MP 06W0000131

AviationSimNet Specification

Release Date: June 2010
Version: 2.2

Authored by the AviationSimNet Standards Working Group
Published by The MITRE Corporation

Sponsor:	The MITRE Corporation	**Contract No.:**	MITRE Technology Program
Dept. No:	F053	**Project No.:**	02MSR055-H3

MITRE

Center for Advanced Aviation System Development
McLean, Virginia

Abstract

This document is a specification for creating and executing distributed Air Traffic Control (ATC) Human-in-the-Loop (HITL) simulations over a public, wide area network. Known as AviationSimNet™[1], this specification was developed mostly by adopting existing industry standards for network communications, in both simulation and voice protocols. This specification builds upon other distributed simulation efforts and will continue to evolve according to the needs of the aviation research community.

The scope of this specification includes definitions of the technologies required to support data and voice inter-communication in a controlled, simulated environment that models ATC simulations. It also includes the required protocols for using those technologies in order to facilitate implementations of this specification over a common network.

This document does not provide instructions on how to adapt any existing simulation capability to comply with the AviationSimNet specification. Rather, it provides the rules and limitations that can be used to achieve such goals. This document also does not address any activities associated with conducting analyses of a simulation execution. AviationSimNet requirements are highlighted throughout this document. Any application that satisfies these requirements is considered "AviationSimNet-compatible."

Section 1 of this document provides an introduction to the scope and technologies covered by AviationSimNet, as well as definitions that are relevant to AviationSimNet. Section 2 covers data communication protocols among simulation applications. Section 3 covers voice communication protocols among live participants of the simulation. Section 4 provides an overview of security measures for using AviationSimNet over a public network. Section 5 addresses performance, throughput, and latency requirements.

AviationSimNet is a collaborative effort among industry, academia, and government agencies. As such, this document was co-authored by the organizations participating in the AviationSimNet Standards Working Group. To date, organizations participating in AviationSimNet include Airline Pilot Association (ALPA), The Boeing Corporation, Center for Applied ATM Research (CAAR) at Embry-Riddle Aeronautical University (ERAU), Crown Consulting, The Federal Aviation Administration (FAA), Lockheed Martin Transportation and Security Solutions, The MITRE Corporation.s Center for Advanced Aviation System Development (CAASD), National Aeronautics and Space Administration (NASA) Ames Research Center, NASA Langley Research Center, Raytheon, and United Parcel Service.

[1] AviationSimNet is a trademark of The MITRE Corporation.

KEYWORDS: Air Traffic Control, Distributed Simulation, Networking, Standards, High Level Architecture, DIS 1278.1a

Acknowledgments

The following organizations are participants in the AviationSimNet Standards Working Group which is responsible for defining this Specification: Airline Pilot Association (ALPA), The Boeing Corporation, Center for Applied ATM Research (CAAR) at Embry-Riddle Aeronautical University (ERAU), Crown Consulting, The Federal Aviation Administration (FAA), Lockheed Martin Transportation and Security Solutions, The MITRE Corporation's Center for Advanced Aviation System Development (CAASD), National Aeronautics and Space Administration (NASA) Ames Research Center, NASA Langley Research Center, Raytheon, and United Parcel Service.

Table of Contents

List of Figures

List of Tables

Section 1

Introduction

This section provides an introduction to the scope and technologies associated with AviationSimNet.

1.1 Background

AviationSimNet was conceived in 2004 by The MITRE Corporation's Center for Advanced Aviation System Development (CAASD), and developed through its internal research and development program. Building on extensive experience in distributed Air Traffic Control (ATC) simulation, CAASD identified the need to extend simulation capabilities beyond the limitations of a single laboratory, and beyond the existing pair-wise establishment of distributed simulations.

Noting advancements in network and communications technologies, as well as industry standards in simulation, CAASD realized that research and advancements in ATC could leverage existing simulation assets from the aviation community, industry and academia. By connecting those assets through such technologies and standards, AviationSimNet can expedite the promotion of concepts from the laboratory to the field.

AviationSimNet is a collaborative effort with industry, academia, and government agencies. As depicted in Figure 1-1, organizations participating in AviationSimNet to date include Airline Pilot Association (ALPA), The Boeing Corporation, Center for Applied ATM Research (CAAR) at Embry-Riddle Aeronautical University (ERAU), Crown Consulting, Federal Aviation Administration's (FAA), Lockheed Martin Transportation and Security Solutions, The MITRE Corporation's Center for Advanced Aviation System Development (CAASD), National Aeronautics and Space Administration (NASA) Ames Research Center, NASA Langley Research Center, Raytheon, and United Parcel Service.

CAASD
FEDERAL AVIATION ADMINISTRATION
NASA
Raytheon
Crown Consulting, Inc.
ups
BOEING
EMBRY-RIDDLE AERONAUTICAL UNIVERSITY
alpa
LOCKHEED MARTIN

Figure 1-1. AviationSimNet Standards Working Group Participants

1.2 Scope for AviationSimNet

The scope of AviationSimNet is Air Traffic Management (ATM) simulation. All applications that adopt this specification are assumed to provide and/or consume data or voice communications associated with simulating aircraft, airspace, surveillance systems, communications systems, navigation systems, operational planning, or other elements of ATM that can be examined in a simulation or training environment. Among other things, such environments are often used for conducting research and development of ATM procedures and decision support systems.

This specification does not provide guidance on what specific types of procedures can be effectively studied through the use of AviationSimNet, however there are no known ATM applications which should be ruled out.

1.3 Human-in-the-Loop Simulation

A human-in-the-loop (HITL) study conducted in the ATC domain implies that humans will be interfacing with a simulated environment and with each other through simulated communications protocols. Those human subjects would be acting in roles associated with ATC such as controllers or pilots. Regardless of the specific roles, the AviationSimNet specification assumes that HITL simulations will always be executed in a mode that's compatible with human interaction. To support this human interface, the rate of simulation time advancement will always be equal to "real-time."

1.4 Hardware and Software

AviationSimNet is designed to take advantage of off-the-shelf technologies already adopted by industry for supporting distributed simulation. These technologies may require the user to build or purchase specific hardware or software elements. Such details are implementation-specific, and as such, no particular hardware platforms or software packages are required by this specification.

1.5 Networking

It is important to make simulations easily accessible to the aviation community, while understanding there are individual security and resource limitations among participants. These needs were addressed by building AviationSimNet on top of the Internet Protocol (IP) standard. As such, it can function across a range of network topologies including local area networks, dedicated wide area networks, or the public Internet.

1.6 Prior Work

The concept of conducting distributed simulation in the ATC domain is not new. CAASD built an architecture to support the Federal Aviation Administration's (FAA) Simulation Capability in 1992 [1]. The FAA's William J. Hughes Technical Center and NASA Ames Research Center conducted a real-time HITL experiment in Free Flight called the *Air-Ground Integration Experiment* in 2001 [2]. NASA Ames also has built the Virtual Airspace Simulation Technology Real-Time (VAST-RT) project as part of their Virtual Aerospace Modeling and Simulation (VAMS) project [3]. All of these projects shared the same goal of distributed ATC simulation, but varied in their technical approach. AviationSimNet's architecture adapts some technical features of these projects in defining the protocols for distributed simulation, but leverages the Internet to make those simulation interactions standard, scalable and adaptable.

Section 2

Version History

Version 1.0 was originally published in October 2005.

Version 2.0 was published in August 2006 and implemented the following changes:

Published by the AviationSimNet Standards Working Group
Better definition for Synchronization Points.
Added mention that this document does not specify any use of Data Distribution Management in HLA.
Replaced "synchronized federate" with "start-up federate" to avoid confusion.
Added mention of Federation Timekeeper where particular duties are specified.
Added "timestamp" as an attribute of the aircraft.
Removed duplicate requirement (28 = 24).
Added mention that federates will "wait" for synchronization point announcements, depending on the join order.
Added requirement that federates can't achive ReadyToRun until they've received the StartSimTime interaction.
Removed the appendix list of requirements.

Version 2.1 removes the ReadyToTerminate Synchronization Point, and redefines how aircraft unique IDs are used.

Section 3

Simulation Data Communications

3.1 Introduction

The data communications for an AviationSimNet simulation must provide the means for exchanging information about the simulated objects and data messages presented by the simulation assets. These objects and messages correlate to entities and events that transpire in a simulated ATC environment. At a higher level, it also coordinates simulation-specific activity so that all the assets are brought together in a single, virtual controlled environment that maintains coordinated state information, such as simulation time as well as startup and shutdown sequencing.

In order to support a distributed simulation over a network composed of heterogeneous participants, this specification sought to adopt an industry standard already devoted to providing these services. That standard is the High Level Architecture (HLA), version 1.3.

HLA was designed to support distributed simulations among heterogeneous systems. It has been standardized by the Institute of Electrical and Electronics Engineers (IEEE) and the U.S. Department of Defense (DOD), and it has been used widely among DOD simulations. HLA is a component-based architecture, and defines terms such as Federate, Federation, Runtime Infrastructure (RTI), and Federation Object Model (FOM). These terms will be used throughout this section. Refer to [4, 5, and 6] at the end of this document for additional information on HLA.

REQ 1	**All participants engaged in data communications shall follow the rules of HLA for exchange of data and simulation coordination.**

By adopting HLA, AviationSimNet subsumes several rules and requirements associated with supporting a data network. This section continues with specifications for how to implement the HLA under the rules of AviationSimNet. It is assumed that the reader is familiar with the rules, interfaces and templates defined in the HLA specification.

3.2 Definitions

3.2.1 Participant

A single organization that maintains a set of one or more AviationSimNet-compatible applications engaged in a simulation is a *participant*.

3.2.2 Simulation Asset

Any simulation capability or part of a simulation capability that complies with the AviationSimNet specification is referred to as a *simulation asset*. The asset may be hardware and/or software based, and may support data and/or voice communications. The asset may furthermore be made up of multiple hardware or software components. Multiple simulation assets may be maintained by a single participant. An AviationSimNet simulation with two or more engaged simulation assets is considered to be a "distributed simulation." Flight simulators, radar displays, target generators and tower simulators are all examples of simulation assets.

3.2.3 Simulation

A collection of simulation assets participating together through voice and/or data communications constitutes a *simulation*. An AviationSimNet simulation can satisfactorily be conducted by one or more participants.

3.2.4 AviationSimNet Runtime Infrastructure

The HLA specification defines the runtime infrastructure (RTI) through its interfaces with simulation components. AviationSimNet refines the HLA definition of the RTI as the one that is using the approved AviationSimNet Federation Object Model. There may be multiple federations that exist in the simulation, but only one uses the AviationSimNet RTI. Mention of the RTI in this document refers to the one used with the AviationSimNet HLA federation. AviationSimNet federates can only communicate with other AviationSimNet federates through this RTI.

3.2.5 AviationSimNet Federate

The HLA specification defines a federate through its association with a runtime infrastrucutre. AviationSimNet further refines that definition such that a federate must be joined to the AviationSimNet RTI. Note that it is possible for an asset to be an HLA federate as a member of a non-AviationSimNet federation. To avoid confusion, those assets are to be distinguished from AviationSimNet federates, which must participate as joined federates in an AviationSimNet federation. Mention of federates in this document refers only to the AviationSimNet definition.

3.2.5.1 Start-Up Federate

A start-up federate is an AviationSimNet federate that participates in all AviationSimNet federation synchronization point activities. Synchronization points are implemented within the HLA RTI and are used in AviationSimNet to put the execution of specific actions of the federates into a controlled order during initialization. Synchronization points are discussed in more detail in Section 3.8.1.1 Synchronization Points.

3.2.5.2 Unsynchronized Federate

An unsynchronized federate is an AviationSimNet federate that does not participate in any AviationSimNet federation synchronization point activities. These federates are less concerned about ordering of initialization activities, and can join the federation at any time.

3.2.6 AviationSimNet Federation

The HLA specification defines a federation as the collection of the FOM, an RTI and a set of federates. AviationSimNet refines that definition such that the federation is the one composed of AviationSimNet federates and the RTI that uses the AviationSimNet FOM. Mention of federation in this document refers to the AviationSimNet definition.

3.2.7 System Time

Federates are allowed to execute from different host computing systems. Each of those systems maintains an internal system clock. That clock represents the *system time* for the computer hosting the federate. The system time has no intrinsic correlation to the operation of AviationSimNet, and therefore the system clocks on each of the hosting computers need not be synchronized with each other.

3.2.8 Logical Time

Each HLA federate maintains its own logical time. *Logical time* is the federate's own value of simulation time at a given moment. Logical time is completely distinct from system time.

3.2.9 Real-Time

All AviationSimNet simulations advance simulation time at a rate called *real-time*. Real-time simulations advance logical time at the same rate as system time. The only exception to this is when the simulation is paused or terminated, in which case logical time advance is halted. This is in contrast to simulations that can run in "fast time", "slow time" and "discrete event time" modes.

3.2.10 Simulation Time

HLA does not enforce the use of a "single federation time." However this document will refer to *simulation time* as being the current logical time of any federate. The context of simulation time does not depend on the frequency with which a federate maintains its own logical time, so the simulation time is effectively a theoretical value.

3.3 Network Topology and Access

The HLA specification does not define how the RTI is implemented at the network layer. Currently, all commercially available implementations of the RTI are only runtime compatible in a federation composed of federates built using a single, common RTI product. Federates built under different RTI implementations are not guaranteed to be runtime compatible, even though the respective RTIs may be fully compliant with the HLA specification.

Given that there are different RTI implementations on the market, the AviationSimNet specification provides guidance in selecting an appropriate RTI with respect to how it should behave over the network formed by AviationSimNet.

REQ 2	**All participants shall agree on using a single version of an RTI implementation.**

Because AviationSimNet must support multiple participants with federates on multiple networks, the RTI must be able to support a federation over a wide area network.

REQ 3	**The HLA RTI implementation shall support a federation that uses TCP/IP network connections and spans multiple networks.**

Those network connections are often filtered by boundary firewalls, therefore the topology for AviationSimNet connections must be architected so they are not blocked by those firewalls. For each TCP/IP connection, there is a client that establishes the connection, and a server that accepts it. If every federate in an AviationSimNet federation were to connect with all other federates, then each federate application would behave as a server for those connections. Since many firewall policies prohibit network connections from so-called "untrusted" sources, AviationSimNet could not operate with each federate acting as a server process behind protected boundaries.

REQ 4	**The HLA RTI shall not require federate applications to accept network connections from other federates or any RTI processes.**

To solve this issue, one server process can be made available, such that all federates connect to that server as clients only. That server process must be hosted on a network that is accessible from each participant network. The location may be dependent on a participant's network security policy and control, but must allow client connections from the participant networks. All data transmission, in either direction, then takes place over those connections.

Figure 3-1 illustrates how network connections will be established through the RTI. All data communications between federates are sent over network connections and relayed to the other federates by the forwarding process.

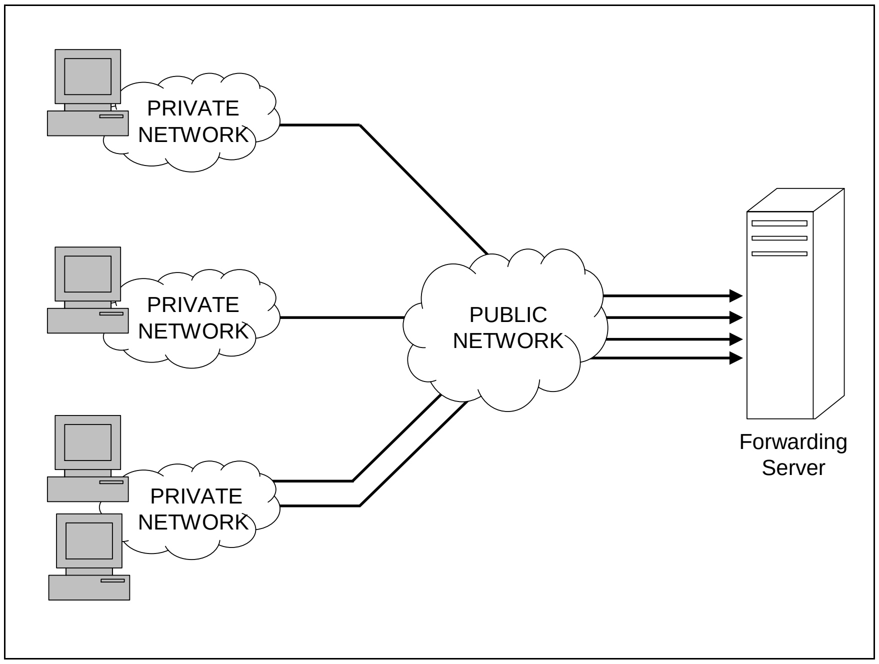


Figure 3-1. Network Topology for Federation Network Connections

Using this topology, the RTI must provide a server that acts as a data forwarder for HLA communications. Each participating federate (or local RTI component) may establish TCP/IP connections to the server only, and the server must accept those connections. Only known assets can connect to the server. Because the federate assets are not permitted to create connections directly among themselves, all data transmissions in either direction must be performed only between the federate and the server, using those established network connections.

3.4 HLA Gateways

In order to satisfy the data communications requirements, any non-HLA assets will need to become HLA compatible. Even existing HLA federates may need adjustments at various levels of implementation to satisfy the AviationSimNet specification. Assets that wish to join an AviationSimNet federation must, at a minimum, be able to either receive information or transfer internal simulation events to external entities. Furthermore, those simulation events must be presented to the federation in accordance with the AviationSimNet FOM [7]. To satisfy these needs, participants can construct "HLA gateways." These gateways relay the asset's internal state to the RTI, and likewise relay RTI events back to the asset as appropriate. Regardless of how much simulation fidelity exists for an asset, it must coordinate all its data communications to other assets via the federate's interfaces with the RTI.

3.5 Federate State

At any given time during the execution, each federate will be in some state that governs what types of activities it can produce or consume. Depending on the type of federate (Synchronized or Unsynchronized), there are six possible states for a federate during a federation execution: Initialization, Declaration, Population, Paused, Unpaused, and Terminated. These six states reflect all of the initialization and runtime states for all AviationSimNet federates, and are illustrated in Figure 3-2. The state model is described in more detail in Section 3.9.

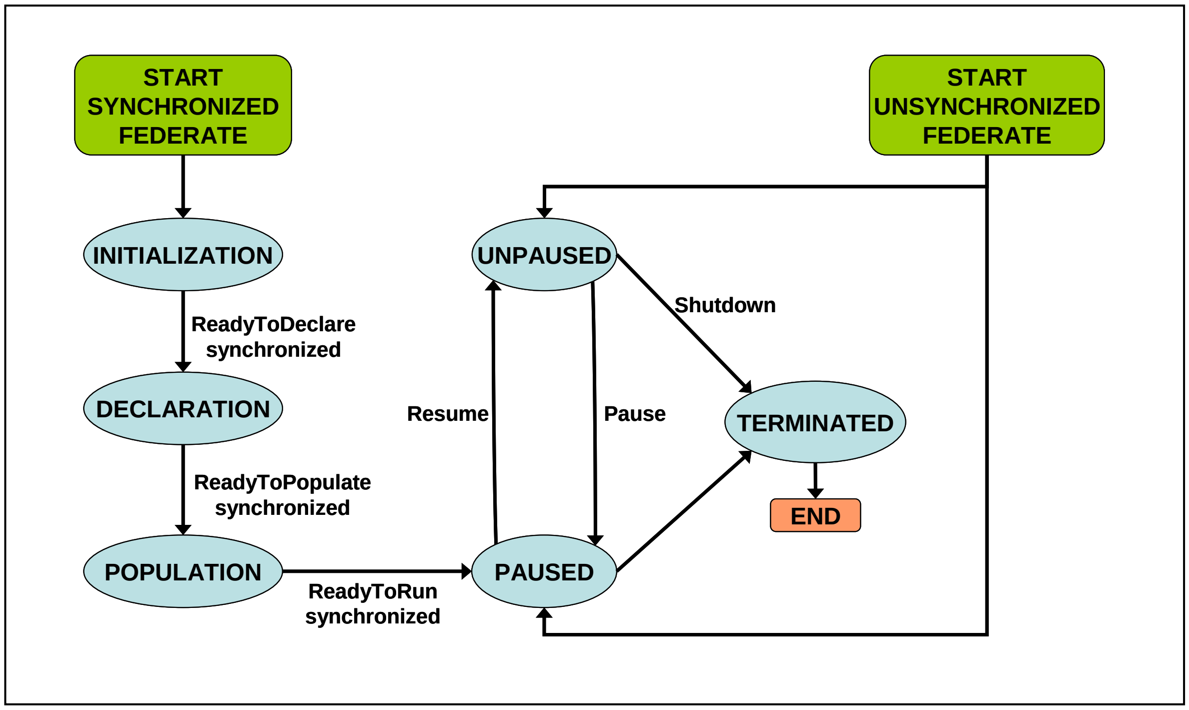

START SYNCHRONIZED FEDERATE
START UNSYNCHRONIZED FEDERATE
INITIALIZATION
ReadyToDeclare synchronized
DECLARATION
ReadyToPopulate synchronized
POPULATION
ReadyToRun synchronized
PAUSED
UNPAUSED
Resume
Pause
Shutdown
TERMINATED
END

Figure 3-2. Federate State Transitions

3.6 Federation Preparation

Prior to any federation execution, all participants must achieve consensus on federation-wide attributes. These attributes are used to associate assets with an intended simulation, and can be RTI-dependent. Primarily, a federation name must be selected and employed by all the participants. All federates must join into the same federation, which is identified during the HLA join command. Participants should adopt a naming convention to ensure all simulations use a unique federation name. Since a single RTI can support multiple simultaneous federations, the federation name is the only distinguishing identifier to match a federate with its intended federation.

Additionally, if any federates are engaged in time management, all participants must agree on an appropriate simulation start time. The start time is the value used to initialize all of the federates' logical clocks. Federates that do not engage in time management or track simulation time can ignore the start time.

REQ 5	**The logical simulation start time shall be approved by all simulation participants.**

Next, the number of expected start-up federates and each of their federate "types" must be known prior to starting the federation. The federate type is a user-supplied name the federate provides to the RTI upon joining. The HLA specification itself does not require unique types, but AviationSimNet uses that value to uniquely identify the joined federates. This step ensures that all federates can be appropriately identified in the simulation. The field specified for the federate type when joining a federation will be used to satisfy this requirement.

REQ 6	**Each federate shall join using a unique federate type to identify itself in the federation.**

REQ 7	**The federate types of all start-up federates shall be known prior to simulation start.**

3.7 Object Model

AviationSimNet supports a single FOM for data communications. The FOM represents the classes of data that can be shared among the federates, giving them a single reference to ensure that data buffers are packaged unambiguously. By conforming to the rules of HLA, all federates must use the same version of the FOM when creating a common federation. Each participant will have access to the AviationSimNet FOM and associated Federation Execution Data (FED). The FED is an HLA input file used by the RTI software to import all the classes of the FOM.

The AviationSimNet FOM is a model maintained independent from this specification. This segregation allows the model to evolve at a schedule separate from these rules and requirements. Users of AviationSimNet are encouraged to review the AviationSimNet FOM for the complete set of supported objects and interactions.

REQ 8	**AviationSimNet FOM versioning shall be explicit with each modification made to the object model.**

REQ 9	**All AviationSimNet federations shall use the same version of the AviationSimNet FOM.**

Full details of the AviationSimNet FOM are not given in this document. The AviationSimNet FOM is a separate document that contains information about the supported objects and interactions available for data exchange. See *AviationSimNet Federation Object Model Version 2.0,* The MITRE Corporation [7] for the AviationSimNet FOM.

The AviationSimNet FOM is evolving and will continue to evolve as needed to incorporate additional ATC capabilities into the AviationSimNet environment and to support the changing needs of its users.

3.7.1 FOM Objects

The FOM contains the definitions of the objects and object attributes available for data communications. The FOM objects are representations of ATC objects that can be modeled in a simulated environment. Aircraft objects in the FOM must have attributes that can properly represent the true trajectory of the flight, including current location in three dimensions, current rate of change, and a timestamp at which those values are correct. The timestamp must be the aircraft publisher's current logical time.

REQ 10	**The FOM shall include an aircraft object, and at least the following aircraft attributes: call sign, aircraft type, current latitude, current longitude, current altitude, current vertical speed, current ground speed, current ground track and timestamp.**

The RTI provides the publishing federate and the discovering federate a common and unique instance handle for each aircraft registered. Federates are encouraged to use that handle as the unique identifier for the aircraft, because the call sign is not guaranteed to be unique. Subscribing federates may use the reflected data from an aircraft update to project aircraft position using dead reckoning.

Note that the FOM will include more ATC type objects and attributes than those listed here.

3.7.2 FOM Interactions

In addition to objects, the AviationSimNet FOM also contains the definitions for interactions representing messages in the simulation. This section details the minimum set of interactions that must be supported by the FOM.

REQ 11	**The FOM shall include interactions for announcing simulation start time (`StartSimTime`), pausing the simulation clock (`Pause`), resuming the simulation clock (`Resume`), and terminating the simulation (`Shutdown`).**

3.7.2.1 StartSimTime Interaction

To ensure that the logical start time is communicated to all federates, it shall be included in a **`StartSimTime`** interaction during federation initialization. Simulation time advancement cannot start until this message event has been published.

REQ 12	**All start-up federates must be joined and have completed subscription activities before the logical simulation start time is announced.**

3.7.2.2 Shutdown Interaction

The **`Shutdown`** interaction is employed to coordinate a graceful federation termination process. It enables a simple means to indicate that the simulation has ended; it should conduct end-of simulation tasks, and no further communications is expected by any other federate. If a start-up federate resigns before this interaction is sent, the federation may continue to run, but in an incomplete state. A federate's early resignation should not interfere with the ability of all other federates to continue with the simulation.

REQ 13	**All federates shall resign from the federation after announcement of the `Shutdown` message.**

3.7.2.3 Pause Interaction

The **`Pause`** interaction is available to provide a means for the simulation to suspend logical time advancement. Any time-aware federates must observe this interaction by halting changes to its local simulation state. Only the Federation Time Keeper shall be allowed to send the Pause Interaction.

REQ 14	**Federates shall not advance logical time after sending or receiving a `Pause` interaction.**

3.7.2.4 Resume Interaction

The **`Resume`** interaction is required to enable a simulation clock to start advancing, both at the start of the simulation, and to un-pause from a suspended mode. Even if there are no federates engaged in time management with the RTI, the **`Resume`** interaction provides a means to indicate to all participants that the simulation is active. Only the Federation Time Keeper shall be allowed to send the Resume Interaction.

REQ 15	**Federates shall advance logical time only after sending or receiving a `Resume` interaction.**

3.7.3 Management Object Model

Management Object Model (MOM) is the standard set of objects and interactions for all FOMs that relate to communicating events for the purpose of managing the federation. The HLA specification includes MOM elements as part of all FOMs.

Pause and Resume interactions define periods of simulation time advancement. If there is a time-regulating federate in the simulation, it may only advance logical time with the RTI after a Resume and before a Pause message. That way, any federate that is not time-constrained can still subscribe to these interactions to know the current status of the simulation clock.

3.8 Federation Roles

The HLA specification leaves the protocols of federation management open to the implementation. These protocols include adopted policies on time management, and startup and shutdown coordination. AviationSimNet explicitly outlines these protocols so that no AviationSimNet federation will ever lack required responsibilities or have duplicated activities. Such protocols are executed through specific federate activities. The responsibilities are identified in two types of federates in the simulation. The **Federation Manager** ensures that initialization and shutdown proceed in an orderly fashion. The **Federation Time Keeper** initializes and controls the advancement of simulation time. There may be any number of federates aside from the Federation Manager and Federation Time Keeper in a federation.

3.8.1 Federation Manager

It is convenient to designate a single federate as having the responsibility of coordinating simulation startup and shutdown activities. AviationSimNet identifies that federate as the Federation Manager. Any federate can be designated to be the Federation Manager for a simulation, but there can be only one per Federation.

The Federation Manager is responsible for ensuring that all start-up federates have joined the simulation before the simulation clock is allowed to start. Furthermore, it coordinates stages of initialization before the clock can be started. These initialization stages are achieved through use of the MOM and HLA synchronization points. The MOM is useful for discovering all other federates when they join the federation. Because all federates join the federation with unique federate types, the Federation Manager can always determine which federates have not yet joined. The Federation Manager must be able to track which start-up federates are joined, regardless of whether they joined prior to or after the Federation Manager.

Additionally, the Federation Manager is responsible for ensuring an orderly termination, and therefore publishes the **`Shutdown`** interaction to indicate that the simulation is complete. All other federates should subscribe to the **`Shutdown`** interaction so that they can receive proper notification to exit from the simulation. The **`Shutdown`** interaction can be triggered by user interaction, or through some failed state observed within the simulation itself.

REQ 16	**Only one federate shall be permitted to announce synchronization points and publish the `Shutdown` interaction.**

3.8.1.1 Synchronization Points

The Federation Manager uses HLA synchronization points to coordinate federation initialization stages. HLA provides synchronization points to allow federates to be notified when some federation-wide condition has been met. Synchronization points are announced to the federation by a single federate. They are achieved by each federate according to the protocol designed by the participants, and once all federates have achieved the same point, the federation is considered to be synchronized. The RTI notifies each federate when synchronization has occurred.

Once the Federation Manager has discovered that all the start-up federates have joined the federation (through subscription to joined federate type names), it announces four synchronization points. The names and purpose of those synchronization points are listed in Table 3-1, AviationSimNet Synchronization Points.

Table 3-1. AviationSimNet Synchronization Points

Synchronization Point	Purpose
`ReadyToDeclare`	Ensures that all start-up federates have joined the simulation before any declaration services are used.
`ReadyToPopulate`	Ensures that all initial publishing and subscribing is complete before any start-up federates send interactions or create objects with the RTI.
`ReadyToRun`	Ensures that all start-up federates have concluded their initialization stages and have received the SimulationStartTime interaction before the simulation clock is started.

Only the Federation Manager is permitted to announce these synchronization points. All other start-up federates must be prepared to accept these announcements. In addition, all start-up federates must achieve each point at the appropriate stage of execution, and acknowledge federation synchronization for each of these points. The Federation Manager will not announce these points until it knows that all start-up federates have joined the federation. The rest of this section refers to the behavior of start-up federates only. Unsynchronized federates ignore these synchronization points, and do not affect the rest of the federation with regard to initialization and shutdown states.

REQ 17	**Synchronization points shall be announced only after all expected start-up federates have joined the federation.**

REQ 18	**All start-up federates must observe the synchronization points labeled `ReadyToDeclare`, `ReadyToPopulate` and `ReadyToRun`.**

3.8.1.1.1 ReadyToDeclare Synchronization Point

The **`ReadyToDeclare`** point is used to define the stage of initialization prior to any federates declaring their interest in publication or subscription (except for the Federation Manager which must subscribe to attributes in the MOM for prompt federate discovery). Each federate must wait for the announcement of **`ReadyToDeclare`**, and achieve that point with the RTI. Once **`ReadyToDeclare`** has been synchronized across the federation, each federate is then permitted to publish and subscribe as necessary. This initialization stage ensures that all start-up federates are joined before any inter-federate dependencies are established.

REQ 19	**Each start-up federate shall await synchronization at `ReadyToDeclare` prior to declaring any publication or subscription activity with the RTI.**

3.8.1.1.2 ReadyToPopulate Synchronization Point

The **`ReadyToPopulate`** point is used to define the stage of initialization prior to any federates creating objects or sending interactions through the RTI. This stage ensures that no simulation event activity is performed before other federates have a chance to declare their interest in those events. Each federate must wait for the announcement of **`ReadyToPopulate`**, and achieve that point with the RTI only after it has completed all its initial publish and subscribe declarations with the RTI. Federates may not create objects or send interactions until **`ReadyToPopulate`** has been synchronized. The Federation Manager ensures that **`ReadyToPopulate`** synchronization occurs only after the federation is synchronized at **`ReadyToDeclare`**. Once the federation has been synchronized at **`ReadyToPopulate`**, the Federation Time Keeper must send the **`StartSimTime`** interaction, notifying other federates of the logical start time of the simulation.

REQ 20	**All start-up federates shall complete their initial publication and subscription declarations before achieving `ReadyToPopulate`.**

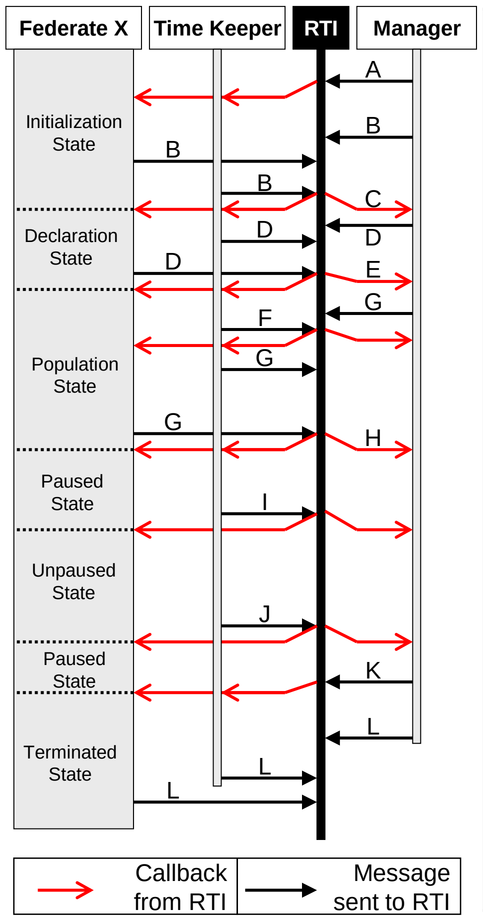
REQ 21	**No start-up federate shall register objects or send interactions until the federation has synchronized at `ReadyToPopulate`.**

3.8.1.1.3 ReadyToRun Synchronization Point

Once each federate is prepared to start running the simulation clock, it must achieve the **`ReadyToRun`** synchronization point. The Federation Manager ensures that **`ReadyToRun`** synchronization occurs only after the **`StartSimTime`** interaction has been sent. Note that AviationSimNet allows object registration and attribute update with the RTI prior to clock start. This is intended to support any necessary cross-federate initialization that should be conducted at the launch of the simulation. The Federation Time Keeper may send the first **`Resume`** interaction only after the federation is synchronized at **`ReadyToRun`**.

REQ 22	**All start-up federates shall achieve `ReadyToRun` only after the federation is synchronized at `ReadyToPopulate` and they have received the `StartSimTime` Interaction.**

An example federation timeline illustrating all the synchronization points is given in Figure 3-3, which shows three federates, the RTI, and some of the messaging between them. Note that the federate state of Federate X is tracked as events and messages transpire. See Section 3.9 for details on state transition.



	Sample Federation Timeline
A	The Manager registers all sync points, which the RTI announces to all other federates
B	Each federate achieves ReadyToDeclare after its internal initialization.
C	The RTI synchronizes ReadyToDeclare, indicating that declaration can begin.
D	Each federate achieves ReadyToPopulate after registering their declarations with the RTI.
E	The RTI synchronizes ReadyToPopulate, indicating that object creation can commence.
F	The Time Keeper sends the StartSimTime message, which the RTI relays to other federates.
G	Each federate achieves ReadyToRun, indicating that it is ready to start the clock.
H	The RTI synchronizes ReadyToRun, indicating that federation is ready to start the clock.
I	The Time Keeper sends the Resume interaction which actually starts the clock.
J	The Time Keeper sends the Pause interaction which pauses the clock.*
K	The Manager sends a Shutdown interaction which the RTI forward to all other federates.
L	Each federate resigns from the federation in the Terminated State.
*****	Step J is optional, and only included here for illustration purposes. The Timekeeper can send any number of pause/resume messages during the simulation.

Figure 3-3. Sample Federation Synchronization

3.8.2 Federation Time Keeper

AviationSimNet is intended to support human-in-the-loop simulations, therefore participating simulations will likely include assets designed for training or human interaction. Such assets should assume the advancement of logical time will proceed at the same rate as system time. Advancing the simulation clock only in real-time greatly simplifies the requirements for observing and managing the time variables in the simulation.

REQ 23	**When simulation time is advancing, it shall only advance at the same rate as real-time.**

Because real-time time advancement is a simple mode of HLA time management, it can be regulated by pausing or resuming a single clock. Managing the AviationSimNet clock is the responsibility of one federate: the Federation Time Keeper. The Time Keeper is responsible for notifying the federation of the simulation start time, and when the simulation clock is running or paused.

REQ 24	**Only the Time Keeper federate shall be permitted to publish the `StartSimTime`, `Pause` and `Resume` interactions.**

Time advancement in the simulation can be temporarily paused and resumed any number of times. This implies that the offset between system time and logical time at the start of the simulation may not be the same as some later point in the simulation.

By assuming that logical time advances at the same rate as the system clock, time-dependent federates are not mandated to invest in a time-constrained policy with the RTI. All that is needed to know about simulation time is the published simulation start time, and when the pause and resume messages are announced. The simulation clock is either advancing or stopped. During initialization, the Federation Time Keeper has the responsibility to announce the simulation start time to other federates.

REQ 25	**The `StartSimTime` interaction shall be sent once after the federation is synchronized at `ReadyToPopulate`.**

Because the simulation is expected to run in real-time, it should only require one federate to be responsible for maintaining that pace. Therefore, for federations that engage in HLA time management services, only the Federation Time Keeper is permitted to be time-regulating with the RTI.

REQ 26	**Only the federate that publishes the `StartSimTime` may be time regulating with the RTI.**

REQ 27	**The FOM shall express all units of time as milliseconds since midnight, 1 January 1970 UT.**

Because the Time Keeper federate has the responsibility to manage simulation time, that federate also has the exclusive privilege of announcing **`Pause`** and **`Resume`** interactions to the federation, in accordance with the simulation clock. Those pause and resume events would likely be triggered by user input.

The Federation Time Keeper federate can be the same asset as the Federation Manager, for the sake of convenience or efficiency, provided there is still only one Federation Time Keeper and one Federation Manager in the federation. It is not necessary to operate both the Federation Manager and the Federation Time Keeper in the same participant location.

3.8.3 All Federates

Any federate in the simulation, whether it is the Federation Manager, the Federation Time Keeper or some other federate, has the ability to join the federation in any order. Any coordination necessary to start assets or applications among the participants is beyond the scope of this document. AviationSimNet assumes there is a way for federate operators to know when a simulation is being readied to execute, and that those communications channels are sufficient to get all of the federates to join within a reasonable time frame. Federates must be allowed to join in any order, and each federate has the authority to create the federation with the RTI. However, the simulation will not proceed until all start-up federates, including the Federation Manager and Federation Time Keeper have each joined the federation. Additonally, start-up federates should suspend declaration and publication activities until it receives announcement of the HLA synchronization points. This is the signal that the Manager Federate has joined, and that all expected start-up federates are also joined.

REQ 28	**Any federate shall have the authority to be the first to create the federation execution.**

REQ 29	**All federates shall be permitted to join the federation in any order.**

Table 3-2 summarizes the capabilities among the various types of AviationSimNet federates.

Table 3-2. Summary of Federate Roles and Responsibilities

Activity	Federation Manager	Federation Time Keeper	Other start-up Federate	Other Unsynchronized Federate
permission to create the federation	Yes	Yes	Yes	Yes
join the federation	Yes	Yes	Yes	Yes
time-regulating	No	Yes	No	No
time-constrained	Possibly	No	Possibly	Possibly
registers synchronization points	Yes	No	No	No
achieves synchronization points	Yes	Yes	Yes	No
publishes **`Pause`** and **`Resume`** interactions	No	Yes	No	No
subscribes **`Pause`** and **`Resume`** interactions	Yes	No	Yes	Yes
publishes **`StartSimTime`** interaction	No	Yes	No	No
subscribes **`StartSimTime`** interaction	Yes	No	Yes	Yes
publishes **`Shutdown`** interaction	Yes	No	No	No

subscribes **`Shutdown`** interaction	No	Yes	Yes	Yes

3.9 State Transitions

This section summarizes explicit state transitions for any federate. Refer to Figure 3-2 for an illustration of the entire lifecycle of an AviationSimNet federate.

3.9.1 Initialization State

Every start-up federate will start in its own initialization state, where it can perform any necessary functions in preparation for coordinating with other federates. Once the federate has properly joined the federation, it must achieve and await synchronization on **`ReadyToDeclare`**, as described in the previous section.

3.9.2 Declaration State

Once the federation is synchronized at **`ReadyToDeclare`**, each start-up federate enters the Declaration state. In this state, the federate is permitted to register all initial publication and subscription interests with the RTI, as described in the previous section. Once those declarations are complete, the federate must achieve and await synchronization on **`ReadyToPopulate`**, as described in the previous section.

3.9.3 Population State

Once the federation is synchronized at **`ReadyToPopulate`**, each start-up federate enters the Population state. In this state, all federates can start registering instances, updating attributes and sending interactions, with the assurance that interested federates have completed their subscription interests in those messages from the previous state. While in this state, the Time Keeper will publish the **`StartSimTime`** interaction to the federation. Once the federate is ready to start running the simulation, and has received the StartSimTime interaction, it must achieve and await synchronization on **`ReadyToRun`**.

3.9.4 Paused State

Once the federation has synchronized at **`ReadyToRun`**, each start-up federate will be in the Paused state. No simulation activity should be performed during this state. In this state, the federate must be able to respond to two types of events. If it receives (or the Federation Time Keeper sends) a **`Shutdown`** interaction, it should immediately enter the Terminate state. If it sends or receives a **`Resume`** interaction, it should enter the Unpaused state. The

Federation Time Keeper should not request any time advances from the RTI during this state, nor is it permitted to send the **`Pause`** interaction during this state.

Unsynchronized federates do not participate in AviationSimNet synchronization activities, and so could enter the simulation directly in a Paused state. Unsynchronized federates are still responsible for subscribing to the Pause, Resume and Shutdown interactions.

3.9.5 Unpaused State

The Unpaused state is the busiest state for any federate. This is when the federation is active, and the simulation clock is enabled. In this state, the federate must be able to respond to two types of events. It must respond to the **`Shutdown`** interaction as it did in the previous state. If it receives (or the Federation Time Keeper sends) a **`Pause`** interaction, it should enter the Paused state. The Federation Time Keeper federate is not permitted to send the **`Resume`** interaction during this state.

Unsynchronized federates do not participate in AviationSimNet synchronization activities, and so could enter the simulation directly in an Unpaused state. There is no current protocol to instruct an unsynchronized federate whether the simulation clock is paused or unpaused when it joins.

3.9.6 Terminate State

The Terminate state is the final state a federate will enter before resigning from the federation. Any federate will enter this state if it sends or receives the **`Shutdown`** interaction, or if it encounters a fatal internal error that forces it to discontinue participation in the federation execution. In this state, the federate should cease all simulation activity, and resign the federation. All federates should attempt to exit the federation gracefully and explicitly.

3.10 HLA Services

3.10.1 Federation Management

AviationSimNet supports all the necessary HLA services associated with federation management, such as creating, joining and resigning federations. It also supports the use of synchronization points as discussed above. AviationSimNet does not enforce any protocols associated with federation save and restore activities.

3.10.2 Time Management

Because the simulation clock only advances in real-time, a minimal amount of HLA time management services are employed by AviationSimNet. Only the Federation Time Keeper is permitted to be time-regulating, but time-regulation is optional, and not required if there are no time-constrained federates. Federates that are time-constrained should slave their logical time to the Time Keeper's advancement with the RTI. AviationSimNet does not require a specific granularity of time regulation updates from the Time Keeper. The Time Keeper may request updates as often as appropriate for the specific simulation, trading performance for precision. The requirements for time management have otherwise already been covered in previous sections.

Furthermore, an AviationSimNet simulation can be conducted without the use of any HLA time management services. The Federation Time Keeper can simply publish the StartSimTime interaction and the Resume and Pause interactions to coincide with time advancement.

3.10.3 Declaration Management

AviationSimNet enforces all initial declaration activities to be performed during a finite initialization stage of the simulation, as described in previous sections. However additional publication and subscription, as well as any alterations or cancellations of those initial declarations can be made at any time during the simulation.

3.10.4 Data Management

Because AviationSimNet is intended to support data communications over a network, it is subject to the difficulties of byte-order differences across computer architectures (i.e., the native address ordering of multi-byte data for a particular CPU family). So-called "little-endian" representations of binary data are not stored the same way as "big-endian" representations. This inequality only comes to bear when CPUs of different byte order attempt to exchange binary data. Because all of the numeric data in the AviationSimNet FOM are specified as binary buffers, they must further be specified with a particular byte order to avoid ambiguity.

Networks protocol communications have adopted "big-endian" representations for all data that is transported over a network. This is also called "network-byte-order." AviationSimNet continues in this tradition by specifying that all binary data buffers in the FOM are to be encoded in Network-byte-order.

REQ 30	**All binary data buffers (update attributes and interaction parameters) shall be represented in network byte order.**

String termination is another common ambiguity in storing ASCII text information in software data buffers. C and C++ applications usually rely on null-terminated string buffers to identify the string length. Since HLA attribute and parameter values provide a buffer length, the null terminator is not necessary. Therefore the byte length of text buffers need only be the length of the string it contains.

REQ 31	**All text-based buffers (update attributes and interaction parameters) shall be represented in ASCII format, and exclude the null terminator character (0x00) as the last byte in the buffer.**

3.10.5 Data Distribution Management

AviationSimNet does not enforce any policies with respect to the Data Distribution Management (DDM) function in the HLA RTI.

3.10.6 Ownership Management

The HLA specification supports the ability to distribute ownership of an object instance among multiple publishing federates. This feature is inherent in the fact that federates own individual attributes and not object instances. For the sake of simplifying object ownership in AviationSimNet, the full set of aircraft attributes (including all inherited attributes) is limited to a single owner at any given moment. This allows federates to "own" entire aircraft objects without the burden of tracking which attributes are locally owned or not owned.

REQ 32	**The federate that owns the `privilegeToDelete` attribute of an aircraft object shall simultaneously own all the available attributes of that aircraft.**

The consequence of this requirement in ownership management is that ownership transfer of any aircraft attributes must be conducted for all attributes. Both the owning federate and prospective owning federate negotiating an ownership transfer can each assume that the set of attributes being transferred is the full set of attributes for the given aircraft object.

Section 4

Audio Network

4.1 Introduction

The audio communications for an AviationSimNet simulation must provide for exchanging audio information among the human participants at various locations. These communications correlate to audible events, such as pilot-controller communications, that transpire in a simulated ATC environment. In order to support a distributed simulation over a network composed of heterogeneous participants, this specification seeks to adopt an industry standard already devoted to providing these capabilities. The standard selected is the IEEE Standard for Distributed Interactive Simulation—Application Protocols, sponsored by the Distributed Interactive Simulation Committee of the Institute of Electrical and Electronics Engineers (IEEE), versions 1278.1-1995 and 1278.1a-1998 [8, 9]. Since IEEE Std 1278.1a-1998 is a supplement to IEEE Std 1278.1-1995, it provides for additional features that may not be required by a particular AviationSimNet simulation. As such, while IEEE Std 1278.1a-1998 is not required for participation in an AviationSimNet simulation, it is highly recommended.

As described in IEEE Std 1278.1a-1998, the standard "defines the format and semantics of data messages, also known as protocol data units (PDUs), that are exchanged between simulation applications and simulation management." The standard "also specifies the communication services to be used with each of the PDUs."

An additional document, authored by the Institute for Simulation and Training of the University of Central Florida, and available from the Simulation Interoperability Standards Organization, is required for use with IEEE Std 1278.1-1995 and IEEE Std 1278.1a-1998. This document is entitled *Enumeration and Bit Encoded Values for Use with Protocols for Distributed Interactive Simulation Applications*.

By adopting this standard, AviationSimNet leverages much work previously completed in the networked audio domain. This section continues with requirements for how to implement the IEEE standards under the rules of AviationSimNet. It is assumed that the reader is familiar with the above referenced specifications.

4.2 Network Topology and Access

The IEEE Std 1278.1-1995 and IEEE Std 1278.1a-1998 specification do not define how data is exchanged at the network layer. Therefore, implementations of the standard may differ from one another in their networking capabilities, while still satisfying the specification. This does not imply that the two implementations are compatible. The AviationSimNet specification provides guidance on an implementation's behavior on the network.

Because AviationSimNet must support multiple participants, on multiple, disparate networks, the audio network must be able to support communications on a public wide area network.

REQ 33	**The audio communications system shall, at a minimum, comply with the IEEE Std 1278.1-1995, and shall support communications on a wide area network.**

Furthermore, all network communications associated with audio networking must be capable of traversing the firewalls protecting AviationSimNet participant's internal networks. This approach cannot assume that the security policies in place will allow inbound network connections to pass through the firewall from an untrusted network.

REQ 34	**The IEEE Std 1278.1-1995 or IEEE Std 1278.1a-1998 implementation shall support a network topology that does not require network connections directly between participant networks.**

To solve this issue for AviationSimNet, a forwarding service shall be established to assume the responsibility of accepting inbound connections from all participant networks. The host providing that service must be accessible from all participant networks and will act as a forwarder for audio data. See Figure 3-1 for a representative network topology.

REQ 35	**There shall be a single host accessible to all audio assets to act as a forwarder of audio data.**

As such, the forwarding service will accept incoming connections from participant networks and will forward data sent from any connected asset to all other connected assets.

REQ 36	**The audio data forwarding system shall send forth incoming communications from any audio asset to all other active audio assets in the simulation.**

4.3 Audio Preparation

Prior to any AviationSimNet simulation exercise, all participants must achieve consensus on select PDU attributes. These attributes, defined in the IEEE standard are used to associate audio entities with a particular simulation. The acceptable values for these attributes are defined in the document *Enumeration and Bit Encoded Values for Use with Protocols for Distributed Interactive Simulation Applications.* At the very least, all participants must use the same values for the protocol version and exercise ID attributes.

REQ 37	**Common protocol version and exercise ID shall be established prior to simulation execution.**

In order to share audio communications data, the set of Site ID, Application ID, and Entity ID attributes must be chosen in such a manner as to guarantee uniqueness of the set. This shall be accomplished through manual participant coordination.

REQ 38	**The set of Site ID, Application ID, and Entity ID attributes shall be unique to every entity in the simulation.**

PDU Type, Protocol Family, and Entity Type must be agreed on between participants, and should reflect the nature of the entity properly. Generally, all values required by the IEEE standards should be set in a manner that reflects the nature of the system being simulated.

Optional fields, such as antenna positions, will need to be coordinated prior to commencement of a simulation exercise.

REQ 39	**Values for optional attributes shall be established prior to execution.**

For implementations of the IEEE Std 1278.1-1995 and IEEE Std 1278.1a-1998 standards that are unable to convert between modulation schemes, encoding schemes, and sample rates, upon receipt of data will need to manually coordinate the proper values for interoperability.

REQ 40	**Values for data format attributes shall be established prior to execution.**

Section 5

Security

The AviationSimNet specification relies on the availability of network connectivity among the involved participants. When using the public Internet, that connectivity is not under the security control of any of the participants in an AviationSimNet simulation, which can give rise to a set of security concerns.

AviationSimNet attempts to address these concerns, while steering clear of mandating any particular networking solution. As described in Sections 3 and 4, the burden of accepting TCP/IP connections from remote sources over the Internet is limited to a single host. This is true for both data and voice communications. This boundary server can be established on a network that is isolated from the assets that are accessing it. This requirement makes no guarantees on the trustworthiness of the data being sent over those connections.

AviationSimNet makes no attempt to protect the use of sensitive data over public networks. In general, protecting, encrypting or authenticating the data and the networks being used by AviationSimNet simulations is outside the scope of this specification.

Section 6

Performance

Because AviationSimNet supports distributed data and voice communications over public, wide-area networks, there are in general no controls over the quality of service on those networks. In fact, the Internet is well known to have widely varying latency and throughput, depending on time of day, and location of packet source and destination. Therefore, AviationSimNet cannot specify or require any overall guarantee of performance when using the Internet or other network beyond the administrative control of the participants.

Performance is also clearly dependent on the simulation scenario being conducted through AviationSimNet. The scenario defines how many voice radios are being used, how many objects are being published, the payload size of attribute updates and interaction events, and the frequency of those updates.

MITRE has conducted performance testing to determine required bandwidth under various configurations of these variables. Those studies revealed that even small bandwidth connections can accommodate typical studies using dozens of aircraft target updates per second, and up to four or five radios. The forthcoming *AviationSimNet Integration Guide* [10] will have more details on this topic.

List of References

1. Maginnis, F. X., et al., November 1992, *FAA National Simulation Capability Architecture – Final Report*, MTR 92W000194, The MITRE Corporation, McLean, VA.
2. Dimeo, Karen, et al., January 2002, *Air-Ground Integration Experiment (AGIE)*, Report No. CT-TN02/06. FAA William J. Hughes Technical Center, Atlantic City International Airport, NJ. http://www.hf.faa.gov/docs/508/docs/wjhtc/tn0206.pdf.
3. Lehmer, R., and S. Malsom, August 2004, *Distributed System Architecture in VAST-RT for Real-Time Airspace Simulation*, AIAA Modeling and Simulation Technologies Conference, Providence, Rhode Island, AIAA-2004-5436.
4. Defense Modeling and Simulation Office (DMSO), 1997, *HLA Interface Specification Version 1.3.*
5. Defense Modeling and Simulation Office (DMSO), 1997, *Object Model Template Specification (Version 1.3.)*
6. Defense Modeling and Simulation Office (DMSO), 1997, *HLA Rules (Version 1.3.)*
7. The MITRE Corporation, 2005, *AviationSimNet Federation Object Model Version 2.0. http://www.aviationsimnet.com.*
8. Institute of Electrical and Electronics Engineers (IEEE), 1995, *Standard for Distributed Interactive Simulation – Application Protocols, IEEE STD 1278.1.*
9. Institute of Electrical and Electronics Engineers (IEEE), 1998, *Standard for Distributed Interactive Simulation – Application Protocols, IEEE STD 1278.1a.*
10. The MITRE Corporation, 2005, *AviationSimNet Integration Guide Version 1.0. http://www.aviationsimnet.com. [forthcoming]*

Glossary

ATC	Air Traffic Control
ALPA	Airline Pilot Association
ATM	Air Traffic Management
CAAR	Center for Applied ATM Research
CAASD	Center for Advanced Aviation System Development
DDM	Data Distribution Management
DOD	Department of Defense
ERAU	Embry-Riddle Aeronautical University
FAA	Federal Aviation Administration
FED	Federation Execution Data
FOM	Federation Object Model
HITL	Human-in-the-Loop
HLA	High Level Architecture
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
MOM	Management Object Model
PDU	Protocol Data Units
RTI	Runtime Infrastructure
UPS	United Parcel Service
VAMS	Virtual Aerospace Modeling and Simulation
VAST-RT	Virtual Airspace Simulation Technology Real-Time
WJHTC	William J. Hughes Technical Center