# Bring Live Sensors to Virtual World via DIS

**Jackie Zhang, Reese Gallagher**
**Infinitas Engineering Inc.**
**Orlando, Florida**
**jackie.zhang@infinitasinc.com**
**reese.gallagher@infintiasinc.com**

**Susan Harkrider**
**NVESD**
**Orlando, Florida**
**susan.m.harkrider.civ@mail.mil**

**Craig Clark**
**KRATOS Defense**
**Orlando, Florida**
**craig.clark@kratosdefense.com**

## ABSTRACT

Integrating Live-Virtual-Constructive (LVC) entities to support military training is the ultimate goal of a Synthetic Training Environment (STE). LVC training incorporates geographically distributed, real-time simulation systems, virtual simulators, and live training devices into a single shared virtual environment to maximize immersive training experiences. Currently, LVC interoperability relies on Distributed Interoperability Simulation (DIS) protocols and High Level Architecture (HLA) for communications amongst training systems. With this as the industry standard, each training device must be able to communicate across the network using DIS protocols or HLA Federation Object Models (FOMs). Integrated Sensor Architecture (ISA) was developed to support Army ground tactical combat by Army's Night Vision and Electronic Sensors Directorate (NVESD). Its architecture was designed to manage the sensor systems of systems (SoS) as well as the distribution of sensor data. ISA brings together a collection of live sensors within an area of operation so they can communicate without requiring physical integration with a set of well-defined protocols in place to ensure interoperability when additional sensors come online to the network. Although ISA was designed to work in the tactical environment, its capability of allowing different sensors to dynamically interact with each other is also very desirable for distributed real-time simulation and training exercises. Our preliminary research shows that in order to bring ISA into the distributed simulation domain, the most crucial step is to make its communication protocols DIS compliant. During the prototyping of an interface that converts ISA sensor data to DIS EntityState Protocal Data Units (PDU), we realized the need for a full complement of DIS sensor message representation. We believe the sensor data can be represented in a new type of DIS PDU, by reusing some ISA classification scheme of capabilities, and having the protocol standardized would enable tactical sensors to be part of the virtual training world.

## ABOUT THE AUTHORS

**Ms. Jackie Zhang** is the CEO and owner of Infinitas Engineering Inc. Her 20+ years of software systems engineering and research and development experiences have been focused on various development phases of CGF systems especially physical modeling and behavioral modeling, cultural modeling, modeling Data Science techniques such as Expert Systems and Case-based and Context-based Reasoning techniques, etc. Ms. Zhang received Bachelor and Master of Science degrees in Computer Science from University of Central Florida. Ms. Zhang incorporated Infinitas Engineering Inc. in 2012 and has been using her CGF modeling skills and experiences to train her simulation engineering team to provide software development and system integration support for various Army contracts.

**Mr. Reese Gallagher** is a simulation engineer & researcher working at Infinitas Engineering Inc. His research background stems from his education in chemistry where he received a Bachelor of Science in chemistry at the University of Florida (UF) and a Masters in chemistry from the University of Central Florida (UCF) where he has co-authored several papers and authored a paper in a peer-reviewed journal. He transitioned into simulation-focused research following additional education from UF in computer science has since been leading IRAD efforts in the MS&T domain at Infinitas Engineering.

**Ms. Susan Harkrider** Susan Harkrider is the Deputy Associate Director for the Modeling & Simulation and Power Portfolio at the US Army DEVCOM C5ISR Research and Technology Integration Directorate. Previously she was the Deputy Director of the Modeling and Simulation Division (MSD) at the CERDEC Night Vision and Electronic Sensors Directorate (NVESD). She has over 30 years' experience supporting modeling and simulation, training and systems engineering efforts for the DoD. She served on several M&S committees, including the Simulation Interoperability Standards Organization (SISO), the RDECOM M&S Senior Working Group (SWG), and the Sensors Working Group at the AMSO M&S Gaps Forum. She currently serves on the I/ITSEC paper committee and is the US voting representative to the NATO M&S Group. She holds a BSE and MSE from the University of Central Florida.

**Mr. Craig Clark** is the Chief Technology Officer (CEO) at Kratos Defense. He is the head of Kratos' Research and Development (R&D) for all of Kratos training and simulation. Over the past 7 years Craig has directed business unit technologies from product operations improvements to research and development. Prior to being the CTO Craig was the Senior Software Architect designed the training application framework for the modeling and simulation systems. He holds a BS in computer science and an MSEM from the University of Central Florida. Craig started his career in the Naval Nuclear Propulsion Program, U.S. Navy. He was a naval instructor and has spent plenty of time underway running propulsion systems. He is a graduate of UCF that has the finest computer science school in the nation on simulation, synthetic design, and training. He worked in telecommunications until 9/11 and subsequently changed his career to defense and training. He became a principal architect for General Dynamics and designed urban operation systems which has trained over a million U.S. Army soldiers and still operates at over 120 different primary military training centers.

# Bring Live Sensors to Virtual World via DIS

**Jackie Zhang, Reese Gallagher**
**Infinitas Engineering Inc.**
**Orlando, Florida**
**Jackie.zhang@infinitasinc.com**
**Reese.gallagher@infintiasinc.com**

**Susan Harkrider**
**NVESD**
**Orlando, Florida**
**Susan.m.harkrider.civ@mail.mil**

**Craig Clark**
**KRATOS Defense**
**Orlando, Florida**
**Craig.clark@kratosdefense.com**

## INTRODUCTION

In the world of Live-Virtual-Constructive (LVC) simulation, the interoperability of a joint distributed exercise is achieved through Distributed Interactive Simulation (DIS) protocols. The basic elements of DIS are Protocol Data Units (PDUs). These PDUs contain information of all simulated entities such as military vehicles, aircrafts, weapon systems, electronic warfare, logistics, collisions and simulation management. With this industry standard, each training device must be able to communicate across the network using these DIS PDUs. In the last decades, the modern world, new technologies have empowered us to allow sensors along with high-tech solutions to delegate complex, critical and dangerous functions to these "smart machines". Sensors are widely used in military training applications for situational awareness when many processes are not feasible or too expensive to be conducted on a real equipment or real physical surrounding.

The U.S. Army's Night Vision and Electronic Sensors Directorate (NVESD) Modeling and Simulation Division (MSD) provides sensor performance modeling using physics-based algorithms of actual sensor performance or platforms, including electro-optic, infrared, acoustic, magnetic, seismic, synthetic, aperture and ground penetrating radar sensors, as well as certain munition effects related to the sensors' capabilities. These sensor models are used for data collection, analysis, concept experiments and capability assessments. Because NVESD uses its authoritative sensory data to model the sensor, these sensors provide credible sensor simulation and are widely used by several DoD programs such as the Army's Maneuver Support Center of Excellence (MSCoE) Capability Development and Integration Directorate (CDID) at Fort Leonard Wood, PM Soldier Maneuver Sights and Long-Range Scout Surveillance System program, and many more (Harkrider, Krapels, Krug & McGlynn). NVESD MSD also developed Integrated Sensor Architecture (ISA), which is a U.S. Army Service-Oriented Architecture (SoA).

> "ISA provides capabilities that enable Soldiers to exchange information between their own sensors and those on other platforms in a fully dynamic and shared environment. ISA enables Army sensors and systems to readily integrate into an existing network and dynamically share information and capabilities to improve situational awareness in a battlefield environment."
>
> --(Kovach & Sadler).

Within an area of operation, ISA brings together a collection of live sensors so they can communicate without requiring physical integration by using a set of well-defined protocols in place to ensure interoperability when additional sensors come online to the network. These sensors come with an extensible data model for each type, coupled with dynamic discovery capabilities, cyber security and sensor management, which can be a great addition to the virtual training world where all Live-Virtual-Constructive systems work together to achieve joint training goals (ISA Data Model Specificaton ). In fact, the STE program already identifies these types of sensors.

Although ISA was designed to work in the tactical environment, its capability of allowing different sensors to dynamically interact with each other is also very desirable for distributed real-time simulation and training exercises. In order to bring ISA sensors into the distributed simulation domain, the most important step is to make its communication protocols DIS compliant. Only through DIS, can the simulated ISA sensors be distinguished from other DIS compatible simulations during exercises while providing adequate and convincing results.

Since ISA has its own extensible sensor data model with well-defined sensory capabilities, as well as a set of tools to facilitate sensor data storage, updates and queries, our research started with a prototyping effort to create a set of methods to convert ISA sensor data into DIS PDUs. To test how well this new PDU works on a distributed simulation exercise, we also developed an interface to pass the newly created EntityState PDU from using ISA data

model, to the DIS network. A typical distributed exercise we used to test our PDUs consists of a Local Area Network (LAN), one instance of Army's One Semi-automated Force (OneSAF), one instance of Bohemia's Virtual Battle Space 4 (VBS4), and ISA. Multiple scenarios were developed to test how an ISA sensor is providing situational awareness message to the DIS network using EntityState PDUs, and how well OneSAF and VBS4 received and decoded the messages to facilitate their decision-making processes. Despite of some initial hiccups on working with ISA's cyber security features and large data dumps, the scenarios worked seamlessly amongst all participating simulation systems. All results were demonstrated to the NVESD technical team.

During the prototyping of converting ISA sensor data to DIS PDUs, we discovered that there isn't a standard PDU designed for sensors. Upon researching on the Modeling, Simulation and Training (MS&T) industry standards, we realized that the industry has been using the Data PDU every time a new sensor is needed to be modeled. This industry practice is not only cumbersome and slow, but it also introduces inconsistencies in simulations where many different types of sensors of different vendors are used. Therefore, we realized the need for a full complement of DIS sensor message representation. We believe the various sensor data can be represented in a new type of DIS PDU, by reusing some ISA classification scheme of capabilities. We also believe that armed with extensive DIS knowledge and a careful designed and implemented Sensor PDU structure, we can have a standardized protocol to enable more tactical sensors to work with all DIS compatible simulation systems.

**Sensor PDU Development**

A Data PDU is 288 bytes in length and is sent out to the DIS network at a frequency of 1 to 30 Hz. This rate can be modified in order to lessen bandwidth impact. Each Data PDU contains a fixed datum dataset which can be used to completely describe whatever data is needed to be transmitted to the DIS network. Data PDU provides an easy way for simulation developers to quickly insert randomly needed data to a PDU so it can be transmitted via DIS. Currently, many sensor messages were put in Data PDUs because there isn't a dedicated Sensor PDU available. We decided to test the validity of creating a new PDU based on the format of Data PDU for sensor messages, and our goal is to design the Sensor PDU using ISA sensor model data because of its well-defined data fields and data structures capture many types of messages using composite data types that are reusable to define PDU fields.

ISA sensor data types has three main categories: Property, Observable and Command, which capture different categories of capabilities for a particular type of sensor. Each category captures a different set of values according to the type of sensor. For example, a type of Property sensor has "Capability" values such as Armed, Auto Focus etc., while a type of Command sensor has "Capabilitiy" values such as Adjust Focus and Cancel.

| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| Property | | | Observable | | | Command | |
| Capability | Capability Enumeration | | Capability | Capability Enumeration | | Capability | Capability Enumeration |
| AC Current Load | 1 | | Area | 10001 | | Add IP Address | 20001 |
| AC Current Source | 2 | | Atmospheric Humidity | 10002 | | Add Subscription | 20002 |
| AC Voltage Load | 3 | | Atmospheric Pressure | 10003 | | Add Zone | 20003 |
| AC Voltage Source | 4 | | Atmospheric Temperature | 10004 | | Adjust Brightness | 20004 |
| Active | 5 | | Atmospheric Wind | 10005 | | Adjust Contrast | 20005 |
| Armed | 6 | | BSO | 10006 | | Adjust Focus | 20006 |
| Attached Orientation | 7 | | Bearing | 10007 | | Adjust Gain | 20007 |
| Attached Position | 8 | | Biological Reading | 10008 | | Adjust Iris Size | 20008 |
| Auto Brightness | 9 | | Chemical Reading | 10009 | | Adjust Level | 20009 |

**Figure 1 Sample ISA Sensor Data**

In addition, a sensor could potentially have multiple capabilities. For example, it may have properties and commands enabled. In order to maximize the PDU usage and to accommodate future sensor types, we believe there should be a family of Sensor PDUs to capture multiple capabilities.

According to DIS 7 (SISO), PDU types 1-72 are well defined and standardized, and 73-128 are currently unused (SISO). We started by creating a PDU family type ID "Sensor PDU", and carefully designed the data fields and data types, and reuse as much as we can from the ISA sensor model representations. Below, we report the result from a recent Independent Research and Development (IRAD) study in which a family of Sensor PDUs are defined, developed, tested in scenarios developed using Unity and ISA.

## SENSOR PDU DEFINITION

***Header*** - Standard header.

***Originating Entity ID & Receiving Entity ID*** - Conveys where the information is being sent to and from. For example, an entity ID associated with a tank sends sensor information to an entity ID associated with a fighter jet.

***Number of Fixed Sensor Datum Records (N) & Number of Variable Sensor Datum Records (M) - Fields that tell*** you how many fixed sensor datum records and variable sensor datum records there will be. This means that the total size of the PDU is variable.

***Fixed Sensor Datum Records*** - *N* number of fixed size sensor records which contain information about the sensor including:

**Capability** - what type of sensor information is conveyed (e.g. information about the sensor, its surroundings, command invocations etc.)

**Sensor ID** - Distinguishes which sensor the data came from. This number would be manually / arbitrarily assigned to be unique.

**Sensor Datum Grouping ID** - Further categorizes the data so that it can be associated with other groups of data (e.g. position information may want to be associated with a battlespace object, or perhaps an event detection so that we know which position is associated with which object or event)

**Fixed Sensor Datum Value** - the actual value or payload of information that fits into 32-bits. Note that this value has no specific type, but that could be determined by capability.

***Variable Sensor Datum Records*** - *M* number of variable size sensor datum records. Here the term *variable* means the actual size of the payload itself. It also contains information about the sensor including:

**Capability** - what type of sensor information is conveyed (e.g. information about the sensor, its surroundings, command invocations etc.).

**Sensor ID** - Distinguishes which sensor the data came from. This number would be manually / arbitrarily assigned to be unique.

**Sensor Datum Grouping ID** - Further categorizes the data so that it can be associated with other groups of data (e.g. position information may want to be associated with a battlespace object, or perhaps an event detection so that we know which position is associated with which object or event).

**Variable Sensor Datum Length** - length of the payload in bits.

**Variable Sensor Datum Value** - the actual value or payload of variable length information. Note that this value has no specific type, but that could be determined by capability.

**Padding** - ensures that the datum is padded to 64-bit alignment.

**Table 1 Proposed Sensor PDU Fields**

| Size (bits) | Sensor Data PDU fields | |
|---|---|---|
| 96 | **PDU Header** | Protocol Version—8-bit enumeration |
| | | Exercise ID—8bit unsigned integer |
| | | PDU Type—8-bit enumeration |
| | | Protocol Family—8-bit enumeration |
| | | Timestamp—32-bit unsigned integer |
| | | Length—16-bit unsigned integer |
| | | Padding—16 bits unused |
| 48 | **Originating Entity ID** | Site—16-bit unsigned integer |
| | | Application—16-bit unsigned integer |
| | | Entity—16-bit unsigned integer |

| | | |
|---|---|---|
| 48 | **Receiving Entity ID** | Site—16-bit unsigned integer |
| | | Application—16-bit unsigned integer |
| | | Entity—16-bit unsigned integer |
| 32 | Number of Fixed Sensor Datum Records (*N*) | 32-bit unsigned integer |
| 32 | Number of Variable Sensor Datum Records (M) | 32-bit unsigned integer |
| 64 | Fixed Sensor Datum #1 | Sensor capability—16-bit enumeration |
| | | Sensor ID—8-bits |
| | | Sensor Datum Grouping ID—8-bits |
| | | Fixed Sensor Datum Value—32-bits |
| . …. | | |
| 64 | Fixed Sensor Datum #*N* | Sensor capability—16-bit enumeration |
| | | Sensor ID—8-bits |
| | | Sensor Datum Grouping ID—8-bits |
| | | Fixed Sensor Datum Value—32-bits |
| $64 + K_1 + P_1$ | Variable Sensor Datum #1 | Sensor capability—16-bit enumeration |
| | | Sensor ID—8-bits |
| | | Sensor Datum Grouping ID—8-bits |
| | | Variable Sensor Datum Length—32-bit unsigned integer ($K_1$) |
| | | Variable Sensor Datum Value—$K_1$ bits |
| | | Padding |
| …. | | |
| $64 + K_1 + P_1$ | Variable Sensor Datum #*M* | Sensor capability—16-bit enumeration |
| | | Sensor ID—8-bits |
| | | Datum Grouping ID—8-bits |
| | | Sensor Variable Datum Length—32-bit unsigned integer ($K_M$) |
| | | Variable Datum Value—$K_M$ bits |
| | | Padding |
| Total Sensor PDU size = $256+N+M+i=1M⌈Ki64⌉64bits$<br>Where<br>*N* is number of fixed sensor datum records<br>*M* is number of variable sensor datum records<br>$K_i$ is length of variable sensor datum value *i* in bits<br>$P_i$ is padding to the 64-bit alignment, which is equal to $⌈Ki64⌉ 64−Ki$<br>$⌈Ki64⌉$ is largest integer < $x+1$ | | |

**Sensor PDU Family**

We designed three different types of Sensor PDUs, by reusing the representation concept of ISA sensory data. A Sensor Command PDU is used for sending commands, while the Sensor Query PDU is used only for sending queries that will not be executed. They are both the same in terms of layout and definition, but the key difference is that when an entity receives the Sensor Command PDU, the entity will actually try to execute the command (or the sensor rather). For example, imagine two exactly identical PDUs, but the only difference is one is a Sensor Command PDU and the other a Sensor Query PDU. In the former case, when the simulated entity receives the PDU it will execute the command, and send a Sensor Data PDU as a response that the command was successful; while in the latter case, only a response showing which sensors can do the command will be returned also in a Sensor Data PDU, but not executing the command.

Both Sensor Command PDU and Sensor Query PDU work with Sensor Data PDU, either as a response PDU to the Query and Command, or just for publishing data in general. An example scenario we used to explain how the new Sensor PDUs can be used on the DIS network.

**Example Scenario Used to help Explain the PDU and Its Use Case**

Suppose there are two simulations participating in an exercise which are communicating via DIS. In one simulation there are a few entities: a tank with a number of sensors attached (though only one is depicted in Figure 1), and some soldiers. The other simulation is of an aircraft in which communication between itself and the tank is required to determine what's going on at the ground level. In reality, this may consist of radio-chatter followed by some exchange of data, but since this is
a simulation, the information must be managed by the simulations separately. In a typical DIS exercise, simulation management PDUs may be used, however these PDUs were never meant for transmitting complex sensor queries, commands, and sensor data. Therefore, we think it is useful to use a Sensor Query, Sensor Command, and Sensor Data PDU respectively. The only differences among these PDUs are their PDU type, which is specified in the DIS header, and their use case.

The exchange of Sensor PDUs are explained by the following simplified versions of the PDUs outlined in Figure 2 (for a rigorous definition of the Sensor PDUs see supporting information). In the scenario depicted in Figure 1, if we suppose that the aircraft wants information about what's going on at the ground level, it may wish to supply the sensor attached to the tank (denoted with sensor ID 1) an observe command. The PDU which is sent must convey that it wishes to do this command, along with some positional information about where to observe. Looking at Figure 2, first and foremost, the PDU must have the standard DIS header. It is non-standard in the sense that the part of the header relating to PDU type (not shown) must now be one of the unused enumerations, but it is otherwise the same. It must supply where the command originated from, and what entity whose attached sensor is receiving the command. In this case the entity ID of the originating command is 1 since we previously denoted this as the entity number of the aircraft, and the receiving entity ID is 2, because that is the entity number of the tank (note that actual entity IDs also contain bits about site and application, but this is left out for simplicity).

Next the PDU must contain the number of Fixed Sensor Datums and Variable Sensor Datums. In the case of the Sensor Command PDU, the data of geo-position (either being a string or a data structure) does not fit into 32-bits, so one Variable Sensor Datum is used. The Sensor Capability within this Variable Sensor Datum, is filled with the OBSERVE_COMMAND--which, in reality, is a 16-bit enumeration that can be further subdivided if needed into observables, properties, commands, etc. Note that the actual definitions of the enumerations are arbitrary (for a full list of possible capability enumerations see supporting information capability table). It should also be noted that since the Sensor Command PDU is being sent instead of the Sensor Query PDU, the OBSERVE_COMMAND will actually be attempted by the sensor, as opposed to just a response as to whether or not a sensor on the tank entity (of ID 1) has the ability to execute observe commands. The Variable Sensor Datum of the Sensor Command PDU also contains the sensor ID, which is important, as there may be many sensors on any one entity. The Sensor Group ID is NULL, but as we will show this is not always the case. Finally, the length X of the variable datum is determined upon PDU packaging, and the padding ensures that the PDU is 64-bit-aligned. In response to the Sensor Command PDU, the simulation logic written for the tank will respond with a Sensor Data PDU. In this case it happens to use a Fixed Sensor Datum since the data for the command acknowledgement is relatively simple. Note however that the originating entity ID and the receiving entity ID are now reversed.

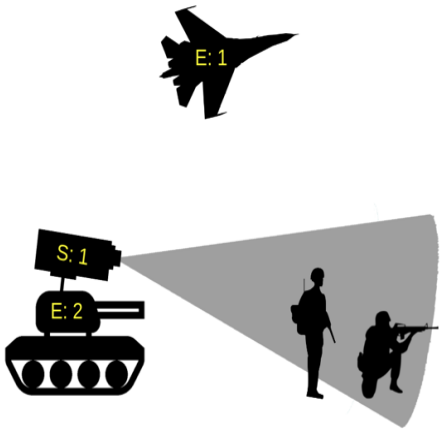


**Figure 2 Example Scenario**

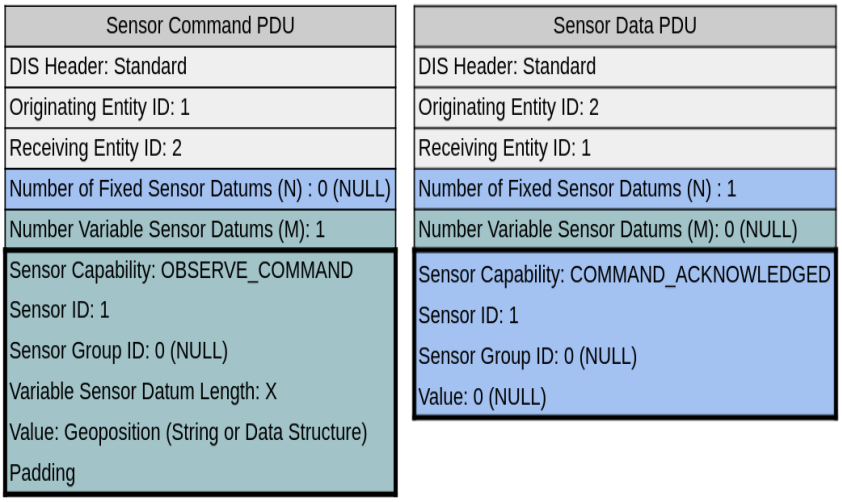


**Figure 3 Sensor Command PDU and Sensor Data PDU**

Since the logic for the tank has responded that its sensor 1 is now observing the geolocation specified, some time may pass before another Sensor Data PDU is sent by the tank, however the aircraft is now listening for this response. After some time, the logic for the tank simulator decides to send information from sensor 1 (outlined in Figure 3), which now contains quite a bit of information. It contains a Fixed Sensor Datum which contains a Sensor Capability: PROPERTY_FOCUS of the camera. The value of the focus is packaged into the 32-bit field, however in the actual implementation one may interpret the bits as a floating-point number (shown in Figure 3 as 83 %).

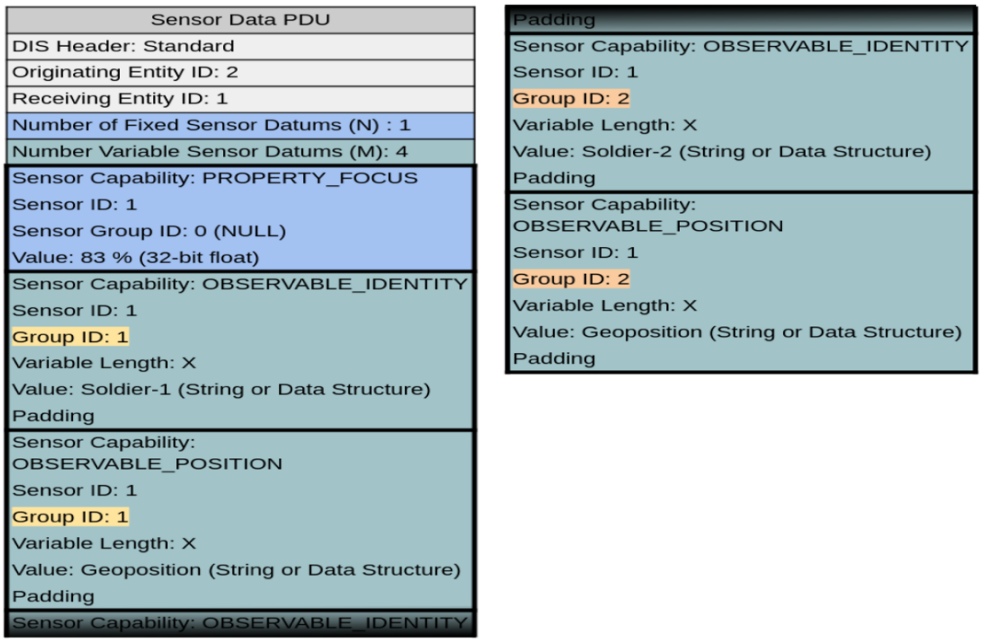


**Figure 4 Sensor Data PDU with Extra Fields**

The PDU also contains four Variable Sensor Datums which sensor 1 has observed. Namely, the Identities which were observed, and their respective positions. The sensor ID is still a key part for each of these datums because there's a possibility that other sensors could have sent similar information (even within the same PDU). Additionally, we see the use of the Sensor Group ID (outlined in Figure 3). This is necessary to associate the observed position with its respective identity. More intuitively we can interpret the whole PDU as follows: the sensor (with ID 1) attached to the tank, observed soldier-1 at some position, and soldier-2 at some other position. Additionally, sensor 1 gave a focus value. Finally, this information was sent from the tank (entity ID 2) to the aircraft (entity ID 1).

**Experiment of Sensor PDUs Using Unity and ISA**

The purpose of the previous example was to demonstrate how these Sensor PDUs might be used in practice by looking at a hypothetical scenario. That example does not convey any of the difficulty associated with implementation and integration with an already existing technology. Here we demonstrate how one might use the Sensor PDUs with an Integrated Sensor Architecture (ISA), and Unity. First a note about why these technologies were chosen. Sensors by themselves are not particularly difficult to integrate with some other existing technology, however, maintaining security, querying, commanding, and listening to many sensors on a network requires a robust system which the ISA provides with their API written in java. Furthermore, the ISA is a real technology which is in no way related to Sensor PDUs (or DIS for that matter). Therefore, we thought it was a good choice for managing sensor data. At the other end, Unity is a general purpose game engine that is used in the military simulation space. Though some other simulation engine could be used by simply writing a networking back end to receive Sensor PDUs, Unity is simple and easy to use and provides the best testbed for creativity and experimentation.

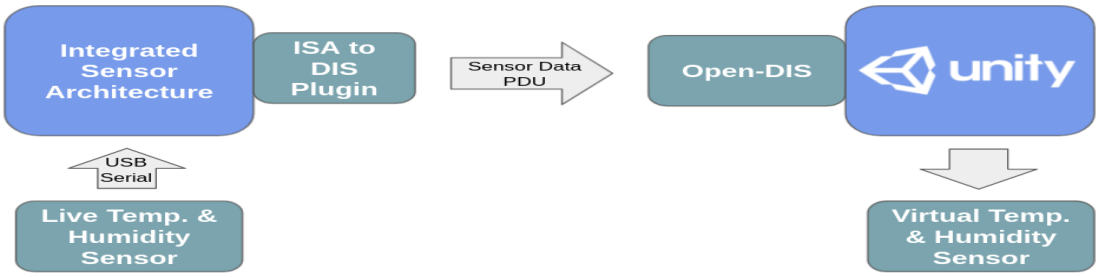


Figure 5 Testbed Design

To begin, the Arduino microcontroller was used with a DHT11 temperature and humidity sensor, and a 16 x 2 backlit LCD (for full schematics see https://create.arduino.cc/projecthub/ThothLoki/portable-arduino-temp-humidity-sensor-with-lcd-a750f4). The code was modified such that temperature and humidity readings were sent to both the LCD and the serial port.

The general flow of the experiment is outlined in Figure 4. The serial port on the Arduino first connected to the computer running an instance of the ISA network controller. A separate process was spawned on this same computer which initialized a component on the ISA network. As part of the protocol of ISA this component made a secure handshake with the network controller already running. Since components are customizable with ISA, a Java serial communication library was used to listen for data (9600 baud rate) on the USB serial port connected to the Arduino. This data was then published on the ISA network. An ISA to DIS plugin was written whose job it was to subscribe to the network for published sensor data, directly querying for the data specified from the universal component identifier (UCI) of the component publishing the sensor data. This plugin was spawned in yet another process. As per the logic written for this ISA to DIS plugin, information was packaged into the Sensor Data PDU. Since the data for temperature and humidity was simple integer values, two Fixed Sensor Datums were packaged in the PDU, one with sensor capability: ATMOSPHERIC_TEMPERATURE and another with ATMOSPHERIC_HUMIDITY.

## RESULTS

Once the information was packaged into the Sensor PDU it was then broadcast over the LAN on port 3000 and received on a separate computer running Unity. Unity's built-in library for listening to UDP packets was used, and these bytes were buffered and sent to a Sensor Data PDU processor. This could then be sent to a mesh of the virtual LCD which displayed the data (Figure 5). The Sensor Data PDU had to be implemented in both Java (for the ISA API) and C# (native language for Unity). This was fairly straightforward for two reasons. The first being that Open-DIS was used, an open-source implementation of the DIS standard protocol in languages such as Java, C++, Python, and C#. The second reason is that the Sensor Data PDU was modeled after the Data PDU. This meant that code could be cleanly modeled after an existing implementation, with the addition of the separate fields for things like sensor capability, sensor ID, and so on.

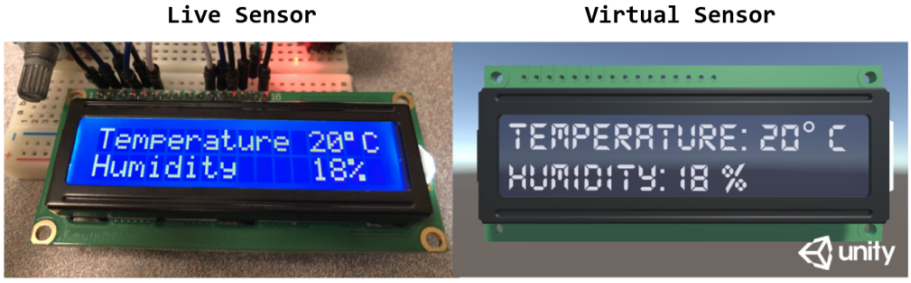


**Figure 6 Live Sensor and Simulated Sensor**

Our goal is to create a new Sensor PDU famly that can be potential used by many types of sensors starting from bringing ISA sensors to the Virtual Training systems such as STE, by utilizing the existing Data PDU and reusing ISA sensor data models. From this IRAD project, we've learned that it is valuable to create a new PDU type just for sensor so there is a standardized method to bring sensory data to the DIS world for interoperability purposes. This PDU provides meaningful data fields in a carefully designed data structure to identify and classify sensor messages on the DIS network. Our use cases provided the evidence that sensory data of a live temperature on the DIS network can be represented in a Sensor PDU and the simulation engine can receive the DIS package and decode the message and update the virtual representation of the sensor with the received data in a timely manner.

We believe that if simulation builders use the newly designed Sensor PDUs to represent various sensory data, there will be a lot of time saved from hacking the Data PDU every time a new type of sensor is needed to participate in distributed exercises. Time saved on using a standard PDU will allow simulation modelers to concentrate on implementing new and advanced features, rather than rewriting DIS code over and over with their own interpretation of how sensory data is represented.

## DISCUSSION AND DIRECTIONS FOR FUTURE RESEARCH

Result of the current effort highlights several important findings as they related to the design and development of a new Sensor PDU family and the process of getting the new PDU type to be standardized. First, using ISA's sensor data model representation as the base of the new PDU design proved to be a great choice because working with sensors requires an understanding of cyber security as well as a robust system for sensor querying, commanding and listening to other sensors on network, and ISA provides all we needed for understanding and reusing sensor data representation.

Second, results showed that a new Sensor PDU family is needed to properly represent sensor capabilities, such as Sensor Command PDU, Sensor Query PDU and Sensor Data PDU. As we have learned, a typical DIS exercise, simulation management PDUs were never meant for transmitting complex sensor queries, commands, and sensor data. Therefore, designing a family PDU that encapsulates all sensor capabilities was needed. We reused the three sensor categories that ISA uses to classify its sensors such as Property, Observable and Command ans created three types of Sensor PDUs that we think can be expanded as needed for future development.

Third, our result demonstrated that testing the new Sensor PDU using a well-defined scenario on an Open DIS-enabled network was effective. We were able to connect the live sensor to ISA securely, and then use the ISA-DIS plug in we developed for NVESD to publish the sensor data to the DIS network. The Sensor PDU contains the sensory data properly representated. The sensor data was received by the Unity engine quickly and simulated sensor data values were updated almost instantly. This result proved that our design of the Sensor PDU has all fields and data types correclt implemented. Future research should explore to data fields with complex sensory data types to represent more advanced sensors that monitors and measure many sources of data such as Atmospheric, Weather, Smoke and Fire, IR Blooming, Dynamic Terrain etc. ISA provides an impressive list of such complext sensory data representations that we can reuse to further our study and implementation.

In addition to the sensors presented on ISA, the U.S. military has also been investing an enormous resources to develop the next generation of optics and weapons that use advanced sensing technologies. One requirement for STE is to allow soldiers to train, rehearse and fight with Goggle sensors that equipped with a heads-up display that utilizes augmented reality to identify potential targets, find ranges and enable synthetic training. Sensors that can track heard and respiration rates and can also detect concussions are in order too. As new sensors are being developed and introduced to the training worls, it is imperitive to bring these features to the military training's DIS world so the advancement can be used with the existing simulation systems.

## CONCLUSION

Sensors are a crusial part of modern military training. The US military organizations such as NVESD has made investments over the years to secure superior sensor modeling technologies to facilitate tactical decision making, theater dominance and winning war outcomings. NVESD's ISA provides credible, realistic sensor models with carefully designed sensory data representation. Recent U.S. Army's STE also demands a system of sensors to detect, communicate and converge fires across domains. STE is going to utilize a suite of well developed simulation systems including Live, Virtual and Constructive training systems, and most of these training systems interoperate over DIS using PDU packets. In order to utilize ISA sensors in STE, a ISA-DIS plugin was developed for NVESD to bring desired sensor data to the DIS world using PDUs. After successfully designed and developed the plug-in which provides a pathway for ISA sensors to interoperate with simulation systems such as OneSAF and VBS4, we discovered a need to develop a new family of DIS PDU, the Sensor PDU. Understanding the validity of ISA sensor models, we decided to reuse its well-designed sensory data representation as much as we can while designing a new PDU family that is modeled after the Data PDU, and to keep its data structure expandable. After carefully selected a set of tools/systems to use, we designed a testbed that utilizes ISA, Open DIS, a live temperature and humidity sensor and Unity game engine. A scenario was designed to test how well the live sensor data was packaged, transmitted to the game engine over a DIS network. Notably, this new Sensor PDU captured and delivered its live data and the Unity-simulated sensor received the data and updated its sensor value accordingly. The findings reported in this paper underscore the importance of a standardized Sensor PDU family to encapsulate existing, developing and future sensor capabilities to the virtual training environment enabled by DIS.

## REFERENCES

Harkrider, S, Krapels, K, Krug, A & McGlynn, L (2016). Sensor Life Cycle Acquisition and Training with Modeling and Simulation. *Journal of Cyber Security and Information Systems*, *Volume 3*.

Kovach, J & Sadler, L (2019). Integrated Sensor Architecture (ISA) Database/Media Storage Tool Software Package Documentation.

ISA Data Model Specification, Release 6.0, April 1st 2020. Retrived from DI2E.

SISO, Simulation Interoperability Standards Organization, The Complete DIS PDU Guide, retrieved from https://www.sisostdsorg.org

AUSA STE Announcement 2021, retrieved from https://www.ausa.org/publications/synthetic-training-environment