

Development of the Artemis Distributed Simulation FOMs

Edwin Z. Crues, Ph.D.
NASA Johnson Space Center
2101 NASA Parkway
Houston, TX 77058
edwin.z.crues@nasa.gov

Paige Whittington
NASA Johnson Space Center
2101 NASA Parkway
Houston, TX 77058
paige.a.whittington@nasa.gov

Keaton Dodd
CACI International
2100 Space Park Drive
Suite 210, Houston, TX 77058
keaton.c.dodd@nasa.gov

Keywords: HLA, SpaceFOM, FOM, Artemis

ABSTRACT: *The National Aeronautics and Space Administration (NASA) is formulating and developing the Artemis Program, a collaboration with domestic commercial and international partners that will establish a long term human presence on the Moon and extend human exploration beyond the Earth-Moon system ahead of exploring Mars. These Artemis partners are developing a portfolio of space and surface systems to support human missions to the lunar surface and beyond. The Artemis systems will provide the mobility, habitation, and logistics infrastructure that will support human exploration and foster robust scientific investigations. Each partner will contribute one or more elements to the Artemis Program with NASA having the overarching responsibility for defining the Artemis architecture and guiding the integration of this complex system of space systems. To successfully accomplish this audacious task, NASA will rely on the development and execution of many complex models and simulations. Many of these simulations will be provided by the Artemis partners. While each of these simulations will provide important insight into the characteristics and performance of an associated system, individually they will not provide insight into the integrated performance of the architecture and the system of systems working in concert to execute a given Artemis mission. To address this need, NASA is developing a distributed simulation capability called the Artemis Distributed Simulation (ADS). ADS's distributed nature supports the complex aggregation of constituent Artemis element simulations. Artemis partner simulations will be able to join into an ADS-based distributed simulation and interact with other Artemis element simulations while limiting the exposure of proprietary designs and data. ADS is defining a distributed simulation capability built on international simulation interoperability standards, specifically the High Level Architecture (HLA) and the Space Reference Federation Object Model (SpaceFOM). While HLA and SpaceFOM provide the substantive necessary technology basis for ADS, additional common datatypes, message definitions, and execution protocols are required. These extensions constitute the ADS Federation Object Model (FOM). This paper describes the fundamental architectural elements of ADS and the FOM extensions needed to support the complex nature of the Artemis Program. This includes the examination of the ADS FOM modules, ADS base datatypes, ADS SpaceFOM Object Class extensions, new ADS Object Classes, and new ADS Interaction Classes.*

1 Introduction

The National Aeronautics and Space Administration (NASA) of the United States of America (US) is developing and executing a new program called Artemis that will establish a long term human presence on the Moon and extend human exploration beyond the Earth-Moon system ahead of exploring Mars. Artemis is a multi-agency, multi-corporation, multi-national program with a diverse set of key elements provided by a variety of partners. An overview to the Artemis Program is provided in Section 2. NASA has the responsibility for establishing and maintaining the Artemis architecture, defining Artemis missions, integrating the partner provided elements, and insuring the overall successful execution of Artemis missions.

*U.S. Government work not protected by U.S. copyright

As with all space flight endeavors, models and simulations will be key to the design, development, analysis, integration, and operation of every constituent Artemis element. This is the traditional approach to creating space systems. However, Artemis will be a complex system of space systems that have to work together seamlessly to execute a variety of exploration missions. As a result, it is impractical to create one huge monolithic simulation to support the Artemis analysis, integration, and mission support needs. To address this challenge, the Artemis program is developing the Artemis Distributed Simulation (ADS) architecture to utilize the authoritative element simulations developed by Artemis partners in any number of coordinated distributed simulations representing the constituent elements of particular Artemis missions [1].

ADS is built on top of commonly used and well tested simulation interoperability standards: the High Level Architecture (HLA) and the Space Reference Federation Object Model (SpaceFOM) (see Section 3). While these standards provide the substantive necessary technology basis for ADS, they are not sufficient for *a priori* interoperability. HLA provides the core simulation interoperability infrastructure and services for effective and efficient distributed simulation execution (see Section 3.1). The SpaceFOM provides additional execution control, time representations, coordinate frame standards, and common base datatypes needed for generalized space systems representations (see Section 3.2). It is expected that distributed simulation systems that adopt HLA and SpaceFOM will have additional needs to extend existing HLA/SpaceFOM capabilities and services and create whole new object models. In fact, the SpaceFOM is designed to be extended.

ADS is being developed and maintained by the NASA Exploration Systems Simulations (NExSyS) team at NASA’s Johnson Space Center in support of NASA’s Exploration Systems Development Mission Directorate’s (ESDMD) Strategy and Architecture Office (SAO). While developing the initial ADS architecture, the ADS team determined they would need to develop ADS specific extensions and additions to the core SpaceFOM provided capabilities. Starting with Section 4, this paper describes these extensions and additions that form the ADS FOM. This includes the examination of the ADS FOM modules, ADS base data types, ADS SpaceFOM Object Class extensions, new ADS Object Classes, and new ADS Interaction Classes. The paper concludes with some general observations and recommendations and provides an outline of future work associated with the ADS FOM.

2 The Artemis Program

It has been over fifty years since humans have placed new footprints on the surface of the Moon with the Apollo 17 mission to Taurus-Littrow in December 1972. NASA is now developing and executing a new program called Artemis that will establish a long term human presence on the Moon and extend human exploration beyond the Earth-Moon system ahead of exploring Mars. Artemis is a multi-agency, multi-corporation, multi-national program with a diverse set of key elements provided by a variety of partners. In an effort to promote the cooperative and peaceful exploration of space within the Artemis Program, NASA established the Artemis Accords in 2020, initially including 7 other signatory nations [2]. As of the writing of this paper, the Artemis signatories number 52. The Artemis Accords provide a common set of principles to enhance the governance of the civil exploration and use of outer space.

In the Artemis Program, NASA has the responsibility for establishing and maintaining the Artemis architecture, defining Artemis missions, integrating the partner provided elements, and insuring the overall successful execution of Artemis missions. NASA’s Moon to Mars Architecture can be explored in detail on NASA’s Artemis website [3]. This includes an Executive Overview [4] and an Architecture Definition Document [5]. The current 2024 Artemis Architecture is composed of many space, ground, and planetary surface systems that will work together to achieve prospective Artemis missions (see Figure 1).

Most of the Artemis elements will be developed and delivered by the Artemis domestic commercial and international agency partners. NASA will be responsible for developing the Artemis Architecture in coordination with its partners, defining the missions, integrating the mission elements, and managing the overall mission operations. The Artemis partners will design, develop, and deliver elements to execute the missions. Along with the elements, the partners will provide the authoritative data that describe their systems including high fidelity models and digital simulations of the systems.

Traditionally, these simulations would be used for standalone analysis of the systems. The resulting data products and associated analysis would be used in a systems engineering process to assess the integrated performance of the systems in their mission. With the emergence of distributed simulation technologies, these

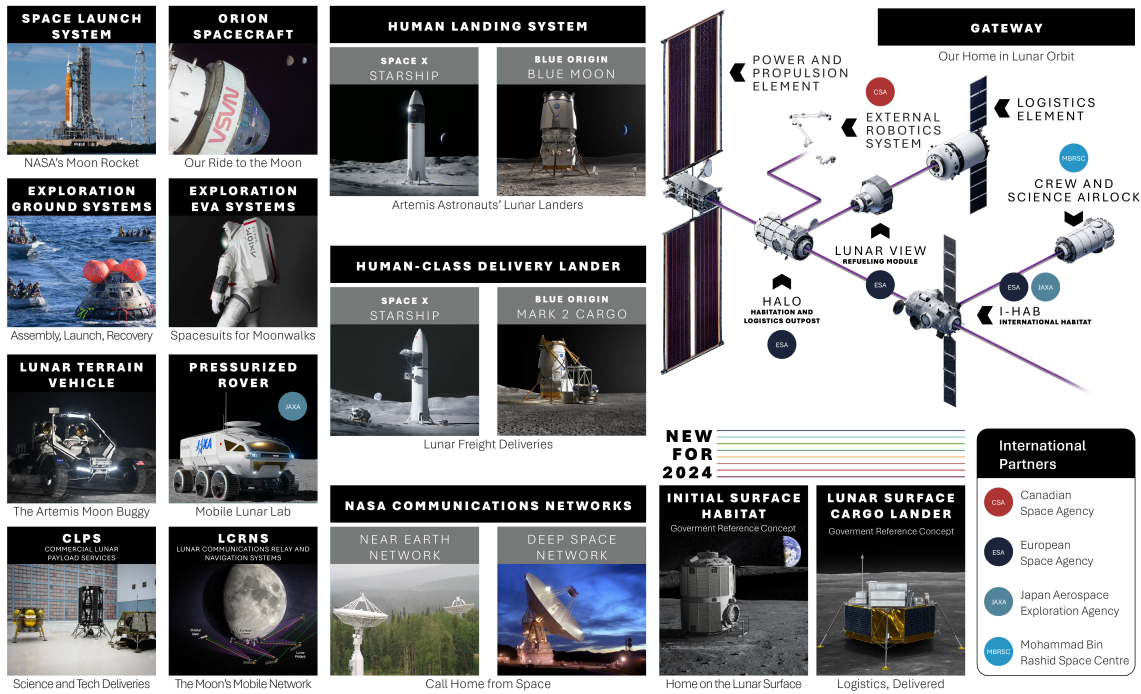


Figure 1: Artemis Elements in Current Architecture[5]

authoritative simulations can be utilized in collaborative execution to provide an integrated analysis of the complex system of space systems. This permits us to observe the emergent behavior of these systems when operating in concert with the other mission elements in the same digital environment. This is something that is difficult if not impossible to do in the traditional systems engineering process using standalone simulation. The key to this approach is the availability of commonly used and well tested simulation interoperability standards.

3 Simulation Interoperability Standards

At this point in the discussion, a reader might ask - What is a Simulation Interoperability Standard and why is it important to ADS? A short definition of simulation interoperability might be the ability of multiple distinct digital simulations running on distributed network connected computers to exchange data and coordinate execution with the intent of simulating a system of systems. This requires a coordinated set of rules and guidelines along with well defined data exchange definitions and execution control protocols. These are usually codified in a set of documents and realized through a common software middleware framework.

Many simulation interoperability solutions have been developed over the years but not all of those were developed formally, published, and recognized as Standards. Standards, by definition, are developed, managed, deployed, and maintained by a recognized technical organization with demonstrated expertise in the area of the standard. This is important because the developing organization will usually ensure the product has input from a diverse set of potential users and that a formal, rigorous, and unbiased standards development process is followed. In addition, the standards development team usually consists of recognized experts in the area of the standard.

The Simulation Interoperability Standards Organization (SISO) was created specifically for the purpose of investigating, developing, managing, deploying, and maintaining simulation interoperability standards [6]. Along with the Institute of Electrical and Electronics Engineers (IEEE), SISO developed the simulation interoperability standard called the High Level Architecture (HLA). SISO also developed the Space Reference Federation Object Model (SpaceFOM) standard as an extension to HLA for use in space systems distributed simulations. ADS

has HLA and SpaceFOM as the foundation of its distributed simulation framework.

3.1 High Level Architecture

HLA is developed and maintained by SISO and is published by the IEEE as standard IEEE-1516 and titled: *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)*. The current version of the HLA standard is IEEE 1516-2010, also known as HLA Evolved.¹ HLA provides a well developed, tested, maintained, and widely used distributed simulation framework with a number of reliable commercially available implementations. The HLA standard outlines the method by which individual simulations, called federates, communicate with each other via the Runtime Infrastructure (RTI) by using pre-defined data exchange definitions called Federation Object Models (FOMs). A set of federates, the RTI, and the required FOMs together make up a single federation.

The HLA standard is published in 3 volumes: Framework and Rules [7]; Federate Interface Specification [8]; and Object Model Template (OMT) Specification [9]. The HLA *Framework and Rules* document defines HLA, its components, and the rules that outline the responsibilities of HLA federates and federations to ensure a consistent implementation. The *Federate Interface Specification* document defines the interface between federates and the underlying software services that support interfederate communication in a distributed simulation domain. The *Object Model Template (OMT) Specification* document defines the format and syntax of HLA object models.

3.1.1 Data Exchange

Data can be exchanged between federates using HLA in the form of objects or interactions. In general, objects persist over time and typically have one or more attributes that describe it. Each attribute of an object can be owned by any joined federate, but only the federate with the object attribute ownership can publish, or send, an attribute update. Additionally, any joined federate can subscribe to an object or attribute in order to receive the corresponding published objects or attributes. Unlike objects, interactions are used to represent something instantaneous or something that only exists for a finite time. Interactions contain parameters and utilize a similar publish and subscribe schema in order to be sent between joined federates. Declaration Management (DM) and Data Distribution Management (DDM) services, as defined in Section 3.1.2, are utilized by joined federates to declare, publish, and subscribe to objects or interaction. Additional details on the management and ownership of objects, attributes, and interactions can be found in the *Federate Interface Specification* document [8].

3.1.2 Management Services

The HLA framework is formulated into a collection of constituent Management Services: Federation, Declaration, Object, Ownership, Time, and Data Distribution. These services are defined in detail in the *Federate Interface Specification* document [8]; but, to help the reader, brief definitions are provided here:

Federation Management Services: refer to the creation, dynamic control, modification, and deletion of a federation execution.

Declaration Management Services (DM): refer to services to declare the intent to generate information.

Object Management Services: deal with the registration, modification, and deletion of object instances and the sending and receipt of interactions.

Ownership Management Services: is used by joined federates and the RTI to transfer ownership of instance attributes among joined federates.

Time Management Services: provide a federation execution with the means to order the delivery of messages throughout the federation execution.

Data Distribution Management Services (DDM): used by joined federates to reduce both the transmission and the reception of irrelevant data.

¹A new HLA Version 4 is expected to be released even before this paper is published.

3.1.3 FOM Modules

Key to the Declaration and Object Management Services are the definitions of the data to be exchanged in an HLA-based federation execution. The format of the definition of these datatypes are covered in the *Object Model Template (OMT) Specification* document [9]. The definitions are stored in eXtensible Markup Language (XML) formatted files referred to as Federation Object Model (FOM) files. In early versions of HLA, all data definitions were collected into a single FOM file. Starting with HLA Evolved (IEEE 1516-2010), a FOM can be broken up into separate but dependent FOM files called FOM Modules. The focus of this paper is to describe the FOM Module extensions developed for ADS.

3.2 SpaceFOM

The SpaceFOM defines a collection of HLA-compliant data constructs, modeling standards, and execution control process standards that support interoperability between simulations in the space domain [10, 11]. It is designed to link simulations of discrete physical entities into distributed collaborative simulations of complex space related systems. The SpaceFOM provides the following:

- Definitions of specific roles and responsibilities of federates within a federation execution;
- Definitions of common datatypes useful in the space domain;
- Definitions of common time lines and time scales needed for time homogeneity;
- Definitions for specific time-stepped focused time management approaches;
- A flexible positioning system, using reference frames for locating arbitrary bodies in space;
- A naming convention for operational reference frames;
- Support for physical entities (e.g., space vehicles and astronauts);
- Support for physical interfaces (e.g., docking ports and sensor locations);
- A synchronized execution control strategy and framework;
- Rules defining the requirements for SpaceFOM compliance;
- A core base set of FOM modules needed for a SpaceFOM-compliant federation execution.

ADS is a SpaceFOM-compliant framework. Simply speaking, this means that ADS adheres to the SpaceFOM rules, roles, and responsibilities while building off of the core SpaceFOM datatypes, Object Classes, and Interaction Classes. The FOM Module level dependencies can be seen in Figure 2. The SpaceFOM Object Class dependencies can be seen in Figure 3. The key object dependencies are on the SpaceFOM *PhysicalEntity*, *DynamicalEntity*, and *PhysicalInterface*. However, there is a deeper dependency on the base SpaceFOM datatypes not shown in the diagrams but discussed in Section 4.

4 Artemis Distributed Simulation

The general purpose of ADS is to create an integration framework for simulation interoperability based on openly available international standards that supports the development and analysis of NASA’s Artemis Program architecture. This effort includes mission analysis, testing, integration, verification, training, and mission support [1]. ADS enables end-to-end Artemis mission analysis using authoritative partner element simulations while providing access to NASA developed element simulations and NASA test, integration, and training facilities.

Several critical elements of the Artemis architecture are already integrated into the ADS framework with the ability to incorporate additional elements as they are added to the architecture or as needed to support studies. Large elements, such as rovers, lander, habitats, and Extravehicular Activity (EVA) suits, are split into individual simulations (federates) in which specific systems and dynamics are simulated. Smaller assets, such

as transport containers, samples, tools, and transferable resources are implemented as objects in the federation where their type, status, and location can be tracked as they are moved around in the federation.

ADS is built on a foundation of HLA and SpaceFOM simulation interoperability standards discussed in Sections 3.1 and 3.2. While these standards are necessary for ADS, they are not sufficient to meet the data exchange needs. Fortunately, HLA and SpaceFOM are designed for extensibility which ADS leverages in the development of ADS specific FOM modules.

The ADS FOM could be specified in a single ADS FOM Module; however, the ADS development team chose to follow the FOM Module pattern of the SpaceFOM and segment FOM specifications into roughly hierarchical modules: Datatypes, Environment, Systems, Graphics, Astronaut, Lander, Rover, etc. These FOM modules contain OMT-compliant ADS-specific definitions of additional datatypes, interaction classes, and object classes.

The ADS FOM Modules are not independent of one another. There is a general hierarchical dependence starting with the SpaceFOM Modules and then the ADS Datatypes FOM Module. The dependency relationships between these FOM Modules can be seen in Figure 2.

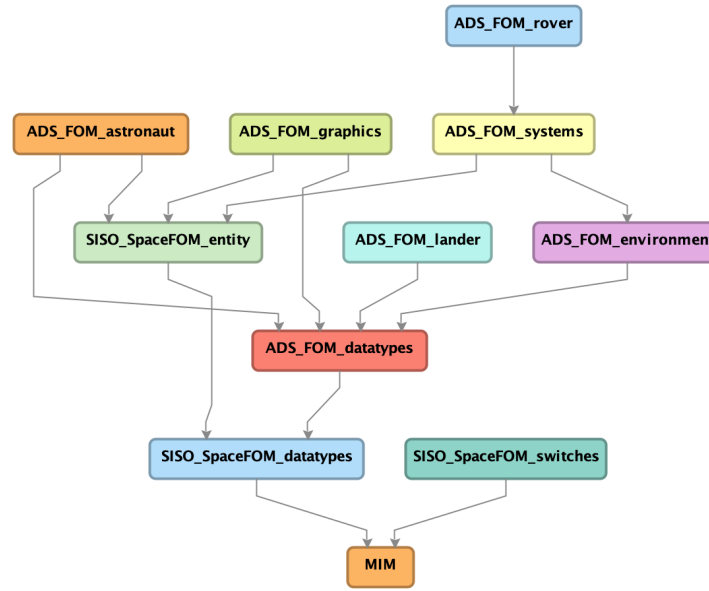


Figure 2: ADS FOM Module Dependencies

This section of the paper will discuss the general purpose and content of each of the principal constituent ADS FOM Modules. The principal ADS FOM Modules are contained within a collection of OMT-compliant XML files:

ADS_FOM_datatypes.xml	Additional base datatypes.
ADS_FOM_environment.xml	Datatypes associated with environmental models.
ADS_FOM_systems.xml	Datatypes associated with generalized space systems.
ADS_FOM_graphics.xml	Datatypes used for driving graphics systems.
ADS_FOM_astronaut.xml	Datatypes used for humans operating in a space environment.
ADS_FOM_landers.xml	Datatypes used by generalized planetary landing systems.
ADS_FOM_rover.xml	Datatypes used by generalized planetary surface mobility systems.

Each of the principal ADS FOM Modules is discussed in the following subsections in this section. The modules define datatypes, HLA Object Classes, and HLA Interaction Classes. Not all modules define all of these HLA constructs. In general, the lower level modules define base datatypes that are either extended in higher level datatypes or included in Object Classes or Interaction Classes. The overall ADS FOM Object Class relationships can be seen in Figure 3. At present there are only a few ADS FOM Interaction Class definitions which are shown in Figure 4, all currently defined in the Environment module. The detailed content of these modules are provided in the Appendices for reference and can be inspected independently of the reading of this paper.

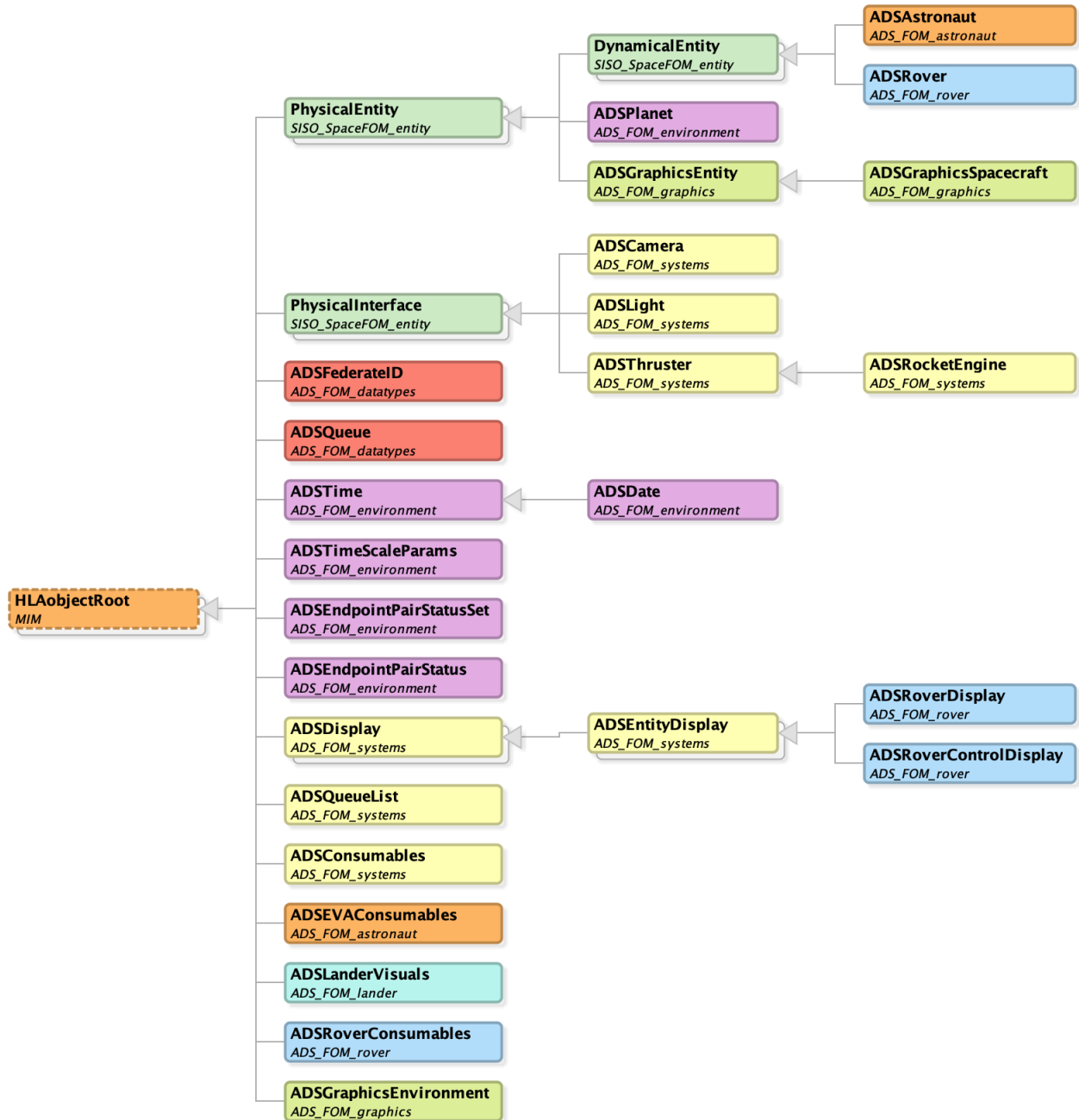


Figure 3: ADS FOM Object Class Hierarchy

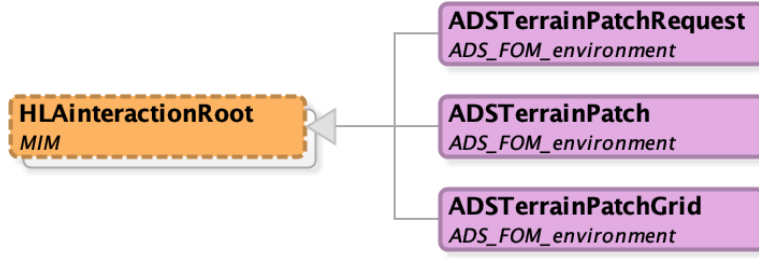


Figure 4: ADS FOM Interaction Class Hierarchy

4.1 ADS FOM Datatypes Module

To maintain a high level of SpaceFOM compliance, the ADS FOM relies heavily on SpaceFOM datatypes, Interaction Classes, and Object Classes. Where necessary, the ADS FOM either extends existing SpaceFOM data constructs or defines new constructs. As shown in Figure 2, the ADS Datatypes Module is directly dependent on the SpaceFOM Datatypes Module using many of the SpaceFOM simple datatypes to create new aggregate data types. The ADS Datatypes Module also creates entirely new datatypes, both simple and aggregate, for the space systems modeling requirements not covered in the SpaceFOM.

4.1.1 Datatypes

The ADS Datatypes Module define 17 new datatypes: 6 Simple, 1 Enumeration, 2 Array, and 8 Fixed Record. These are datatypes not already defined in HLA or SpaceFOM that fill a semantic requirement for ADS. The Simple types are: *ADSCount16*, *ADSCount32*, *ADSCount64*, *ADSHeatRate*, *ADSIndex*, *ADSPressure*. The one Enumeration is *ADSEulerSequence*. The Array types are *ADSEulerAngles* and *ADSPixelPoint*. The Fixed Record types are: *ADSPOCInfo*, *ADSEulerAttitude*, *ADSCalendarDate*, *ADSEndPointPairMapping*, *ADSPositionVector*, *ADSQueueState*, *ADSGraphicsState*, and *ADSEngineState*. For detailed semantic descriptions of these types, see the tables in Appendix A.

4.1.2 Object Classes

As shown in Figure 3, the ADS Datatypes Module only creates two new HLA Object Classes: *ADSFederateID* and *ADSQueue*. *ADSFederateID* provides base identifying information common to all ADS federates. This information is used to identify any joining federate and provide basic contact and version information. *ADSQueue* is used as a base FOM object for managing individual resource queues. More detail on the semantics of these Systems Object Classes can be found in Appendix A.

4.2 ADS FOM Environment Module

Beyond the core datatypes defined in the ADS Datatypes Module, the ADS FOM defines additional datatypes that are associated with modeling the environmental aspects of space, planetary surfaces, and the Artemis systems in the ADS Environment Module. As shown in Figure 2, the ADS Environment Module is directly dependent on the ADS Datatypes Module and, through dependency relationships, SpaceFOM.

4.2.1 Datatypes

The ADS Environment Module defines 7 new datatypes: 3 Enumeration, 1 Array, and 3 Fixed Record. These are datatypes specific to modeling the ADS space environment. The 3 new Enumeration datatypes are: *ADSTerrainPatchFormat*, *ADSTimeEpoch*, and *ADSTimeScale*. Two deal with the designation of time and are used in the *ADSTime* Object Class described below. The third enumerates possible digital terrain formats used for ADS terrain patch data. This module defines one new Array datatype *ADSTerrainPatchArray* also related to digital terrain data requests. The remaining three fixed record datatypes are: *ADSGeodeticPosition*, *ADSTerrainPatch*, and *ADSTimeStamp*. More detailed information on these datatypes is presented in Appendix B.

4.2.2 Interaction Classes

The ADS Environment Module defines 3 new HLA Interaction Classes: *ADSTerrainPatchGrid*, *ADSTerrainPatch*, and *ADSTerrainPatchRequest*. The *ADSTerrainPatchRequest* Interaction Class is used to request a specific patch of digital terrain data. The *ADSTerrainPatch* Interaction Class is used to provide a specific patch of digital terrain in response to a *ADSTerrainPatchRequest*. Finally, the *ADSTerrainPatchGrid* interaction class is used to provide a 3x3 of digital terrain data. The patch location is the middle of the center patch in the 3x3 grid. More detail on these Interaction Classes can be found in Appendix B

4.2.3 Object Classes

The ADS Environment Module defines 6 new HLA Object Classes: *ADSDate*, *ADSEndpointPairStatus*, *ADSEndpointPairStatusSet*, *ADSPlanet*, *ADSTime*, and *ADSTimeScaleParams*. The *ADSTime* Object Class provides a time value with an identified representation. For specificity, this time representation explicitly carries both the scale and epoch definitions. The *ADSDate* Object Class is a calendar date that extends *ADSTime*. The *ADSTimeScaleParams* Object Class is used to publish the offsets used to relate commonly used time scales. Specifically, *ADSTimeScaleParams* provides the offset between Terrestrial Time (TT) and Atomic Time (TAI), the leap second offset between Universal Time (UTC) and TAI, and the correlation factor (DUT1) to keep UTC synchronized within 0.9 seconds the Earth rotational time (UT1).

ADSEndpointPairStatus provides additional information used to inform the status of a line-of-sight intersection between an origin point and a destination point. *ADSEndpointPairStatusSet* contains a set of line-of-sight blockage statuses between specified endpoint pairs. The map contains a means for mapping into the blockage status array using the endpoint names. *ADSPlanet* provides the descriptive parameters for a planet as an extension of a SpaceFOM *PhysicalEntity* Object Class.

More detail on these Object Classes can be found in Appendix B.

4.3 ADS FOM Systems Module

The ADS FOM also includes datatypes associated with modeling spacecraft and spacecraft systems. Many of the more general systems modeling concepts are captured in the ADS Systems Module. As shown in Figure 2, the ADS Systems Module is directly dependent on the SpaceFOM Entity module and the ADS Environment Module.

4.3.1 Datatypes

The ADS Systems Module defines 10 new datatypes: 3 Simple and 7 Fixed Record. These are datatypes specific to modeling ADS systems. The three simple datatypes are related to electrical systems and are: *ADSVoltage*, *ADSCurrent*, and *ADSResistance*. The 7 Fixed Record datatypes are: *ADSBattery*, *ADSCircuitSwitch*, *ADSEVAState*, *ADSLightIndicator*, *ADSSimpleCamera*, *ADSSimpleSwitch*, and *ADSRangeCamera*. More detail on the semantics of these datatypes can be found in Appendix C.

4.3.2 Object Classes

The ADS Systems Module defines 8 new Object Classes: *ADSCamera*, *ADSConsumables*, *ADSDisplay*, *ADSEntityDisplay*, *ADSLight*, *ADSQueueList*, *ADSRocketEngine*, and *ADSThruster*. *ADSConsumables* is used as a base object for tracking and exchanging habitation systems consumable resources between ADS systems. It is expected that this base class will evolve and new classes will extend from it as the ADS architecture matures. *ADSDisplay* is a base Object Class definition for generalized displays used across the ADS systems. In general, this is intended for the base information associated with any DCApp based display, a NASA open source Display and Control Application [12], but does not necessarily need to be restricted to DCApp. *ADSEntityDisplay* extends *ADSDisplay* by associating a display with a named physical entity.

The remaining Object Classes are extensions of the SpaceFOM *PhysicalInterface*. *ADSCamera* provides a general camera description specified in association with a named *PhysicalInterface*. *ADSLight* provides a general light description including key parameters describing the status and traits of a light source. The final Object Classes are related: *ADSThruster* and *ADSRocketEngine*. *ADSThruster* defines the basic rocket thruster

design and performance parameters and is intended as a common base class for any specific rocket thruster class. *ADSRocketEngine* extends *ADSThruster* and defines additional data related to gimbal state.

More detail on the semantics of these Systems Object Classes can be found in Appendix C.

4.4 ADS FOM Graphics Module

In addition to modeling the physical behavior of constituent Artemis elements, the ADS also provides support for visualization of the complex Artemis mission scenarios. The ADS Graphics FOM Module defines the datatypes used to provide planetary status values for visualization in participating ADS federates. This includes flags and environment descriptors needed when running graphics in ADS to control certain toggles and get statuses of terrain objects. As shown in Figure 2, the ADS FOM Graphics Module depends directly on the SpaceFOM Entity module and the ADS FOM Environment Module.

4.4.1 Object Classes

The ADS FOM Graphics Module defines 3 Object Classes: *ADSGraphicsEnvironment*, *ADSGraphicsEntity*, and *ADSGraphicsSpacecraft*. The principal Object Class is *ADSGraphicsEnvironment* and is used to manage the state of a planetary surface graphics environment. It consists of the following 5 attributes: *sun_hide*, *sun_blocked*, *rock_highlight*, *rock_diameter*, and *rock_moveable_location*. Attributes *sun_hide*, *sun_blocked*, and *rock_highlight* are boolean flags used to respectively toggle the visibility of the Sun, indicate if the Sun is blocked by terrain, and highlight the movable rock in the graphics scene. The remaining two attributes are sizing and position values for the movable rock in the graphics scene. The *rock_diameter* attribute defines the characteristic sizing of a moveable rock model using a SpaceFOM *Length* data type. The *rock_moveable_location* attribute defines the location of the moveable rock using an ADS *ADSPositionVector* datatype.

The two remaining Graphics Module Object Classes are *ADSGraphicsEntity* and *ADSGraphicsSpacecraft*. Note that *ADSGraphicsEntity* extends the SpaceFOM *DynamicalEntity* Object Class. At this point in the ADS FOM development, *ADSGraphicsEntity* is simply a generalized marker base class for graphical objects based on the SpaceFOM *PhysicalEntity*/*DynamicalEntity* objects. It is still to be determined if this has any real value. One possibility future attribute may be information on the associated graphics model. *ADSGraphicsSpacecraft* extends *ADSGraphicsEntity* with an additional *thrust* attribute. This is used to display and size the graphical display of rocket engine plumes.

More detail on these Graphics Module Object Classes can be found in Appendix D.

4.5 ADS FOM Astronaut Module

At its core, the Artemis program is a human space exploration program. ADS is designed to support a wide variety of Artemis simulation needs, including human-in-the-Loop (HITL). As a result, humans will also need to be represented in ADS federation executions. The ADS FOM Astronaut Module provides the definitions for the human aspects of HITL systems.

4.5.1 Datatypes

The ADS FOM Astronaut Module defines 5 additional datatypes: 3 Simple, 1 Enumerated, and 1 Fixed Record. The 3 Simple data types are *ADSHeartRate*, *ADSPressuremmHg*, and *ADSRespirationRate*. The remaining two datatypes are the Enumeration *ADSGender* and the Fixed Record *ADSBloodPressure*. The semantic details of these datatypes can be see in Appendix E.

4.5.2 Object Classes

The ADS FOM Astronaut Module defines 2 additional Object Classes: *ADSAstronaut* and *ADSEVAConsumables*. Object Class *ADSAstronaut* extends the SpaceFOM *DynamicalEntity* class and provides the additional identifying and metabolic information for an astronaut. Object Class *ADSEVAConsumables* provides key parameters associated with a human in a space system closed environment or on an ExtraVehicular Activity (EVA). These consumable attributes include: *location*, *status*, *metabolic_rate*, *O2*, *CO2*, *H2O*, *O2.consumption_rate*,

CO2_production_rate, *H2O_production_rate*, and *heat_production_rate*. The semantic details of these Object Class attributes can be seen in Appendix E.

4.6 Other Modules

The preceding general datatypes, Interaction Classes, and Object Classes provide the foundation used to describe Artemis elements in ADS. The intent is to build the constituent Artemis element ADS data models upon this foundation. Simply stated, all federates participating in an ADS federation execution are expected to base their published Interactions and Object instances on these predefined ADS FOM Module definitions wherever possible and only define new datatypes, Interaction Classes, and Objects Classes when the predefined types are insufficient. This increases the likelihood of *a priori* interoperability with any existing ADS federates and federation executions while enabling the use of new federates in exploring an evolving set of Artemis mission scenarios.

Two categories of space systems of particular interest to Artemis are landing systems and planetary surface mobility systems. The ADS FOM includes two FOM Modules for these: `ADS_FOM_lander.xml` and `ADS_FOM_rover.xml`

4.6.1 ADS FOM Lander Module

The ADS Lander FOM Module currently defines only a single additional Object Class called *ADSLanderVisuals*. This defines a collection of lander systems states that are used in the visual display of a generalized landing vehicle. The ADS development team expects this class to be used as a pattern for design specific implementations as the Artemis architecture and Artemis landing systems evolve. More detail on these Lander Module Object Classes can be found in Appendix F.

4.6.2 ADS FOM Rover Module

The ADS Rover FOM Module currently defines 4 additional Object Classes intended for use in planetary surface mobility systems: *ADSRover*, *ADSRoverConsumables*, *ADSRoverDisplay*, and *ADSRoverControlDisplay*. *ADSRover* extends the SpaceFOM *DynamicalEntity* and provides additional entity state data useful to planetary surface systems. *ADSRoverConsumables* provides consumables tracking and data exchange for planetary surface systems. The final two Object Classes extend from *ADSEntityDisplay*: *ADSRoverDisplay* and *ADSRoverControlDisplay*. *ADSRoverDisplay* defines base state parameters for a simple display on a two person unpressurized rover. *ADSRoverControlDisplay* defines control data for a generalized human operated rover. More detail on these Graphics Module Object Classes can be found in Appendix G.

5 Conclusions

This paper provides an overview of the ADS FOM and examines the ADS FOM modules, ADS base data types, ADS SpaceFOM Object Class extensions, new ADS Object Classes, and new ADS Interactions Classes. Any Artemis element simulations intended for use in an ADS-compliant federation execution will need to use the ADS FOM as a base for common element data definitions. This practice increases the likelihood of *a priori* interoperability with any existing ADS federates and the utility of those simulations in exploring new Artemis mission scenarios. This will also enable the use of those simulation in integrated Artemis analysis, verification, or test scenarios.

6 Future Work

It is important to note that ADS is a work in progress. The ADS development team is actively expanding the ADS FOM, adjusting content, clarifying semantics, and modifying the architecture to track the Artemis program. As new Artemis elements are proposed and accepted, element specific FOM Modules will be created. We are currently working on specific FOM additions for the Human Landing Systems (HLS) vendor systems and the Lunar Terrain Vehicle (LTV) systems. The experience gained from incorporating these systems will further fuel the evolution of the ADS FOM and ADS architecture.

Acronyms

ADS	Artemis Distributed Simulation
DDM	Data Distribution Management
DM	Declaration Management
EVA	ExtraVehicular Activity
FOM	Federation Object Model
HITL	Human In The Loop
HLA	High Level Architecture
HLS	Human Landing System
IEEE	Institute for Electrical and Electronics Engineers
JSC	Johnson Space Center
LTV	Lunar Terrain Vehicle
NASA	National Aeronautics and Space Administration
NExSyS	NASA Exploration Systems Simulations team
OMT	Object Model Template
SISO	Simulation Interoperability Standards Organization
SpaceFOM	Space Reference Federation Object Model
US	United States of America
XML	eXtensible Markup Language

References

- [1] K. Dodd, P. Whittington, E. Crues, J. Gentile, and N. Trevino, “Using distributed simulation capabilities to support studies and mission operations for NASA Artemis Program,” in *IEEE 2025 Aerospace Conference*, IEEE, IEEE, March 2025.
- [2] National Aeronautics and Space Administration, “The Artemis Accords.” <https://www.nasa.gov/artemis-accords/>.
- [3] National Aeronautics and Space Administration, “Moon to Mars Architecture.” <https://www.nasa.gov/moontomarsarchitecture/>.
- [4] National Aeronautics and Space Administration, “Moon to Mars Architecture - Executive Overview,” Architecture Concept Review NP-2024-01-3212-HQ, National Aeronautics and Space Administration, 300 Hidden Figures Way SW, Washington, DC 20546, 2023.
- [5] Exploration Systems Development Mission Directorate, “Moon to Mars Architecture - Architecture Definition Document,” Architecture Concept Review NP-2024-01-3212-HQ, National Aeronautics and Space Administration, 300 Hidden Figures Way SW, Washington, DC 20546, 2023.
- [6] Simulation Interoperability Standards Organization, “Simulation Interoperability Standards Organization.” <https://www.sisostandards.org>.

- [7] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC), “*IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules*,” Tech. Rep. IEEE-1516-2010, The Institute of Electrical and Electronics Engineers, 2 Park Avenue, New York, NY 10016-5997, August 2010.
- [8] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC), “*IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Federate Interface Specification*,” Tech. Rep. IEEE-1516.1-2010, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, August 2010.
- [9] Simulation Interoperability Standards Organization/ Standards Activities Committee (SISO/SAC), “*IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Object Model Template (OMT) Specification*,” Tech. Rep. IEEE-1516.2-2010, The Institute of Electrical and Electronics Engineers, 3 Park Avenue, New York, NY 10016-5997, August 2010.
- [10] E. Crues, D. Dexter, A. Falcone, A. Garro, M. Madden, B. Möller, and D. Ronnfeldt, *Standard for Space Reference Federation Object Model (SpaceFOM)*. No. SISO-STD-018-2020, P.O. Box 781238, Orlando, FL 32878-1238, USA: Simulation Interoperability Standards Organization (SISO), October 2019.
- [11] E. Z. Crues, D. Dexter, A. Falcone, A. Garro, and B. Möller, “Spacefom - a robust standard for enabling a-priori interoperability of hla-based space systems simulations,” *Journal of Simulation*, vol. 16, no. 6, pp. 624–644, 2022.
- [12] National Aeronautics and Space Administration, Johnson Space Center, Software, Robotics & Simulation Division, Simulation and Graphics Branch (ER7), “Display and Control Application (DCApp).” <https://github.com/nasa/dcapp>, September 2023.

Author Biographies



Edwin Z. {Zack} Crues: received B.S., M.S., and Ph.D. degrees in Aerospace Engineering from the University of Texas, Austin in 1983, 1985, and 1989 respectively. Zack has over 35 years of professional experience in developing spacecraft simulation and simulation technologies internationally. Zack currently works at NASA Johnson Space Center in Houston, Texas as a Space Systems Simulation Architect in the Simulation and Graphics Branch where he leads the development of simulation technologies and the application of those technologies in the simulation of NASA’s current and proposed crewed spacecraft.



Paige Whittington: received her B.S. degree in Aerospace Engineering from The University of Texas at Austin and a M.S. in Astrodynamics and Space Applications from Purdue University. Paige currently works at NASA Johnson Space Center in Houston, TX as a member of the NASA Exploration Systems Simulations (NExSyS) team. Paige is the technical chair for the Simulation Exploration Experience (SEE) international student distributed simulation collaboration and is a developer for lunar landing simulations and the Artemis Distributed Simulation.



Keaton Dodd: received a B.S. in Aerospace Engineering from the Texas A&M University College Station in 2020. Keaton has 4 years of experience in the Spacecraft systems and simulation field. Keaton currently works at the NASA Johnson Space Center in Houston, Texas as a Spacecraft Systems and Dynamic Simulation Engineer for CACI International. Keaton heads the Artemis Distributed Simulation project development and contributes to lunar rover and NASA Gateway simulation development.

Appendices

A ADS FOM Datatypes Module

While the SpaceFOM provides a number of datatypes useful in the description of complex space systems, the SpaceFOM was not designed to provide a complete set of all possible datatypes for space systems. The need to develop additional datatypes is both expected and supported. The ADS Datatypes Module depends directly on the SpaceFOM Datatypes Module using many of the SpaceFOM simple and aggregate datatypes to create new aggregate data types. The ADS Datatypes Module also creates new datatypes, both simple and aggregate, for the space systems modeling requirements not covered in the SpaceFOM. The ADS Datatypes Module define 17 new datatypes and two new Object Classes.

A.1 Data Types

The following tables describe the content of the ADS FOM's base datatypes.

A.1.1 Simple

Name	Base Type	Units	Semantics
ADSIndex	HLAinteger64LE		Used to represent and index into an array of data.
ADSCount64	HLAinteger64LE		An arbitrary 64 bit integer count.
ADSCount32	HLAinteger32LE		An arbitrary 32 bit integer count.
ADSCount16	HLAinteger16LE		An arbitrary 16 bit integer count.
ADSHeatRate	HLAfloat64LE	Joule/second (J/s)	A measure of the time rate of change of energy/heat.
ADSPressure	HLAfloat64LE	Pascal (Pa)	A measure of pressure.

Table 1: ADS FOMs Base Simple Datatypes

A.1.2 Enumerations

Name	Base Type	Semantics
ADSEulerSequence	HLAinteger16LE	These represent the principal Euler rotation sequences for an Euler angle representation of attitude.

Table 2: ADS FOMs Base Enumerated Datatypes

A.1.3 Arrays

Name	Base Type	Units	Semantics
ADSEulerAngles	Angle	Radians (r)	A 3 vector of angles used to represent an Euler sequence. The actual sequence is defined elsewhere.
ADSPixelPoint	ADSIndex		An (x,y) integer point paring that corresponds to a location in a 2-dimensional array of pixels. This does not imply any specific direction as positive and pixel values may be both positive and negative. The (0,0) pixel point is the origin of the pixel raster.

Table 3: ADS FOMs Base Array Datatypes

A.1.4 Fixed Records

Name	Fields	Semantics
ADSPOCInfo	name, phone, email, affiliation, updated	Contains the general information used to identify a point of contact (POC) for an ADS federate.
ADSEulerAttitude	sequence, angles	A representation of an object attitude using Euler angels. Both the Euler sequence and the Euler angle values are specified.
ADSCalendarDate	seconds, minute, hour, day, month, year, day_number	The calendar date and time representation of an associated time value.
ADSEndPointPairMapping	origin, destination, index	Contains the information necessary to map between two endpoint names and the indexed location into an associated blockage status array.
ADSPositionVector	parent_reference_frame, position	A position vector expressed in the defined parent reference frame.
ADSQueueState	name, size, count	The state of a queue.
ADSGraphicsState	name, graphics_modeling_frame, state	The state of a named entity associated with a federation execution represented with respect to some understood graphics modeling frame (GMF). The state consists of a position, velocity, attitude, and attitude rate expressed in some known 'parent reference frame'. This is a subset of what is usually contained in a Space-FOM PhysicalEntity. In most cases, the PhysicalEntity representation should be used. The ADSGraphicsState is only intended for publishing reduced/simplified states for graphics rendering.
ADSEngineState	isp, m_dot, v.e, thrust, gimbal_pitch, gimbal_yaw	Basic rocket engine design and performance parameters. While this data type can be used as is for exchanging relevant rocket engine information, it should be the base for any specific rocket engine classes.

Table 4: ADS FOMs Base Fixed Record Datatypes

A.2 Object Classes

Name	Attributes	Semantics
ADSFederateID	name, version, poc	Base identifying information common to all Artemis Distributed Simulation (ADS) federates. This information is used to identify any joining federate and provide basic contact and version information. It is intended as a base object to be extended with additional information that may be pertinent to a specific federate. However, this base information is always present.
ADSQueue	name, size, poc	This is a base FOM object for managing individual queues. It is the same as the ADSQueueState type but intended for object use and general extension.

Table 5: ADS FOMs Base Object Classes

B ADS FOM Environment Module

The ADS Environment Module defines additional datatypes that are associated with modeling the environmental aspects of space, planetary surfaces, and the Artemis systems. The ADS Environment Module defines 7 new datatypes, 3 new Interaction Classes, and 6 Object Classes.

B.1 Data Types

B.1.1 Enumerations

Name	Base Type	Semantics
ADSTerrainPatchFormat	HLAinteger16LE	Indicator for the terrain patch format.
ADSTimeScale	HLAoctet	The principal time scales recognized by an Artemis Distribute Simulation (ADS) federation execution. Currently, those are Terrestrial Time (TT), Atomic Time (TAI), and Universal Coordinated Time (UTC).
ADSTimeEpoch	HLAoctet	The origin or zero reference point for a time representation.

Table 6: ADS FOMs Environment Enumerated Datatypes

B.1.2 Arrays

Name	Base Type	Units	Semantics
ADSTerrainPatchArray	ADSTerrainPatch	–	A 9 element array of terrain patches that constitute a 3x3 grid.

Table 7: ADS FOMs Environment Array Datatypes

B.1.3 Fixed Records

Name	Fields	Semantics
ADSGeodeticPosition	latitude, longitude, altitude	A geodetic position based on latitude, longitude, and altitude.
ADSTerrainPatch	name, data	A specified patch of terrain.
ADSTimeStamp	scale, epoch, time	A time value with a particular representation. For specificity, this time representations explicitly carries both the scale and epoch of the time representation. The base time scale for SpaceFOM is the Terrestrial Time (TT) time scale. By definition, SpaceFOM TT time is expressed with respect to the Truncated Julian Date (TJD) epoch. Note that many TT representations are expressed with respect to the J2000 epoch. Be careful to account for differences in representations due to epoch.

Table 8: ADS FOMs Environment Fixed Record Datatypes

B.2 Interaction Classes

Name	Attributes	Semantics
ADSTerrainPatchRequest	name, location, format	This interaction is used to request a 'terrain patch' from the ADS environment federate.
ADSTerrainPatch	name, location, format, data	Digital terrain data for a specified patch.
ADSTerrainPatchGrid	name, location, format, grid_data	A 3x3 grid of terrain data. Used to provide a 'patch' of data about an identified central position. The central position should be in the center patch (1,1).

Table 9: ADS FOMs Environment Interaction Classes

B.3 Object Classes

Name	Attributes	Extends	Semantics
ADSTime	scale, epoch, time	–	A time value with a particular representation. For specificity, this time representations explicitly carries both the scale and epoch of the time representation. The base time scale for SpaceFOM is the Terrestrial Time (TT) time scale. By definition, SpaceFOM TT time is expressed with respect to the Truncated Julian Date (TJD) epoch. Note that many TT representations are expressed with respect to the J2000 epoch. Be careful to account for differences in representations due to epoch.
ADSDate	date	ADSTime	The calendar data associated with particular time representation.
ADSTimeScaleParams	tai_offset, delta_at, dut1	–	This object is used to publish the offsets used to relate commonly used time scales: TAI offset, Delta AT, and DUT1. Note that TAI offset from TT is almost always a constant 32.184 second offset ($TAI = TT - 32.184$). Delta AT is the offset between TAI and UTC to account for the change in Earth rotation over time. It is applied as an integer leap second to keep UTC within the sidereal time time determined by the actual Earth rotation ($UTC = TAI - \Delta AT$). DUT1 is a correlation factor to keep UTC within 0.9 seconds of UT1 ($UTC = UT1 - DUT1$).

Name	Attributes	Extends	Semantics
ADSEndpointPairStatusSet	map, blockage_status	–	Contains a set of line-of-site blockage status between specified endpoint pairs. The map contains a means for mapping into the blockage_status array using the endpoint names.
ADSEndpointPairStatus	origin, destination, position, length, intersection, intersected	–	Additional information use to inform the status of a line-of-site intersection between an origin point and a destination point.
ADSPlanet	equatorial_radius, mean_radius, polar_radius, density, equatorial_gravity	SpaceFOM:: PhysicalEntity	Descriptive parameters for a planet as an extension of a SpaceFOM PhysicalEntity.

Table 10: ADS FOMs Environment Object Classes

C ADS FOM Systems Module

The ADS Systems Module provides datatypes for general systems modeling concepts. The ADS Systems Module depends directly on the SpaceFOM Entity module and the ADS Environment Module. The ADS Systems Module defines 10 new datatypes and 8 new and extended Object Classes.

C.1 Data Types

C.1.1 Simple

Name	Base Type	Units	Semantics
ADSVoltage	HLAfloat64LE	Volt (V)	Electrical voltage.
ADSCurrent	HLAfloat64LE	Ampere (A)	Electrical amperage.
ADSResistance	HLAfloat64LE	Ohms (ohm)	Electical resistance measured in Ohms.

Table 11: ADS FOMs Systems Simple Datatypes

C.1.2 Fixed Records

Name	Fields	Semantics
ADSBattery	charge, voltage, current, power	Parameters that represent a batter state.
ADSSimpleSwitch	state, voltage, current	Represents the state of a simple electrical switch.
ADSCircuitSwitch	state, voltage, current, indicator, positive_trip, negative_trip, trip_reset	Represents the state of an electrical circuit switch.
ADSLightIndicator	indicator, state	A light indicator state. Primarily used in association with a display. The light itself has a state.
ADSEVAState	active, latitude, longitude	The state of an EVA activity.

Name	Fields	Semantics
ADSSimpleCamera	pan, tilt, zoom	A simple camera description. Primarily used by displays. More detailed information would be available in an associated ADSCamera Object Class if there is one.
ADSRangeCamera	pan, tilt, zoom, homing, fixate, range, range_rate	Range finder camera state data.

Table 12: ADS FOMs Systems Fixed Record Datatypes

C.2 Object Classes

Name	Attributes	Extends	Semantics
ADSDisplay	name, time	–	This is a base Object Class definition for generalized displays used across the Artemis Distributed Simulation (ADS) systems. In general, this is intended for the base information associated with any DCAApp based display but does not necessarily need to be restricted to DCAApp.
ADSEntityDisplay	entity_name	ADSDisplay	An extension to the ADSDisplay object class that associates a display with a specific PhysicalEntity.
ADSCamera	pan, tilt, zoom, spin, scale_factor, focal_length, principle_point, skew	SpaceFOM::PhysicalInterface	General camera description. The pan, tilt, and spin parameters are applied in addition to the camera mounting frame specified in the PhysicalInterface base object class.
ADSLight	active, temperature, intensity, active_command, temperature_command, intensity_command	SpaceFOM::PhysicalInterface	Key parameters describing the status and traits of a light source. This object also contains the command state as well.
ADSThruster	isp, m_dot, v_e, thrust	SpaceFOM::PhysicalInterface	Basic rocket thruster design and performance parameters. While this object class can be used as is for exchanging relevant rocket thruster information, it should be the base for any specific rocket thruster classes.
ADSRocketEngine	gimbal_pitch, gimbal_yaw	ADSThruster	Basic rocket engine design and performance parameters. While this object class can be used as is for exchanging relevant rocket engine information, it should be the base for any specific rocket engine classes.
ADSQueueList	tank, sample	–	A list of queues used to manage logistics flow through the architecture.
ADSConsumables	grey_water, potable_water	–	Consumable data exchange.

Table 13: ADS FOMs Environment Object Classes

D ADS FOM Graphics Module

The ADS Graphics Module defines the datatypes used to provide planetary status values for visualization in participating ADS federates. The ADS Graphics Module directly depends on SpaceFOM Entity module and the ADS Environment Module and defines 3 extended Object Classes.

D.1 Object Classes

Name	Attributes	Extends	Semantics
ADSGraphicsEnvironment	sun_hide, sun_blocked, rock_highlight, rock_diameter, rock_moveable_location	SpaceFOM:: DynamicalEntity	This object is used to manage the state of a planetary surface graphics environment.
ADSGraphicsEntity	–	SpaceFOM:: PhysicalEntity	A generalized marker base class for graphics specific PhysicalEntities. It is still TBD if this has any real value.
ADSGraphicsSpacecraft	thrust	ADSGraphics- Entity	A graphics specific representation of a spacecraft. It is still TBD if this has any real value.

Table 14: ADS FOMs Graphics Object Classes

E ADS FOM Astronaut Module

The ADS Astronaut Module provides datatypes for the human aspects of Artemis HITL systems. The ADS Astronaut Module defines 5 new datatypes and 2 new and derived Object Classes.

E.1 Data Types

E.1.1 Simple

Name	Base Type	Units	Semantics
ADSHeartRate	HLAinteger16LE	beats per minute (bpm)	The medical heart rate of a being.
ADSPressuremmHg	HLAinteger16LE	millimeters of Mercury (mmHg)	Pressure measured in millimeters of mercury (mmHg). This is typically used for measuring blood pressure.
ADSRespirationRate	HLAinteger16LE	breaths per minute	A beings respiration rate. The number of breath a being takes per minute.

Table 15: ADS FOMs Astronaut Simple Datatypes

E.1.2 Enumerations

Name	Base Type	Semantics
ADSGender	HLAoctet	The genetic physiological gender of a being. This is not meant for recording gender identity but physiological gender for metabolic data recording.

Table 16: ADS FOMs Astronaut Enumerated Datatypes

E.1.3 Fixed Records

Name	Fields	Semantics
ADSBloodPressure	systolic, diastolic	Blood pressure information.

Table 17: ADS FOMs Systems Fixed Record Datatypes

E.2 Object Classes

Name	Attributes	Extends	Semantics
ADSAstronaut	first_name, middle_name, last_name, location, event, gender, blood_pressure, heart_rate, respiration, on_eva	SpaceFOM:: DynamicalEntity	Identifying and metabolic information for an astronaut PhysicalEntity. It extends DynamicalEntity in case there are applicable forces and torques.
ADSEVAConsumables	location, status, metabolic_rate, O2, CO2, H2O, O2_consumption_rate, CO2_production_rate, H2O_production_rate, heat_production_rate	—	Consumable data exchange.

Table 18: ADS FOMs Astronaut Object Classes

F ADS FOM Lander Module

The ADS Lander Module provides a collection of lander systems states that are used in the visual display of a generalized landing vehicle. Currently, the ADS Lander Module defines one new Object Class.

F.1 Object Classes

Name	Attributes	Extends	Semantics
ADSLanderVisuals	hldm_graphics_state, hldm_engine, hldm_flight_att, hlram_graphics_state, hlram_engine, hlram_inert_state, hlram_struct_att, hlram_launch_att, hltv_graphic_state, hltv_engine, plume_origin, plume_angles	–	Lander Object.

Table 19: ADS FOMs Lander Object Classes

G ADS FOM Rover Module

The ADS Rover Module currently defines 4 additional Object Classes intended for use in planetary surface mobility systems.

G.1 Object Classes

Name	Attributes	Extends	Semantics
ADSRoverConsumables	potable_water, solar_illumination	–	Rover consumables object.
ADSRover	ups_position, utc_date, charge, odometer	SpaceFOM:: DynamicalEntity	Rover object class definition.
ADSRoverDisplay	geo_position, battery, switch1, light1, ev1, ev2, range_finder_camera	ADSEntityDisplay	A display base for a planetary surface rover.
ADSRoverControlDisplay	relpor_vel, por_range, por_range_rate, e_brake_ind, brake_applied_ind, attitude_angles, bulldozing_ind, bulldozing_scale, cruise_target_vel, ground_clearance, teleported, elapsed_time, hsi_sun_angle, hsi_earth_angle, projected_lle, speed_factor_ind, steer_factor_ind	ADSEntityDisplay	Rover control display information.

Table 20: ADS FOMs Rover Object Classes