarDetectionCoefficient	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether scalar detection coefficient is requested.
QuerySnowBurialDepthOffset	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether snow burial depth offset is requested.
QueryThermalContrast	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether thermal contrast is requested.
QueryTripDetonationWire	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether trip detonation wire is requested.
QueryWaterBurialDepthOffset	Minefield Query	Data Filter	1998: 5.3.10.2.g	False	Specifies whether water burial depth offset is requested.
RequestingEntityIdentifier	Minefield Query	Requesting Entity ID	1998: 5.3.10.2.c	**(Not Optional)**	Identifies the entity that requested the information from the minefield federate.
RequestIdentifier	Minefield Query	Request ID	1998: 5.3.10.2.d	**(Not Optional)**	Identifies the minefield query request.
RequestedMineType	Minefield Query	Requested Mine Type	1998: 5.3.10.2.h	**(Not Optional)**	Identifies the type of mine being queried by the requesting federate.
SensorTypes	Minefield Query	Sensor Types	1998: 5.3.10.2.j	**(Not Optional)**	Specifies the types of sensors requesting the data.

7.9.2.2 MinefieldData Interaction Class

The MinefieldData interaction class provides the mine data that satisfies a MinefieldQuery interaction specification. It shall contain only mines of the requested type and within the requested region. The MinefieldData interaction shall be sent even if no mines match the query. Multiple MinefieldData interactions may be utilized to satisfy the query. The MinefieldData interaction shall include all of the data requested in the corresponding MinefieldQuery interaction. The RequestingEntityIdentifier, RequestIdentifier, and RecordSequenceNumber uniquely identify a MinefieldData interaction.

Table 45 MinefieldData Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
GroundBurialDepthOffset	Minefield Data	Ground Burial Depth Offset	1998: 5.3.10.3.o	0	Specifies the offset of the origin of the mine coordinate system with respect to the ground surface. Ground burial depth offset is specified as a positive measurement in meters below the ground surface along the up vector. If any of the three burial depth offsets is published, ground burial depth offset and mine orientation shall also be published. If a terrain database does not include the ground surface (such as under a water feature), an arbitrary ground burial depth offset shall be specified.
Fusing	Minefield Data	Fusing	1998: 5.3.10.3.w	{0,0,0} (No fuse, No fuse, No anti-handling device)	Specifies the primary and secondary fuse and anti-handling device for each mine.
MineEmplacementTime	Minefield Data	Mine Emplacement Time	1998: 5.3.10.3.u	0	Specifies the real-world Coordinated Universal Time (UTC) emplacement time of the mine.
MineEntityIdentifier	Minefield Data	Mine Entity ID	1998: 5.3.10.3.v	**(Not Optional)**	Identifies the mine entity identifier. The MineEntityID in conjunction with the MinefieldID form the unique identifier for each mine.
MinefieldIdentifier	Minefield Data	Minefield ID	1998: 5.3.10.3.b	**(Not Optional)**	Identifies the minefield to which the mines belong.
MineLocation	Minefield Data	Mine Location	1998: 5.3.10.3.n	**(Not Optional)**	Specifies the location of the mine relative to the minefield location.
MineOrientation	Minefield Data	Mine Orientation	1998: 5.3.10.3.r	{0,0,0} (Psi, Theta, Phi)	Specifies the orientation of the center axis direction of fire of the mine, relative to the minefield Coordinate System. If any of the three burial depth offsets is published, mine orientation shall also be published.
MineType	Minefield Data	Mine Type	1998: 5.3.10.3.l	**(Not Optional)**	Specifies the type of each mine contained within the minefield interaction.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
NumberOfRecords	Minefield Data	Number of Mines	1998: 5.3.10.3.h	**(Not Optional)**	Specifies the total number of minefield records being published in response to a Minefield Query interaction.
NumberTripDetonationWires	Minefield Data	Number of Wires	1998: 5.3.10.3.z	*Required if tripwires are used, otherwise the default value is Empty*	Specifies the number of trip detonation wires that exist for each mine. I sub n designates the trip wire count for mine n of N. Default is that no mine has a trip wire (all counts are equal to 0), but if any one mine has a trip wire, all counts shall be published.
NumberWireVertices	Minefield Data	Number of Vertices	1998: 5.3.10.3.aa	*Required if the sum of values in NumberTripDetonationWires is greater than 0, otherwise the default value is Empty*	Specifies the number of vertices for each trip wire. A set of I sub n vertex counts is given for each mine n of N, if I sub n > 0. If I sub n is 0, the vertex count set is empty. Each non empty set of vertex counts (I sub n > 0) follows the previous non empty set. J sub i designates the vertex count of tripwire i of I sub n. The size of this parameter is the sum of the values in the NumberTripDetonationWires parameter.
PaintScheme	Minefield Data	Paint Scheme	1998: 5.3.10.3.y	0 (Other)	Specifies the camouflage scheme/color of the mine.
RecordSequenceNumber	Minefield Data	PDU Sequence Number	1998: 5.3.10.3.g	**(Not Optional)**	Specifies the number of the current record in a sequence of minefield records published in response to a Minefield Query interaction.
Reflectance	Minefield Data	Reflectance	1998: 5.3.10.3.t	0	Specifies the local dielectric difference between the mine and the surrounding soil.
RequestIdentifier	Minefield Data	Request ID	1998: 5.3.10.3.f	**(Not Optional)**	Identifies the matching response to a request for mine information from the minefield simulation made by means of a Minefield Query interaction.
RequestingEntityIdentifier	Minefield Data	Requesting Entity ID	1998:5.3.10.3.c	**(Not Optional)**	Identifies the entity that requested the information from the minefield simulation in QRP (Query Response Protocol) mode.

Parameter Name	**DIS PDU**	**DIS Field**	**IEEE 1278 Reference**	**Default Value (if optional)**	**Definition**
ScalarDetectionCoefficient	Minefield Data	Scalar Detection Coefficient	1998: 5.3.10.3.x	0	Specifies the coefficient to be utilized to insure proper correlation between detectors located on different simulation platforms. In statistically based detection system applications, the detection system simulation will generally compare a random number against an internally calculated probability of detection. M (number of sensors) coefficients are specified for each of the N mines (MxN values formatted with M values for mine 1 followed by M values for mine 2, etc.). If not published, the default value of zero indicates that the mines are always detected. If any one value is published all MxN values shall be published.
SensorTypes	Minefield Data	Sensor Types	1998: 5.3.10.3.m	**(Not Optional)**	In QRP mode, specifies the requesting sensor types which were specified in the minefield query whereas in Heartbeat mode, specifies the sensor types that are being served by the minefield.
SnowBurialDepthOffset	Minefield Data	Snow Burial Depth Offset	1998: 5.3.10.3.q	0	Specifies the offset of the origin of the mine coordinate system with respect to the snow surface. Snow burial depth offset is specified as a positive measurement in meters below the snow surface along the up vector. If any of the three burial depth offsets is published, ground burial depth offset and mine orientation shall also be published. The value of the snow burial depth offset shall be set to the value of the ground burial depth offset to indicate there is no snow for the mine to be buried in.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ThermalContrast	Minefield Data	Thermal Contrast	1998: 5.3.10.3.s	0	Specifies the temperature difference between the mine and the surrounding soil in degrees Centigrade. In the case of a buried mine, the delta temperature shall be measured between the ground surface above the mine and the surrounding ground surface temperature.
WaterBurialDepthOffset	Minefield Data	Water Burial Depth Offset	1998: 5.3.10.3.p	0	Specifies the offset of the origin of the mine coordinate system with respect to the water surface. Water burial depth offset is specified as a positive measurement in meters below the water surface along the up vector. If any of the three burial depth offsets is published, ground burial depth offset and mine orientation shall also be published. The value of the water burial depth offset shall be set to the value of the ground burial depth offset to indicate there is no water for the mine to be buried in.
WireVertices	Minefield Data	Vertex	1998: 5.3.10.3.ab	*Required if the sum of values in NumberWireVertices is greater than 0, otherwise the default values is Empty*	Specifies the locations of vertices in a trip wire. A set of J sub i locations of vertices is given for each trip wire i of I sub n, if I sub n > 0 and J sub i > 0. If J sub i is 0, the set is empty. Each non empty set of locations of vertices (I sub n > 0 and J sub i > 0) follows the previous non empty set. The size of this parameter is the sum of the values in the NumberWireVertices parameter.

7.9.2.3 MinefieldResponseNACK Interaction Class.

The MinefieldNACK interaction class is used to request retransmission of MinefieldData interactions in cases where one or more MinefieldData interactions were not received by a federate.

Table 46 MinefieldResponseNACK Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
MinefieldIdentifier	Minefield Response NACK	Minefield ID	1998: 5.3.10.4b	**(Not Optional)**	Identifies the minefield to which this query is addressed.
MissingRecordNumbers	Minefield Response NACK	Missing PDU Sequence Numbers	1998: 5.3.10.4.f	**(Not Optional)**	Specifies the record numbers that were not received in a sequence of minefield records.
RequestIdentifier	Minefield Response NACK	Request ID	1998: 5.3.10.4.d	**(Not Optional)**	Identifies the minefield query request.
RequestingEntityIdentifier	Minefield Response NACK	Requesting Entity ID	1998: 5.3.10.4.c	**(Not Optional)**	Identifies the entity that requested the information from the minefield federate.

7.9.2.4 MinefieldObjectTransaction Interaction Class

This interaction class is a subclass of ArealObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a MinefieldObject instance. The ArealObjectTransaction base class is represented as a scaffolding class in the Minefield FOM module for HLA 2010. The full description of the ArealObjectTransaction interaction class can be found in the Synthetic Environment Module (see section 7.8.2.2).

Table 47 MinefieldObjectTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BreachedStatus	Areal Object State	Object AppearanceType	1998: 5.3.11.5.h	**(Not Optional)**	Specifies the breached appearance of the minefield.
MineCount	Areal Object State	Object AppearanceObject Type	1998: 5.3.11.5.h	**(Not Optional)**	Specifies the number of mines in the minefield.

7.9.2.5 MinefieldLaneMarkerTransaction Interaction Class

This interaction class is a subclass of LinearObjectTransaction that supports the conveyance of common parameters associated with requests sent to environmental manager federate(s) to create, modify, or delete a MinefieldLaneMarkerObject instance. The LinearObjectTransaction base class is represented as a scaffolding class in the Minefield FOM module for HLA 2010. The full description of the LinearObjectTransaction interaction class can be found in the Synthetic Environment Module (see section 7.8.2.4).

Table 48 MinefieldLaneMarkerTransaction Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SegmentRecords	Linear Object State	Segment Appearance, Segment Location, Segment Orientation, Segment Length, Segment Width, Segment Height, Segment Depth	1998: 5.2.48.a-i, 5.3.11.4.j	**(Not Optional)**	Description of a lane marker linear object.

7.10 Communication Module

The Communication module is used to simulate radio communications. It defines classes that simulate radio transmitters and receivers as well as the radio signals that are transmitted between them. These classes support the simulation of both audio and data transmission by radio and Tactical Data Links (TDLs). The content of a radio transmission may be conveyed in its entirety in real-time, or may be conveyed by reference to a prerecorded database.

The RadioTransmitter and RadioReceiver classes both derive from the EmbeddedSystem class of the Base module. The Communication module also depends on the Base module for datatypes.

7.10.1 Object Classes

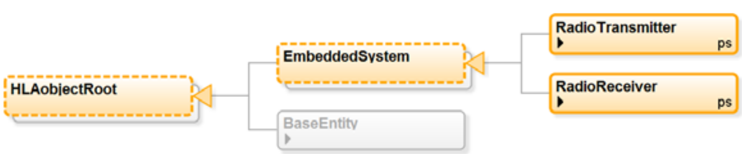

Figure 11 Communication Module Object Class Structure

7.10.1.1 RadioTransmitter Object Class

This object class is a subclass of EmbeddedSystem that provides electromagnetic properties of radio transmitting systems for the purpose of both simulated radio reception and electronic warfare. The EmbeddedSystem base class is represented as a scaffolding class in the Communication FOM module for HLA 2010. The full description of the EmbeddedSystem interaction class can be found in the Base Module (see section 7.5.1.2).

Four types of fields are incorporated in this object class: state/identification, electromagnetic characteristics, modulation, and cryptography. Each federate updating instance attributes of this object class or its subclasses shall provide the state/identification fields RadioIndex, RadioSystemType, and TransmitterOperationalStatus. Each federate updating instance attributes of this class or its subclasses shall guarantee that a unique RadioIndex / HostObjectIdentifier combination is provided for each radio instance. The one remaining state/identification field, RadioInputSource, shall be treated as an optional field.

The electromagnetic characteristics data consists of the WorldLocation, AntennaPatternData, Frequency, FrequencyBandwidth, and TransmittedPower attributes. All of the electromagnetic characteristics shall be provided by each federate updating instance attributes of this object class or its subclasses except AntennaPatternData, which is only required for non-OmniDirectional pattern types. The attributes Frequency and FrequencyBandwidth shall be updated together. All of the modulation, and cryptography attributes shall be treated as optional fields by federates updating instance attributes of this object class or its subclasses.

Table 49 RadioTransmitter Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AntennaPatternData	Transmitter	Antenna Pattern Parameters	1995: 5.3.8.1.h, 5.3.8.1.i, 5.3.8.1.r	(*not required for Omni Directional source*)	Specifies the radiation pattern from the antenna.
Cryptographic Mode	Transmitter	Crypto Key ID	1995: 5.3.8.1.o	Baseband Encryption	Indicates baseband or diphase mode.
CryptoSystem	Transmitter	Crypto System	1995: 5.3.8.1.n	Other	Identifies the cryptographic equipment used.
EncryptionKeyIdentifier	Transmitter	Crypto Key ID	1995: 5.3.8.1.o	*Required if CryptoSystem is specified.*	Key identifier number. The transmitter and receiver shall be considered to be using the same key if these numbers match.
Frequency	Transmitter	Frequency	1995: 5.3.8.1.j	**(Not Optional)**	Center frequency of the radio transmissions.
FrequencyBandwidth	Transmitter	Transmit Frequency Bandwidth	1995: 5.3.8.1.k	**(Not Optional)**	Bandpass of the radio transmissions, specified in hertz.
RadioIndex	Transmitter	Radio ID	1995: 5.3.8.1.c	**(Not Optional)**	Specifies the identification number for each radio on a given host. This ID shall not change during an exercise.
RadioInputSource	Transmitter	Input Source	1995: 5.3.8.1.f	Other	Specifies which position (pilot, gunnery officer, etc.) or data port provided the input for the transmission.
RadioSystemType	Transmitter	Radio Entity Type	1995: 5.3.8.1.d	**(Not Optional)**	Kind, Country, Domain, Category, Nomenclature Version, and Nomenclature of the DIS Radio Type. This ID shall not change during an exercise.
RFModulation SystemType	Transmitter	Modulation Type: System	1995: 5.3.8.1.m	Other	The radio system type associated with this transmitter.
RFModulation Type	Transmitter	Modulation Type: Major	1995: 5.3.8.1.m	Other	Classification of the modulation type.
SpreadSpectrum	Transmitter	Modulation Parameters	1995: 5.3.8.1.q	SpreadSpectrumType field is None, SINCGARSModulation field is not present	Describes the spread spectrum characteristics of the transmission, such as frequency hopping or other spread spectrum transmission modes.
StreamTag	Transmitter	Entity ID, Radio ID	1995: 5.3.8.1b,c	No associated audio stream	A globally unique identifier for the associated audio stream

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
TimeHopInUse	Transmitter	Modulation Type: Spread Spectrum: Time Hop	1995: 5.3.8.1.m	False	True if a time hop transmit algorithm is in use.
TransmittedPower	Transmitter	Power	1995: 5.3.8.1.l	**(Not Optional)**	The average power being transmitted in units of decibel-milliwatts.
TransmitterOperationalStatus	Transmitter	Transmit State	1995: 5.3.8.1.e	**(Not Optional)**	On/Off state of the transmitter as an enumeration.
WorldLocation	Transmitter	Antenna Location	1995: 5.3.8.1.g	**(Not Optional)**	Location of the antenna in world coordinates.

This object class is intended to describe only the radio gear itself and not the contents of any messages carried by this system. The contents of the messages transmitted by the radio are provided as discrete events by the RadioSignal interaction class. The attributes for this object class are derived from the Transmitter PDU as described in IEEE Std 1278.1™-1995 [6] Section 5.3.8.1.

The RadioIndex field is used principally for conversion between HLA and DIS. See Section 7.10.2.1 for details on the mapping between the RPR FOM and DIS addressing schemes for radio numbers.

This object class continues to follow a DIS RadioTransmitters positioning model that relies on two WorldLocation fields. The first version of position data is derived from the Spatial attribute of the host entity, a second version of WorldLocation is included as part of the RadioTransmitter itself. The RadioTransmitter's WorldLocation is intended to be a low-resolution value that is not dead reckoned. It provides a basis by which RadioTransmitters can be geographically filtered out of a simulation without having to process the host's information. Once engaged, the host entity's Spatial attribute, and the RelativePosition data from EmbeddedSystem are used to fine-tune the radio's position.

The StreamTag identifies a real-time audio stream. When this stream is transmitted using a series of EncodedAudioRadioSignal interactions, they shall all use the same StreamTag. Conversely all signal interactions with the same StreamTag compose the same stream. This StreamTag shall be different from all other StreamTags in use at that time. A real-time audio stream shall be associated with one or more RadioTransmitter instance objects. The StreamTag attribute of the RadioTransmitter object class shall identify its associated real-time audio stream.

In RPR FOM version 1.0, the transmitter`s string name is embedded in the audio packet. This increases the load and coding complexity of the audio channel. It also is less general, in that a given audio stream can only be associated with a single object, while it would be useful to use one networked audio stream for two purposes. In order to support multiple uses of the same audio stream by multiple radio transmitters, the following changes have been made to the RadioTransmitter object class and the EncodedAudioRadioSignal interaction class. HostRadioIndex parameter has been removed from the EncodedAudioRadioSignal interaction and replaced with a 64 bit identifier, StreamTag. StreamTag shall be unique across the federation execution. This StreamTag is referenced by the StreamTag attribute of the RadioTransmitter object class.

The RadioTransmitter's StreamTag attribute need not change unless the radio becomes associated with a different stream. The value of the StreamTag attribute is only valid when the value of the TransmitterOperationalStatus is "On and Transmitting". It is valid for a RadioTransmitter interaction's StreamTag to identify a stream that is not currently active, i.e., no EncodedAudioRadioSignal interactions are currently being sent. If the associated RadioTransmitter Object is "On and Transmitting" and there are no associated EncodedAudioRadioSignal interactions, then the receiver shall be receiving an unmodulated carrier.

A single stream should be used to represent identical audio signals transmitted simultaneously by multiple transmitters. This single stream shall be identified by the use of the same StreamTag value in each RadioTransmitter Object.

StreamTags are globally unique 64-bit integers for relating voice information (EncodedAudio interaction) to current state information (RadioTransmitter object). This mechanism replaces the HostRadioIndex string for efficiency reasons but only for the EncodedAudio interactions.

StreamTags are built directly from DIS PDU variables. Specifically these variables are site, application, entity and radio numbers (SAER). StreamTags are created and destroyed with the creation and destruction of Transmitter PDUs. StreamTags are used on incoming EncodedAudio interactions in order to look up the proper RadioTransmitter object. See section 7.3.1.

The pseudo code for creating a StreamTag on a Big Endian machine is as follows. That is the site variable shall be in the lowest physical memory location.

```
StreamTag = (site << 48) | (app << 32) | (entity << 16) | radio;
```

It is still the burden of the creator of the DIS Transmitter PDU to ensure the uniqueness of this quadruplet of numbers (SAER). As per the DIS standard, the site, application, entity (SAE) triplet of numbers shall identify a valid host vehicle (DIS Entity State PDU) to which this radio is attached – even if this radio is in the possession of an infantryman.

7.10.1.2 RadioReceiver Object Class

This object class is a subclass of EmbeddedSystem that provides state information for a particular radio receiver in order to support radio network monitors, data loggers, and similar applications for use in debugging, supervision, and after-action review. The EmbeddedSystem base class is represented as a scaffolding class in the Communication FOM module for HLA 2010. The full description of the EmbeddedSystem interaction class can be found in the Base Module (see section 7.5.1.2).

Each federate updating instance attributes of this object class or its subclasses shall always provide the RadioIndex and ReceiverOperationalStatus. Each federate updating instance attributes of this object class or its subclasses shall guarantee that a unique RadioIndex / HostObjectIdentifier combination is provided for each radio instance. The remaining parameters shall also be provided whenever the radio is in a receiving state.

Table 50 RadioReceiver Attributes

Attribute Name	**DIS PDU**	**DIS Field**	**IEEE 1278 Reference**	**Default Value (if optional)**	**Definition**
RadioIndex	Receiver	Radio ID	1995: 5.3.8.3.c	**(Not Optional)**	Specifies the identification number for each receiver on a given host. This ID shall not change during an exercise.
ReceivedPower	Receiver	Received Power	1995: 5.3.8.3.e	*Required when Receiver Operational Status is OnAndReceiving*	The radio frequency power received after applying any propagation loss and antenna gain, in decibel milliwatts.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivedTransmitterIdentifier	Receiver	Transmitter	1995: 5.3.8.3.g	*Required when Receiver Operational Status is OnAndReceiving*	Object Instance ID of the transmitter currently being received.
ReceiverOperationalStatus	Receiver	Receiver State	1995: 5.3.8.3.d	**(Not Optional)**	On/Off state of the receiver as an enumeration.

This object class is intended to describe only the radio gear itself and not the contents of any messages carried by this system. The contents of the messages received by the radio are provided as discrete events by the RadioSignal interaction class. The attributes for this object class are derived from the Receiver PDU as described in IEEE Std 1278.1™-1995 [6] Section 5.3.8.3.

Strictly speaking, the RadioIndex field is only needed for conversion between HLA and DIS. See Section 7.10.2.1 for details on the mapping between the RPR FOM and DIS addressing schemes for radio numbers.

7.10.2 Interaction Classes

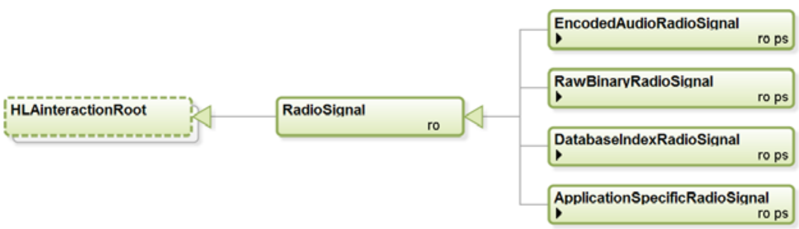

Figure 12 Communication Module Interaction Class Structure

7.10.2.1 RadioSignal Interaction Class

The RadioSignal interaction is a way to send data over a radio. This can be u-law encoded voice or tactical data such as Improved Data Modem (IDM) or Tactical Digital Information Link-J (TADIL-J) data. The RadioSignal interaction does not contain any state information however. Instead the RadioSignal interaction shall point back to the RadioTransmitter object of Section 7.10.1.1 in order to get current state information such as frequency, power, bandwidth, etc.

The mechanism that relates a RadioSignal to a RadioTransmitter is dependent on which subclass is used.

- DatabaseIndex, ApplicationSpecific and RawBinary interactions use the HostRadioIndex mechanism.
- EncodedAudio uses the StreamTag mechanism.

The HostRadioIndex is an RTIObjectIDStruct which is a unique string that identifies the name of the RadioTransmitter object. The StreamTag is a 64-bit integer representation of a RadioTransmitter object. StreamTags are essentially concatenations of the following 16-bit integers – site, application, entity and radio numbers (SAER).

Both of these mechanisms, HostRadioIndex and StreamTags, shall be globally unique. These mechanisms are not optional. They shall be properly filled in before issuing the RadioSignal interaction.

Creation and destruction of HostRadioIndexes and StreamTags is locked to the creation and destruction of RadioTransmitter objects of section 7.10.1.1. RadioSignal interactions use HostRadioIndexes and StreamTags to point back to the controlling RadioTransmitter object. Note that a particular radio may transmit any one of these four subclasses at any time.

The layout of the subclasses is derived from the Signal PDU described in Section 5.3.8.2 of IEEE Std 1278.1™-1995 [6]. However, the RPR FOM changed the DIS structure by creating the four subclasses for each encoding type.

The radio addressing scheme used in the RPR FOM differs from that used in DIS. However, all of the information needed for a translation to DIS is still supplied. Under DIS, the Signal PDU included an Entity ID that specified the entity carrying the radio and a Radio ID which distinguished multiple radios on the same entity. Since radios are object instances in their own right in the RPR FOM (see Sections 7.10.1.1 and 7.10.1.2), the most direct means of addressing them under HLA is through their globally unique RTI object instance ID.

When using HostRadioIndex, it is sometimes necessary to reconstruct the SAE numbers of the original host vehicle to which a radio is attached. The following steps shall be taken to perform this operation.

1. The HostRadioIndex of the RadioSignal interaction is used to look up the RadioTransmitter object that emitted this signal.
2. The HostObjectIdentifier of the RadioTransmitter (in EmbeddedSystem) is then used to look up the object of the entity that is carrying this radio. Not only does the host entity provide the absolute location of the radio, but its EntityIdentifier is also equivalent to the Entity ID needed by the Signal PDU.
3. Finally, the RadioTransmitter's RadioIndex attribute is used as an equivalent to the Radio ID field needed to complete the DIS radio addressing scheme.

When using StreamTags, it is not necessary to reconstruct the SAE numbers of the host vehicle. The StreamTag already contains this information.

7.10.2.2 EncodedAudioRadioSignal Interaction Class

This interaction class is a subclass of RadioSignal that is used to transmit encoded audio data to other simulation participants. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 51 EncodedAudioRadioSignal Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AudioData	Signal	Entity ID, Radio ID, Encoding Scheme, Sample Rate, Samples, Data Length, Data	1995: 5.3.8.2.b, 5.3.8.2.c, 5.3.8.2.d, 5.3.8.2.f, 5.3.8.2.h, 5.3.8.2.g, 5.3.8.2.i	**(Not Optional)**	Specification of an encoded audio signal.

The EncodedAudio interaction used to be inefficient in that it would break up the audio data into separate parameters. This would cause additional Central Processing Unit (CPU) loading. Also, the previous definition did not make sense without ALL of the fields. For example, without the encoding type, the data is meaningless. There is no situation in which a subset of parameters in an audio interaction could be sent, so there is nothing gained by splitting the data up. The voice data represents about 50% of the traffic in a typical exercise, so efficiency considerations were given substantial weight in reducing

EncodedAudio to a single parameter. Consequently all the data within the EncodedAudio class is now within one parameter - "AudioData."

StreamTags are part of the AudioData complex data type. StreamTags are used to get back to the current state information (RadioTransmitter objects) when given transient voice information (EncodedAudio interaction).

7.10.2.3 RawBinaryRadioSignal Interaction Class

This interaction class is a subclass of RadioSignal that is used to transmit raw binary data to other simulation participants. The values for TacticalDataLinkType and TDLMessageCount shall default to “other” and “zero” respectively. The sender shall provide all other parameters.

Table 52 RawBinaryRadioSignal Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
HostRadioIndex	Signal	Entity ID and Radio ID	1995: 5.3.8.2.b, 5.3.8.2.c	**(Not Optional)**	Object instance ID of the embedded system host.
DataRate	Signal	Sample Rate	1995: 5.3.8.2.f	**(Not Optional)**	Bits per second for the binary signal
SignalDataLength	Signal	Data Length	1995: 5.3.8.2.g	**(Not Optional)**	Length of transmission in bits.
SignalData	Signal	Data	1995: 5.3.8.2.i	**(Not Optional)**	Information contents of this transmission.
TacticalDataLinkType	Signal	Tactical Data Link (TDL) Type	1995: 5.3.8.2.e	Other	Tactical data link enumeration.
TDLMessageCount	Signal	Encoding Scheme	1995: 5.3.8.2.d	Zero	Number of tactical data link messages contained in this transmission.

7.10.2.4 DatabaseIndexRadioSignal Interaction Class

This interaction class is a subclass of RadioSignal that represents the transmittal of pre-recorded voice data or other messages that can be represented by using a pre-defined database. The values for TacticalDataLinkType and TDLMessageCount shall default to “other” and “zero” respectively. The sender shall provide all other parameters.

Table 53 DatabaseIndexRadioSignal Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
HostRadioIndex	Signal	Entity ID and Radio ID	1995: 5.3.8.2.b, 5.3.8.2.c	**(Not Optional)**	Object instance ID of the embedded system host.
DatabaseIndex	Signal	Data	1995: 5.3.8.2.i	**(Not Optional)**	Index into database of messages.
Duration	Signal	Data	1995: 5.3.8.2.i	**(Not Optional)**	Duration, in milliseconds, of transmitted signal.
StartOffset	Signal	Data	1995: 5.3.8.2.i	**(Not Optional)**	The offset, in milliseconds from the start of the stored signal, that the signal is replayed from.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
TacticalDataLinkType	Signal	TDL Type	1995: 5.3.8.2.e	Other	Defines type of tactical data link.
TDLMessage Count	Signal	Encoding Scheme	1995: 5.3.8.2.d	Zero	Number of tactical data link messages contained in this transmission.

7.10.2.5 ApplicationSpecificRadioSignal Interaction Class

This interaction class is a subclass of RadioSignal that is used for any case not satisfied by the other subclasses of the RadioSignal interaction class. It is similar to RawBinaryRadioSignal but includes an extra UserProtocolID field that allows the application to translate the encoding scheme for each transmission. The values for TacticalDataLinkType and TDLMessageCount shall default to "other" and "zero" respectively. The sender shall provide all other parameters.

Table 54 ApplicationSpecificRadioSignal Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
HostRadioIndex	Signal	Entity ID and Radio ID	1995: 5.3.8.2.b, 5.3.8.2.c	**(Not Optional)**	Object instance ID of the embedded system host.
DataRate	Signal	Sample Rate	1995: 5.3.8.2.f	**(Not Optional)**	Bits per second for the binary signal
SignalDataLength	Signal	Data Length	1995: 5.3.8.2.g	**(Not Optional)**	Length of transmission in bits.
SignalData	Signal	Data	1995: 5.3.8.2.i	**(Not Optional)**	Information contents of this transmission.
TacticalDataLinkType	Signal	TDL Type	1995: 5.3.8.2.e	Other	Tactical data link enumeration.
TDLMessage Count	Signal	Encoding Scheme	1995: 5.3.8.2.d	Zero	Number of tactical data link messages contained in this transmission.
UserProtocol ID	Signal	Data	1995: 5.3.8.2.i	**(Not Optional)**	User protocol identification number.

7.11 Distributed Emission Regeneration Module

This module involves the representation of lasers, active electromagnetic emissions, and acoustic emissions including active countermeasures. Emitting entities simulate their emitter and output real-time operational parameters. Receiving entities can then regenerate the transmitted signal based upon the simulated emitter output data and stored database.

The Designator, EmitterSystem, and IFF classes all derive from the EmbeddedSystem class of the Base module. The Distributed Emission Regeneration module also depends on the Base module for datatypes.

7.11.1 Object Classes

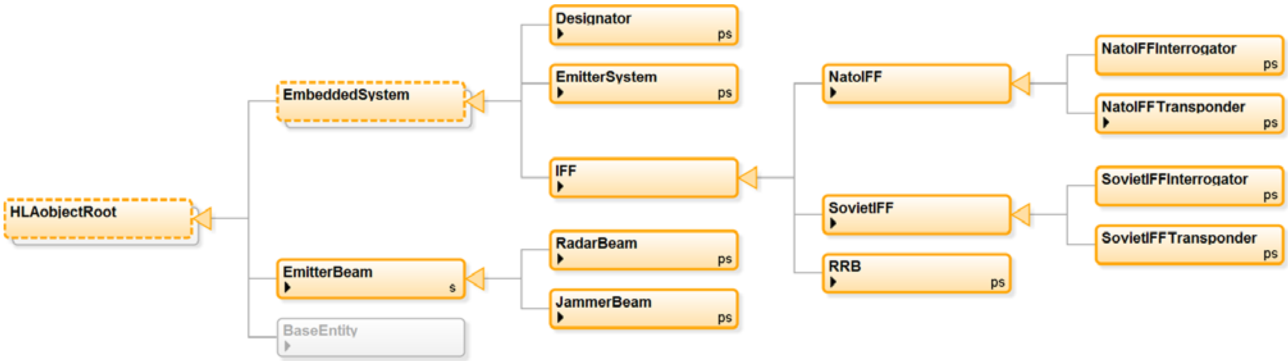

Figure 13 Distributed Emission Regeneration Module Object Class Structure

7.11.1.1 Designator Object Class

The Designator object class is a subclass of EmbeddedSystem that is used to describe the behaviors of targeting system illuminations such as those used in laser-guided weapon engagement. The EmbeddedSystem base class is represented as a scaffolding class in the Distributed Emission Regeneration FOM module for HLA 2010. The full description of the EmbeddedSystem interaction class can be found in the Base Module (see section 7.5.1.2).

The Designator provides a representation at the target site (e.g., the laser spot on a target) instead of the source emission system (e.g., the laser targeting system itself). If a federate updating instance attributes of this object class or its subclasses does not supply the DesignatedObjectIdentifier attribute, the default behavior shall treat the designator as if it was not located on an entity.

In addition to supporting a location in the World Coordinate System, this object class is also capable of dead reckoning the relative spot location. The use of the spot dead reckoning algorithms provided for within the IEEE Std 1278.1™ specification is not widespread and its use is not recommended. The spot dead reckoning parameters provided are not believed to be broadly applicable as discontinuous spot translation can potentially create infinite accelerations. Spot dead reckoning is implemented exactly as described in DIS. It has been identified that this might not provide a complete solution; however, it preserves compatibility and consistency with DIS.

The DesignatedObjectIdentifier, DeadReckoningAlgorithm, RelativeSpotLocation, and SpotLinearAccelerationVector shall be treated as optional fields. All other attributes in Table 55 are mandatory and shall be provided by federates updating instance attributes of this object class or its subclasses.

Table 55 Designator Attributes

Attribute Name	DIS PDU	DIS Equivalent	IEEE 1278 Reference	Default Value (if optional)	Definition
CodeName	Designator	Code Name	1995: 5.3.7.2.c	**(Not Optional)**	Identifies the code name for the designator system.
DesignatedObjectIdentifier	Designator	Designated Entity ID	1995: 5.3.7.2.d	When this attribute is not updated, it indicates that there is no designated object.	Object instance ID of the entity that is currently being designated.

Attribute Name	DIS PDU	DIS Equivalent	IEEE 1278 Reference	Default Value (if optional)	Definition
DesignatorCode	Designator	Designator Code	1995: 5.3.7.2.e	**(Not Optional)**	The designator code being used by the designating entity.
DesignatorEmissionWavelength	Designator	Designator Wavelength	1995: 5.3.7.2.g	**(Not Optional)**	The wavelength of the designator system, in microns.
DesignatorOutputPower	Designator	Designator Power	1995: 5.3.7.2.f	**(Not Optional)**	The output power of the designator system, in watts.
DesignatorSpotLocation	Designator	Designator Spot Location	1995: 5.3.7.2.i	**(Not Optional)**	Location of the Designator Spot in DIS World Coordinate System.
DeadReckoningAlgorithm	Designator	DRAlgorithm	1995: 5.3.7.2.j.1	Static	Algorithm used to dead reckon the position of the designator spot.
RelativeSpotLocation	Designator	Designator Spot With Respect to Designated Entity	1995: 5.3.7.2.h	All Zeros	Designator spot with respect to the designated entity's coordinate system when the spot is on an entity.
SpotLinearAccelerationVector	Designator	Entity Linear Acceleration	1995: 5.3.7.2.j.2	All Zeros	The linear acceleration used to dead reckon the position of the designator spot.

The attributes for this object class are derived from the Designator PDU as described in IEEE Std 1278.1™-1995 [6] Section 5.3.7.2.

7.11.1.2 EmitterSystem Object Class

The EmitterSystem object class is a subclass of EmbeddedSystem that provides electromagnetic properties of radars, jammers, and other electronic warfare (EW) systems not covered elsewhere in the EmbeddedSystem hierarchy. The EmbeddedSystem base class is represented as a scaffolding class in the Distributed Emission Regeneration FOM module for HLA 2010. The full description of the EmbeddedSystem interaction class can be found in the Base Module (see section 7.5.1.2).

This object class has no optional attributes; federates updating instance attributes of this object class or its subclasses shall provide all of the attributes specified in Table 56.

Table 56 EmitterSystem Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EmitterFunctionCode	Electro magnetic Emissions	Emitter System: Function	1995: 5.2.11, 5.3.7.1.e.3	**(Not Optional)**	Specifies the function for a particular emitter as an enumeration.
EmitterType	Electro magnetic Emissions	Emitter System: Emitter Name	1995: 5.2.11, 5.3.7.1.e.3	**(Not Optional)**	Emitter type specified as an enumeration.
EmitterIndex	Electro magnetic Emissions	Emitter ID Number	1995: 5.2.11, 5.3.7.1.e.3	**(Not Optional)**	Specifies the identification number for each emitter system on a given host. This ID shall not change during an exercise.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EventIdentifier	Electro magnetic Emissions	Event ID	1995: 5.3.7.1.c	**(Not Optional)**	Used by the generating federate to associate EmitterSystem and EmitterBeam changes.

The attributes for this object class are derived from the Electromagnetic Emission (EE) PDU as described in IEEE Std 1278.1™-1995 [6] Section 5.3.7.1. Unlike the DIS structure, the RPR FOM divides the emission into two parts: an emitter system, and a series of emitter beams. The EmitterSystem object class represents the properties of the electromagnetic system itself while the EmitterBeam object class provides the beam data. (Although radio transmitters are sometimes used as detectable EW systems, the RadioTransmitter object class is actually the proper EmbeddedSystem object class for those systems.)

The EventIdentifier allows correlation of EmitterSystem and EmitterBeam data. Each change in the electromagnetic emission characteristics updates the EventIdentifier in both object classes. The composite of these two emitter components is then re-assembled on the receiving side.

7.11.1.3 IFF Object Class

The IFF object class is a subclass of EmbeddedSystem that is designed to provide a basis for the properties of cooperative Identification Friend or Foe (IFF) systems, Air Traffic Control (ATC) beacons and transponder systems, collision avoidance, and NAVAIDS (Navigation Aid) systems. The EmbeddedSystem base class is represented as a scaffolding class in the Distributed Emission Regeneration FOM module for HLA 2010. The full description of the EmbeddedSystem interaction class can be found in the Base Module (see section 7.5.1.2).

Each federate updating an instance of the IFF object class or its subclasses shall always provide the EventIdentifier, Layer2DataAvailable, SystemName, SystemType, SystemOn, SystemIsOperational, and SystemMode attributes. If the Layer2DataAvailable attribute is set to true, then the BeamAzimuthCenter, BeamAzimuthSweep, BeamElevationCenter, BeamElevationSweep, BeamSweepSync, FundamentalParameterData, SecondaryOperationalDataParameter1, and SecondaryOperationalDataParameter2 shall be provided.

Table 57 IFF Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BeamAzimuthCenter	IFF/ATC/ NAVAIDS	Beam Data – Azimuth Center	1998: 5.2.39.a, 5.3.7.4.2.b	*Required when Layer2DataAvailable is true.*	Specifies the azimuth center to describe the scan volume. Measured in relation to the emitter coordinate system. Radians 32-bit float point. Update condition for this attribute is described in IEEE Std 1278.1™-1995 [6] section 4.5.6.2.2.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BeamAzimuthSweep	IFF/ATC/ NAVAIDS	Beam Data – Azimuth Sweep	1998: 5.2.39.b, 5.3.7.4.2.b	*Required when Layer2DataAvailable is true.*	Specifies the azimuth sweep half-angle to describe the scan volume. Measured in relation to the emitter coordinate system. Radians 32-bit float point. Update condition for this attribute is described in IEEE Std 1278.1™-1995 [6] section 4.5.6.2.2.
BeamElevationCenter	IFF/ATC/ NAVAIDS	Beam Data – Elevation Center	1998: 5.2.39.c, 5.3.7.4.2.b	*Required when Layer2DataAvailable is true.*	Specifies the elevation center to describe the scan volume. Measured in relation to the emitter coordinate system. Radians 32-bit float point. Update condition for this attribute is described in IEEE Std 1278.1™-1995 [6] section 4.5.6.2.2.
BeamElevationSweep	IFF/ATC/ NAVAIDS	Beam Data – Elevation Sweep	1998: 5.2.39.d, 5.3.7.4.2.b	*Required when Layer2DataAvailable is true.*	Specifies the elevation sweep half-angle to describe the scan volume. Measured in relation to the emitter coordinate system. Radians 32-bit float point. Update condition for this attribute is described in IEEE Std 1278.1™-1995 [6] section 4.5.6.2.2.
BeamSweepSync	IFF/ATC/ NAVAIDS	Beam Data – Sweep Sync	1998: 5.2.39.e, 5.3.7.4.2.b	*Required when Layer2DataAvailable is true.*	When non-zero, specifies the percentage of time a scan is through its pattern from the origin. A 32-bit float point.
EventIdentifier	IFF/ATC/ NAVAIDS	Event ID	1995: 5.2.18, 5.3.7.4.1.c	**(Not Optional)**	Generated by the issuing federate to associate related events.
FundamentalParameterData	IFF/ATC/ NAVAIDS	Fundamental Parameter Data Set	1998: 5.2.45, 5.3.7.4.2.d	*Required when Layer2DataAvailable is true.*	The specification of the fundamental energy radiation characteristics of an IFF/ATC/NAVAIDS emission.
Layer2DataAvailable	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Information Layers	1998: 5.2.42.c, 5.3.7.4.1.f	**(Not Optional)**	This field shall specify which additional layers are present in the IFF/ATC/NAVAIDS PDU
SecondaryOperationalDataParameter1	IFF/ATC/ NAVAIDS	Secondary Operational Data – Parameter 1	1998: 5.2.57.a, 5.3.7.4.2.c	*Required when Layer2DataAvailable is true.*	Additional fields for IFF PDU – dependent upon system type.
SecondaryOperationalDataParameter2	IFF/ATC/ NAVAIDS	Secondary Operational Data – Parameter 2	1998: 5.2.57.a, 5.3.7.4.2.c	*Required when Layer2DataAvailable is true.*	Additional fields for IFF PDU – dependent upon system type.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SystemMode	IFF/ATC/ NAVAIDS	System ID – System Mode	1998: 5.2.58.c, 5.3.7.4.1.d, SISO-REF-010 10.3	**(Not Optional)**	The mode of operation for the named system.
SystemName	IFF/ATC/ NAVAIDS	System ID – System Name	1998: 5.2.58.b, 5.3.7.4.1.d, SISO-REF-010 10.2	**(Not Optional)**	Specifies the particular named type of system.
SystemType	IFF/ATC/ NAVAIDS	System ID – System Type	1998: 5.2.58.a, 5.3.7.4.1.d, SISO-REF-010 10.1	**(Not Optional)**	Specifies the general type of emitting system.
SystemIsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 0	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Indicate system on/off: Off: 0 On: 1
SystemIsOperational	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 7	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Indicate system operational: System Failed: 1 System Operational: 0

7.11.1.4 NatoIFF Object Class

This object class is a subclass of IFF that provides state information about worldwide civilian Air Traffic Control Radar Beacon (ATCRB) systems that are used by all countries (e.g., Modes A, C and S), as well as state information for specific allied military IFF systems (e.g., Modes 1, 2, 4 and 5) that are only available to U.S., NATO and other allied countries. The term "NatoIFF" is not appropriate for this object class but was retained in order to maintain compatibility with widely used draft versions of RPR FOM 2.0. This object class has no optional attributes, federates updating an instance of this object class or its subclasses shall provide all of the attributes specified in Table 58.

Table 58 NatoIFF Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AlternateMode4	IFF/ATC/ NAVAIDS	System ID – Change/Options – bit 1	1998: 5.2.58.d, 5.3.7.4.1.d	**(Not Optional)**	No: 0 Yes: 1
Mode1Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 1	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Mode1IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Mode1IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Mode1IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Mode2Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 2	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Mode2IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Mode2IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Mode2IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Mode3AEnabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 3	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Mode3AIsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Mode3AIsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Mode3AIsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Mode4Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 4	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Mode4IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Mode4IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Mode4IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Mode4PseudoCrypto	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bits 0-11	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	If value = 0-4094, valid crypto value.
Mode4PseudoCryptoAvailable	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bits 0-11	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	If value = 4095, no crypto available.
Mode5CEnabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 5	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Mode5CIsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Mode5CIsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Mode5CIsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ModeSEnabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 6	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
ModeSIsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
ModeSIsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
ModeSIsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
ModeSIsTcasI	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 12	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Traffic Alert/Collision Avoidance System TCAS I: 0 TCAS II: 1

7.11.1.5 NatoIFFInterrogator Object Class

This attributeless object class is a specialization of the NatoIFF class used to denote IFF interrogator systems.

7.11.1.6 NatoIFFTransponder Object Class

This object class is a specialization of the NatoIFF class used to denote IFF transponder systems. This object class has no optional attributes, federates updating an instance of this object class or its subclasses shall provide all of the attributes specified in Table 59.

Table 59 NatoIFFTransponder Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EmergencyOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Modifier bit 1	1998: 5.2.42.d, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
IdentSquawkFlashOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Modifier bit 2	1998: 5.2.42.d, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Mode1Code	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bits 0-5	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Two octal values.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Mode2Code	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bits 0-11	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Four octal values.
Mode3ACode	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bits 0-11	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Four octal values.
Mode5CAltitude	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bits 0-11	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Bit 0=0; bits 1-11 positive alt. Bit 0=1; bits 1-11 negative alt. Bits 1-11; 0-1260 100 ft. incr. Bits 0-11 = 4095, no alt. value
Mode5CAltitudeAvailable	IFF/ATC/ NAVAIDS	System ID – Change/Options – bit 2	1998: 5.2.58.d, 5.3.7.4.1.d	**(Not Optional)**	No: 0 Yes: 1
StiOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Modifier bit 3	1998: 5.2.42.d, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1

7.11.1.7 SovietIFF Object Class

This object class is a subclass of IFF that provides state information about Soviet IFF systems. The former Soviet Union developed their own transponder and interrogator systems and many of these systems are still in use by its successor states and countries that have obtained Soviet designed systems. This object class has no optional attributes, federates updating an instance of this object class or its subclasses shall provide all of the attributes specified in Table 60.

Table 60 SovietIFF Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Parameter1Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 1	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Parameter1IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Parameter1IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Parameter1IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Parameter2Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 2	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Parameter2IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Parameter2IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Parameter2IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 2 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Parameter3Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 3	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Parameter3IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Parameter3IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Parameter3IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 3 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Parameter4Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 4	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Parameter4IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Parameter4IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Parameter4IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 4 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Parameter5Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 5	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Parameter5IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Parameter5IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
Parameter5IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 5 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1
Parameter6Enabled	IFF/ATC/ NAVAIDS	Fundamental Operational Data – System Status bit 6	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Capable: 0 Not Capable: 1
Parameter6IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 14	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
Parameter6IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 15	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Parameter6IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 6 bit 13	1998: 5.2.42.e, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1

7.11.1.8 SovietIFFInterrogator Object Class

This attributeless object class is a specialization of the SovietIFF class used to denote IFF interrogator systems.

7.11.1.9 SovietIFFTransponder Object Class

This attributeless object class is a specialization of the SovietIFF class used to denote IFF transponder systems.

7.11.1.10 RRB Object Class

This object class is a subclass of IFF that provides state information for Outfit RRB transponder systems. This object class has no optional attributes, federates updating an instance of this object class or its subclasses shall provide all of the attributes specified in Table 61.

Table 61 RRB Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Code	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bits 0-4	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Specifies the decimal code
PowerReduction	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 11	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Power reduction off: 0 Power reduction on: 1
IsDamaged	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 14	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	No Damage: 0 Damage: 1
IsMalfunctioning	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 15	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	No Malfunction: 0 Malfunction: 1
IsOn	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 13	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Off: 0 On: 1

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RadarEnhancement	IFF/ATC/ NAVAIDS	Fundamental Operational Data – Parameter 1 bit 12	1998: 5.2.42.a, 5.3.7.4.1.f	**(Not Optional)**	Radar Enhancement Off: 0 Radar Enhancement On: 1

7.11.1.11 EmitterBeam Object Class

Emitter beams define the electromagnetic characteristics of the emission emanating from an emitter system (see Section 7.11.1.2). The attributes for this object class are derived from the Electromagnetic Emission (EE) PDU as described in IEEE Std 1278.1™-1995 [6] Section 5.3.7.1. The emitter beam attributes describe the fundamental parameter data of the emission. Emitter beams are associated with a specific instance of an emitter system. A reference to the emanating emitter system is required in order to determine the beam's full characteristics. The emitter system is required primarily for spatial correlation, but it might also be required for database lookups. The BeamAzimuthCenter, BeamAzimuthSweep, BeamElevationCenter, BeamElevationSweep, and SweepSynch are optional parameters and shall default to the value zero for beam functions where a scan volume does not apply (e.g., target tracking beam). All other fields in Table 62 are mandatory and shall be provided by federates updating instance attributes of this object class or its subclasses.

Table 62 EmitterBeam Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BeamAzimuthCenter	Electromagnetic Emissions	Beam Azimuth Center	1995: 5.2.22.f, 5.3.7.1.e.4.iv	Zero	This attribute specifies the azimuth center angle of the beam's scan-volume relative to the emitter system. This attribute in conjunction with BeamElevationCenter, azimuth sweep, and BeamElevationSweep describes the scan volume covered by the emitter beam scan.
BeamAzimuthSweep	Electromagnetic Emissions	Beam Azimuth Sweep	1995: 5.2.22.g, 5.3.7.1.e.4.iv	Zero	This attribute specifies the azimuth sweep of the beam's scan-volume relative to the azimuth center. This attribute in conjunction with BeamElevationCenter, azimuth sweep, and BeamElevationSweep describes the scan volume covered by the emitter beam scan.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
BeamElevationCenter	Electro magnetic Emissions	Beam Elevation Center	1995: 5.2.22.h, 5.3.7.1.e.4.iv	Zero	This attribute specifies the elevation center angle of the beam's scan-volume relative to the emitter system. This attribute in conjunction with BeamElevationCenter, azimuth sweep, and BeamElevationSweep describes the scan volume covered by the emitter beam scan.
BeamElevationSweep	Electro magnetic Emissions	Beam Elevation Sweep	1995: 5.2.22.i, 5.3.7.1.e.4.iv	Zero	This attribute specifies the elevation sweep of the beam's scan-volume relative to the BeamElevationCenter. This attribute in conjunction with BeamElevationCenter, azimuth sweep, and elevation sweep describes the scan volume covered by the emitter beam scan.
BeamFunctionCode	Electro magnetic Emissions	Beam Function	1995: 5.3.7.1.e.4.v	**(Not Optional)**	This enumerated attribute specifies the beam's function. It serves as a general data filter.
BeamIdentifier	Electro magnetic Emissions	Beam ID Number	1995: 5.3.7.1.e.4.ii	**(Not Optional)**	This attribute specifies a unique database number assigned to differentiate between otherwise similar or identical emitter beams within an emitter system.
BeamParameterIndex	Electro magnetic Emissions	Beam Parameter Index	1995: 5.3.7.1.e.4.iii	**(Not Optional)**	This attribute specifies a beam parameter index number that shall be used by receiving entities in conjunction with the emitter name attribute (EmitterSystem object class) to provide a pointer to the stored database parameters required to regenerate the beam.
EffectiveRadiatedPower	Electro magnetic Emissions	ERP	1995: 5.2.22.c, 5.3.7.1.e.4.iv	**(Not Optional)**	This attribute specifies the EffectiveRadiatedPower for the emission in dBm. For a radar or a noise jammer, this attribute shall indicate the peak of the transmitted power. Thus, it includes peak transmitter power, transmission line losses, and peak of the antenna gain.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EmissionFrequency	Electro magnetic Emissions	Frequency	1995: 5.2.22.a, 5.3.7.1.e.4.iv	**(Not Optional)**	This attribute specifies the frequency of the emission in hertz. Frequency modulation for a particular emitter and mode shall be derived from database parameters by the reflecting federate.
EmitterSystemIdentifier	Electro magnetic Emissions	Emitter System.Emitter ID Number	1995: 5.3.7.1.e.3	**(Not Optional)**	This attribute specifies a reference to the emitter system object from which the beam is emanating.
EventIdentifier	Electro magnetic Emissions	Event ID	1995: 5.3.7.1.c	**(Not Optional)**	Used by the generating federate to associate EmitterSystem and EmitterBeam changes.
FrequencyRange	Electro magnetic Emissions	Frequency Range	1995: 5.2.22.b, 5.3.7.1.e.4.iv	**(Not Optional)**	This attribute specifies the bandwidth of the frequencies corresponding to the Frequency attribute. Thus, if, for operational purposes, the Frequency is supposed to be a single number, then the Frequency Range shall be zero. Specified in hertz.
PulseRepetitionFrequency	Electro magnetic Emissions	PRF	1995: 5.2.22.d, 5.3.7.1.e.4.iv	**(Not Optional)**	This attribute specifies the average PulseRepetitionFrequency of the emission in hertz. PulseRepetitionFrequency modulation for a particular emitter and mode shall be derived from database parameters by the reflecting federate.
PulseWidth	Electro magnetic Emissions	Pulse Width	1995: 5.2.22.e, 5.3.7.1.e.4.iv	**(Not Optional)**	This attribute specifies the average pulse width of the emission in microseconds. Pulse modulation for a particular emitter and mode shall be derived from database parameters by the reflecting federate.
SweepSynch	Electro magnetic Emissions	Beam Sweep SYNC	1995: 5.2.22.j, 5.3.7.1.e.4.iv	Zero	This attribute is provided to allow a receiver to synchronize its regenerated scan pattern to that of the emitter. This attribute when employed specifies the percentage of time a scan is through its pattern from its origin. The pattern and origin data are derived from database parameters.

7.11.1.12 RadarBeam Object Class

The RadarBeam object class is a subclass of EmitterBeam that represents all electromagnetic emitter beams, whose functions are not represented in other object classes (e.g., JammerBeam or RadioTransmitter). Primary examples would be search, acquisition, and tracking. To support the tracking function, the RadarBeam extends the EmitterBeam with attributes that indicate which entities are being tracked by the beam. If supplied, the TrackObjectIdentifiers attribute shall indicate the identity of the targets being tracked. For a single-track emitter system, this field shall be used to identify the target the system is tracking. If the system is tracking a target cluster, then all the targets in the cluster shall be identified in this attribute. The system shall not indicate a target(s) in this field if the system determines that the track has been physically offset from the target(s) by jamming. The attributes in this object class are optional.

Table 63 RadarBeam Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
HighDensityTrack	Electro magnetic Emissions	High Density Track/Jam	1995: 5.3.7.1.e.vii	False	This field is used to indicate whether or not the receiving federates can assume that all targets that are in the scan pattern that the sending emitter can track are being tracked.
TrackObjectIdentifiers	Electro magnetic Emissions	Track/Jam	1995: 5.3.7.1.e.viii	Empty	This attribute identifies the targets in an emitter.

7.11.1.13 JammerBeam Object Class

The JammerBeam object class is a subclass of EmitterBeam that represents those Electromagnetic emitter beams whose function is jamming other electromagnetic emitter beams. The JammerBeam extends the EmitterBeam with attributes that indicate which object instances (radar beams or other jam beams) are being jammed by the beam. If supplied, the JammedObjectIdentifiers attribute shall indicate the emitters the system is attempting to jam. The attributes in this object class are optional.

Table 64 JammerBeam Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
JammingModeSequence	Electro magnetic Emissions	Jamming Mode Sequence	1995: 5.3.7.1.e.4.viii	0	Indicates the jamming mode technique or series of techniques being applied.
JammedObjectIdentifiers	Electro magnetic Emissions	Track/Jam	1995: 5.3.7.1.e.4.viii	Empty	This attribute identifies the emitters a system is attempting to jam.
HighDensityJam	Electro magnetic Emissions	High Density Track/Jam	1995: 5.3.7.1.e.4.vii	False	This field is used to indicate whether or not the receiving federates can assume that all targets that are in the scan pattern that the sending emitter can jam are being jammed.

7.12 Underwater Acoustics Module

The Underwater Acoustics Module provides definitions for the objects and interactions required to share acoustic signature state with distributed simulation federates. These acoustic signatures are the sounds produced by a target entity which can be detected by passive sonar sensors. These acoustic signatures can be produced as unintentional byproducts of ship's systems such as propulsion machinery or physical activities or they can be intentionally generated by an acoustic transducer such as an active sonar or depth monitor. Active acoustic signatures are those sounds reflected from a vessel when it is impacted by sounds from an active sonar. Active sonar signatures (i.e., reflections) are not included in this module or elsewhere in RPR2. They have to be regenerated by the active sonar model. While this module provides objects to describe the direct output produced by active sonars, they are provided to stimulate models of passive sonar receivers rather than to support the functioning of an active sonar. However, it may be possible to use them to support bistatic sonar operation. Rather than express all the information about a vessel's acoustic signature as object attributes and interaction parameters, the UA Module relies heavily on database indices that allow federates to look up values in a shared predistributed database. Since propulsion noise is the most common unintentional sound associated with vessels, a PropulsionNoise object is provided to express that component of the acoustic signature. An AdditionalPassiveActivities object is provided to allow other steady state components of the signature to be looked up in database. An AcousticTransient interaction is provided to represent transient sounds like an explosion or dropping a wrench on a steel deck. Each acoustic projector array is represented by an Active Sonar object which ties back to its host entity via the HostObjectIdentifier of its parent class, EmbeddedSystem. Active pings or acoustic pulses are represented by ActiveSonarBeam objects. Multiple and repeating pulses can be combined to maintain a single persistent beam that illuminates a sector with acoustic energy. ActiveSonarBeams are associated with the ActiveSonar that generated them.

The UnderwaterAcousticsEmission class derives from the EmbeddedSystem class of the Base module. The Underwater Acoustics module also depends on the Base module for datatypes.

7.12.1 Object Classes

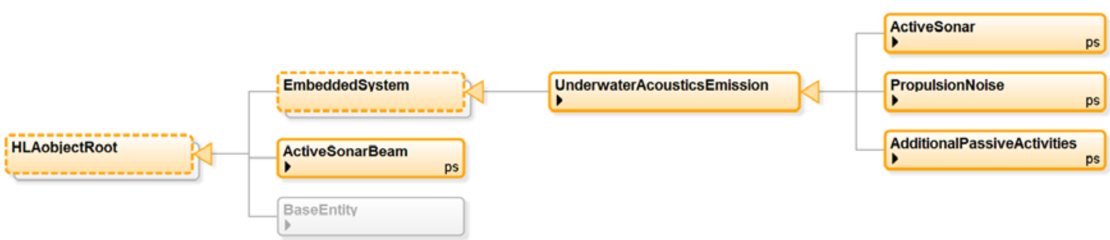

Figure 14 Underwater Acoustics Module Object Class Structure

7.12.1.1 UnderwaterAcousticsEmission Object Class

This object class is a subclass of EmbeddedSystem that is used to distribute the sounds emitted by a platform in the maritime environment. The EmbeddedSystem base class is represented as a scaffolding class in the Underwater Acoustics FOM module for HLA 2010. The full description of the EmbeddedSystem interaction class can be found in the Base Module (see section 7.5.1.2).

The emissions represented by this object class and its subclasses include not only the pings transmitted intentionally by active sonar, fathometers, and acoustic communication systems, but also the passive signatures emitted unintentionally by all mechanical systems. These emissions are used during undersea warfare scenarios to detect, classify, and track hostile forces when electronic warfare mechanisms are unavailable. Following the approach used in DIS 1998, this object class and its sub-classes define the operational states that lead to the acoustic emission instead of modeling the sounds themselves. This approach has the benefit of allowing subscribing federates to model the emissions at their own level of fidelity. Using multiple levels of fidelity across the federation is especially important for hardware-in-the-loop stimulation systems where isolating a few features of interest is often a performance requirement.

However, the multiple fidelity approach has the adverse side-effect of requiring all federation members to coordinate their models and databases during the federation development process.

The UnderwaterAcousticsEmission object class and its sub-classes model those components of the sound that are persistent. These types of emissions include steady-state propulsion noise, steady-state noise from auxiliary equipment, and multiple-ping active sonars. Sounds that occur aperiodically (such as torpedo tube floodings, hatch slams, and wrench drops) are distributed by the AcousticTransient interaction class described in section 7.12.2.1. This design limits network updates to changes in the platform's operating mode. Normally the type of operating mode changes that affect the UnderwaterAcousticsEmission class can be expected only a few times per hour. Because the data is persistent and changes are infrequent, a request based mechanism for the persistent data is better than periodically heartbeating the state from a bandwidth perspective. Representing this persistency using an HLA object class allows for the use of the HLA mechanisms for requesting attribute updates. This is a natural choice for the requests over a custom interaction based request or heartbeat approach.

Table 65 UnderwaterAcousticsEmission Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EventIdentifier	Underwater Acoustics		1998: 5.3.7.3.c	*Only required if associated ActiveSonar Beams exist.*	This field is used to coordinate changes between an active sonar system and the ActiveSonarBeam associated with it.

Following the model used for EmitterBeam, the active sonar beam characteristics are communicated separately using the ActiveSonarBeam object class. Normally the EventIdentifier is only required when coordinating updates between UnderwaterAcousticsEmission and its associated ActiveSonarBeams. However, some passive sonar simulations include an optional EventIdentifier as an exercise coordination tool.

7.12.1.2 ActiveSonar Object Class

This object class is a subclass of UnderwaterAcousticsEmission that describes the operational status of active sonar, fathometer, and/or acoustic communication systems. It is intended to communicate the steady state configuration information about the embedded system, rather than detailed information about the emission itself. Details about the active emissions are distributed using the ActiveSonarBeam object class.

The EventIdentifier from the UnderwaterAcousticsEmission super object class shall be provided with each update of any attributes of the ActiveSonar object. This is used to coordinate updates between the ActiveSonar and its associated ActiveSonarBeams. Federates updating an instance of this object class or its subclasses shall provide values for all of the attributes; there are no optional fields.

Table 66 ActiveSonar Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcousticName	Underwater Acoustics	Acoustic Name	1998: 5.2.35.a, 5.3.7.3.q	**(Not Optional)**	This field is used to specify the sub-system for a particular acoustic emitter. Typical systems include BQQ-5, SSQ-62, SQR-89, etc.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
FunctionCode	Underwater Acoustics	Function	1998: 5.2.35.b, 5.3.7.3.q	**(Not Optional)**	This field shall describe the function of the acoustic system. Examples include: platform search/detect/track, navigation, mine hunting, weapon search/detect/track/attack.
AcousticIdentifier	Underwater Acoustics	Acoustic Identification	1998: 5.2.35.c, 5.3.7.3.q	**(Not Optional)**	This field is used specify an acoustic emitter identification number relative to a specific system. This field allows the differentiation of multiple systems on an entity, even if in some instances two or more of the systems could have identical acoustic emitter types. Numbering of systems begins with the value 1.

7.12.1.3 PropulsionNoise Object Class

This object class is a subclass of UnderwaterAcousticsEmission that describes the steady state component of unintended passive emissions that are normally associated with the power plant. Federates updating instance attributes of this object class or its subclasses shall provide values for the Passive Parameter Index and the Propulsion Plant Configuration. The HullMaskerOn, and ShaftRateData and shall be treated as optional fields.

Table 67 PropulsionNoise Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
HullMaskerOn	Underwater Acoustics	Propulsion Plant Configuration	1998: 5.3.7.3.f	False	A hull masker is a counter-measure system that camouflages the propulsion noise of a platform. This field indicates the current state of the hull masker sub-system.
PassiveParameterIndex	Underwater Acoustics	Passive Parameter Index	1998: 5.3.7.3.e	**(Not Optional)**	This field indicates the database entry that shall be used to model the passive signature of the entity.
PropulsionPlantConfiguration	Underwater Acoustics	Propulsion Plant Configuration	1998: 5.3.7.3.f	**(Not Optional)**	This field specifies the operating mode of the propulsion plant.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ShaftRateData	Underwater Acoustics	Number of Shafts, Current Shaft RPM (Revolutions Per Minute), Ordered Shaft RPM, Shaft RPM Rate of Change	1998: 5.3.7.3.g, 5.3.7.3.j, 5.3.7.3.k, 5.3.7.3.l,	Unknown	This field defines the shaft speed information for each platform. Shafts are defined from port to starboard locations looking from the stern to the bow.

The PropulsionNoise object class defines passive signatures through references to a common federation database (such as CSDB, ATaS, or PSDB). Currently, each federation has to define this association at federation development time. The PassiveParameterIndex allows an individual platform to support multiple versions of the acoustic signature based on details about its manufacturing. With an acoustic signature version, the PropulsionPlantConfiguration distinguishes between unique operating modes. One example of an operating mode change within a signature type would be a diesel submarine converting from battery to snorkeling operations.

Typically, passive signatures databases are parameterized by the speed of the shafts that turn the propellers. Shaft speed gives the simulation a metric for the force being applied to the water by the propulsion plant. The ShaftRateData field allows federates updating instance attributes of this object class to specify the current speed, ordered speed, and rate of change values for each shaft (in revolutions per minute). This combination of values allows for a continuous shaft rate interpolation between network updates. If the shaft rate is not provided by the federate updating instance attributes of the object class or its subclasses, the reflecting federates are responsible for defining a "typical" behavior derived from their own database and the entity's current velocity. This is necessary because many legacy platform simulations do not model their own acoustic emissions and therefore have no idea how fast their shafts are spinning. This compromise allows an agent external to the legacy simulation to manage the issuing of the PropulsionNoise information.

7.12.1.4 AdditionalPassiveActivities Object Class

This object class is a subclass of UnderwaterAcousticsEmission that describes the steady state component of non-propulsion passive emissions such as oil pumps, air filters, and other auxiliary equipment. Unlike propulsion noise, additional passive activities have no dependence on platform speed or shaft rate. The duration of additional passive activities is not required to be infinitely long, but they are usually considered to be more than a few minutes. The AcousticTransient interaction class is has almost the same structure, but it is used for short duration events.

Federates updating instance attributes of this object class or its subclasses shall provide values for the ActivityCode. All of the other attributes shown in Table 68 shall be treated as optional fields.

Table 68 AdditionalPassiveActivities Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ActivityCode	Underwater Acoustics	Additional Passive Activity (APA) Parameter Index	1998: 5.3.7.3.m	**(Not Optional)**	This field indicates the database entry that shall be used to model the passive signature of the entity.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ActivityParameter	Underwater Acoustics	Additional Passive Activity (APA) Value	1998: 5.3.7.3.n	Zero	Current state of this activity.
IsSilent	Underwater Acoustics	Additional Passive Activity (APA) Parameter Index	1998: 5.3.7.3.m	False	This field is used to indicate on/off status of this activity as indicated by the first two bits of the DIS APA Parameter Index.

The AdditionalPassiveActivities object class defines passive signatures through references to a common federation database such as the Additional Narrowband Database (ANDB). Currently, each federation has to define this association at federation development time. The ActivityCode identifies a particular version of the additional passive activity to be modeled. The ActivityParameter provides a single integer parameter that can be used to control the acoustic emission. The interpretation of the ActivityParameter field depends on which additional passive activities database used.

7.12.1.5 ActiveSonarBeam Object Class

This object class describes the acoustic configuration for one transmitter beam in an active sonar, a fathometer, and/or an acoustic communication systems. This construct allows many ActiveSonarBeam(s) to be associated with a single ActiveSonar (see Section 7.12.1.2). A reference to the emanating sonar system is required in order to determine the beam's full characteristics. The active sonar system is required primarily for spatial correlation, but it might also be required for database lookups.

The ActiveEmissionParameterIndex, ActiveSonarIdentifier, BeamIdentifier, and EventIdentifier shall be provided by all federates updating instance attributes of this object class. All of the other attributes shown in Table 69 shall be treated as optional fields.

Table 69 ActiveSonarBeam Attributes

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ActiveEmissionParameterIndex	Underwater Acoustics	Fundamental Data Parameters. Active Parameter Indexes	1998: 5.2.59a, 5.3.7.3.s.3	**(Not Optional)**	This field indicates the database entry that shall be used to model the active mode of this transmitter beam.
ActiveSonarIdentifier	Underwater Acoustics	Acoustic Emitter System. Acoustic ID Number	1998: 5.3.7.3.q	**(Not Optional)**	This attribute specifies a reference to the active sonar object from which the beam is emanating.
AzimuthBeamwidth	Underwater Acoustics	Fundamental Data Parameters. Horizontal Beamwidth	1998: 5.2.59.d, 5.3.7.3.s.3	Zero	This field specifies the horizontal beamwidth of the main beam. Beamwidth is measured at the 3 dB point down from maximum response axis. A value of zero indicates omni-directional.

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AzimuthCenter	Underwater Acoustics	Fundamental Data Parameters . Main Beam Center Azimuth (Horizontal Bearing)	1998: 5.2.59.c, 5.3.7.3.s.3	Zero	This field specifies the horizontal angle of the maximum response axis for this beam. The value is specified in radians from own ship heading with positive being used for the clockwise direction.
BeamIdentifier	Underwater Acoustics	Beam ID Number	1998: 5.3.7.3.s.2	**(Not Optional)**	This field specifies a unique number assigned to differentiate between multiple emitter beams within an Underwater Acoustics (UA) emitter system.
ElevationBeamwidth	Underwater Acoustics	Fundamental Data Parameters . D/E Beam Width (Vertical Beam Width)	1998: 5.2.59.f , 5.3.7.3.s.3	Zero	This field specifies the vertical beamwidth of the main beam. Beamwidth is measured at the 3 dB point down from maximum response axis. A value of zero indicates omni-directional.
ElevationCenter	Underwater Acoustics	Fundamental Data Parameters . Main Beam D/E	1998: 5.2.59.e, 5.3.7.3.s.3	Zero	This field specifies the vertical angle of the maximum response axis for this beam. The value is specified in degrees from the horizontal with positive being used for the downward direction.
EventIdentifier	Underwater Acoustics	Event ID	1998: 5.3.7.3.c	**(Not Optional)**	This field is used to coordinate changes between an active sonar beam and the ActiveSonar object associated with it.
ScanPattern	Underwater Acoustics	Fundamental Data Parameters . Scan Pattern	1998: 5.2.59.b, 5.3.7.3.s.3	Scan pattern not used	An enumeration of Scan Patterns to be used if applicable.

This object class uses EventIdentifier to coordinate updates between each ActiveSonar and all of the ActiveSonarBeams that it emits. When the ActiveSonar changes its basic configuration it shall update its own EventIdentifier. The ActiveSonarBeams associated with this change shall, at a minimum, copy their EventIdentifiers from the ActiveSonar to show their understanding of the new mode. Although this does not guarantee that all sonar changes will be received in a single network update (like DIS), it does provide a cause and effect relationship between the beams and their sonar system.

7.12.2 Interaction Classes

Figure 15 Underwater Acoustics Module Interaction Class Structure

7.12.2.1 AcousticTransient Interaction Class

This interaction class describes the transient component of non-propulsion passive emissions such as torpedo tube flooding, hatch slams, and wrench drops. The duration of a transient event is not required to be infinitesimally small, but they are usually considered to be less than a few minutes. The AdditionalPassiveActivities object class has almost the same structure, but it is used for long duration events. Unlike AdditionalPassiveActivities, the AcousticTransient does not support the concept of repeated transmission with a maintenance of state information between updates.

Senders of this interaction class shall provide values for the ActivityCode and HostObjectIdentifier. All of the other parameters shown in Table 70 shall be treated as optional fields.

Table 70 AcousticTransient Parameters

Attribute Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ActivityCode	Underwater Acoustics	APA Parameter Index	1998: 5.3.7.3.m	**(Not Optional)**	This field indicates the database entry that shall be used to model the passive signature of the entity.
ActivityParameter	Underwater Acoustics	APA Value	1998: 5.3.7.3.n	Zero	Current state of this activity.
HostObjectIdentifier	Underwater Acoustics	Emitting Entity ID	1998: 5.3.7.3.b	**(Not Optional)**	Object instance ID of the host that created this sound.
RelativePosition	N/A	N/A	N/A	All Zeros	Location of the sound with respect to the host's coordinate system.

The AcousticTransient interaction class defines passive signatures through references to a common federation database such as the Additional Narrowband Database (ANDB). Currently, each federation has to define this association at federation development time. The ActivityCode identifies a particular version of the additional passive activity to be modeled. The ActivityParameter provides a single integer parameter that can be used to control the acoustic emission. The interpretation of the ActivityParameter field depends on which additional passive activities database is used.

7.13 Warfare Module

This module defines the interaction classes related to weapons, expendables, and to any type of explosion whether or not it is related to munitions.

The Warfare module depends on the Base module for datatype definitions.

7.13.1 Interaction Classes

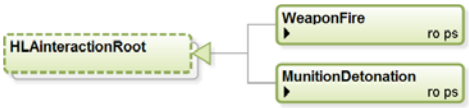

Figure 16 Warfare Module Interaction Class Structure

The DIS protocols allowed for two types of munitions. In general, small munitions were tracked at just the launch and impact points using the Fire PDU and Detonation PDU. (Issuing rules for these PDUs are described in Section 4.5.3 of IEEE Std 1278.1™-1995 [6].) Simulation developers also had the option of tracking larger weapon (torpedoes, missiles, etc.) through their transit by treating them as independent entities. The WeaponFire interaction alerts simulation participants of each weapon firing for either type of weapon. WeaponFire contains sufficient information so that the weapon can be tracked off-line without creating a corresponding Munition. The MunitionDetonation interaction alerts simulation participants when the weapon is detonated, and includes information used in battle damage assessment models.

7.13.1.1 WeaponFire Interaction Class

The Weapon Fire interaction alerts all subscribed simulation participants when a weapon is fired. The interaction class shall be issued regardless of whether the munition will be tracked off-line, or simulated on-line using a corresponding Munition entity. The form of this interaction class closely follows the layout of the FirePDU described in Section 5.3.4.1 of IEEE Std 1278.1™-1995 [6]. The "FireControlSolutionRange," "FireMissionIndex," "QuantityFired," "TargetObjectIdentifier," "MunitionObjectIdentifier" and "RateOfFire" are optional fields; senders of this interaction class shall provide values for all other parameters.

Table 71 WeaponFire Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EventIdentifier	Fire	Event ID	1995: 5.3.4.1.e	**(Not Optional)**	ID generated by the firing entity to associate related fire and detonation interactions.
FireControlSolutionRange	Fire	Range	1995: 5.3.4.1.j	Zero	Range in meters assumed by firing entity in computing the fire control solution. Zero if range is unknown or inapplicable.
FireMissionIndex	Fire	Fire Mission Index	1995: 5.3.4.1.f	NO_FIRE_MISSION	This attribute is intended for after-action review purposes and can be used to denote a specific assignment given to the weapon system, i.e., the fire mission, with each assignment being given a different value. If used, the Fire Mission Index shall be set to a value other than NO_FIRE_MISSION.
FiringLocation	Fire	Location in World Coordinates	1995: 5.3.4.1.g	**(Not Optional)**	The location, in world coordinates, from which the munition was launched.
FiringObjectIdentifier	Fire	Firing Entity ID	1995: 5.3.4.1.b	**(Not Optional)**	Object instance ID of the entity firing the munition.
FuseType	Fire	BurstDescriptor: Fuse	1995: 5.2.7.c, 5.3.4.1.h	**(Not Optional)**	The type of fuse specified by a 16-bit enumeration.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
InitialVelocityVector	Fire	Velocity	1995: 5.3.4.1.i	**(Not Optional)**	Velocity of the fired munition at the point when externally visible effects become apparent (e.g., exhaust plume or muzzle flash).
MunitionObjectIdentifier	Fire	Munition ID	1995: 5.3.4.1.d	No Specific Entity	Object instance ID of the fired munition, if an object instance is registered. Used only for tracked munitions.
MunitionType	Fire	BurstDescriptor: Munition	1995: 5.2.7.a, 5.3.4.1.h	**(Not Optional)**	Kind, Country, Domain, Category, Subcategory, Specific, and Extra fields of the munition being fired.
QuantityFired	Fire	BurstDescriptor: Quantity	1995: 5.2.7.d, 5.3.4.1.h	One	The number of rounds fired in the burst.
RateOfFire	Fire	Burst Descriptor: Rate	1995: 5.2.7.d, 5.3.4.1.h	Zero	Rate of fire in rounds per minute when quantity > 1. Zero otherwise.
TargetObjectIdentifier	Fire	Target Entity ID	1995: 5.3.4.1.c	No Specific Entity	Object instance ID of the intended target (if any).
WarheadType	Fire	BurstDescriptor: Warhead	1995: 5.2.7.b, 5.3.4.1.h	**(Not Optional)**	The type of warhead specified by a 16-bit enumeration.

7.13.1.2 MunitionDetonation Interaction Class

The MunitionDetonation interaction alerts all simulation participants that a weapon has detonated. The MunitionDetonation could have been preceded by a WeaponFire interaction when the munition was fired, or could stand on its own as in the case of a mine detonation. The form of this interaction class closely follows the layout of the Detonation PDU used by DIS. The “ArticulatedPartData,” “DetonationResultCode,” “QuantityFired,” “TargetObjectIdentifier,” “MunitionObjectIdentifier,” and “RateOfFire” parameters shall be always treated as optional fields. The sender shall provide data for “FiringObjectIdentifier” and “FinalVelocityVector” for all munitions other than mines. All other parameters shall be unconditionally required.

Table 72 MunitionDetonation Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ArticulatedPartData	Detonation	Articulation Parameters	1995: 5.2.5, 5.3.4.2.l	Empty	Articulated Parts info is included when the firer determines an articulated part of the target entity has been affected by the detonation.
DetonationLocation	Detonation	Location in World Coordinates	1995: 5.3.4.2.g	**(Not Optional)**	The location, in world coordinates, at which the munition detonated.
DetonationResultCode	Detonation	Detonation Result	1995: 5.3.4.2.j	Other	The type of detonation (Entity Impact, Ground Impact, Entity Proximate Detonation, etc.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
EventIdentifier	Detonation	Event ID	1995: 5.3.4.2.e	**(Not Optional)**	ID generated by the firing entity to associate related fire and detonation interactions.
FiringObjectIdentifier	Detonation	Firing Entity ID	1995: 5.3.4.2.b	*Required if the munition type is not a mine.*	Object instance ID of the entity that fired the munition.
FinalVelocityVector	Detonation	Velocity	1995: 5.3.4.2.f	*Required if the munition type is not a mine.*	The velocity vector of the munition at the moment of the detonation.
FuseType	Detonation	BurstDescriptor: Fuse	1995: 5.2.7.c, 5.3.4.2.h	**(Not Optional)**	The type of fuse specified by a 16-bit enumeration.
MunitionObjectIdentifier	Detonation	Munition ID	1995: 5.3.4.2.d	No Specific Entity	Object instance ID of the fired munition, if an object instance is registered. Used only for tracked munitions.
MunitionType	Detonation	BurstDescriptor: Munition	1995: 5.2.7.a, 5.3.4.2.h	**(Not Optional)**	Kind, Country, Domain, Category, Subcategory, Specific, and Extra fields of the detonated munition.
QuantityFired	Detonation	BurstDescriptor: Quantity	1995: 5.2.7.d, 5.3.4.2.h	One	The number of rounds fired in the burst.
RateOfFire	Detonation	Burst Descriptor: Rate	1995: 5.2.7.d, 5.3.4.2.h	Zero	Rate of fire in rounds per minute when quantity > 1. Zero otherwise.
RelativeDetonationLocation	Detonation	Location in Entity Coordinates	1995: 5.3.4.2.i	*Required if TargetObjectIdentifier is provided.*	The location, in coordinates relative to the target entity, at which the munition detonated.
TargetObjectIdentifier	Detonation	Target Entity ID	1995: 5.3.4.2.c	No Specific Entity	Object instance ID of the intended target (if any).
WarheadType	Detonation	BurstDescriptor: Warhead	1995: 5.2.7.b, 5.3.4.2.h	**(Not Optional)**	The type of warhead specified by a 16-bit enumeration.

7.14 Logistics Module

The Logistics module defines interactions that represent repair, service, and resupply logistic activities. These interactions support both requests by simulation objects in need of service, and responses by simulation objects which are able to provide the service.

The Logistics module depends on the Base module for datatype definitions.

7.14.1 Interaction Classes

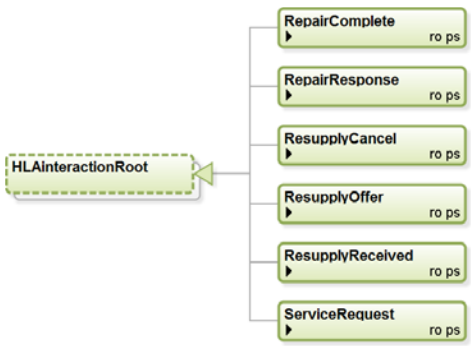

Figure 17 Logistics Module Interaction Class Structure

The logistics interaction classes are used to represent the discrete events associated with the request for logistic services and transfer of supplies between two objects. Each interaction class mirrors the purpose and data of a logistics family PDU defined within DIS 1995. Four (ServiceRequest, ResupplyOffer, ResupplyReceived, and ResupplyCancel) are used to simulate resupply. Three (ServiceRequest, RepairResponse and RepairComplete) are used to simulate repair. The representation of logistics services with the logistics interaction classes re-uses the state transition and timers defined in Clause 4.5.4 of IEEE Std 1278.1™-1995 [6].

7.14.1.1 RepairComplete Interaction Class

An instance of the RepairComplete interaction class shall be published by the federate simulating a repairing object upon the completion of the repair as described in IEEE Std 1278.1™-1995 [6] Sections 4.5.4.3 and 4.5.4.9. Federates publishing this interaction class shall provide values for all parameters; there are no optional parameters.

Table 73 RepairComplete Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivingObject	Repair Complete	Receiving Entity ID	1995: 5.3.5.5.b	**(Not Optional)**	Object requesting repair.
RepairingObject	Repair Complete	Repairing Entity ID	1995: 5.3.5.5.c	**(Not Optional)**	The object performing the repair.
RepairType	Repair Complete	Repair	1995: 5.3.5.5.d	**(Not Optional)**	One of the enumerated repairs.

7.14.1.2 RepairResponse Interaction Class

An instance of the RepairResponse interaction class shall be published by the federate simulating the object receiving repairs on receipt of a RepairComplete interaction, as described in IEEE Std 1278.1™-1995 [6] Sections 4.5.4.3 and 4.5.4.10. Federates publishing this interaction class shall provide values for all parameters; there are no optional parameters.

Table 74 RepairResponse Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivingObject	Repair Response	Receiving Entity ID	1995: 5.3.5.6.b	**(Not Optional)**	Object requesting repair.
RepairingObject	Repair Response	Repairing Entity ID	1995: 5.3.5.6.c	**(Not Optional)**	Object performing the repair.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RepairResult Code	Repair Response	Repair Result	1995: 5.3.5.6.d	**(Not Optional)**	One of the enumerated repair results.

7.14.1.3 ResupplyCancel Interaction Class

An instance of ResupplyCancel interaction class shall be published by the federate simulating the resupplying or receiving object when conditions for resupply are no longer met, as described in IEEE Std 1278.1™-1995 [6] Sections 4.5.4.2 and 4.5.4.7. Federates publishing this interaction class shall provide values for all parameters; there are no optional parameters.

Table 75 ResupplyCancel Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivingObject	Resupply Cancel	Receiving Entity ID	1995: 5.3.5.4.b	**(Not Optional)**	Object that has requested resupply.
SupplyingObject	Resupply Cancel	Supplying Entity ID	1995: 5.3.5.4.c	**(Not Optional)**	Supplying object.

7.14.1.4 ResupplyOffer Interaction Class

An instance of the ResupplyOffer interaction class shall be published by the federate simulating the object identified in a previously published ServiceRequest interaction, as described in IEEE Std 1278.1™-1995 [6] Sections 4.5.4.2.2 and 4.5.4.5. Federates publishing this interaction class shall provide values for all parameters; there are no optional parameters.

Table 76 ResupplyOffer Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivingObject	Resupply Offer	Receiving Entity ID	1995: 5.3.5.2.b	**(Not Optional)**	Object that has requested resupply.
SupplyingObject	Resupply Offer	Supplying Entity ID	1995: 5.3.5.2.c	**(Not Optional)**	Supplying object.
SuppliesData	Resupply Offer	Number of Supply Types and Supplies	1995: 5.3.5.2.e	**(Not Optional)**	List of offered supplies.

7.14.1.5 ResupplyReceived Interaction Class

An instance of the ResupplyReceived interaction class shall be published by the federate simulating the identified receiving object to report what supplies have been transferred, as described in IEEE Std 1278.1™-1995 [6] Sections 4.5.4.2.1 and 4.5.4.6. Federates publishing this interaction class shall provide values for all parameters; there are no optional parameters.'

Table 77 ResupplyReceived Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivingObject	Resupply Received	Receiving Entity ID	1995: 5.3.5.3.b	**(Not Optional)**	Object that has requested resupply.
SupplyingObject	Resupply Received	Supplying Entity ID	1995: 5.3.5.3.c	**(Not Optional)**	Supplying object.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
SuppliesData	Resupply Received	Number of Supply Types Supplies	1995: 5.3.5.3.e	**(Not Optional)**	List of supplies taken by receiving entity.

7.14.1.6 ServiceRequest Interaction Class

An instance of the ServiceRequest interaction class shall be published by the federate simulating an object requesting repair or resupply when appropriate conditions for repair or resupply exist, as described in IEEE Std 1278.1™-1995 [6] Section 4.5.4.4. Federates publishing this interaction class shall only publish the SuppliesData parameter if the ServiceType parameter is assigned to Resupply. Values for all other parameters shall be provided.

Table 78 ServiceRequest Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RequestingObject	Service Request	Requesting Entity ID	1995: 5.3.5.1.b	**(Not Optional)**	Object that has requested service.
ServicingObject	Service Request	Servicing Entity ID	1995: 5.3.5.1.c	**(Not Optional)**	Object able to provide the requested service.
ServiceType	Service Request	Service Type Requested	1995: 5.3.5.1.d	**(Not Optional)**	Type of requested service.
SuppliesData	Service Request	Number of Supply Types and Supplies	1995: 5.3.5.1.f	*Required if ServiceType is Resupply. Otherwise, this parameter is not sent.*	List of type and number of supplies requested. The unit measure depends on the supply type and shall use the SI units of measurement used for such supplies.

7.15 Simulation Management Module

The purpose of the Simulation Management module is to define interaction classes that to allow exercise personnel to manage an exercise by requesting that certain actions be taken by another or all federates, or by a federate that owns a specific entity.

This module depends on the Base module for datatype definitions.

7.15.1 Interaction Classes

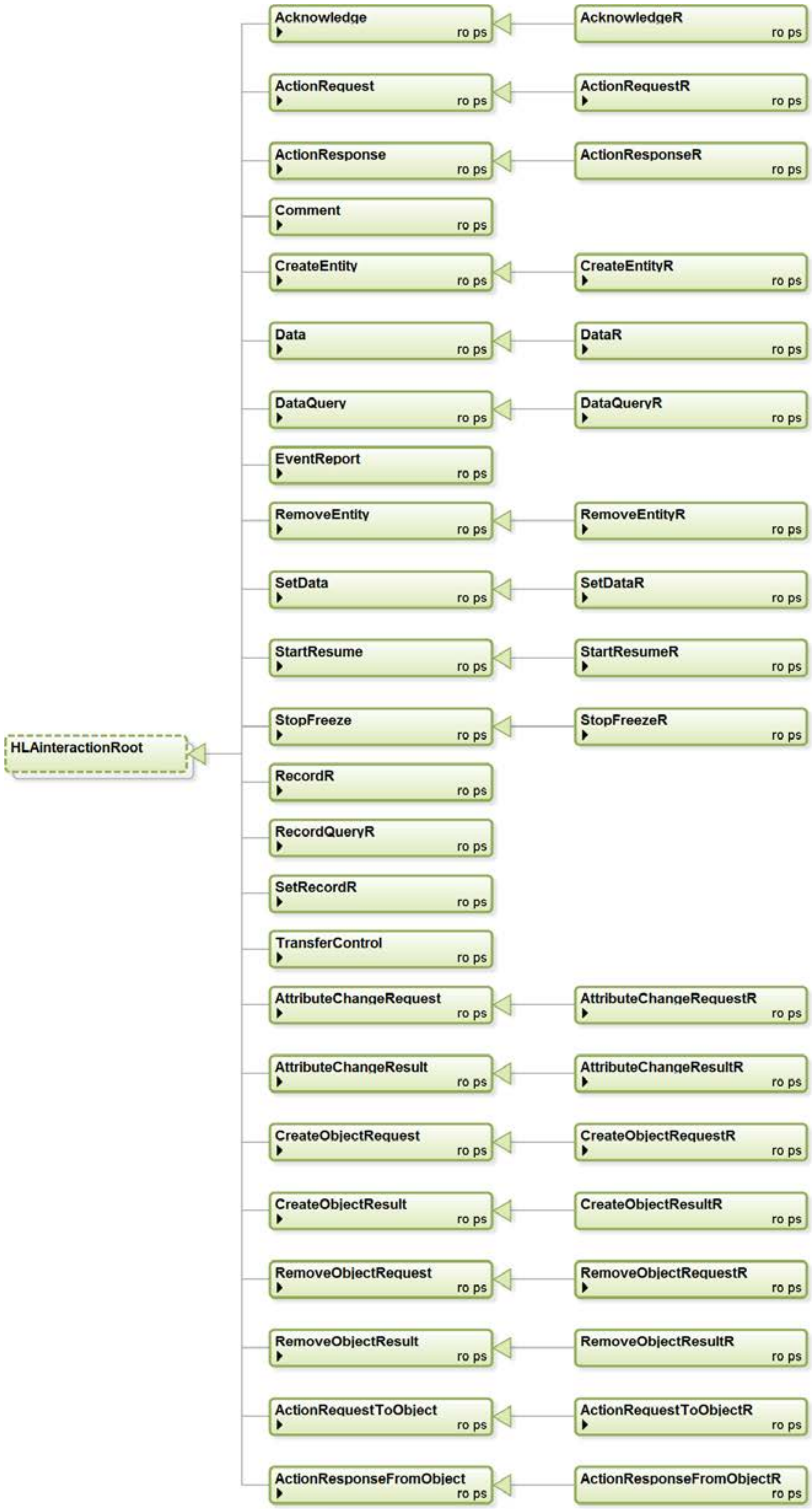

Figure 18 Simulation Management Module Interaction Class Structure

7.15.1.1 DIS Simulation Management Family

Although the RTI provides many simulation management (SIMAN) functions, some DIS capabilities, such as the group addressing scheme supported by the EntityIdentifier, are not supported directly by the RTI services. This family provides a direct porting of the DIS SIMAN PDUs into a set of HLA interaction classes. State diagrams for these interaction classes are the same, in each case, as the corresponding diagrams in IEEE Std 1278.1™-1995 [6] Section 4.4.5. In each case, the interaction class name and parameter names were developed to closely match those of DIS. Although federates are not required to support this family of interaction classes, failure to do so could limit the system's ability to interact with federates derived from legacy DIS simulation management systems. A negative response to simulation management requests shall be provided as a minimum to support this family.

DIS supported multiple levels of characteristic visibility. Characteristics described in the PDU structures were "public" values with full visibility to all simulations. However, through the Simulation Management (SIMAN) PDUs, another set of "private" characteristics could also be manipulated. These "private" characteristics were generally considered to be those components of the entity that had no interoperability impact during an exercise, but that might be required for data collection or after-action review. In the interest of supporting the transition of legacy DIS simulations into the HLA, these visibility rules are maintained by the RPR FOM.

The interaction classes supported within the DIS Simulation Management Family are complementary to those supported within the HLA Simulation Management Family. It is recommended that RPR FOM federates support both capabilities to ensure maximal interoperability with RPR FOM management facilities.

In addition to supporting addressing of a single entity, the DIS addressing scheme also supports group addressing. In 1278.1-1995 [6] three values are defined with the symbolic names ALL_SITES, ALL_APPLIC, and ALL_ENTITIES. These values can be used as the site identifier, application identifier, or entity identifier fields of Receiving Entity ID in a DIS SIMAN PDU. This addressing scheme is supported directly by the ReceivingEntity parameter in the RPR FOM SIMAN interaction classes.

Many of these interaction classes have subclasses to support SIMAN with reliability. The requesting interaction classes add an additional parameter specifying the level of reliability required for the response. The responding interaction classes do not add any parameters. The authoritative explanation for SIMAN with reliability is in 1278.1a-1998 4.5.11.

7.15.1.1.1 Acknowledge Interaction Class

This interaction class is sent in return to some type of SIMAN request (StartResume, StopFreeze, CreateEntity, RemoveEntity, TransferControl) made by the originating entity. It identifies which request the acknowledge is in response to, the actual type of request that was made, and the response to this request. All parameters in this interaction class shall be required. The value of the RequestIdentifier comes from the originating request and allows the recipient of the acknowledgement to match it with an original request.

Table 79 Acknowledge Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Acknowledge	Originating Entity ID	1995: 5.2.29.b, 5.3.6.5.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this response.
ReceivingEntity	Acknowledge	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.5.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient of this response.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RequestIdentifier	Acknowledge	Request ID	1995: 5.3.6.5.d	**(Not Optional)**	This field matches this response with the specific StartResume, StopFreeze, CreateEntity or RemoveEntity interaction sent by the simulation manager.
AcknowledgeFlag	Acknowledge	Acknowledge Flag	1995: 5.3.6.5.b	**(Not Optional)**	Enumeration for the acknowledgement.
ResponseFlag	Acknowledge	Response Flag	1995: 5.3.6.5.c	**(Not Optional)**	The type of response made to the interaction by the recipient.

Although some of the functionality of the Acknowledge interaction class is simplified by the federate's ability to specify reliable transportation under HLA, a response message is still required to allow federates to reject simulation management requests.

7.15.1.1.2 AcknowledgeR Interaction Class

This interaction class is a subclass of Acknowledge that provides a reliable response to a StartResumeR, StopFreezeR, CreateEntityR, or RemoveEntityR interaction class. The AcknowledgeR interaction class does not add any parameters to the parent class Acknowledge interaction class.

7.15.1.1.3 ActionRequest Interaction Class

This interaction class requests an entity to perform some type of action. An entity or application responds to this interaction with an ActionResponse interaction. The ActionRequest closely resembles the Action Request PDU with the primary differences being the lack of a PDU header, a number indicating how many fixed datums and a number indicating how many variable length datums are contained in the request. The "FixedDatums" and "VaribleDatumsSet" are used in conjunction with the action that is being requested, and vary with each type of action being requested. The "OriginatingEntity", "ReceivingEntity", "RequestIdentifier", and "ActionRequestCode" shall be considered required fields in this interaction class.

Table 80 ActionRequest Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Action Request	Originating Entity ID	1995: 5.2.29.b, 5.3.6.6.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this request.
ReceivingEntity	Action Request	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.6.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RequestIdentifier	Action Request	Request ID	1995: 5.3.6.6.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
ActionRequestCode	Action Request	Action ID	1995: 5.3.6.6.c	**(Not Optional)**	Enumeration that specifies the specific action requested.
FixedDatums	Action Request	Fixed Datum	1995: 5.2.10, 5.3.6.6.d	Empty	A set of fixed length data items where each element specifies the type of item (enumeration) and its value (specific data type).
VariableDatumSet	Action Request	Variable Datum	1995: 5.2.10, 5.3.6.6.d	Empty	A set of variable length data items where each element specifies the type of item (enumeration), its length, and its value (specific data type).

7.15.1.1.4 ActionRequestR Interaction Class

This interaction class is a subclass of ActionRequest that requests an entity to perform some type of action using reliable protocol. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 81 ActionRequestR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Action Request-R	Required Reliability Service	1998: 5.3.12.6 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.5 ActionResponse Interaction Class

This interaction class returns a response to an earlier ActionRequest interaction. The “RequestIdentifier” originates from the ActionRequest interaction, and the “ResultStatus” contains the status of the requested action. The “FixedDatums” and the “VariableDatumSet” contain any relevant information that is being returned in response to the requested action. This interaction class closely mimics the DIS ActionResponse PDU, and has the same differences as the ActionRequest interaction class. The “OriginatingEntity”, “ReceivingEntity”, ”RequestIdentifier”, and “RequestStatus” shall be considered required fields in this interaction class. The “FixedDatums” and “VariableDatumSet” vary with the type of action that was requested.

Table 82 ActionResponse Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Action Response	Originating Entity ID	1995: 5.2.29.b, 5.3.6.7.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this response.
ReceivingEntity	Action Response	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.7.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient of this response.
RequestIdentifier	Action Response	Request ID	1995: 5.3.6.7.b	**(Not Optional)**	This field matches this response with the specific ActionRequest interaction sent by the simulation manager.
RequestStatus	Action Response	Request Status	1995: 5.3.6.7.c	**(Not Optional)**	Enumeration identifying the status of the response
FixedDatums	Action Response	Fixed Datums	1995: 5.2.10, 5.3.6.7.d	Empty	A set of fixed length data items where each element specifies the type of item (enumeration) and its value (specific data type).
VariableDatumSet	Action Response	Variable Datums	1995: 5.2.10, 5.3.6.7.d	Empty	A set of variable length data items where each element specifies the type of item (enumeration), its length, and its value (specific data type).

7.15.1.1.6 ActionResponseR Interaction Class

The ActionResponseR interaction is a subclass of ActionResponse that is sent by a simulation to respond to an ActionRequestR interaction class. The ActionResponseR interaction class does not add any parameters to the parent interaction class.

7.15.1.1.7 Comment Interaction Class

The Comment interaction class is used to insert messages and information into a log stream and closely matches the structures used by the Data interaction class. This information is usually unsolicited in nature. The information contained within the Interaction should be used for commenting purposes only.

Table 83 Comment Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Comment	Originating Entity ID	1995: 5.2.29.b, 5.3.6.12.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this interaction.
ReceivingEntity	Comment	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.12.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this interaction. Group addressing is allowed.
VariableDatumSet	Comment	Variable Datum	1995: 5.2.32, 5.3.6.12.d	Empty	A set of variable length data items where each element specifies the type of item (enumeration), its length, and its value (specific data type).

7.15.1.1.8 CreateEntity Interaction Class

This interaction class is used to request the creation of a new entity and closely matches the Create Entity PDU. All parameters in this request shall be required. The results of the CreateEntity are returned via an Acknowledgement interaction. The "EntityNumber" contained within the "ReceivingEntity" can contain two types of valid values. If the value is between 1 – 65533 this is the exact entity number requested to be created. If the value contains 65534, it is a placeholder value for the recipient to use the next entity number it has available. The numbers 0 and 65535 are invalid for EntityNumbers. The number 0 is reserved for applications, and the number 65535 is reserved to mean all entities.

Table 84 CreateEntity Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Create	Originating Entity ID	1995: 5.2.29.b, 5.3.6.1.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this request.
ReceivingEntity	Create	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.1.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.
RequestIdentifier	Create	Request ID	1995: 5.3.6.1.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.

7.15.1.1.9 CreateEntityR Interaction Class

This interaction class is a subclass of CreateEntity that is used to request the creation of a new entity using reliable protocol. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 85 CreateEntityR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Create Entity-R	Required Reliability Service	1998: 5.3.12.1 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.10 Data Interaction Class

Data interactions are usually solicited responses to DataQuery and SetData interactions, and closely resemble DIS Data PDU. The "OriginatingEntity", "ReceivingEntity", and "RequestIdentifier" parameters shall be required and the contents of the "FixedDatums" and "VariableDatumSet" parameters vary with the type of data being sent with the interaction. The value of the "RequestIdentifier" parameter shall come from the originating solicitation of the Data interaction, and at least one "FixedDatum" or "VariableDatumSet" parameter should be present, but it is not required. An example of a DataInteraction that would contain no fixed or variable information would be a response to a set data that a simulator does not model any of the requested datums contained in the SetData. It is also possible for the receiver to get multiple responses to the same request since the reply could contain multiple variable length datums that are greater than the maximum size allowed for a single network packet.

Table 86 Data Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Data	Originating Entity ID	1995: 5.2.29.b, 5.3.6.10.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this response.
ReceivingEntity	Data	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.10.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient of this response.
RequestIdentifier	Data	Request ID	1995: 5.3.6.10.d	**(Not Optional)**	This field matches this response with the specific SetData or DataQuery interaction sent by the simulation manager.
FixedDatums	Data	Fixed Datum	1995: 5.2.10, 5.3.6.10.c	Empty	A set of fixed length data items where each element specifies the type of item (enumeration) and its value (specific data type).
VariableDatumSet	Data	Variable Datum	1995: 5.2.10, 5.3.6.10.c	Empty	A set of variable length data items where each element specifies the type of item (enumeration), its length, and its value (specific data type).

7.15.1.1.11 DataR Interaction Class

This interaction class is a subclass of Data that requests an entity to perform some type of action using reliable protocol. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 87 DataR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Action RequestR	Required Reliability Service	1998: 5.3.12.10 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.12 DataQuery Interaction Class

SIMAN DataQuerys allow for the solicitation of data, and takes the place of DIS Data Query PDU. The "TimeInterval" parameter allows the originator to request the receiving entity to periodically send the requested information at the specified interval. The field "TimeInterval" shall be optional. If this value is zero or it is not provided, the recipient need only respond to the Data Query with a single Data interaction. The "OriginatingEntity", "ReceivingEntity", "RequestIdentifier" parameters shall be required. The "FixedDatum" and "VariableDatums" parameters indicate the attributes being queried and at least one of these parameters should be present but are considered optional. A query that contains no fixed or variable datums is not forbidden, but it is not a good practice, since it does nothing more than consume processing and network resources.

Table 88 DataQuery Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Data	Originating Entity ID	1995: 5.2.29.b, 5.3.6.8.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this request.
ReceivingEntity	Data	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.8.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient of this request. Group addressing is allowed.
RequestIdentifier	Data	Request ID	1995: 5.3.6.8.c	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
TimeInterval	Data	Time Interval	1995: 5.3.6.8.b	Zero	Timestamp indicating the amount of time that shall elapse between continued responses to this request. Represented as a timestamp as described in section 5.2.31 of IEEE Std 1278.1™-1995 [6].
FixedDatumIdentifiers	Data	Fixed Datum	1995: 5.2.10, 5.3.6.8.d	Empty	The set of datum identifiers that specify the requested fixed length datums.
VariableDatumIdentifiers	Data	Variable Datum	1995: 5.2.10, 5.3.6.8.d	Empty	The set of datum identifiers that specify the requested variable length datums.

7.15.1.1.13 DataQueryR Interaction Class

The DataQueryR Interaction is a subclass of DataQuery that is used to request data from a remote federate using reliable protocol. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 89 DataQueryR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Data Query-R	Required Reliability Service	1998: 5.3.12.8b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.14 EventReport Interaction Class

When a significant event occurs on a managed entity, the entity reports these incidents to the simulation manager through Event Reports. This interaction class closely mimics the Event Report PDU utilized by DIS. The “OriginatingEntity”, “ReceivingEntity”, and “EventType” shall be required. The contents of the

"FixedDatums" and "VariableDatumSet" are contingent on the type of event being reported and vary with the EventType.

Table 90 EventReport Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	Event Report	Originating Entity ID	1995: 5.2.29.b, 5.3.6.11.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this report.
ReceivingEntity	Event Report	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.11.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this report. Group addressing is allowed.
EventType	Event Report	Event Type	1995: 5.3.6.11.b	**(Not Optional)**	Enumeration indicating the type of event that caused the issuance of the Event Report Interaction
FixedDatums	Event Report	Fixed Datum	1995: 5.2.10, 5.3.6.11.d	Empty	A set of fixed length data items where each element specifies the type of item (enumeration) and its value (specific data type).
VariableDatumSet	Event Report	Variable Datum	1995: 5.2.10, 5.3.6.11.d	Empty	A set of variable length data items where each element specifies the type of item (enumeration), its length, and its value (specific data type).

7.15.1.1.15 RemoveEntity Interaction Class

This interaction class is used to request the removal of an entity and closely matches the DIS Remove Entity PDU. All parameters in this request shall be required. This interaction class differs from the native HLA mechanism by the wide variety of responses possible in the Acknowledge interaction.

Table 91 RemoveEntity Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	RemoveEntity	Originating Entity ID	1995: 5.2.29.b, 5.3.6.2.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating the request.
ReceivingEntity	RemoveEntity	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.2.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RequestIdentifier	RemoveEntity	Request ID	1995: 5.3.6.2.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.

7.15.1.1.16 RemoveEntityR Interaction Class

This interaction class is a subclass of RemoveEntity that is used to request the removal of an entity using reliable protocol. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 92 RemoveEntityR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Remove Entity-R	Required Reliability Service	1998: 5.3.12.2 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction.

7.15.1.1.17 SetData Interaction Class

The SetData Interaction is used to request a remote federate to set the value of specified data. It is often used to initialize new entities once they've been created. This interaction class maps to the DIS Set Data PDU. "OriginatingEntity", "ReceivingEntity", and "RequestIdentifier" shall be required. The "FixedDatum" and "VariableDatums" indicate the attributes being queried and at least one of these parameters should be present but are considered optional and vary from interaction to interaction.

Table 93 SetData Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	SetData	Originating Entity ID	1995: 5.2.29.b, 5.3.6.9.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating the request.
ReceivingEntity	SetData	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.9.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RequestIdentifier	SetData	Request ID	1995: 5.3.6.9.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
FixedDatums	SetData	Fixed Datum	1995: 5.2.10, 5.3.6.9.c	Empty	A set of fixed length data items where each element specifies the type of item (enumeration) and its value (specific data type).
VariableDatumSet	SetData	Variable Datum	1995: 5.2.10, 5.3.6.9.c	Empty	A set of variable length data items where each element specifies the type of item (enumeration), its length, and its value (specific data type).

7.15.1.1.18 SetDataR Interaction Class

The SetDataR interaction is a subclass of SetData that is used to reliably request a remote federate to set the value of specified data. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 94 SetDataR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Set Data-R	Required Reliability Service	1998: 5.3.12.9b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.19 StartResume Interaction Class

This interaction informs federates that they should begin updating particular entities. It differs from the native HLA mechanisms for the behavior because it supports the DIS mechanisms for entity addressing and time of day. This interaction class closely follows the format of the DIS Start/Resume PDU. All parameters in this interaction class shall be required.

Table 95 StartResume Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	StartResume	Originating Entity ID	1995: 5.2.29.b, 5.3.6.3.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating the request.
ReceivingEntity	StartResume	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.3.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RealWorldTime	StartResume	Real-World Time	1995: 5.2.8, 5.3.6.3.b	**(Not Optional)**	Greenwich Mean Time (GMT) that the entity shall be started/resumed at.
RequestIdentifier	StartResume	Request ID	1995: 5.3.6.3.d	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
SimulationTime	StartResume	Simulation Time	1995: 5.2.8,5.3.6.3.c	**(Not Optional)**	The simulation time that the entity or entities shall use when they start/resume.

The “RealWorldTime” parameter is the time that the start shall take effect, and when it takes affect the “SimulationTime” indicates what the current simulation time will be. For example, if the real world time is set to 17:00 GMT and the simulation time is 11:00, when the wall clock on the simulator reaches 17:00, it starts its simulation, and uses 11:00 as its simulation time. But, just like any other standard, there is always an exception. While there was no specific rule in DIS to indicate this, common usage holds that simulations shall start immediately if the RealWorld time is set to 0:00.

7.15.1.1.20 StartResumeR Interaction Class

This interaction is a subclass of StartResume that informs federates using reliable protocol that they shall begin updating particular entities. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 96 StartResumeR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Start/Resume-R	Required Reliability Service	1998: 5.3.12.3d	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.21 StopFreeze Interaction Class

This interaction informs federates that they shall stop updating particular entities. It differs from the native HLA mechanisms for the behavior because it supports the DIS mechanisms for entity addressing and time of day. This interaction class closely follows the format of the DIS Stop/Freeze PDU. All parameters in this interaction class shall be required.

Table 97 StopFreeze Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
OriginatingEntity	StopFreeze	Originating Entity ID	1995: 5.2.29.b, 5.3.6.4.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating the request.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ReceivingEntity	StopFreeze	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.4.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.
RequestIdentifier	StopFreeze	Request ID	1995:5.3.6.4.e	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
RealWorldTime	StopFreeze	Real-World Time	1995: 5.2.8,5.3.6.4.b	**(Not Optional)**	GMT that the entity shall be stopped/frozen.
Reason	StopFreeze	Reason	1995: 5.3.6.4.c	**(Not Optional)**	8 bit enumeration indicating the reason why the entity/simulation is frozen.
ReflectValues	StopFreeze	Frozen Behavior	1995: 5.3.6.4.d	**(Not Optional)**	True if the entities should continue to reflect incoming attributes while stopped/frozen.
RunInternalSimulationClock	StopFreeze	Frozen Behavior	1995: 5.3.6.4.d	**(Not Optional)**	True if the entities should continue to run their internal simulation clock when stopped/frozen.
UpdateAttributes	StopFreeze	Frozen Behavior	1995: 5.3.6.4.d	**(Not Optional)**	True if the entities should continue to update outgoing attributes while stopped/frozen.

Like the StartResume, common usage here holds that simulations shall stop immediately if the RealWorld time is set to 0:00.

7.15.1.1.22 StopFreezeR Interaction Class

This interaction is a subclass of StopFreeze that informs federates using reliable protocol that they shall stop updating particular entities. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 98 StopFreezeR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Stop/Freeze-R	Required Reliability Service	1998: 5.3.12.4e	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.1.23 RecordR Interaction Class

RecordR interactions are usually solicited responses to RecordQueryR and SetRecordR interactions, and closely resemble the DIS Record PDU. The “OriginatingEntity”, “ReceivingEntity”, and “RequestIdentifier” shall be required parameters and the contents of “Records” will vary with the type of data being sent with the interaction. The value of the “RequestIdentifier” shall come from the originating solicitation of the RecordR interaction. The “EventType” shall be an optional field that indicates the type of event being reported. The “ResponseSerialNumber” shall be an optional field used to indicate the serial number when more than one PDU is used to report record values. The “RecordSetData” field shall contain one “RecordSetList” field, but it is not required that the list have any records in it. An example of a RecordR Interaction that would contain no records would be a response to a SetRecordR in which the federate issuing the RecordR Interaction does not wish to model any of the Records contained within the SetRecordR. It is possible for the receiver to get multiple responses to the same request since the reply could contain lots of records that in total are greater than the maximum size allowed for a single network packet.

Table 99 RecordR Parameters

Parameter Name	**DIS PDU**	**DIS Field**	**IEEE 1278 References**	**Default Value (if optional)**	**Definition**
OriginatingEntity	Record-R	Originating Entity ID	1995: 5.2.29.b 1998: 5.3.12.15.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this response.
ReceivingEntity	Record-R	Receiving Entity ID	1995: 5.2.29.c 1998: 5.3.12.15.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient of this response.
RequestIdentifier	Record-R	Request ID	1998: 5.3.12.15.b	**(Not Optional)**	This field matches this response with the specific RecordQueryR or SetRecordR interaction sent by the simulation manager.
EventType	Record-R	Event Type	1998: 5.3.12.15.d	Zero	32-bit enumeration identifying the type of event reported.
AcknowledgementProtocol	Record-R	Required Reliability Service	1998: 5.3.12.15 c	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction
ResponseSerialNumber	Record-R	Response Serial Number	1998: 5.3.12.15.e	Zero	32 bit integer used to indicate the serial number of the Record Interaction when more than one Record interaction is used to report record values.
RecordSetData	Record-R	Record Sets	1998: 5.2.55, 5.3.12.15 f.	**(Not Optional)**	A set of “RecordSets” where each set specifies the type of its Records, its serial number, the number of records in the set. Actual Record lengths and values are stored in the “RecordStruct” structure.

7.15.1.1.24 RecordQueryR Interaction Class

SIMAN RecordQueryRs allow for the solicitation of data, and takes the place of DIS Record Query PDU. The “EventType” shall be an optional field that may specify the type of event that should prompt the issuance of a RecordR interaction from the receiving federate. If the value of “EventType” is zero or it is not provided, the reporting shall be periodic based on the interval specified in the “TimeInterval” field. The “TimeInterval” parameter allows the originator to request the receiving entity to periodically send the requested information at the specified interval. The field “TimeInterval” shall be optional. If this value is zero or it is not provided, the recipient need only respond to the RecordQueryR with a single RecordR interaction. The “OriginatingEntity”, “ReceivingEntity”, “RequestIdentifier” shall be required parameters. The “RecordIdentifiers” indicate the data being queried and at least one of these parameters should be present but are considered optional. A query that contains no “RecordIdentifiers” is not forbidden, but it is not good practice, since it does nothing more than consume processing and network resources.

Table 100 RecordQueryR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 References	Default Value (if optional)	Definition
OriginatingEntity	Record Query-R	Originating Entity ID	1995: 5.2.29.b 1998: 5.3.12.13.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating this request.
ReceivingEntity	Record Query-R	Receiving Entity ID	1995: 5.2.29.c 1998: 5.3.12.13.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.
RequestIdentifier	Record Query-R	Request ID	1998: 5.3.12.13.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
TimeInterval	Record Query-R	Time	1998: 5.3.12.13.e	Zero	Timestamp indicating the amount of time that shall elapse between continued responses to this request or a time for time-based reporting.
EventType	Record Query-R	Event Type	1998: 5.3.12.13.d	Zero	Specifies the type of event that will cause a Record Interaction to be sent.
AcknowledgementProtocol	Record Query-R	Required Reliability Service	1998: 5.3.12.13 c	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction
RecordIdentifiers	Record Query-R	Record ID	1998: 5.2.54, 5.3.12.13.f	Empty	The set of record identifiers that specify the requested records.

7.15.1.1.25 SetRecordR Interaction Class

Requests for the federate responsible for simulating an entity to set or change parameter values of an entity shall be made with a SetRecordR interaction. If appropriate to the request, the recipient of this interaction may call Update Attribute Values to notify the federation of said state changes. This interaction maps to the DIS Set Record PDU. “OriginatingEntity”, “ReceivingEntity”, “RequestIdentifier”, and “RecordSetData” shall be required parameters. The “RecordSetData” field shall contain one “RecordSetList” field indicating the attributes being set. This field contains a length specifying the number of RecordR sets. There should be at least one RecordSetR, although this is not required. The number of Records in each RecordSetR can vary based on the RecordSetStruct’s “NumberOfRecords” field. The length of each RecordR in a RecordSetR can vary based on the RecordStruct’s “Length” field. This mechanism is used to allow the federate responsible for simulating the entity to maintain control of the values and reject or modify request. It also allows values that are not associated with attributes to be set.

Table 101 SetRecordR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 References	Default Value (if optional)	Definition
OriginatingEntity	Set Record-R	Originating Entity ID	1995: 5.2.29.b, 1998: 5.3.12.14.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating the request.
ReceivingEntity	Set Record-R	Receiving Entity ID	1995: 5.2.29.c 1998: 5.3.12.14.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.
RequestIdentifier	Set Record-R	Request ID	1998: 5.3.12.14.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
AcknowledgementProtocol	Set Record-R	Required Reliability Service	1998: 5.3.12.14 d	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction
RecordSetData	Set Record-R	Record Sets	1998: 5.2.55, 5.3.12.14 c.	**(Not Optional)**	A set of “RecordSets” where each set specifies the type of its Records, its serial number, the number of records in the set. Actual Record lengths and values are stored in the “RecordStruct” structure.

7.15.1.1.26 TransferControl Interaction Class

Requests to transfer control of an entity are communicated with a TransferControl interaction. This interaction maps to the Transfer Control Request PDU. “OriginatingEntity”, “ReceivingEntity”, “RequestIdentifier”, and “RecordSetData” shall be required parameters. The “RecordSetData” field shall contain data as described in Table 101. The use of this interaction is described in section 7.15.1.1.25.

Table 102 TransferControl Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 References	Default Value (if optional)	Definition
OriginatingEntity	Transfer Control Request	Originating Entity ID	1995: 5.2.29.b, 1998: 5.3.9.3.a	**(Not Optional)**	The DIS Entity ID triplet of the entity or application originating the request.
ReceivingEntity	Transfer Control Request	Receiving Entity ID	1995: 5.2.29.c 1998: 5.3.9.3.a	**(Not Optional)**	The DIS Entity ID triplet of the intended recipient(s) of this request. Group addressing is allowed.

Parameter Name	DIS PDU	DIS Field	IEEE 1278 References	Default Value (if optional)	Definition
RequestIdentifier	Transfer Control Request	Request ID	1998: 5.3.9.3.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.
TransferType	Transfer Control Request	Transfer Type	1998: 5.3.9.3.d	**(Not Optional)**	The type of transfer (for example, entity push or entity pull).
TransferEntity	Transfer Control Request	Transfer Entity ID	1998: 5.3.9.3.e	**(Not Optional)**	The ID of the entity being transferred.
RecordSetData	Transfer Control Request	Record Sets	1998: 5.2.55, 5.3.9.3.f.	**(Not Optional)**	A set of “RecordSets” where each set specifies the type of its Records, its serial number, the number of records in the set. Actual Record lengths and values are stored in the “RecordStruct” structure.

7.15.1.2 HLA Simulation Management Family

For new HLA simulations, the DIS addressing schemes and redundant SIMAN interaction classes might not be appropriate in all cases. This set of interaction classes re-defines the DIS simulation management functions in a scheme that is closer to the underlying HLA architecture than those provided in Section 7.15.1.1. In each case, these interaction classes assume that the intelligence required to create, delete, or change object instances resides at the remote site responsible for modeling that object instance. Graceful methods of refusal or redefinition of the requests are therefore required. These interaction classes support this functionality.

The interaction classes supported within the DIS Simulation Management Family are complementary to those supported within the HLA Simulation Management Family. It is recommended that RPR FOM federates support both capabilities to insure maximal interoperability with RPR FOM management facilities.

Many of these interaction classes have subclasses to support SIMAN with reliability. The requesting interaction classes add an additional parameter specifying the level of reliability required for the response. The responding interaction classes do not add any parameters.

7.15.1.2.1 AttributeChangeRequest Interaction Class

This interaction class, in conjunction with AttributeChangeResult, is used to request a remote federate to set the value of specified instance attributes. Unlike the HLA’s attribute transfer mechanism, AttributeChangeRequest provides a means for the owning federate to evaluate and possibly modify the change request. It is intended to replace the functionality of the DIS SetData PDU with a more generic HLA alternative. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 103 AttributeChangeRequest Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectIdentifiers	SetData	Receiving Entity ID	1995: 5.2.29.c, 5.3.6.9.a	**(Not Optional)**	Recipients as a list of object instance ID's
AttributeValueSet	SetData	Fixed Datums	1995: 5.2.10, 5.3.6.9.c	**(Not Optional)**	The set of attributes and their values that the recipients are asked to update.

7.15.1.2.2 AttributeChangeRequestR Interaction Class

This interaction class, in conjunction with AttributeChangeResultR, provides a reliable mechanism for the manipulation of object instance attributes owned by another federate. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 104 AttributeChangeRequestR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Set Data-R	Required Reliability Service	1998: 5.3.12.9 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.2.3 AttributeChangeResult Interaction Class

This interaction is issued in response to an AttributeChangeRequest to indicate the success, failure, or redefinition of a remote attribute manipulation. Unlike the HLA's attribute transfer mechanism, AttributeChangeResult provides a means for the owning federate to evaluate and possibly modify the change request. It is intended to replace the functionality of the DIS Data PDU with a more generic HLA alternative. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 105 AttributeChangeResult Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectIdentifier	Data	Originating Entity ID	1995: 5.2.29.b, 5.3.6.10.a	**(Not Optional)**	Recipients as a list of object instance ID's
AttributeChangeResult	Data	Fixed Datums	1995: 5.2.10, 5.3.6.10.c	**(Not Optional)**	Indicates ability to comply.
AttributeValueSet	Data	Fixed Datums	1995: 5.2.10, 5.3.6.10.c	**(Not Optional)**	The set of attributes and their values that the recipient has been able to update.

7.15.1.2.4 AttributeChangeResultR Interaction Class

This interaction is a subclass of AttributeChangeResult that is issued in response to an AttributeChangeRequestR. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 106 AttributeChangeResultR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Data-R	Required Reliability Service	1998: 5.3.12.10 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.2.5 CreateObjectRequest Interaction Class

This interaction class, in conjunction with CreateObjectResult, provides a mechanism for the remote initialization of new object instances. Unlike the HLA's attribute transfer mechanism, CreateObjectRequest provides a means for the simulation that registers the new object instance to evaluate and possibly modify the initial conditions. It is intended to replace the functionality of the DIS Create Entity PDU with a more generic HLA alternative. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 107 CreateObjectRequest Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectClass	Create Entity	N/A	none	**(Not Optional)**	Type of object class to register.
AttributeValueSet	Create Entity	N/A	none	**(Not Optional)**	Initial set of attribute/value pairs.
RequestIdentifier	Create Entity	Request ID	1995: 5.3.6.1.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.

7.15.1.2.6 CreateObjectRequestR Interaction Class

This interaction class is a subclass of CreateObjectRequest that, in conjunction with CreateObjectResultR, provides a reliable mechanism for the remote initialization of new object instances. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 108 CreateObjectRequestR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Create Entity-R	Required Reliability Service	1998: 5.3.12.1 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.2.7 CreateObjectResult Interaction Class

This interaction class is issued in response to a CreateObjectRequest to indicate the success, failure, or redefinition of a remote initialization of new object instances. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 109 CreateObjectResult Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
CreateObject Result	Acknowledge	Response Flag	1995: 5.3.6.5.c	**(Not Optional)**	Indicates ability to comply.
RequestIdent ifier	Acknowledge	Request ID	1995: 5.3.6.5.d	**(Not Optional)**	This field matches this response with the specific CreateObject interaction sent by the simulation manager.

7.15.1.2.8 CreateObjectResultR Interaction Class

This interaction class is a subclass of CreateObjectResult that is issued in response to a CreateObjectRequestR interaction class. This interaction class does not add any parameters to the parent interaction class.

7.15.1.2.9 RemoveObjectRequest Interaction Class

This interaction class, in conjunction with RemoveObjectResult, provides a mechanism for the remote deletion of existing object instances. It is intended to replace the functionality of the DIS Remove PDU with a more generic HLA alternative. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 110 RemoveObjectRequest Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectIdentifi ers	N/A	N/A	N/A	**(Not Optional)**	Object instances to delete as a list of object instance ID's
RequestIdent ifier	Remove Entity	Request ID	1995: 5.3.6.2.b	**(Not Optional)**	The Request ID is a monotonically increasing integer identifier inserted by the Simulation Manager into all Simulation management interactions. It is used as a unique identifier to identify the latest in a series of competing requests and identifying acknowledgements.

7.15.1.2.10 RemoveObjectRequestR Interaction Class

This interaction class is a subclass of RemoveObjectRequest that, in conjunction with RemoveObjectResultR, provides a reliable mechanism for the remote deletion of existing object instances. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 111 RemoveObjectRequestR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
Acknowledg ementProtoc ol	Action Request-R	Required Reliability Service	1998: 5.3.12.6	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.2.11 RemoveObjectResult Interaction Class

This interaction class is issued in response to a CreateObjectRequest to indicate the success or failure, of a remote deletion. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 112 RemoveObjectResult Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
RemoveObjectResult	Remove	N/A	none	**(Not Optional)**	Indicates ability to comply.
RequestIdentifier	Remove	Request ID	1995: 5.3.6.2.b	**(Not Optional)**	This field matches this response with the specific RemoveObject interaction sent by the simulation manager.

7.15.1.2.12 RemoveObjectResultR Interaction Class

This interaction class is a subclass of RemoveObjectResult that is issued in response to a CreateObjectRequestR interaction class. This interaction class does not add any parameters to the parent interaction class.

7.15.1.2.13 ActionRequestToObject Interaction Class

This interaction class, in conjunction with ActionResponseFromObject, provides a mechanism for requesting object instances to perform enumerated actions. It is intended to replace the functionality of the DIS ActionRequest PDU with a more generic HLA alternative. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 113 ActionRequestToObject Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
ObjectIdentifiers	N/A	N/A	N/A	**(Not Optional)**	The list of object instances that are the intended recipients of this interaction.
ActionRequestCode	Action Request	Action ID	1995: 5.3.6.6.c	**(Not Optional)**	The action that the recipient(s) are intended to perform.

7.15.1.2.14 ActionRequestToObjectR Interaction Class

This interaction class is a subclass of ActionRequestToObject that, in conjunction with ActionResponseFromObjectR, provides a mechanism for requesting object instances to perform enumerated actions. The AcknowledgementProtocol parameter shall be a required parameter of this interaction class.

Table 114 ActionRequestToObjectR Parameters

Parameter Name	DIS PDU	DIS Field	IEEE 1278 Reference	Default Value (if optional)	Definition
AcknowledgementProtocol	Action Request-R	Required Reliability Service	1998: 5.3.12.6 b	**(Not Optional)**	This field shall identify the level of reliability service to be used for this transaction

7.15.1.2.15 ActionResponseFromObject Interaction Class

This interaction is issued in response to an ActionRequestToObject to indicate the success or failure, of an action request. Senders of this interaction class shall provide values for all parameters; there are no optional fields.

Table 115 ActionResponseFromObject Parameters

Parameter Name	**DIS PDU**	**DIS Field**	**IEEE 1278 Reference**	**Default Value (if optional)**	**Definition**
ActionResult	Action Response	Request Status	1995: 5.3.6.7.c	**(Not Optional)**	The status of the request that the recipient has been asked to perform.

7.15.1.2.16 ActionResponseFromObjectR Interaction Class

This interaction is a subclass of ActionResponseFromObject that is issued in response to an ActionRequestToObjectR interaction class. This interaction class does not add any parameters to the parent interaction class.

8 Procedures

In many cases, the HLA RTI has incorporated features from DIS directly into the data distribution architecture. However, the mapping from DIS to HLA and back again is not always clear to developers. In the spirit of supporting the transition of legacy DIS systems, this section defines procedures that use the HLA application programmer's interface, rather than object or interaction classes, to implement elements of DIS functionality.

8.1 Implementation of Transfer Control

8.1.1 Differences in Transfer Control Mechanisms in DIS and HLA

DIS explicitly allows transfer of entity state simulations and environmental process simulations via an arbitrated process that is initiated with the Transfer Control PDU. DIS also allows the transfer of simulation of emissions, IFF, designator, transmitter, receiver, etc. This is done via a non-arbitrated process that involves the acquiring application start sending, and the relinquishing application stop upon receipt of the PDUs. This obviously does not map in to the HLA realm because only one federate can update an object instance attribute and all ownership transfer is arbitrated. In the following process descriptions, the terms “arbitrated” and “non-arbitrated” refer to the corresponding DIS process behaviors.

DIS allows for three types of arbitrated transfer control for the initiating application:

1. Push: divesting ownership of an entity.
2. Pull: acquiring ownership of an entity.
3. Mutual Exchange: swapping ownership of entities atomically.

HLA supports only two: push and pull. Mutual exchange can be implemented in HLA as a combination of push and pull; however, the operation is not atomic.

In DIS, transfers occur at the Entity level; HLA supports ownership transfer of object instance attributes. RPR FOM provides only for “object instance ownership transfer”, which is defined to be ownership transfer of all of an object instance’s attributes, including “privilegeToDelete” within a single transfer operation.

If the functionality that is provided by Record and Set Record PDUs outlined in DIS 1998 is required by a federate, then the corresponding Record and SetRecord interaction classes shall be used during the analogous points in the transfer control process. As these interaction classes do not directly affect the outcome of the transfer control process, their usage is not expanded upon here.

8.1.2 Arbitrated Transfer Control Process Definitions

This section describes both the Push and Pull arbitrated transfer control processes from the perspective of both the Relinquishing Federate and the Acquiring Federate for BaseEntity and EnvironmentProcess. Note that “Relinquishing” and “Acquiring” refer to the role of the Federate in the process, assuming they desire the transfer.

8.1.2.1 Arbitrated Transfer of Control – Push

This section defines the roles of the relinquishing and acquiring federates for transferring control in the push case. See Figure 19 for a state diagram of Arbitrated Transfer Control - Push. See in addition Figure 20 for an example of Arbitrated Transfer Control – Push initiated between two DIS applications and relayed in the HLA realm through HLA gateways.

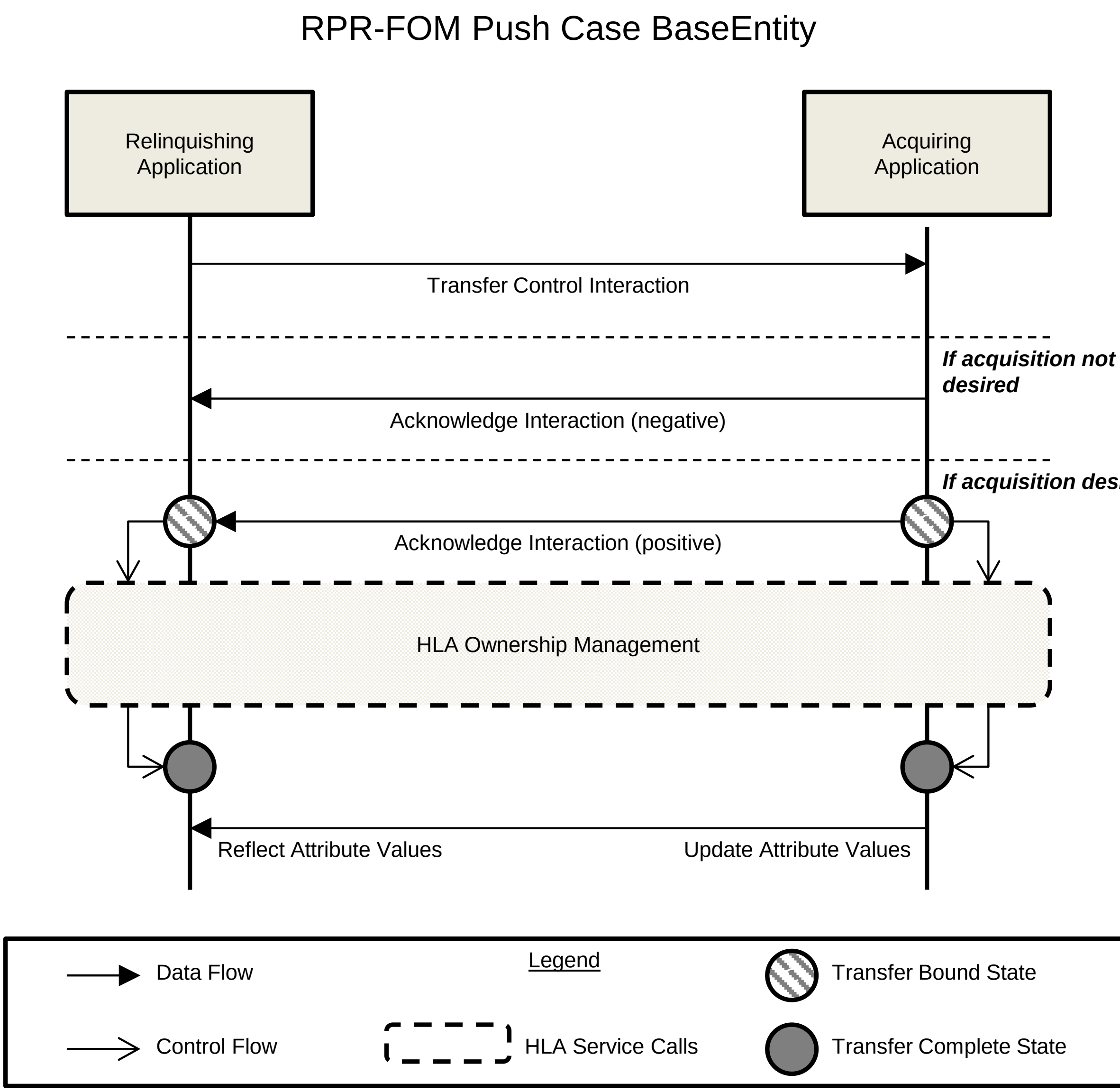

Figure 19 Arbitrated Transfer Control - Push

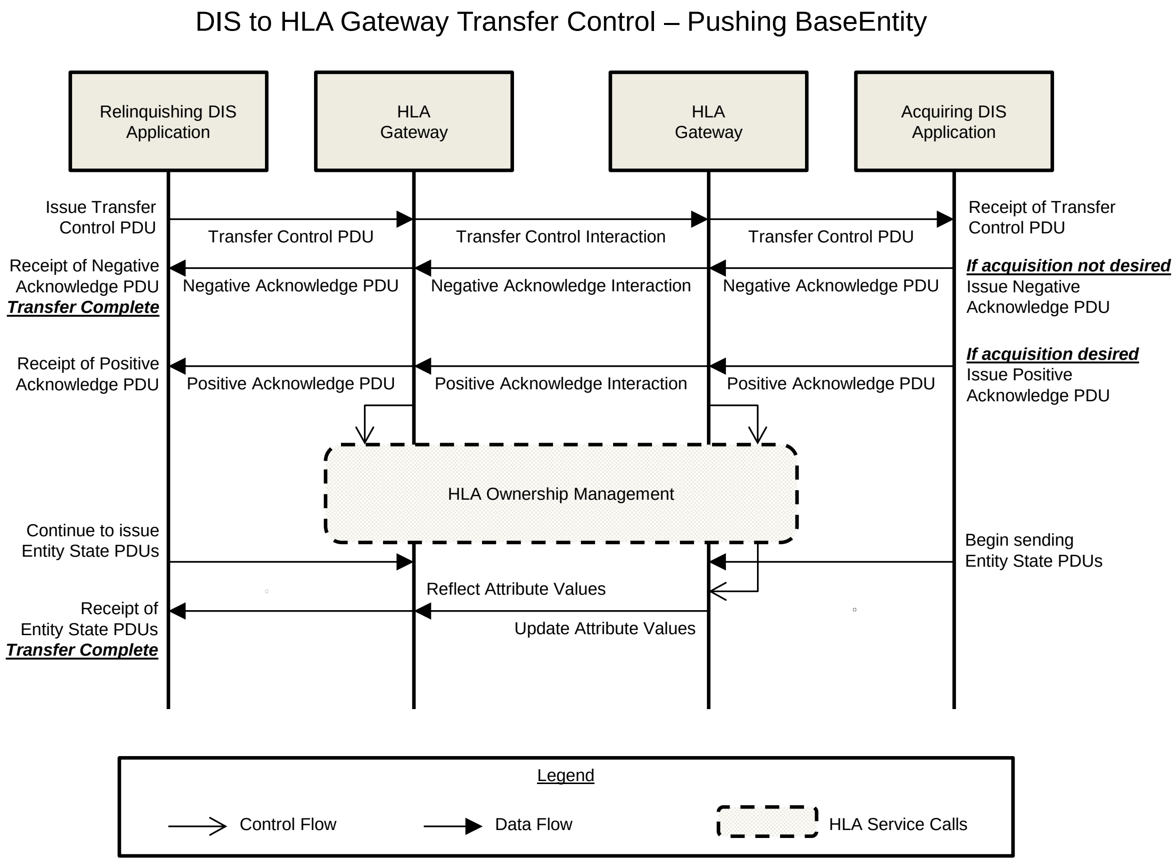

Figure 20 Arbitrated Transfer Control – Push: DIS/HLA Gateway Process

8.1.2.1.1 Push: Relinquishing Federate

To initiate divestiture of the ownership of an object instance's attributes, the Relinquishing Federate shall send a TransferControl Interaction that specifies the object instance to be transferred.

Two things can happen in response to the divestiture request:

1. Acquiring Federate accepts transfer: This is indicated by receipt of a positive Acknowledge Interaction. This binds the transfer to occur. It is important to note that the transfer has not yet occurred, simply that it will at some point in the future.
2. Acquiring Federate denies transfer: This is indicated by receipt of a negative Acknowledge Interaction. This ends the transfer control process.

Once the transfer is bound, the Relinquishing Federate shall use the HLA Ownership Management services to divest ownership of all attributes of the object instance, including "privilegeToDelete". The precise details of the HLA Ownership Management services are not described here, but can be found in the relevant HLA standards.

The Relinquishing Federate shall continue to update the instance attributes normally, until completion of the ownership transfer. When the ownership transfer is complete, the Relinquishing Federate shall cease to update the object instance's attributes, as it no longer owns them. The transfer control process is complete upon receipt of a *Reflect Attribute Values* callback for the object instance.

8.1.2.1.2 Push: Acquiring Federate

The Acquiring Federate is notified of the initiation of a transfer control Push process by receipt of a TransferControl Interaction. All federates that are subscribed to the TransferControl interaction class will receive this callback, so they have to examine the contents to determine if they are the Acquiring Federate. Each federate shall check the receiving entity ID portion of the interaction to see if the divestiture is directed at them. If it is not, they shall not respond. Only the Acquiring Federate shall respond to the divestiture request.

Once the Acquiring Federate has determined its role in the divestiture, it shall respond in one of two ways:

1. Accept: If the Acquiring Federate wishes to take ownership of the entity being offered, it shall respond by sending a positive Acknowledge Interaction. This binds the transfer to occur. It is important to note that the transfer has not yet occurred, simply that it will at some point in the future.
2. Deny: If the Acquiring Federate does not wish to take ownership of the entity being offered, it shall respond by sending a negative Acknowledge Interaction. This ends the transfer control process.

The Acknowledge Interaction shall be constructed using the information found in the TransferControl Interaction received.

Once the transfer is bound, the Acquiring Federate shall use the HLA Ownership Management services to acquire ownership of all attributes of the object instance, including “privilegeToDelete”. The precise details of the HLA Ownership Management services are not described here, but can be found in the relevant HLA standards.

The Acquiring Federate will continue to receive a *Reflect Attribute Values* callbacks for the object instance from the Relinquishing Federate, until completion of the ownership transfer. When the ownership transfer is complete, the Acquiring Federate makes an *Update Attribute Values* call for the object instance.

8.1.2.2 Arbitrated Transfer of Control – Pull

This section defines the roles of the relinquishing and acquiring federates for transferring control in the pull case. See Figure 21 for a state diagram of Arbitrated Transfer Control - Pull. See in addition Figure 22 for an example of Arbitrated Transfer Control - Pull initiated between two DIS applications and relayed in the HLA realm through HLA gateways.

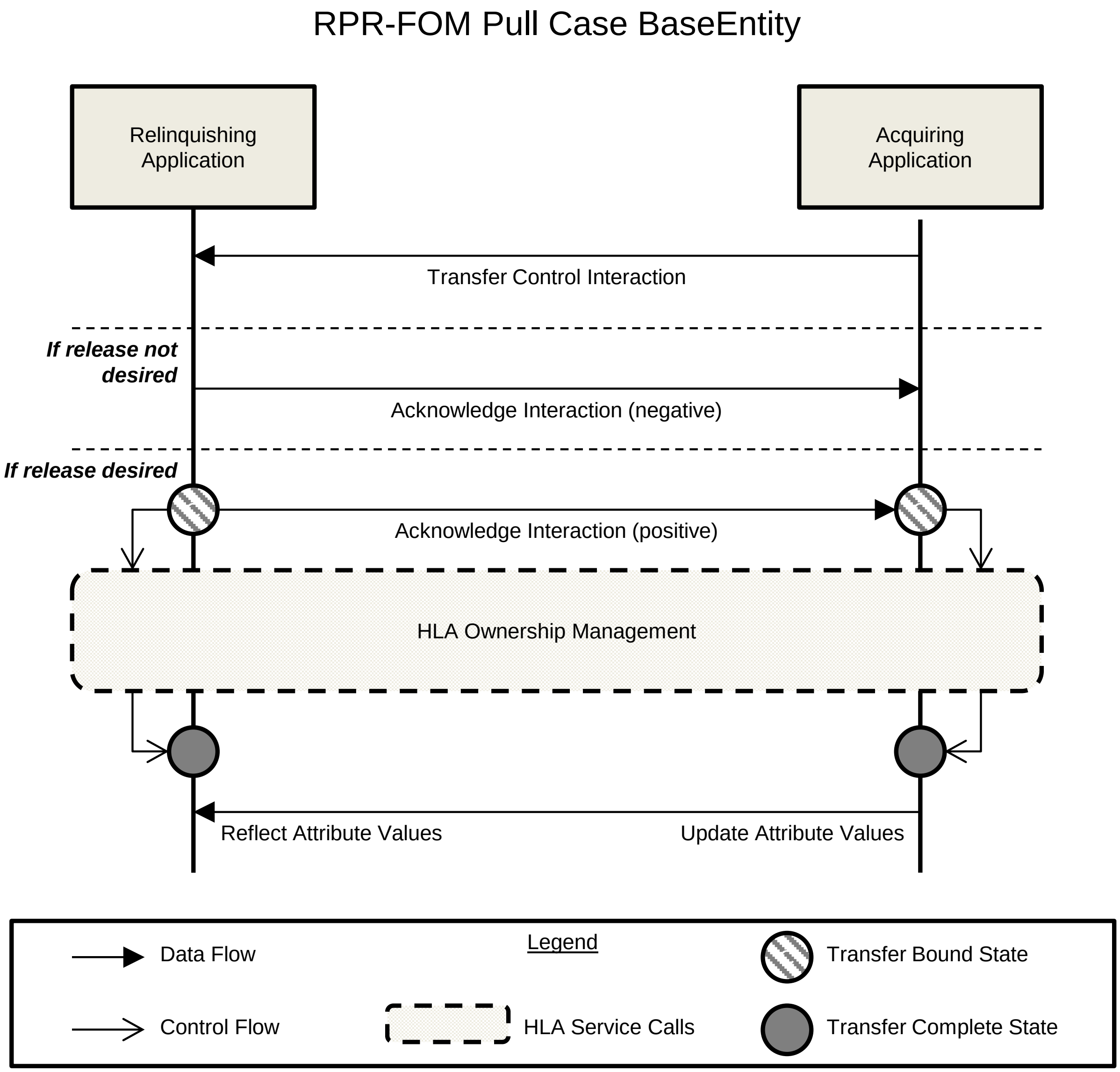

Figure 21 Arbitrated Transfer Control - Pull

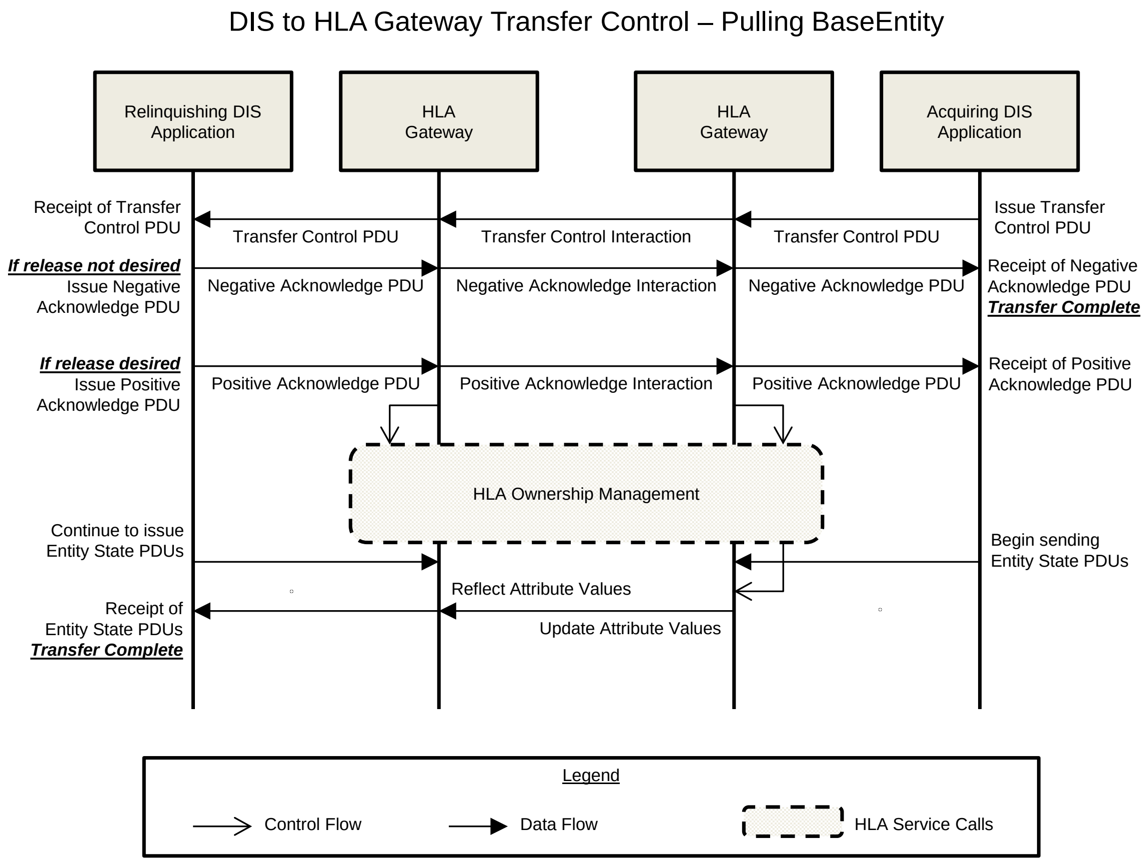

Figure 22 Arbitrated Transfer Control – Pull: DIS/HLA Gateway Process

8.1.2.2.1 Pull: Acquiring Federate

To initiate acquisition of the ownership of an object instance's attributes, the Acquiring Federate shall send a TransferControl Interaction that specifies the object instance to be transferred.

Two things can happen in response to the acquisition request:

1. Relinquishing Federate accepts transfer: This is indicated by receipt of a positive Acknowledge Interaction. This binds the transfer to occur. It is important to note that the transfer has not yet occurred, simply that it will at some point in the future.
2. Relinquishing Federate denies transfer: This is indicated by receipt of a negative Acknowledge Interaction. This ends the transfer control process.

Once the transfer is bound, the Acquiring Federate shall use the HLA Ownership Management services to acquire ownership of all attributes of the object instance, including "privilegeToDelete". The precise details of the HLA Ownership Management services are not described here, but can be found in the relevant HLA standards.

The Acquiring Federate will continue to receive a *Reflect Attribute Values* callbacks for the object instance from the Relinquishing Federate, until completion of the ownership transfer. When the ownership transfer is complete, the Acquiring Federate makes an *Update Attribute Values* call for the object instance.

8.1.2.2.2 Pull: Relinquishing Federate

The Relinquishing Federate is notified of the initiation of a transfer control Pull process by receipt of a TransferControl Interaction that specifies an entity it owns as the one to be transferred. All federates that are subscribed to the TransferControl interaction class could receive this interaction, so they have to examine the entity to be transferred to determine if they are the owner of it and thus the Relinquishing Federate. Each federate shall check the TransferEntity parameter of the TransferControl Interaction to see if the transfer is directed at them. If it is not, they shall not respond. Only the Relinquishing Federate shall respond to the transfer request.

Once the Relinquishing Federate has determined its role in the transfer, it shall respond in one of two ways.

1. <u>Accept:</u> If the Relinquishing Federate wishes to give up ownership of the entity, it shall respond by sending a positive Acknowledge Interaction. This response binds the transfer to occur. It is important to note that the transfer has not yet occurred, simply that it will at some point in the future.
2. <u>Deny:</u> If the Relinquishing Federate does not wish to give up ownership of the entity, it shall respond by sending a negative Acknowledge Interaction. This ends the transfer control process.

The Acknowledge Interaction shall be constructed using the information found in the TransferControl Interaction received.

Once the transfer is bound, the Relinquishing Federate shall use the HLA Ownership Management services to divest ownership of all attributes of the object instance, including “privilegeToDelete”. The precise details of the HLA Ownership Management services are not described here, but can be found in the relevant HLA standards.

The Relinquishing Federate shall continue to update the instance attributes normally, until completion of the ownership transfer. When the ownership transfer is complete, the Relinquishing Federate shall cease to update the object instance’s attributes, as it no longer owns them. The transfer control process is complete upon receipt of a *Reflect Attribute Values* callback for the object instance.

8.1.3 Transfer Control Definitions for All Other RPR FOM Object classes

This section describes both the Push and Pull non-arbitrated transfer control processes from the perspective of both the Relinquishing Federate and the Acquiring Federate for all other RPR FOM object classes. Note that “Relinquishing” and “Acquiring” refer to the role of the Federate in the process assuming they desire the transfer.

8.1.3.1 General Rules for Non-Arbitrated Transfer Control

The processes outlined in this section apply to all non-BaseEntity and non-EnvironmentProcess RPR-FOM object classes (referred to as non-arbitrated). Any additional specific rules are defined in the next section. See Figure 23 for a state diagram of Non-arbitrated Transfer Control.

In DIS, when non-arbitrated PDUs are transferred, the acquiring application just starts issuing PDUs. It is up to the DIS applications themselves to determine who remains simulating them. In fact, nothing disallows two simulations from sending PDUs representing the same thing simultaneously.

In HLA, when a simulation desires to make updates to an object instance it does not own, it shall obtain ownership by pulling. The next two sections describe this pull process in detail from both the acquiring and divesting federate’s perspectives. There is no corresponding push process. All transfer of control of non-arbitrated objects shall be performed via a pull.

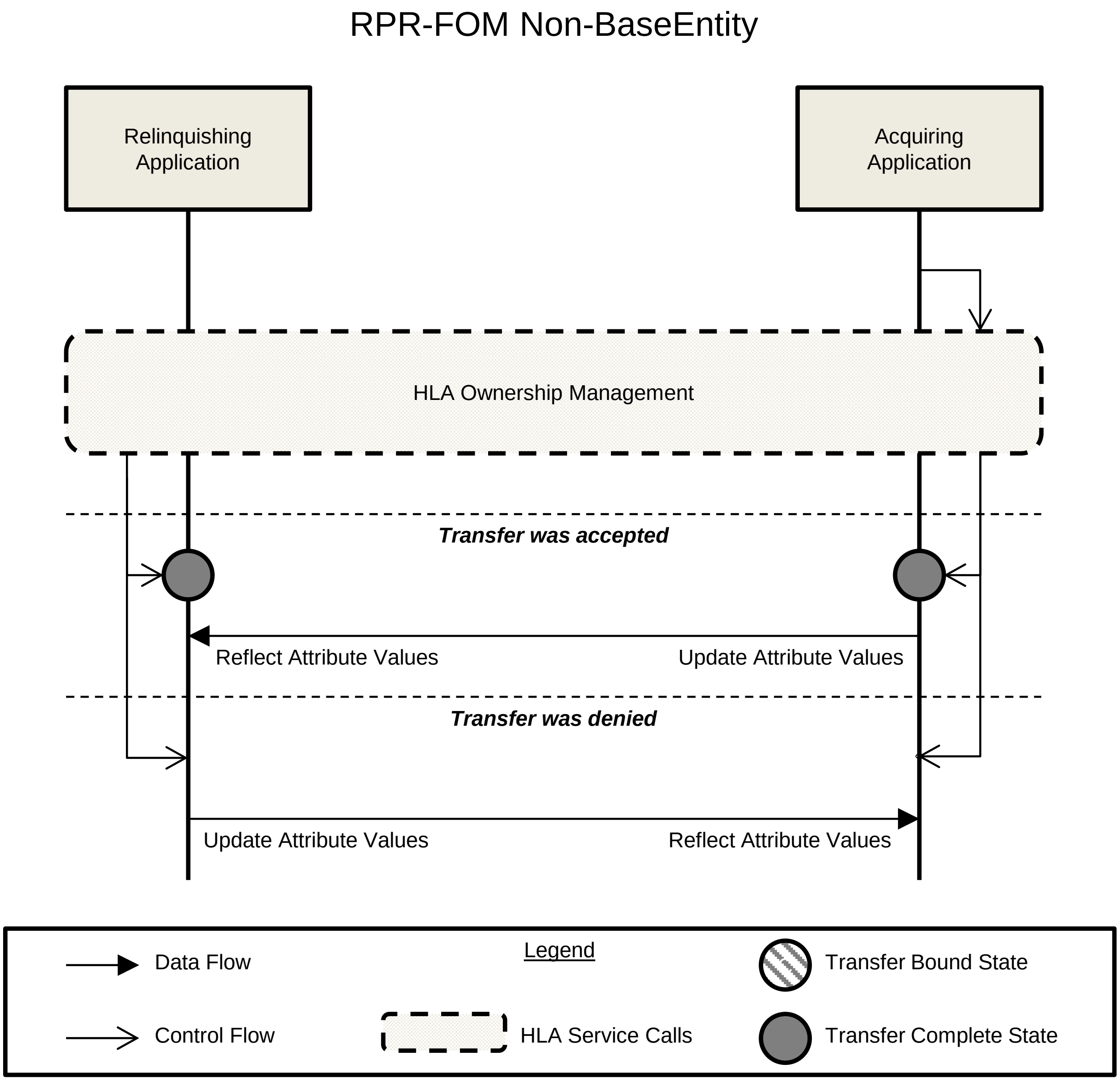

Figure 23 Non-arbitrated Transfer Control

8.1.3.1.1 Pull: Acquiring Federate

To initiate acquisition of the ownership of an instance of a non-arbitrated object class, the Acquiring Federate shall use the HLA Ownership Management services to acquire ownership of all attributes of the object instance, including "privilegeToDelete". The precise details of the HLA Ownership Management services are not described here, but can be found in the relevant HLA standards.

Two things can happen:

1. Relinquishing Federate accepts transfer: All attributes of the desired object instance, including "privilegeToDelete", are transferred to the Acquiring Federate.
2. Relinquishing Federate denies transfer: None of the attributes of the desired object instance, including "privilegeToDelete", are transferred to the Acquiring Federate.

8.1.3.1.2 Pull: Relinquishing Federate

The Relinquishing Federate is notified of the initiation of a transfer control process for a non-arbitrated object instance via the HLA Ownership Management services. The precise details of the HLA Ownership Management services are not described here, but can be found in the relevant HLA standards. The Relinquishing Federate shall either accept the transfer or deny the transfer.

8.1.3.2 Specific Rules for Non-Arbitrated Transfer Control

Where multiple instances of an object are involved in a transfer of control, they shall be treated as a group, i.e., the Acquiring federate shall not transfer a particular object instance without transferring all associated object instances. For example, in the case of emitter systems and beams that use multiple object instances in HLA, the Acquiring federate shall repeat the process described in section 8.1.3.1 for all object instances owned by the Relinquishing federate. The Relinquishing federate shall provide the same response for all object instances requested.

9 Mapping from DIS Fields back to the RPR FOM

9.1 Entity Information / Interaction Family

9.1.1 Entity State PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Entity ID	BaseEntity	EntityIdentifier
Force ID	PhysicalEntity	ForceIdentifier
Number of Articulation Parameters	PhysicalEntity	size of ArticulatedParametersArray
Entity Type	BaseEntity	EntityType
Alternative Entity Type	PhysicalEntity	AlternativeEntityType
Entity Linear Velocity	BaseEntity	Spatial
Entity Location	BaseEntity	Spatial
Entity Orientation	BaseEntity	Spatial
Entity Appearance	BaseEntity	Spatial
	EnvironmentalEntity	OpacityCode
	PhysicalEntity	DamageState, EngineSmokeOn, FlamesPresent, HatchState, Immobilized, PersonStanceCode, PowerPlantOn, RampDeployed, SmokePlumePresent, TentDeployed, TrailingEffectsCode, CamouflageType, FirePowerDisabled, IsConcealed
	Platform	AfterburnerOn, AntiCollisionLightsOn , BlackOutBrakeLightsOn, BlackOutLightsOn, BrakeLightsOn, FormationLightsOn, HatchState, HeadLightsOn, InteriorLightsOn, LandingLightsOn, LauncherRaised, NavigationLightsOn, RampDeployed, RunningLightsOn, SpotLightsOn, TailLightsOn
	Sensor	AntennaRaised, BlackoutLightsOn, InteriorLightsOn, LightsOn, MissionKill
	Munition	LauncherFlashPresent
	CulturalFeature	ExternalLightsOn, InternalHeatSourceOn, InternalLightsOn
Dead Reckoning Parameters: Dead Reckoning Algorithm	BaseEntity	Spatial
Dead Reckoning Parameters: Entity Linear Acceleration	BaseEntity	Spatial
Dead Reckoning Parameters: Entity Angular Velocity	BaseEntity	Spatial

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
EntityMarking	PhysicalEntity	Marking
Capabilities	PhysicalEntity	HasAmmunitionSupplyCap, HasFuelSupplyCap, HasRecoveryCap, HasRepairCap
	Lifeform	PrimaryWeaponState, SecondaryWeaponState
Articulation Parameters	PhysicalEntity	ArticulatedParametersArray

9.1.2 Entity State Update PDU

In DIS, the Entity State Update PDU provided the ability to send updates for those components of an entity's state that are likely to change frequently without having to also send more "static" data like entity types and markings in every update. The ability to selectively send updates only for attributes whose values are out of date is inherent to HLA. No additional classes or attributes were needed in the RPR FOM to implement this DIS capability.

9.1.3 Collision PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Issuing Entity ID	CollisionInteraction	IssuingObjectIdentifier
Colliding Entity ID	CollisionInteraction	CollidingObjectIdentifier
Event ID	CollisionInteraction	EventIdentifier
Collision Type	CollisionInteraction	CollisionType
Velocity	CollisionInteraction	IssuingObjectVelocityVector
Mass	CollisionInteraction	IssuingObjectMass
Location	CollisionInteraction	CollisionLocation

9.1.4 Collision-Elastic PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N.A.	
Issuing Entity ID	Collision	IssuingObjectIdentifier
Colliding Entity ID	Collision	CollidingObjectIdentifier
Collision Event ID	Collision	EventIdentifier
Contact Velocity	Collision	IssuingObjectVelocityVector
Mass	Collision	IssuingObjectMass
Location Of Impact	Collision	CollisionLocation
Collision Intermediate Result-XX	Collision-Elastic	IntermediateResultXX
Collision Intermediate Result-XY	Collision-Elastic	IntermediateResultXY
Collision Intermediate Result-XZ	Collision-Elastic	IntermediateResultXZ
Collision Intermediate Result-YY	Collision-Elastic	IntermediateResultYY
Collision Intermediate Result-YZ	Collision-Elastic	IntermediateResultYZ
Collision Intermediate Result-ZZ	Collision-Elastic	IntermediateResultZZ
Unit Surface Normal	Collision-Elastic	UnitSurfaceNormal
Coefficient Of Restitution	Collision-Elastic	CoefficientOfRestitution

9.2 Warfare Family

9.2.1 Fire PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Firing Entity ID	WeaponFire	FiringObjectIdentifier
Target Entity ID	WeaponFire	TargetObjectIdentifier
Munition ID	WeaponFire	MunitionObjectIdentifier
Event ID	WeaponFire	EventIdentifier
Fire Mission Index	WeaponFire	FireMissionIndex
Location In World Coordinates	WeaponFire	FiringLocation
Burst Descriptor: Munition	WeaponFire	MunitionType
Burst Descriptor: Warhead	WeaponFire	WarheadType
Burst Descriptor: Fuse	WeaponFire	FuseType
Burst Descriptor: Quantity	WeaponFire	QuantityFired
Burst Descriptor: Rate	WeaponFire	RateOfFire
Velocity	WeaponFire	InitialVelocityVector
Range	WeaponFire	FireControlSolutionRange

9.2.2 Detonation PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Firing Entity ID	MunitionDetonation	FiringObjectIdentifier
Target Entity ID	MunitionDetonation	TargetObjectIdentifier
Munition ID	MunitionDetonation	MunitionObjectIdentifier
Event ID	MunitionDetonation	EventIdentifier
Velocity	MunitionDetonation	FinalVelocityVector
Location in World Coordinates	MunitionDetonation	DetonationLocation
Burst Descriptor: Munition	MunitionDetonation	MunitionType
Burst Descriptor: Warhead	MunitionDetonation	WarheadType
Burst Descriptor: Fuse	MunitionDetonation	FuseType
Burst Descriptor: Quantity	MunitionDetonation	QuantityFired
Burst Descriptor: Rate	MunitionDetonation	RateOfFire
Location in Entity Coordinates	MunitionDetonation	RelativeDetonationLocation
Detonation Result	MunitionDetonation	DetonationResultCode
Number of Articulation Parameters	MunitionDetonation	size of ArticulatedParametersData
Articulation Parameters	MunitionDetonation	ArticulatedParametersData

9.3 Logistics Family

9.3.1 Service Request PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Requesting Entity ID	ServiceRequest	RequestingObject
Servicing Entity ID	ServiceRequest	ServicingObject
Service Type Requested	ServiceRequest	ServiceType
Number of Supply Types	ServiceRequest	size of SuppliesData
Supplies	ServiceRequest	SuppliesData

9.3.2 Resupply Offer PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Receiving Entity ID	ResupplyOffer	ReceivingObject
Supplying Entity ID	ResupplyOffer	SupplyingObject

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Number of Supply Types	ResupplyOffer	size of SuppliesData
Supplies	ResupplyOffer	SuppliesData

9.3.3 Resupply Received PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Receiving Entity ID	ResupplyReceived	ReceivingObject
Supplying Entity ID	ResupplyReceived	SupplyingObject
Number of Supply Types	ResupplyReceived	size of SuppliesData
Supplies	ResupplyReceived	SuppliesData

9.3.4 Resupply Cancel PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Receiving Entity ID	ResupplyCancel	ReceivingObject
Supplying Entity ID	ResupplyCancel	SupplyingObject

9.3.5 Repair Complete PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Receiving Entity ID	RepairComplete	ReceivingObject
Repairing Entity ID	RepairComplete	RepairingObject
Repair	RepairComplete	RepairType

9.3.6 Repair Response PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Receiving Entity ID	RepairResponse	ReceivingObject
Repairing Entity ID	RepairResponse	RepairingObject
Repair Result	RepairResponse	RepairResultCode

9.4 Simulation Management Family

9.4.1 Create Entity PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	CreateEntity	OriginatingEntity
Receiving Entity ID	CreateEntity	ReceivingEntity
Request ID	CreateEntity	RequestIdentifier
	CreateObjectRequest	RequestIdentifier

9.4.2 Remove Entity PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	RemoveEntity	OriginatingEntity
Receiving Entity ID	RemoveEntity	ReceivingEntity
	RemoveObjectRequest	ObjectIdentifiers

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Request ID	RemoveEntity	RequestIdentifier
	RemoveObjectRequest	RequestIdentifier

9.4.3 Start/Resume PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	StartResume	OriginatingEntity
Receiving Entity ID	StartResume	ReceivingEntity
Real-World Time	StartResume	RealWorldTime
Simulation Time	StartResume	SimulationTime
Request ID	StartResume	RequestIdentifier

9.4.4 Stop/Freeze PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	StopFreeze	OriginatingEntity
Receiving Entity ID	StopFreeze	ReceivingEntity
Real-World Time	StopFreeze	RealWorldTime
Reason	StopFreeze	Reason
Frozen Behavior	StopFreeze	RunInternalSimulationClock, UpdateAttributes, ReflectValues
Request ID	StopFreeze	RequestIdentifier

9.4.5 Acknowledge PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	Acknowledge	OriginatingEntity
Receiving Entity ID	Acknowledge	ReceivingEntity
Acknowledge Flag	Acknowledge	AcknowledgeFlag
Response Flag	Acknowledge	ResponseFlag
	CreateObjectRequest	CreateObjectResult
	RemoveObjectRequest	RemoveObjectResult
Request ID	Acknowledge	RequestIdentifier
	CreateObjectRequest	RequestIdentifier
	RemoveObjectRequest	RequestIdentifier

9.4.6 Action Request PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	ActionRequest	OriginatingEntity
Receiving Entity ID	ActionRequest	ReceivingEntity
	ActionRequestToObject	ObjectIdentifiers
Request ID	ActionRequest	RequestIdentifier
Action ID	ActionRequest	ActionRequestCode
	ActionRequestToObject	ActionRequestCode
Number Fixed Datums	ActionRequest	size of FixedDatums
Number Variable Datums	ActionRequest	size of VariableDatumSet
Fixed Datum	ActionRequest	FixedDatums
Variable Datums	ActionRequest	VariableDatumSet

9.4.7 Action Response PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	ActionResponse	OriginatingEntity
Receiving Entity ID	ActionResponse	ReceivingEntity
Request ID	ActionResponse	RequestIdentifier
Request Status	ActionResponse	RequestStatus
	ActionResponseFromObject	ActionResult
Number Fixed Datums	ActionResponse	size of FixedDatums
Number Variable Datums	ActionResponse	size of VariableDatumSet
Fixed Datums	ActionResponse	FixedDatums
Number Variable Datums	ActionResponse	VariableDatumSet

9.4.8 Data Query PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	DataQuery	OriginatingEntity
Receiving Entity ID	DataQuery	ReceivingEntity
Request ID	DataQuery	RequestIdentifier
Time Interval	DataQuery	TimeInterval
Number Fixed Datums	DataQuery	size of FixedDatums
Number Variable Datums	DataQuery	size of VariableDatumSet
Fixed Datum	DataQuery	FixedDatums
Variable Datum	DataQuery	VariableDatumSet

9.4.9 Set Data PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	SetData	OriginatingEntity
Receiving Entity ID	SetData	ReceivingEntity
	AttributeChangeRequest	ObjectIdentifiers
Request ID	SetData	RequestIdentifier
Number Fixed Datums	SetData	size of FixedDatums
	AttributeChangeRequest	size of AttributeValueSet
Number Variable Datums	SetData	size of VariableDatumSet
Fixed Datum	SetData	FixedDatums
	AttributeChangeRequest	AttributeValueSet
	CreateObjectRequest	AttributeValueSet
Variable Datums	SetData	VariableDatumSet

9.4.10 Data PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	Data	OriginatingEntity
Receiving Entity ID	Data	ReceivingEntity
Request ID	Data	RequestIdentifier
Number Fixed Datums	Data	size of FixedDatums
	AttributeChangeResult	AttributeValueSet
Number Variable Datums	Data	size of VariableDatumSet
FixedDatum	Data	FixedDatums

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
	AttributeChangeResult	AttributeValueSet, AttributeChangeResult
Variable Datums	Data	VariableDatumSet

9.4.11 Event Report PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	EventReport	OriginatingEntity
Receiving Entity ID	EventReport	ReceivingEntity
Event Type	EventReport	EventType
Number Fixed Datums	EventReport	size of FixedDatums
Number Variable Datums	EventReport	size of VariableDatumSet
Fixed Datum	EventReport	FixedDatums
Variable Datums	EventReport	VariableDatumSet

9.4.12 Comment PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	Comment	OriginatingEntity
Receiving Entity ID	Comment	ReceivingEntity
Number Fixed Datums	Comment	size of FixedDatums
Number Variable Datums	Comment	size of VariableDatumSet
Variable Datum	Comment	VariableDatumSet

9.5 Distributed Emission Regeneration Family

9.5.1 Electromagnetic Emissions PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Emitting Entity ID	EmbeddedSystem	HostObjectIdentifier
Event ID	EmbeddedSystem	EventIdentifier
	EmitterBeam	EventIdentifier
State Update Indicator	EmbeddedSystem	computed data
Number of Systems	EmbeddedSystem	computed data
System Data Length	EmbeddedSystem	computed data
Number of Beams	EmitterBeam	computed data
Emitter System: Emitter name	EmitterSystem	EmitterType
Emitter System: Function	EmitterSystem	EmitterFunctionCode
Emitter System: Emitter ID number	EmitterSystem	EmitterIndex
Location (with respect to entity)	EmbeddedSystem	RelativeLocation
Beam Data Length	EmitterBeam	computed data
Beam ID Number	EmitterBeam	BeamIdentifier
Beam Parameter Index	EmitterBeam	BeamParameterIndex
Fundamental Parameter Data: Frequency	EmitterBeam	EmissionFrequency
Fundamental Parameter Data: Frequency Range	EmitterBeam	FrequencyRange

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Fundamental Parameter Data: Effective Radiated Power	EmitterBeam	EffectiveRadiatedPower
Fundamental Parameter Data: Pulse Repetition Frequency	EmitterBeam	PulseRepetitionFrequency
Fundamental Parameter Data: Pulse Width	EmitterBeam	PulseWidth
Fundamental Parameter Data: Beam Azimuth Center	EmitterBeam	BeamAzimuthCenter
Fundamental Parameter Data: Beam Azimuth Sweep	EmitterBeam	BeamAzimuthSweep
Fundamental Parameter Data: Beam Elevation Center	EmitterBeam	BeamElevationCenter
Fundamental Parameter Data: Beam Elevation Sweep	EmitterBeam	BeamElevationSweep
Fundamental Parameter Data: Beam Sweep Sync	EmitterBeam	SweepSynch
Beam Function	EmitterBeam	BeamFunctionCode
Number of Targets in the Track/Jam Field	RadarBeam	size of TrackObjectIdentifiers
	JammerBeam	size of JammedObjectIdentifiers
High Density Track/Jam	RadarBeam	HighDensityTrack
	JammerBeam	HighDensityJam
Jamming Mode Sequence	JammerBeam	JammingModeSequence
Track/Jam: Site/Applic/Entity	RadarBeam	TrackObjectIdentifiers
	JammerBeam	JammmedObjectIdentifiers
Track/Jam: Emitter ID	EmitterBeam	EmitterIndex (of emitter targeted by jammer)
Track/Jam: Beam ID	JammerBeam	BeamIdentifier (of beam targeted by jammer)

9.5.2 Designator PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Designating Entity ID	EmbeddedSystem	HostObjectIdentifier
Code Name	Designator	CodeName
Designated Entity ID	Designator	DesignatedObjectIdentifier
Designator Code	Designator	Designator Code
Designator Power	Designator	DesignatorOutputPower
Designator Wavelength	Designator	DesignatorEmissionWavelength
Designator Spot with Respect to Designated Entity	Designator	RelativeSpotLocation
Designator Spot Location	Designator	DesignatorSpotLocation
Dead Reckoning Algorithm	Designator	DeadReckoningAlgorithm
Entity Linear Acceleration	Designator	SpotLinearAccelerationVector

9.5.3 Underwater Acoustic PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Emitting Entity ID	EmbeddedSystem	HostObjectIdentifier
		EntityIdentifier
	AcousticTransient	HostObjectIdentifier

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Event ID	UnderwaterAcousticsEmission	EventIdentifier
	ActiveSonarBeam	EventIdentifier
State/Change Update Indicator	N/A	
Passive Parameter Index	PropulsionNoise	PassiveParameterIndex
Propulsion Plant Configuration	PropulsionNoise	PropulsionPlantConfiguration
Number of Shafts (s)	PropulsionNoise	length of ShaftRateData
Shaft RPM	PropulsionNoise	ShaftRateData.CurrentShaftRate
		ShaftRateData.OrderedShaftRate
Shaft RPM Rate of Change	PropulsionNoise	ShaftRateData.ShaftRateOfChange
Number of Additional Passive Activities (APA)	N/A	N/A
Additional Passive Activity	AdditionalPassiveActivities	ActivityCode
		IsSilent
	AcousticTransient	ActivityCode
APA Value	AdditionalPassiveActivities	ActivityParameter
	AcousticTransient	ActivityParameter
Number of Emitter Systems	N/A	N/A
Emitter System Data Length	N/A	N/A
Number of Beams	N/A	N/A
Acoustic System		
Location	EmbeddedSystem	RelativePosition
Beam Data Length	N/A	N/A
Beam ID Number	ActiveSonarBeam	BeamIdentifier
Fundamental Data Parameters	ActiveSonarBeam	ActiveEmissionParameterIndex
		AzimuthBeamwidth
		AzimuthCenter
		ElevationBeamwidth
		ElevationCenter
		ScanPattern

9.5.4 IFF/ATC/NAVAIDS PDU

Layer 1:

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Emitting Entity ID	EmbeddedSystem	HostObjectIdentifier
Event ID	EmbeddedSystem	EventIdentifier
	EmitterBeam	EventIdentifier
Location (with respect to entity)	EmbeddedSystem	RelativeLocation
System: Type	EmitterSystem IFF	EmitterFunctionCode SystemType
System: Name	EmitterSystem IFF	EmitterIndex SystemName
System: Mode	IFF	SystemMode
System: Change/Options	N/A	
System Status	IFF	SystemIsOperational ApplicableModes
Alternate Parameter 4	NatoIFF	AlternativeMode4
Information Layers	IFF	Layer2DataAvailable
Modifier	N/A	

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Parameter 1	NatoIFF NatoIFFTransponder	Mode1Enabled Mode1IsDamaged Mode1IsMalfunctioning Mode1IsOn Mode1Code
Parameter 2	NatoIFF NatoIFFTransponder	Mode2Enabled Mode2IsDamaged Mode2IsMalfunctioning Mode2IsOn Mode2Code
Parameter 3	NatoIFF NatoIFFTransponder	Mode3Enabled Mode3IsDamaged Mode3IsMalfunctioning Mode3IsOn Mode3Code
Parameter 4	NatoIFF	Mode4Enabled Mode4IsDamaged Mode4IsMalfunctioning Mode4IsOn
Parameter 5	NatoIFF NatoIFFTransponder	Mode5CEnabled Mode5CIsDamaged Mode5CIsMalfunctioning Mode5CIsOn Mode5CAltitude
Parameter 6	NatoIFF	ModeSEnabled ModeSIsDamaged ModeSIsMalfunctioning ModeSIsOn

Layer 2:

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Layer Header	N/A	
Beam Data: Azimuth Center Azimuth Sweep Elevation Center Elevation Sweep Sweep Sync	IFF	 BeamAzimuthCenter BeamAzimuthSweep BeamElevationCenter BeamElevationSweep SweepSync
Secondary Operation Data Parameter 1 Parameter 2 Number of Fund. Data Sets	 IFF	 SecondaryOperationalDataParameter1 SecondaryOperationalDataParameter2
Fundamental Parameter Set ERP Frequency PgRF Pulse Width Burst Length Applicable Modes	 IFF	 FundamentalParameterData ERP, Frequency, PgRF, PulseWidth, BurstLength, ApplicableModes

9.5.5 Supplemental Emission/Entity State PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Originating ID		
Infrared Signature Representation Index	PhysicalEntity	InfraredSignatureIndex
Acoustic Signature Representation Index	PhysicalEntity	AcousticSignatureIndex
Radar Cross-Section Signature Rep. Index	PhysicalEntity	RadarCrossSectionSignatureIndex
Number of Propulsion Systems	N/A	
Number of Vectoring Nozzle Systems	N/A	
Propulsion System Data Power Setting Engine RPM	PhysicalEntity	 PowerSetting EngineRPM
Vectoring Nozzle System Data Horizontal Deflection Angle Vertical Deflection Angle	PhysicalEntity	 HorizontalDeflectionAngle VerticalDeflectionAngle

9.6 Radio Communications

9.6.1 Transmitter PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Entity Identifier	EmbeddedSystem	EntityIdentifier
Entity Identifier	RadioTransmitter	StreamTag (Bits 16-63)
Radio ID	RadioTransmitter	RadioIndex
Radio ID	RadioTransmitter	StreamTag (Bits 0-15)
Radio Entity Type	RadioTransmitter	RadioSystemType
Transmit State	RadioTransmitter	TransmitterOperationalStatus
Input Source	RadioTransmitter	RadioInputSource
Antenna Location	RadioTransmitter	WorldLocation
Relative Antenna Location	EmbeddedSystem	RelativeLocation
Antenna Pattern Type	RadioTransmitter	AntennaPatternData
Antenna Pattern Length	RadioTransmitter	size of AntennaPatternData
Frequency	RadioTransmitter	Frequency
Transmit Frequency Bandwidth	RadioTransmitter	FrequencyBandwidth
Power	RadioTransmitter	TransmittedPower
Modulation Type: Spread spectrum	RadioTransmitter	SpreadSpectrum
Modulation Type: Major	RadioTransmitter	RFModulationType
Modulation Type: Detail	RadioTransmitter	RFModulationType
Modulation Type: System	RadioTransmitter	RFModulationSystemType
CryptoSystem	RadioTransmitter	CryptoSystem
CryptoKeyID	RadioTransmitter	CryptographicMode, EncryptionKeyIdentifier
Length of Modulation Parameters	RadioTransmitter	size of SINCGARSModulation field of SpreadSpectrum attribute
Modulation Parameter #1 ... #N	RadioTransmitter	SpreadSpectrum (SINCGARSModulation field)

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Antenna Pattern Parameter #1 ... #N	RadioTransmitter	AntennaPatternData

9.6.2 Signal PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Entity ID	EncodedAudioRadioSignal	AudioData (StreamTag field)
	RawBinaryRadioSignal	HostRadioIndex
	DatabaseIndexRadioSignal	HostRadioIndex
	ApplicationSpecificRadioSignal	HostRadioIndex
Radio ID	EncodedAudioRadioSignal	AudioData (StreamTag field)
	RawBinaryRadioSignal	HostRadioIndex
	DatabaseIndexRadioSignal	HostRadioIndex
	ApplicationSpecificRadioSignal	HostRadioIndex
Encoding Scheme	EncodedAudioRadioSignal	Bits 0-13: AudioData (EncodingType field) Bits 14-15: N/A
	RawBinaryRadioSignal	TDLMessageCount
	DatabaseIndexRadioSignal	TDLMessageCount
	ApplicationSpecificRadioSignal	TDLMessageCount
TDL Type	RawBinaryRadioSignal	TacticalDataLinkType
	DatabaseIndexRadioSignal	TacticalDataLinkType
	ApplicationSpecificRadioSignal	TacticalDataLinkType
Sample Rate	EncodedAudioRadioSignal	AudioData (SampleRate field)
	RawBinaryRadioSignal	DataRate
	ApplicationSpecificRadioSignal	DataRate
Data Length	EncodedAudioRadioSignal	AudioData (DataLength field)
	RawBinaryRadioSignal	SignalDataLength
	ApplicationSpecificRadioSignal	SignalDataLength
Samples	EncodedAudioRadioSignal	AudioData (SampleCount field)
Data	EncodedAudioRadioSignal	AudioData (Data field)
	RawBinaryRadioSignal	SignalData
	DatabaseIndexRadioSignal	DatabaseIndex, Duration, StartOffset
	ApplicationSpecificRadioSignal	SignalData, UserProtocolID

9.6.3 Receiver PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Entity ID	EmbeddedSystem	HostObjectIdentifier
Radio ID	RadioReceiver	RadioIndex
Receiver State	RadioReceiver	ReceiverOperationalStatus
Received Power	RadioReceiver	ReceivedPower
Transmitter Entity ID	RadioReceiver	ReceivedTransmitterIdentifier
Transmitter Radio ID	RadioReceiver	ReceivedTransmitterIdentifier

9.6.4 Intercom Signal PDU

The Intercom Protocol is not supported by the RPR FOM.

9.6.5 Intercom Control PDU

The Intercom Protocol is not supported by the RPR FOM.

9.7 Entity Management

9.7.1 Aggregate State PDU

DIS Fields	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Aggregate ID	BaseEntity	EntityIdentifier
Force ID	AggregateEntity	ForceIdentifier
Aggregate State	AggregateEntity	AggregateState
Aggregate Type	BaseEntity	EntityType
Formation	AggregateEntity	Formation
Aggregate Marking	AggregateEntity	AggregateMarking
Dimensions	AggregateEntity	Dimensions
Orientation	BaseEntity	Orientation
Center of Mass	BaseEntity	WorldLocation
Velocity	BaseEntity	VelocityVector
Number of DIS Aggregates		
Number of DIS Entities		
Number of Silent Aggregate Types		
Number of Silent Entity Types	AggregateEntity	NumberOfSilentEntities
Aggregate ID List	AggregateEntity	SubAggregateIdentifiers
Entity ID List	AggregateEntity	EntityIdentifiers
Silent Aggregate System List	AggregateEntity	SilentAggregates
Silent Entity System List	AggregateEntity	SilentEntities
Number of Variable Datum Records	AggregateEntity	NumberOfVariableDatums
Variable Datum Records	AggregateEntity	VariableDatums

9.7.2 IsGroupOf PDU

The AggregateState and IsGroupOf PDUs describe aggregate units in DIS. The RPR FOM represents the AggregateState PDU with the AggregateEntity object class (see section 7.7.1.1). The IsGroupOf PDU contained some additional information about aggregate units that the AggregateState PDU does not represent. This information included Group Entity Descriptor (GED) fields used to convey entity-specific parameters, including relative positioning. The RPR FOM does not directly support the functionality of the GED fields. Federations that require the kind of entity-specific data that GED fields contained can extend the RPR FOM to meet their needs.

The RPR FOM does provide an alternative way to communicate relative positioning information for entity aggregations. The RelativeSpatial attribute of the BaseEntity object class represents the relative position of an entity with respect to a specific host entity. Each entity of an aggregated unit would specify its location using the RelativeSpatial attribute to give its relative position within the unit. A separate aggregate instance would be the host entity and it would communicate the parameters that apply to the unit as a whole.

9.7.3 Transfer Control Request PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	N/A
Originating Entity ID	TransferControl	OriginatingEntity

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Receiving Entity ID	TransferControl	ReceivingEntity
Request ID	TransferControl	RequestIdentifier
Transfer Type	TransferControl	TransferType
Transfer Entity ID	TransferControl	TransferEntity
Number Of Record Sets	TransferControl	RecordSetData
Record Set	TransferControl	RecordSetData

9.7.4 IsPartOf PDU

The IsPartOf PDU allows an entity (*part entity*) to become part of another entity (*host entity*). The RPR FOM and DIS implementations of this function are fundamentally different and are neither compatible nor interoperable. This is primarily because the DIS IsPartOf PDU functionality is non-implementable as described in DIS 1998. Instead, the RPR FOM implements a simplified IsPartOf function. Some data fields from the DIS IsPartOf PDU are, however, used. The IsPartOf concept is optional to implement in both DIS and the RPR FOM.

9.7.4.1 DIS IsPartOf PDU

The DIS IsPartOf function envisioned one simulation requesting another simulation to let one of its entities become part of a requestor's entity. The other simulation's entity then became a *part entity* of the requestor's *host entity*. This request has the option of the other simulation rejecting the request. The IsPartOf PDU is sent once to make the request and the receiving simulation sends an Acknowledge PDU to indicate acceptance or rejection. If it accepted the request, the receiving simulation would stop sending the Entity State PDU for what is now a "part entity" of another simulation's *host entity*. If rejected, the requested entity does not become a *part entity*. It is important to note that the data describing the details of the relationship between the *part entity* and the *host entity* contained in the IsPartOf PDU are "proposed" details of the relationship if the request was accepted. The IsPartOf PDU is sent only once and there is no description of how the details (e.g., nature, position, named location) of an established IsPartOf relationship would be provided or updated.

In DIS, it is clear that the IsPartOf function does not allow an IsPartOf relationship to be established between two entities that are owned by the same simulation. (A simulation does not send PDUs to itself.)

There is no clear indication in DIS of what constitutes the *part entity* information that would be maintained by the *host entity* simulation or the *part entity*. The ownership of a *part entity* is also unclear as no mention is made of ownership in the DIS IsPartOf PDU description. Finally, the DIS IsPartOf PDU includes both the case of a physical association (e.g., the *part entity* could be inside or in physical contact with the *host entity*), or a functional association (e.g., the *part entity* is an aircraft flying in formation with a *host entity*.)

9.7.4.2 RPR FOM IsPartOf Function

The RPR FOM IsPartOf function allows an entity (*part entity*) to establish a relationship with another entity (*host entity*). The IsPartOf relationship could be physical (e.g., the *part entity* could be inside or in physical contact with the *host entity*), or it could be functional (e.g., the *part entity* is an aircraft flying in formation with a *host entity*.) A federate merely indicates that one of its entities is now a *part entity* of another *host entity*. Both the *part* and *host entities* remain as separate entities.

Unlike DIS, a RPR FOM IsPartOf relationship can either be established between (1) two local entities in a single federate or (2) a local entity owned by one federate and a remote entity owned by a different federate. The owner of the *part entity* in an IsPartOf relationship is the one that sends IsPartOf functional data. (A *host entity* does not send any IsPartOf functional data unless it happens to also be the *part entity* in another IsPartOf relationship.

When an IsPartOf relationship involves a local *part entity* and a remote *host entity*, the *host entity* federate has no responsibilities or control over the *part entity*. Information is provided by the *part entity* federate to indicate the physical or functional IsPartOf relationship to the *host entity*.

If the IsPartOf function is implemented for publishing, it does not require a negotiation to establish an IsPartOf relationship between two entities, in the case where each entity is owned by a different federate.

The following general requirements shall apply:

1. A specific local *part entity* can only be associated with one specific local or remote *host entity* at a time.
2. A *host entity* can be the *host entity* in multiple IsPartOf relationships simultaneously. A *host entity* in one or more IsPartOf relationships can also simultaneously be a *part entity* in one other IsPartOf relationship.
3. The IsPartOf functional data indicating the details of the IsPartOf relationship is only sent by the *part entity,* and not the *host entity*.
4. An entity is not considered a *part* or *host entity* except when the entity is involved in an IsPartOf relationship.
5. When an entity is not presently the *part entity* in an IsPartOf relationship, then all IsPartOf functional data shall be set to zero.
6. The IsPartOf functional data consists of the following attributes and other data:
    - IsPartOf attribute
    - RelativeSpatial attribute
    - HostRTIObjectIdentifier
    - HostEntityIdentifier
7. The Spatial Attribute shall continue to be sent even if an IsPartOf relationship exists.
8. The IsPartOf functional data shall always be sent for the *part entity* so long as an IsPartOf relationship exists.
9. If an IsPartOf relationship presently exists and a new relationship is being established with a different *host entity*, then the new relationship merely replaces the existing relationship. There is no separate process to terminate the previous relationship.
10. The following IsPartOf functional requirements are not included in the RPR FOM and would be covered in federation agreements:
    - A requirement to negotiate the establishment of an IsPartOf relationship between two federates including deciding which federate (i.e., the one with the *part entity* or the *host entity*) is allowed to initiate the negotiation, or whether either federate may do so.
    - Allowing a federate with simulation management authority to require that an IsPartOf relationship be established or terminated between two entities.
    - A requirement to allow other than the federate that owns the *part entity* to terminate the associated IsPartOf relationship.
    - Whether the transfer of ownership of a *part entity* is allowed and, if so, what affect this would have on the establishment of, or an existing, IsPartOf relationship. This includes the possible transfer of ownership of the specific RelativeSpatial, Spatial, or IsPartOf BaseEntity attributes.
    - The extent to which a *part entity* has to account for changes to the *host entity* beyond computing the relative spatial relationship (e.g., the *host entity* is indicated to be destroyed but remains as an active entity in the federation).

9.7.4.3 Establishment of an IsPartOf Relationship

When a federate desires to establish an IsPartOf relationship between one of its local entities (a *part entity*) and a local or remote *host entity*, the federate shall set the IsPartOf attributes and fields of its BaseEntity object as follows:

1. <u>IsPartOf attribute</u>.

   a. *HostEntityIdentifier*. This identifier shall be set to the EntityIdentifier of the host entity.
   b. *HostRTIObjectIdentifier*. This identifier shall be set to the RTIObjectIdentifier of the *host entity*.
   c. *Relationship*. Both the Nature and Position subfields are optional. Each subfield shall either be set to Other (0) or to a valid enumeration.
   d. *NamedLocation*. Both the StationName and StationNumber subfields are optional. Each subfield shall either be set to Other (0) or to a valid enumeration. The usage of the StationName alternatives ‘On station RNG/BRG’ (15) and ‘On station - x,y,z’ (16) is discouraged. The RelativeSpatial attribute shall be used to specify the relative spatial location and orientation from the host entity.
2. RelativeSpatial attribute. The fields of this attribute shall be set to indicate the relative spatial relationship to the host entity.

9.7.4.4 Termination of an IsPartOf Relationship

An existing IsPartOf relationship shall be terminated when the federate that owns the BaseEntity object representing the *part entity* sets all IsPartOf functional data to zero, or when a new IsPartOf relationship is established between the *part entity* and a different *host entity*. The IsPartOf relationship shall also be terminated if the *host entity* is deactivated and no longer exists in the federation.

9.7.4.5 IsPartOf Receipt Requirements

If a federate subscribes to IsPartOf information for a remote entity that is a *part or host entity* in an ISPartOf relationship, the following requirements shall apply:

1. There is no requirement to process all received IsPartOf functional data.
2. If the IsPartOf attribute Station Name subfield is set to On station RNG/BRG (15), it shall be ignored and either the RelativeSpatial or Spatial attribute shall be used. (On station RNG/BRG does not provide sufficient information to establish a RelativeSpatial location.)
3. The RelativeSpatial attribute may be ignored even though other IsPartOf functional data is used.
4. A receiving federate can choose to compute a relative spatial location based on the Spatial attribute of the *host entity* indicated by the HostRTIObjectIdentifier and the Spatial attribute of the *part entity*.

9.7.4.6 Other Considerations

No extensive systems analysis has been documented regarding the implementation of the IsPartOf concept either in DIS or for the RPR FOM. The initial functionality might not be adequate for all common cases, such as its use in a multi-resolution environment where the *part entity* and *host entity* federates might not be at the same level of fidelity.

The use of relative spatial information to assist in properly positioning a *part entity* in relation to a *host entity* becomes more problematic when the host entity is a large object, such as an aircraft carrier, and the *part entity* is a much smaller object, such as a fighter aircraft, especially if high-fidelity visual models are involved. It also becomes more problematic when the *part entity* and the *host entity* are owned by two different federates (i.e., one is a local entity and the other is a remote entity).

All relative spatial information is based on the spatial location of entities. Such locations represent the centroid of a bounding volume around each entity. The bounding volume of an entity does not include its real-world articulated and attached parts. Bounding volume information is not required to be exchanged between federates. (In the case of an aircraft carrier, the superstructure would be technically considered part of the bounding volume as it is not an articulated or attached part.) It could be that the centroid of the aircraft carrier entity is above or below the flight deck. This will have an effect on the use of relative spatial information between a *part entity* and the *host entity* as related to, for example, visual models.)

Regardless of whether a receiving federate is using a remote *part entity's* spatial or relative spatial information to position the entity in relationship to a *host entity*, it will adjust such positional information further "to make it look real" from its perspective (e.g., a high-fidelity visual model of an aircraft carrier local *host entity* will not let a remote entity representing a landing aircraft disappear into the deck.)

The present independence of each federate being able to decide to become a *part entity* of a *host entity* relies on federation agreements to maintain control of a given situation (e.g., multiple aircraft attempting to simultaneously land on an aircraft carrier or standing by ready to refuel in close proximity to a tanker aircraft). Although a federation agreement could invoke a requirement to have negotiated IsPartOf relationships to resolve such issues, since there is no centralized, coordinated FOM development activity worldwide, one federation's solution might not be adequate for another federation. In this case, a federate might have to further develop changes to handle the situation when involved in a different federation. The matter of when it would be better, for example, to simply transfer ownership of a *part entity* to the federate that owns the *host entity* instead of using the IsPartOf function has not been fully explored.

Finally, note that the IEEE Std 1278.1™-2012 [8] DIS Standard provides the capability to associate (or disassociate) two or more entities with each other that represent either a physical or functional association between them, including providing details of the association. This is independent of the use of the IsPartOf PDU and is accomplished using the Entity Association Variable Parameter (VP) record. This association can be between two local entities or a local and remote entity. This VP record, together with additional VP records, can provide an alternative approach to implementing the IsPartOf function. This is summarized here:

1. Adapting the DIS Entity Association capability for use in a FOM can be more practical approach than using the RPR FOM IsPartOf function, such as the case where it is desired to indicate an aircraft in a formation.
2. The availability of the VP record in the Entity State PDU permits the easy inclusion of a new RelativeSpatial VP record and a new HostEntityIdentifier VP record that would be compatible with the corresponding RPR FOM data. (Note: The implementation of these two new VP records could be used independently of any entity association or IsPartOf function.)

FOM developers need to look at all options available, including the capabilities of IEEE Std 1278.1™-2012, to assess the best approach to providing physical and functional association information between entities in an HLA environment whether as part of an IsPartOf function or some other function.

9.8 Minefield

9.8.1 Minefield State PDU

DIS Fields	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Minefield ID	Minefield	MinefieldID
Minefield Sequence Number	Minefield	MinefieldSequenceNumber
Force ID	Minefield	ForceID
Minefield Type	Minefield	MineTypes
Number of Mine Types	Minefield	
Minefield Location	Minefield	MinefieldLocation
Minefield Orientation	Minefield	MinefieldOrientation
Appearance	Minefield	MinefieldAppearanceType
Protocol Mode	Minefield	ProtocolMode
Perimeter Point Coordinates	Minefield	PerimeterPointCoordinates
Mine Type	Minefield	MineTypes

9.8.2 Minefield Query PDU

DIS Fields	FOM Class	FOM Attribute(s)/Parameter(s)
Minefield ID	MinefieldQuery	MinefieldID

DIS Fields	FOM Class	FOM Attribute(s)/Parameter(s)
Requesting Entity ID	MinefieldQuery	RequestingEntityID
Request ID	MinefieldQuery	RequestID
Number of Perimeter Points	MinefieldQuery	size of PerimeterPoints data
Number of Sensors Types	MinefieldQuery	size of SensorTypes data
Data Filter	MinefieldQuery	QueryGroundBurialDepthOffset
		QueryWaterBurialDepthOffset
		QuerySnowBurialDepthOffset
		QueryMineOrientation
		QueryThermalContrast
		QueryReflectance
		QueryMineEmplacementAge
		QueryTripDetonationWire
		QueryFusing
		QueryScalarDetectionCoefficient
		QueryPaintScheme
Requested Mine Type	MinefieldQuery	RequestedMineType
Requested Perimeter Point Coordinates	MinefieldQuery	PerimeterPoints
Sensor Types	MinefieldQuery	SensorTypes

9.8.3 Minefield Data PDU

DIS Fields	FOM Class	FOM Attribute(s)/Parameter(s)
Minefield ID	MinefieldData	MinefieldID
Requesting Entity ID	N/A	N/A
Minefield Sequence Number	MinefieldData	MinefieldSequenceNumber
Request ID	N/A	N/A
PDU Sequence Number	N/A	N/A
Number of PDUs	N/A	N/A
Number of Mines in this PDU (n)	MinefieldData	size of various data elements
Number of Sensor Types (m)	MinefieldData	size of SensorTypes
Data Filter	N/A	N/A
Mine Type	MinefieldData	MineType
Sensor Types	MinefieldData	SensorTypes
Mine Location	MinefieldData	MineLocation
Ground Burial Depth Offset	MinefieldData	GroundBurialDepthOffset
Water Burial Depth Offset	MinefieldData	WaterBurialDepthOffset
Snow Burial Depth Offset	MinefieldData	SnowBurialDepthOffset
Mine Orientation	MinefieldData	MineOrientation
Thermal Contrast	MinefieldData	ThermalContrast
Reflectance	MinefieldData	Reflectance
Mine Emplacement Time	MinefieldData	MineEmplacementTime
Mine Entity ID	MinefieldData	MineEntityID
Fusing	MinefieldData	Fusing
Scalar Detection Coefficient	MinefieldData	ScalarDetectionCoefficient
Paint Scheme	MinefieldData	PaintScheme
Number of Trip/Detonation Wires	MinefieldData	NumberTripDetonationWires
Number of Vertices	MinefieldData	NumberWireVertices
Vertex	MinefieldData	WireVertices

9.8.4 MinefieldResponseNACK

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Minefield ID	MinefieldResponseNack	MinefieldID
Requesting Entity ID	MinefieldResponseNack	RequestingEntityID
Request ID	MinefieldResponseNack	RequestID
Number of Missing PDU(s)	MinefieldResponseNack	
Missing PDU Sequence Number(s)	MinefieldResponseNack	MissingRecordNumbers

9.9 Synthetic Environment Family

9.9.1 Environmental Process PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Environmental Process ID	EnvironmentProcess	ProcessIdentifier
Environment Type	EnvironmentProcess	Type
Model Type	EnvironmentProcess	ModelType
Environment Status	EnvironmentProcess	EnvironmentProcessActive
Number of Environment Records	EnvironmentProcess	EnvironmentRecData
Environment Record	EnvironmentProcess	EnvironmentRecData

9.9.2 Gridded Data PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Environmental Simulation Application ID	GriddedData	GridIdentifier
Field Number	N/A	
PDU Number	N/A	
PDU Total	N/A	
Coordinate System	GriddedData	CoordinateSystem
Number of Grid Axes	GriddedData	NumberOfGridAxes
Constant Grid	GriddedData	ConstantGrid
Environment Type	GriddedData	EnvironmentType
Orientation	GriddedData	Orientation
Sample Time	GriddedData	SampleTime
Total Values	GriddedData	TotalValues
Vector Dimension	GriddedData	VectorDimension
Grid Axis Descriptor	GriddedData	GridAxisInfo
Grid Data	GriddedData	GridDataInfo

9.9.3 Point Object State PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Object ID	EnvironmentObject	ObjectIdentifier
Referenced Object ID	EnvironmentObject	ReferencedObjectID
Update Number	N/A	
Force ID	EnvironmentObject	ForceIdentifier
Modifications	N/A	
Object Type	EnvironmentObject	ObjectType
Object Location	PointObject	Location

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Object Orientation	PointObject	Orientation
Object Appearance	PointObject	PercentComplete, DamagedAppearance, ObjectPreDistributed, Deactivated, Smoking, Flaming
Requestor ID	EnvironmentObjectTransaction	RequestingIdentifier
Receiving ID	EnvironmentObjectTransaction	ReceivingIdentifier

9.9.4 Linear Object State PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Object ID	EnvironmentObject	ObjectIdentifier
Referenced Object ID	EnvironmentObject	ReferencedObjectID
Update Number	N/A	
Force ID	EnvironmentObject	ForceIdentifier
Number of Segments	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Requestor ID	EnvironmentObjectTransaction	RequestingIdentifier
Receiving ID	EnvironmentObjectTransaction	ReceivingIdentifier
Object Type	EnvironmentObject	ObjectType
Segment Number	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Modification	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Appearance	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Location	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Orientation	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Length	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Width	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
Segment Height	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords
Segment Depth	BreachableLinearObject, BreachObject, ExhaustSmokeObject, MinefieldLaneMarker	SegmentRecords

9.9.5 Areal Object State PDU

DIS Field	FOM Class	FOM Attribute(s)/Parameter(s)
PDU Header	N/A	
Object ID	EnvironmentObject	ObjectIdentifier
Referenced Object ID	EnvironmentObject	ReferencedObjectID
Update Number	N/A	
Force ID	EnvironmentObject	ForceIdentifier
Modifications	N/A	
Object Type	EnvironmentObject	ObjectType
Object Appearance	ArealObject	PercentComplete, DamagedAppearance, ObjectPreDistributed, Deactivated, Smoking, Flaming
Number of Points	ArealObject	PointsData
Requestor ID	EnvironmentObjectTransaction	RequestingIdentifier
Receiving ID	EnvironmentObjectTransaction	ReceivingIdentifier
Object Location	ArealObject	PointsData

9.10 Simulation Management with Reliability

The following SIMAN with Reliability PDUs have a corresponding SIMAN PDU. All fields map to the same parameters of the interaction class that the corresponding SIMAN PDU's fields do. That interaction class is subclassed with an attributeless subclass whose name is the same as its parent's with a capital 'R' appended.

- Acknowledge-R PDU
- Action Response-R PDU

The following SIMAN with Reliability PDUs have a corresponding SIMAN PDU with the addition of a Required Reliability Service field. This Required Reliability Service field maps to the AcknowledgementProtocol parameter of the subclass of the interaction that its corresponding SIMAN PDU maps to. This interaction class has the same name as its parent with a capital 'R' appended. All of the other fields map to the same parameters of the interaction class that the corresponding SIMAN PDU's field do. The reliable interactions (those SIMAN interactions that end in R) are still sent best effort between federates. They should be transmitted / received in a similar manner described from reliable PDUs in the 1278.1a specification.

- Action Request-R PDU
- Create Entity-R PDU
- Data-R PDU
- Data Query-R PDU
- Remove Entity-R PDU
- Set Data-R PDU

- Start/Resume-R PDU
- Stop/Freeze-R PDU

Appendix A. Bibliography (Informative)

SISO-ADM-002-2011, SISO Policies and Procedures, 11 April 2011.

SISO-ADM-005-2011, Policy for: The Style and Format of SISO Documents, 13 June 2011.

SISO-STD-001.1-1999, Real-time Platform Reference Federation Object Model (RPR FOM 1.0), 24 August 1999.

SISO-STD-001-1999, Guidance, Rationale & Interoperability Modalities for the RPR FOM (GRIM 1.0), 24 August 1999.

Appendix B. Differences Between RPR FOM 1.0 and RPR FOM 2.0 (Informative)

New Object Classes

- ActiveSonar
- ActiveSonarBeam
- AdditionalPassiveActivities
- AggregateEntity
- ArealObject
- BreachObject
- BreachableLinearObject
- BreachablePointObject
- BurstPointObject
- CraterObject
- EnvironmentObject
- EnvironmentProcess
- ExhaustSmokeObject
- GriddedData
- IFF
- LinearObject
- Minefield
- MinefieldData
- MinefieldLaneMarkerObject
- MinefieldObject
- NatoIFF
- NatoIFFInterrogator
- NatoIFFTransponder
- OtherArealObject
- OtherLinearObject
- OtherPointObject
- PointObject
- PropulsionNoise
- RRB
- RibbonBridgeObject
- SovietIFF
- SovietIFFInterrogator
- SovietIFFTransponder
- StructureObject
- UnderwaterAcousticsEmission

Added Attributes

- BaseEntity
    - IsPartOf
    - RelativeSpatial
    - Spatial
- Lifeform
    - ComplianceState
- PhysicalEntity
    - AcousticSignatureIndex
    - InfraredSignatureIndex

- LiveEntityMeasuredSpeed
- PropulsionSystemsData
- RadarCrossSectionSignatureIndex
- VectoringNozzleSystemData

- RadioTransmitter
  - SpreadSpectrum
  - StreamTag

Removed Attributes

- BaseEntity
  - AccelerationVector
  - AngularVelocityVector
  - DeadReckoningAlgorithm
  - IsFrozen
  - Orientation
  - VelocityVector
  - WorldLocation
- RadioTransmitter
  - FrequencyHopInUse
  - ModulationParameters
  - PsuedoNoiseSpectrumInUse

New Interaction Classes

- AcknowledgeR
- AcousticTransient
- ActionRequestR
- ActionRequestToObjectR
- ActionResponseFromObjectR
- ActionResponseR
- ArealObjectTransaction
- AttributeChangeRequestR
- AttributeChangeResultR
- BreachObjectTransaction
- BreachableLinearObjectTransaction
- BreachablePointObjectTransaction
- BurstPointObjectTransaction
- CollisionElastic
- CraterObjectTransaction
- CreateEntityR
- CreateObjectRequestR
- CreateObjectResultR
- DataQueryR
- DataR
- EnvironmentObjectTransaction
- ExhaustSmokeObjectTransaction
- LinearObjectTransaction
- MinefieldData
- MinefieldLaneMarkerObjectTransaction
- MinefieldObjectTransaction
- MinefieldQuery
- MinefieldResponseNACK
- OtherArealObjectTransaction
- OtherLinearObjectTransaction

- OtherPointObjectTransaction
- PointObjectTransaction
- RecordQueryR
- RecordR
- RemoveEntityR
- RemoveObjectRequestR
- RemoveObjectResultR
- RibbonBridgeObjectTransaction
- SetDataR
- SetRecordR
- StartResumeR
- StopFreezeR
- StructureObjectTransaction
- TransferControl

Added Parameters

- EncodedAudioRadioSignal
    - AudioData

Removed Parameters

- EncodedAudioRadioSignal
    - HostRadioIndex
    - SampleCount
    - SignalData
    - SignalDataLength
    - SignalSampleRate
    - TransmitterSignalEncodingType

New Datatypes[5]

- *Basic*
    - RPRunsignedInteger16BE
    - RPRunsignedInteger32BE
    - RPRunsignedInteger64BE
    - RPRunsignedInteger8BE
- *Simple*
    - AccelerationMeterPerSecondSquaredFloat32
    - AngleDegreeFloat32
    - AngleRadianFloat32
    - AngularVelocityRadianPerSecondFloat32
    - BitRateBitPerSecondUnsignedInteger32
    - BitsUnsignedInteger16
    - ClockTimeHourInteger32
    - DepthMeterFloat32

[5] In DoD HLA 1.3 there is only a distinction between basetypes, defined in the OMT Specification, and enumerated and complex datatypes defined as part of the object model. In addition, in DoD HLA 1.3 arrays are not distinct datatypes but follow from attribute/parameter/field cardinality other than 1, and can be preceded by a field indicating the number of elements. Hence all HLA 1516 basic, simple, and array datatypes defined in RPR FOM 2.0 are new. The DoD HLA 1.3 complex datatypes are classified in HLA 1516 as either fixed record or variant record datatypes.

  - Float32
  - Float64
  - FrequencyHertzFloat32
  - FrequencyHertzUnsignedInteger64
  - Integer16
  - Integer32
  - InterrogationsPerSecondFloat32
  - LengthMeterFloat32
  - MassKilogramFloat32
  - MeterFloat32
  - MeterFloat64
  - MineDielectricDifference
  - MineIdentifier
  - Octet
  - PercentFloat32
  - PercentUnsignedInteger32
  - PowerRatioDecibelMilliwattFloat32
  - PowerWattFloat32
  - RevolutionsPerMinuteFloat32
  - RevolutionsPerMinuteInteger16
  - SpeedChangeRateRPMPerSecondInteger16
  - TemperatureDegreeCelsiusFloat32
  - TimeMicrosecondFloat32
  - TimeMillisecondUnsignedInteger32
  - TimeSecondInteger32
  - TimestampUnsignedInteger32
  - TransponderModeCAltitude100-FootInteger16
  - UnsignedInteger16
  - UnsignedInteger32
  - UnsignedInteger64
  - UnsignedInteger8
  - VelocityDecimeterPerSecondInteger16
  - VelocityMeterPerSecondFloat32
  - WavelengthMicronFloat32
- *Enumerated*
  - AcknowledgementProtocolEnum8
  - ActiveSonarEnum16
  - ActiveSonarFunctionCodeEnum8
  - ActiveSonarScanPatternEnum16
  - AggregateStateEnum8
  - BreachedStatusEnum8
  - ChemicalContentEnum32
  - ComplianceStateEnum32
  - ConstituentPartNatureEnum16
  - ConstituentPartPositionEnum16
  - ConstituentPartStationNameEnum16
  - EnvironmentDataCoordinateSystemEnum16
  - EnvironmentDataRepresentationEnum16
  - EnvironmentDataSampleTypeEnum16
  - EnvironmentGridAxisTypeEnum8
  - EnvironmentGridTypeEnum8
  - EnvironmentModelTypeEnum8
  - EnvironmentRecordTypeEnum32
  - FormationEnum32
  - IffAlternateMode4Enum8
  - IffApplicableModesEnum8

  - IffOperationalParameter1Enum8
  - IffOperationalParameter2Enum8
  - IffSystemModeEnum8
  - IffSystemNameEnum16
  - IffSystemTypeEnum16
  - MinefieldFusingEnum32
  - MinefieldLaneEnum8
  - MinefieldPaintSchemeEnum32
  - MinefieldProtocolEnum8
  - MinefieldSensorTypeEnum32
  - MinefieldStatusEnum8
  - MinefieldTypeEnum8
  - PropulsionPlantEnum8
  - RPRboolean
  - SpreadSpectrumEnum16
  - TransferTypeEnum8
  - VisibleSideLocationEnum32
- *Array*
  - AntennaPatternVariantStructLengthlessArray
  - ArticulatedParameterStructLengthlessArray
  - BreachableSegmentStructLengthlessArray
  - BreachedStatusArray8
  - BreachStructLengthlessArray
  - ClockTimeStructLengthlessArray
  - CoefficientsLengthlessArray1Plus
  - DatumIdentifierLengthlessArray
  - TemperatureDegreeCelsiusFloat32LengthlessArray
  - DepthMeterFloat32LengthlessArray
  - EntityTypeStructLengthlessArray
  - EnvironmentRecStructArray
  - ExhaustSmokeStructLengthlessArray
  - FixedDatumStructLengthlessArray
  - Float32Array1Plus
  - FundamentalParameterDataStructLengthlessArray
  - GridAxisStructLengthlessArray
  - GridDataStructLengthlessArray
  - Integer16Array1Plus
  - MarkingArray11
  - MarkingArray31
  - MineDielectricDifferenceLengthlessArray
  - MinefieldLaneMarkerStructLengthlessArray
  - MinefieldPaintSchemeLengthlessArray
  - MinefieldSensorTypeLengthlessArray
  - MineFusingStructLengthlessArray
  - MineIdentifierLengthlessArray
  - MissingRecordNumbersLengthlessArray1Plus
  - OctetArray
  - OctetArray1Plus
  - OctetArray2
  - OctetArray3
  - OctetArray4
  - OctetArray7
  - OctetArray8
  - OctetPadding32Array
  - OctetPadding64Array
  - OrientationStructLengthlessArray

- PerimeterPointStructLengthlessArray
- PropulsionSystemDataStructLengthlessArray
- RecordSetStructArray1Plus
- RecordStructArray
- RPRUserDefinedTag
- ShaftDataStructLengthlessArray1Plus
- SignalDataLengthlessArray1Plus
- SilentAggregateStructLengthlessArray
- SilentEntityStructLengthlessArray
- SupplyStructLengthlessArray
- UnsignedInteger16Array1Plus
- UnsignedInteger32LengthlessArray
- UnsignedInteger64Array1Plus
- UnsignedInteger8LengthlessArray
- VariableDatumStructLengthlessArray
- VectoringNozzleSystemDataStructLengthlessArray
- WorldLocationStructLengthlessArray

- *Fixed Record*
  - AggregateMarkingStruct
  - AudioDataTypeStruct
  - BreachStruct
  - BreachableSegmentStruct
  - COMBICStateRecStruct
  - Cone1GeomRecStruct
  - Cone2GeomRecStruct
  - ConstituentPartRelationshipStruct
  - DimensionRateStruct
  - DimensionStruct
  - Ellipsoid1GeomRecStruct
  - Ellipsoid2GeomRecStruct
  - EntityCoordinateVectorStruct
  - EnvironmentObjectTypeStruct
  - EnvironmentRecStruct
  - EnvironmentTypeStruct
  - ExhaustSmokeStruct
  - FlareStateRecStruct
  - FundamentalParameterDataStruct
  - GaussPlumeGeomRecStruct
  - GaussPuffGeomRecStruct
  - GridAxisStruct
  - GridDataStruct
  - GridValueType0Struct
  - GridValueType1Struct
  - GridValueType2Struct
  - IrregularGridAxisStruct
  - IsPartOfStruct
  - Line1GeomRecStruct
  - Line2GeomRecStruct
  - LinearSegmentStruct
  - MineFusingStruct
  - MinefieldLaneMarkerStruct
  - NamedLocationStruct
  - PerimeterPointStruct
  - PlumeDimensionRateStruct
  - PlumeDimensionStruct
  - Point2GeomRecStruct

- PropulsionSystemDataStruct
- RecordSetStruct
- RecordStruct
- RectVol1GeomRecStruct
- RectVol2GeomRecStruct
- RectVol3GeomRecStruct
- RelativeRangeBearingStruct
- ShaftDataStruct
- SilentAggregateStruct
- SilentEntityStruct
- SpatialFPStruct
- SpatialFVStruct
- SpatialRPStruct
- SpatialRVStruct
- SpatialStaticStruct
- Sphere1GeomRecStruct
- Sphere2GeomRecStruct
- UniformGeomRecStruct
- VectoringNozzleSystemDataStruct

- *Variant Record*
  - EnvironmentRecVariantStruct
  - GridAxisTypeVariantStruct
  - GridDataRepresentationVariantStruct
  - SpatialVariantStruct
  - SpreadSpectrumVariantStruct
  - StationNameLocationVariantStruct

Renamed Datatypes

- *Variant Record*
  - AntennaPatternVariantStruct (was AntennaPatternStruct)
  - ParameterValueVariantStruct (was ParameterValueStruct)
  - RFModulationTypeVariantStruct (was RFModulationTypeStruct)

Removed Datatypes

- ModulationStruct

Changed Datatypes

- *Array*[6]
  - AttributeValuePairStructArray1Plus (was AttributeValueSetStruct)
    - Replaced AttributeSetCount and AttributePairs fields by array datatype
  - RTIobjectId (was RTIObjectIdStruct)
    - Replaced ID field of Datatype “string” by array datatype of “HLAASCIIchar”
  - RTIobjectIdArray (was RTIObjectIdArrayStruct)
    - Replaced Length and ID fields by array datatype
    - This changes the Datatype of Length (array number_of_elements) from “unsigned short” to an unsigned 32-bit integer
  - VariableDatumStructArray (was VariableDatumSetStruct)

[6] Applies to both HLA 2000 and HLA 2010 OMT files. For HLA 1.3 OMT, arrays are converted to complex datatypes matching HLA 2000 and HLA 2010 encoding.

        - Replaced NumberOfVariableDatums and VariableDatums fields by array datatype
        - Changed Cardinality of VariableDatums (array content) from “1+” to “0+”

- *Fixed Record*
    - ArticulatedParameterStruct
        - Removed Padding[7]
    - AttributeValuePairStruct (was AttributePairStruct)
        - Replaced ValueLength and Value by NumberOfBytes-A-Value (array datatype)
        - Changed Datatype of Value (array content) from “any” to “Octet”
        - Changed Cardinality of Value (array content) from “1” to “0+”
        - Added PaddingTo32 field
    - BeamAntennaStruct
        - Removed Padding1 and Padding2[7]
    - SINCGARSModulationStruct
        - Removed ClearChannel
        - Removed Padding
        - Removed StartOfMessage
    - SphericalHarmonicAntennaStruct
        - Changed Datatype of Order from “char” to an unsigned 32-bit integer
        - Added Padding
    - VariableDatumStruct
        - Replaced DatumLength and DatumValue by DatumValue (array datatype)
        - Added DatumLength specifying the length of the variable datum in bits.
        - Changed Datatype of DatumValue (array content) from “any” to an unsigned 64-bit integer
        - Changed Cardinality of DatumValue (array content) from “1” to “1+”

---

[7] Applies to both HLA 2000 and HLA 2010 OMT files. For HLA 1.3 OMT, explicit padding fields are introduced where necessary to ensure identical encoding.