CALYTRIX PROFESSIONAL WHITE PAPER SERIES

Systems Integration Primer for DIS/HLA Simulations

In a perfect world all DIS/HLA simulations would immediately work together. The reality, however, at the coalface of simulation integration is quite different. This technical paper introduces the reader to the fundamentals of distributed simulation; the various technical tasks associated with systems integration; and highlights many of the pitfalls to be avoided.

Systems Integration Primer for DIS/HLA Simulations: **More than just blue string**

Take two simulations, add an experienced simulation engine, sprinkle with standards and stir.

The task of setting up and deploying an LVC environment still remains a systems engineering problem. While the fundamental challenge of standards compliance may have been, for the most part, resolved there is still a significant engineering task to ensure that the connected systems work in a consistent manner without introducing any negative lessons or unfair advantages between the participating systems.

Overview:

In modern training centers, particular in a military context, the use of multiple simulations to deliver a complete training solution is common place. For example, constructive simulation environments (e.g. OneSAF or MASA Sword) are routinely used to drive computer generated forces which in-turn are integrated with virtual first person environments (e.g. VBS or Steel Beasts) as well as stimulating live Command & Control (C2) systems in the real world. Collectively, the simulation world, this type of systems-of-systems approach is referred to as LVC and is an underpinning capability in modern complex military training.

At a marketing level the integration of different simulations, almost always from different vendors, is trivialized to almost 'plug and play' and point to simple standards compliance (notably DIS & HLA, see below), however the realities are quite different. An analogy would be to say that you could take the engine out of any car and put it inside any other. While technically this might be correct the integration costs and the mechanics bill would be substantial. To a an extent the same is true of connecting simulation systems.

While it can be reasonably straightforward to connect two or more simulations over a network and exchange data via DIS or HLA (the network layer), this is just the first step. The systems integrator must also consider:

- Which integration standard to use,
- Entity mappings,

- Entity ownership,
- Entity aggregation,
- Damage models,
- Correlated terrain, and
- ‘Game’ box sizes.

Each of those items need to be considered when looking to integrate LVC systems. This paper introduces the various tasks required to successfully integrate simulation systems in order to deliver a unified LVC environment.

A Background in Distributed Simulation:

The various standards that support distributed simulation exist to allow otherwise separate applications and simulators to work together in a richer, broader environment.

There are several reasons why you may wish to connect simulations together:

> **Linking People:** Often highly specialized simulations or Part-Trask Trainers (PTT) are used to provide focused training on a dedicated training area. To enable trainees to co-operate in a larger training environment, it is often desirable to link these simulations together to create a broader training curriculum and/or team training environment. Individual simulations providing specialized training services can be linked with other simulations to deliver a different set of specialized services to an extended audience, bringing together a larger group of participants who can benefit from interaction with each other.
>
> **Linking Capabilities:** Similarly, when considering a smaller training audience one simulation can provide a particular set of features while another provides a complimentary set. A more effective training solution is achieved by having the two separate systems communicate with one another. One such setup might have OneSAF being used to model semi-automated ground forces while the JSAF simulation environment is used to model maritime assets. Combining otherwise standalone simulation software allows users to leverage both the expertise and investment in individual systems, and in this example conduct an effective joint activity.

However, while many simulation systems have the means to communicate with other instances of the same application (via a proprietary protocol), linking them to other software suites is a more difficult proposition. This is why the Distributed Interactive Simulation (DIS) and High-Level Architecture (HLA) international standards exist.

DIS and HLA prescribe a way for generic simulation systems to exchange common data in pursuit of integration with one another. Via DIS or HLA, one system should be able to visualize and interact with entities created, modelled and controlled by an entirely separate application (both physically and in terms of the underlying software).

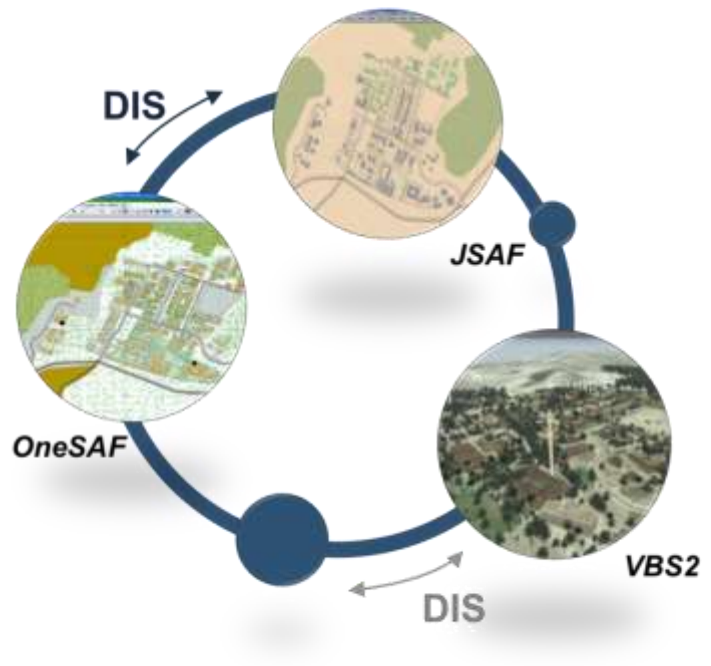


FIGURE 1: SEPARATE APPLICATIONS COMMUNICATING OVER DIS

What is DIS?

The DIS protocol is an open standard based around the binary encoding of messages for exchange of entity and event data among disparate simulation systems. Heavily geared towards the military domain, the standard defines the binary structure of packets called Protocol Data Units (PDU) to communicate simulation information.

PDUs describe the status and location of simulation entities, describe a munitions detonation events, encapsulate a radio transmission and so forth. When information needs to be communicated, a PDU is created and sent out to the network where other DIS enabled systems pick it up for interpretation and visualization.

Despite the PDU message set being quite extensive, the basics of DIS are simple. At a fundamental level, PDUs are created and sent out via broadcast or multicast networks for other clients to consume. Entities are created when an individual client detects an `EntityState` PDU sent from another simulation for an entity it has no prior knowledge of. A system will also send out `EntityState` PDUs with the relevant information for all entities it has created locally and controls.

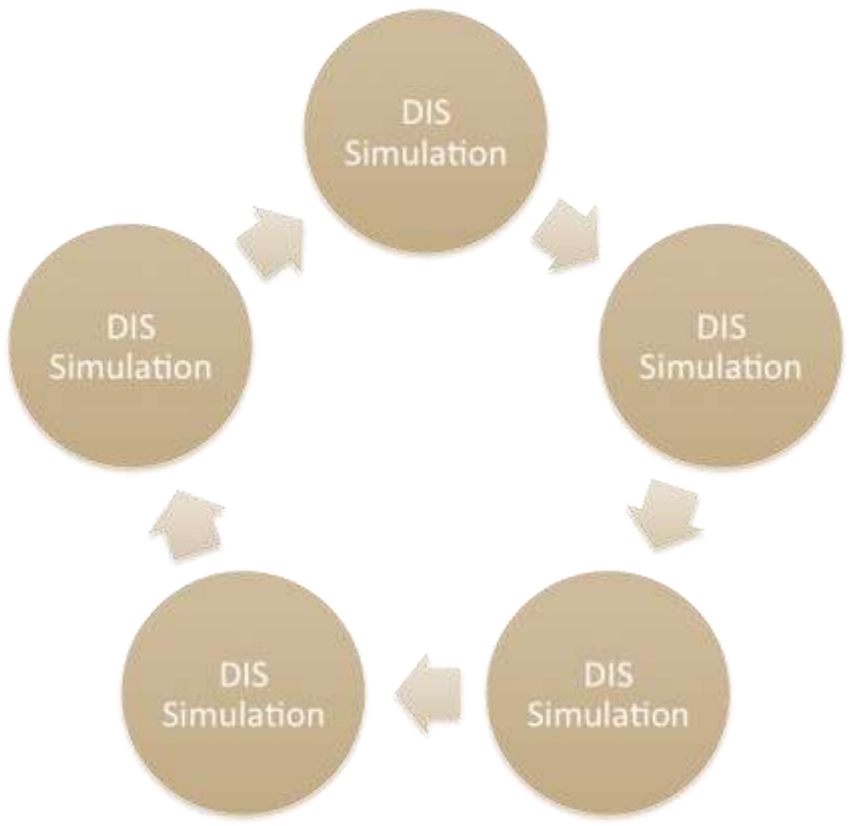


FIGURE 2: DIS SIMULATIONS COMMUNICATING

To join a distributed simulation activity, a DIS client need only connect to the appropriate network and begin listening or sending information on the network. The DIS standard itself makes little to no provision for additional simulation services such as explicit entity lifecycle management, coordinated management of time, data distribution management and filtering, etc.

The Simulation Interoperability and Standards Organization (SISO) currently maintain DIS as an open standard in which anyone can participate.

What is HLA?

The High-Level Architecture (HLA) is another open, international distributed simulation standard. Developed after DIS with a goal of standardizing some additional simulation services (e.g. timing, repeatability etc.), HLA takes a different approach to the specification of a simulation standard.

Where the DIS protocol specifies the binary structure of messages communicated over the network, HLA abstracts this layer and instead specifies an Application Program Interface (API) that defines a set of simulation services. The central component in an HLA distributed simulation is the Run-Time Infrastructure (RTI). Rather than having each simulation send out entity information directly to the network as per DIS, in HLA individual simulation applications (known as federates) all communicate with an RTI component that then handles the process of passing the information to all the other federates.

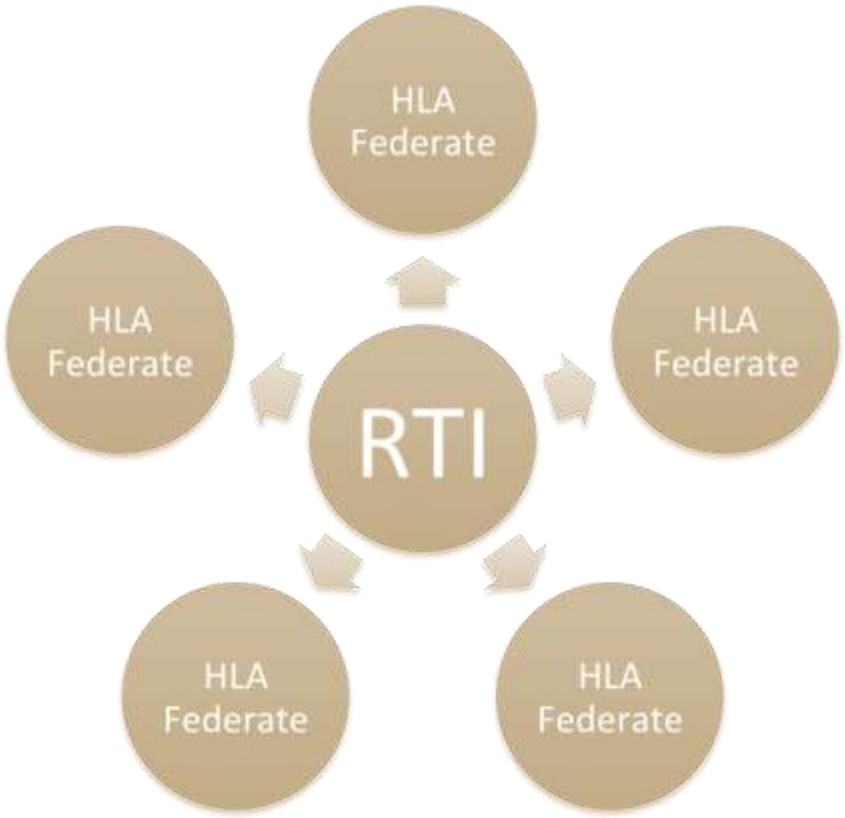


FIGURE 3: CONCEPTUALLY CENTRALIZED COMMUNICATION THROUGH THE RTI

As all communication between federates is routed through the RTI, this component can provide additional simulation services. The HLA prescribes a number of services that federates can avail themselves of through the RTI. These include (but are not limited to):

- Full entity lifecycle control,
- Co-ordinated time management,
- Ownership transfer of individual pieces of an entity for co-operative modeling by multiple federates of a single entity,
- Data filtering based on a publication and subscription framework, and
- Advanced Data filtering based on abstract regions within a simulation world.

The other primary difference between an HLA federation and a DIS-based distributed simulation is that the HLA is model agnostic. In DIS, the communications model is fixed (individual PDUs – the data sent between systems - are defined by the standard). In HLA, when a federation is created, a file describing the data to be exchanged must also be provided. This file is called a Federation Object Model (FOM). This means that while DIS is squarely targeted at the military domain and has a fixed data message set, HLA can be used in any user-defined situation by defining a new FOM and is a more general-purpose distributed simulation framework.

The Realtime-Platform Reference (RPR) FOM (RPR-FOM) is a standard HLA model designed to provide a structure somewhat similar to DIS, but in HLA constructs. Each DIS PDU is reflected in the RPR-FOM data model. It defines the structure of an object model that conceptually parallels the common concepts from a DIS network, to some extent reusing various constructs (such as an enumeration – discussed below). Both DIS and HLA are maintained by SISO as open, international standards and are formal standards under the Institute of Electrical and Electronics Engineers (IEEE).

Which to Select?

It is often joked that one of the great things about standards is that there are so many to choose from, and this is true also in the simulation domain.

In the LVC domain, a simulation integrator will come across both DIS and HLA compliant systems and will often be required to connectthese systems. So it isn't so much a question of which standard is better, as they both present pros and cons, but rather working within the practical constraints that both systems presents. It should be noted that many systems support both standards and some commonly used gateways such as Calytrix™ LVC Game allow the user to easily switch between DIS, HLA and any supported HLA FOM.

Nonetheless, there are a number of factors to be considered when selecting between DIS and HLA:

- DIS tends to be much simpler as it is a wire standard over a multicast network, while HLA requires the installation and configuration of an RTI;
- If the HLA simulation is not a RPR FOM derivative a mechanism will need to be found to translate its message calls across to the DIS PDU network. This may require the development of a custom gateway to translate the FOM data types into their equivalent and fixed DIS PDU types;
- A good understanding of the network type. For example, DIS is a multicast protocol and will run on a LAN or specially configured WAN (a non-trivial configuration task), while HLA can deliver a point-to-point service;
- Cost may be a factor. HLA requires an RTI and while there are a number of free RTIs available, notably the Portico RTI, the bulk of RTIs are commercial. RTIs are usually licensed on a per-federate (or simulation) basis and these costs can quickly add up;
- There may be some interoperability issues between the various commercial RTIs and the version of HLA (1.3, 1516 and 1516E) that exist. While a lot of work has been done to address these issues it is still a factor to consider; and
- Both DIS and HLA come with different support tools for diagnostics, logging, gateways and network bridging. The availability of tools may impact the decision process.

The above points are just a few of the considerations that need to be taken into account when selecting between DIS and HLA, noting that in many instances a gateway can be used to bridge between the two. Indeed, the decision will often be driven by the choice of simulation in use, rather than the underlying protocols. However, once the simulation systems are 'talking' the next challenge is to make the integration meaningful between the various systems.

Enumerations and Mappings

The process of linking together separate simulation systems involves the exchange of entity information in some common format and typically this is done using DIS or HLA. Although they are considerably different in the way they facilitate communication, there is one common thread that ties them together with regard to application integration: Enumerations.

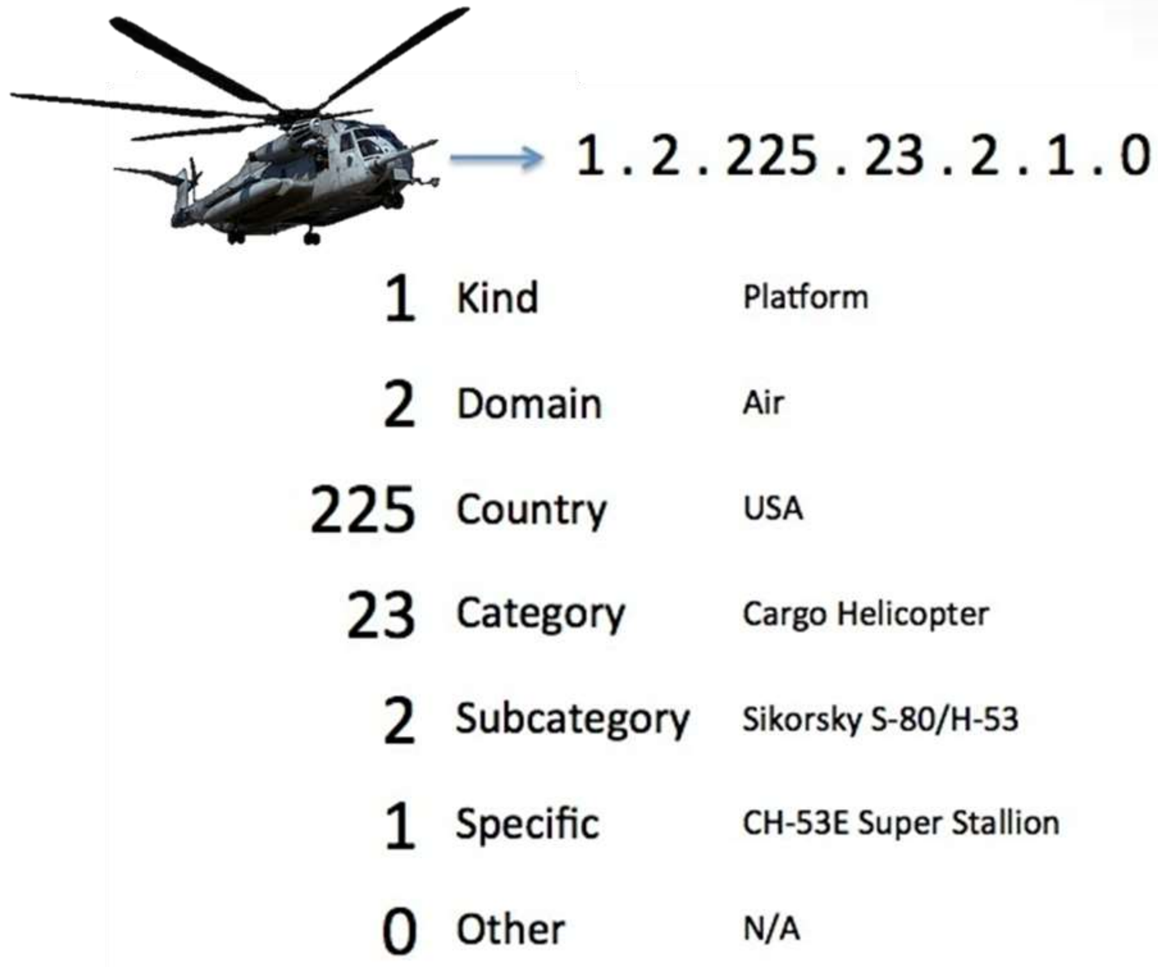
When sharing information about an entity between systems, applications need a way to specify *what* that entity is. If there is an entity representing an M1A1 Tank in one application, that information needs to be communicated to the other application so that it can display that entity using its local representation of an M1A1. Ensuring that the entities appear correctly across many simulation systems is perhaps the key to enabling integration.

An Enumeration is a series of numbers that forms the common language for communicating what an entity is. When describing an entity, a simulation system will include its enumeration value to delineate its kind. Other systems can then use this number to determine what local representation they should use for the entity to ensure that it appears correctly. Enumerations are a core part of the DIS specification, and although they are not part of the HLA specification itself, they are part of the RPR-FOM that is used as the common interoperability model for standard military simulations.

As an example, the Enumeration for a US CH-53 Helicopter is as follows:

```
    1.2.225.23.2.1.0
```

Reviewing these numbers from left to right the categorizations move from broad to increasingly specific. The following figure shows how this enumeration is broken down:



1	Kind	Platform
2	Domain	Air
225	Country	USA
23	Category	Cargo Helicopter
2	Subcategory	Sikorsky S-80/H-53
1	Specific	CH-53E Super Stallion
0	Other	N/A

Figure 4: Enumeration Breakdown

One of the primary tasks a systems integrator must complete when linking together two systems is to ensure that the mappings for each system are correct. A mapping is a link between an internal representation of an entity and the enumeration value to use for it.

There are two types of mappings: incoming and outgoing. Outgoing mappings make a link describing which enumeration to use for a local model when an application creates an entity it wants to share. For each entity-type in a system, there can only be a single outgoing enumeration mapping (as an entity can only be advertised as a single type). For systems receiving remote information about entities, they must map from the incoming enumeration to the local representation. A single local type could be used to represent many different remote types (for example, there are many variations on M1A1 tanks, but in many games there is just one model). Thus, there can be many incoming mappings for different enumerations to a single local type.

Common Mapping Problems

When attempting to integrate many separate simulation systems into the same synthetic environment, integrators often face a number of largely misunderstood or forgotten problems. This section outlines some of the primary integration concerns that can cause integration issues in multi-system environments.

Entity Mapping Disconnect

When integrating separate simulation systems, an integrator often has to deal with the mismatch of purpose. Some systems, such as OneSAF, have a rich set of entities for describing ground troops, but a minimal set for sea assets. Other systems such as JSAF have a broader sea assets. An entity or munitions type that doesn't exist in a remote system cannot be displayed in the local system, thus the level of achievable integration is typically defined as limited by the amount of overlap that exists between any two given systems.

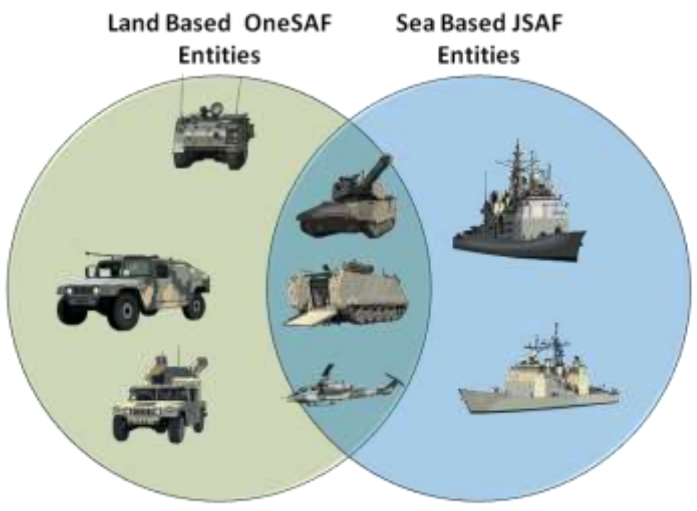


Figure 5: Entity Overlap between OneSAF AS and JSAF AS

One approach to dealing with the problem is to identify an appropriate replacement entity. In an individual exercise scenario, decisions about how close an entity mapping has to be in order to be considered "*correct*" is made by the designers of that training activity. Typically, compromises on correctness are made in order to present a situation that is approximately visually correct, with the errata able to be provided in the initial briefing for activity participants. The level of tolerance for what defines "correct" mapping varies considerably depending on the particular approach of individual exercise administrators and the nature of the activity (training versus analysis). This approach is by its very nature rather ad-hoc.

Using the above OneSAF / JSAF example, the following rules are typically applied when determining the mappings for the exercise construct:

1. **EXACT MAPPING**

OneSAF Model	← →	JSAF Model
ASLAV	← →	ASLAV

2. **NEAREST MAPPING**

OneSAF Model	← →	JSAF Generic Model
ASLAV	← →	M2 Bradley

3. **NO MAPPING**

OneSAF Model	← →	n/a
ASLAV	← →	n/a

In this process, the compromise typically occurs at step 2, where a "nearest acceptable alternative" is chosen. In the example above, while the Bradley may provide an appropriate synthesis in that it has the same turret, whether or not it is acceptable depends largely on the desired use. For training purposes when integrating with a system that has limited visualization capabilities such as JSAF, this may be acceptable as it presents very little difference to the end user. However, when using a 3D virtual simulation such as VBS3, the difference might become immediately apparent and may introduce negative lessons.

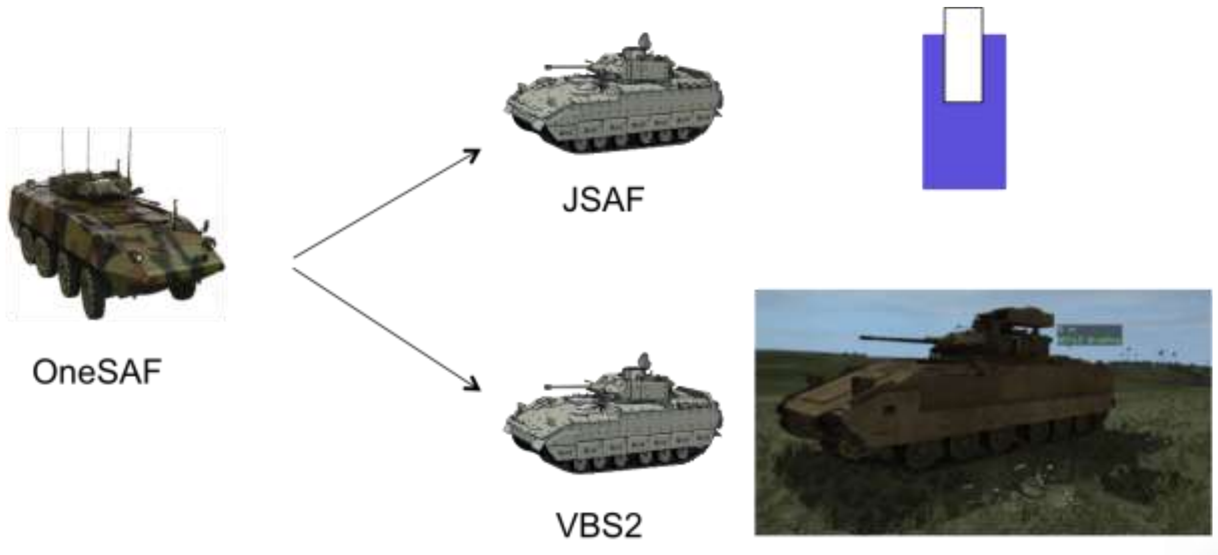


FIGURE 6: THE EFFECT OF ALTERED MAPPINGS

Whether or not the use of a nearest mapping alternative is acceptable depends entirely on the parameters of the situation for which the entity is being configured. For a short-lived situation such as an exercise, the use of the nearest alternative may be deemed adequate. Depending on the focus of the exercise (land, air, sea) the priorities for what must be mapped and what is optional will change.

For situations that are attempting to create an ongoing standard set of mappings, the use of the closest match may not be desirable as it creates a false expectation about the completeness of that set and as further simulation systems, with richer sets of representations become available, the ability to display remote information correctly changes.

This task of mapping enumerations grows as more systems are connected. Often the integrator is forced back to the lowest common denominator, which is often the 3D visual system(s) as this is where mapping mismatches are far more pronounced.

Missing Munitions and Mappings Disconnect

Another common and less obvious problem when attempting to integrate many different simulation systems is the effect of incorrect munitions mappings.

Much like entities, information about munitions transferred between systems hinges heavily on the use of an enumeration to describe their type.

Typically, a firing event is generated specifying an enumeration that describes the type of munitions that were discharged, the originating entity of the event and the target entity and location (where the target entity is not provided, indirect fire). Unlike the case for entities, it is not vitally important that a receiving simulation system have an appropriate mapping from the enumeration to a local munition type in order to display the event. It is very common for individual munitions not to be displayed in a system. Firing lines from the source to target might be displayed, but visualization of the individual munition, e.g, a 155mm artillery shell, is uncommon[1].

Munition mappings become important when attempting to calculate damage. All damage for an entity is calculated by the simulation system that created it. Thus, if you have a tank in VBS2 firing on a tank in Steel Beasts, it is Steel Beasts that decides whether the target tank receives any damage or not.

For these calculations to occur correctly, the receiving simulation system must:

- Have knowledge of the munition type (so that it can make the appropriate calculations), and
- Have the appropriate mapping from the munition enumeration to the local type.

[1] For munitions that are larger, such as torpedoes or missiles, a separate entity will often be created to represent them. This entity itself has an enumeration type and is treated by most simulation systems the same as any other type of entity.

If either of these aspects are missing, it will result in what appears to be indestructible entities as the simulation system that created the entity becomes unable to calculate the required damage because it either can't represent the munition and doesn't know how it affects local entities, or it can't translate from the enumeration to the appropriate local munition type.

Exasperating the problem is the expectation of how multi-system entities interact. If a relatively simple simulation system is firing on tanks from a complex system, those entities may appear much harder to kill than ones created in the local simulation.

As en example, consider Steel Beasts virtual 3D environment: it provides extremely high fidelity modelling of ballistics impacts on armored vehicles. The level of accuracy and effort required to destroy a tank in Steel Beasts will be much higher than another simulation like VBS2 where complex ballistic impacts are not as important. Where the firing entity is modelled in VBS2, this could mean that due to the fidelity mismatch, the remote Steel Beasts tanks appear much more difficult to hit and destroy than others created by other VBS2 instances. This is despite there being no direct visual way to distinguish between a local tank and a remote representation of a Steel Beasts tank.

These are some primary examples of how the results of misaligned mappings can have a serious effect on the ability to achieve true integration between many systems. Just because two systems are able to connect to a DIS or HLA network, does not mean they can integrate with one another in a meaningful way. Ensuring that scenarios are built using systems that are able to adequately fulfil the desired functionality and making sure that time is allocated to ensuring that the mappings used by all systems are aligned and correct are the only ways to combat the often unrealized side effects of multi-system integration.

Canonical References and Standards for Enumerations

Given the sheer volume of potential interpretations for various enumeration combinations, the Simulation Interoperability and Standards Organization (SISO) has produced a standard document that codifies the appropriate values to use for various enumerations when mapping them to and from their respective types.

The *Enumeration and Bit Encoded Values for use with Protocols for Distributed Interactive Simulation Applications* (SISO-REF-010-2006) document, colloquially referred to as the EBV, defines a set of standard enumeration values that simulations should use for communication. The intent behind this document is to help ensure that the interoperation of separate systems is as smooth as possible by defining the set of known and proper enumeration values.

Despite the presence of a canonical source for enumeration values such as the EBV, it is still common to find situations in which there are departures. While comprehensive, there will always remain systems not covered by the EBV. Where the EBV does not provide a solution, developers and integrators are often forced to make up a new enumeration value for the entity, platform or munitions they are trying to represent. As the standard grows, this problem reduces, but this process is a gradual one.

Further affecting the force and effect of a canonical set of mappings, such as the EBV, is the compromises made over time to achieve higher levels of interoperability. It is common for integrations to depart from the standard in order to achieve higher levels of interoperability (see the discussion on nearest type mapping in the section titled, Entity Mapping Disconnect). Depending on how widely used such systems are, these settings can often become something of a smaller de-facto standard, as other systems attempt to use them as the benchmark to integrate with. This in turn causes these inaccurate settings to gain wider use, thus giving them more weight. Arguments such as “we need to integrate with system X, and system X uses this value” become common and departures from the standard as defined by the EBV prevail with recognized updates to the EBV lagging common practice.

The effective use of enumerations as a mechanism for communicating type information for platforms, entities and munitions helps to aid the coupling and integration of otherwise standalone simulation systems. In common use, there are a number of factors to consider when deciding on an approach for implementing such integration, and these have been outlined above. Another area that is largely forgotten, but has a substantial impact on multi-system integration is that of terrain.

The Effect of Terrain

In order to enable meaningful integration accurate correlated terrain is required. Uncorrelated terrain across systems can impact integration in many ways, from simple visual mismatches to the inability to inflict damage on entities and movement capabilities.

When starting an exercise in a given simulation system, a user or administrator must specify the terrain box that the system will use. This refers to the world in which the simulated entities will operate. It may be desert, it may be urban or it may be coastline based or any combination of these and more.

When attempting to get many systems to communicate and operate in a larger synthetic environment, it is vitally important to ensure that they are all using a terrain set that matches the other systems. This ensures that various terrain features such as hills, mountains, water and buildings all appear in the same location across all systems. Having terrain that matches across systems is known as terrain correlation.

The problem is that each specific simulation system will likely require terrain in a format that is specific to it. Due to this, the common term of a “correlated terrain set” refers to a set of files that contain terrain that is identical, but in many different file formats (one for each target simulation system).

Without correlated terrain, a number of problems can arise.

Visual Disconnect

All entity movement is communicated by describing the location of an entity within the world (such as latitude, longitude and elevation). If each of the systems have different terrain for a location, visual disconnects can quickly arise. For example, consider two tank troops advancing on each other in the same area of the world. The separate systems are using two different sets of terrain, with one displaying the tanks moving through an urban area while the other displays entities moving across open territory. In such situations it is not uncommon to see entities apparently floating above the terrain in one system due to mismatches in terrain altitudes. While the entities may see each other, they will not appear to be playing in the same area due to differences in surrounding terrain features and structures.

As all modelling (damage and movement) of an entity is done by the creating system, further visually strange behavior can arise. Where one system will sense a building and attempt to send its entities around it, the other system will have no notion that a building exists in that location and be perfectly content to have its entities pass directly through.

Damage Effects

Another major effect of a lack of correlated terrain occurs in the calculation of damage effects. As highlighted in the previous section, all damage calculations are performed by the system that created the entity. Consider for a moment a situation where one simulation believes that its target is at ground level where the elevation is sea-level. It is firing at a remote entity, and in the remote simulation system, the target appears to be at an elevation of 10 meters. This scenario is visually depicted in Figure 7 below:

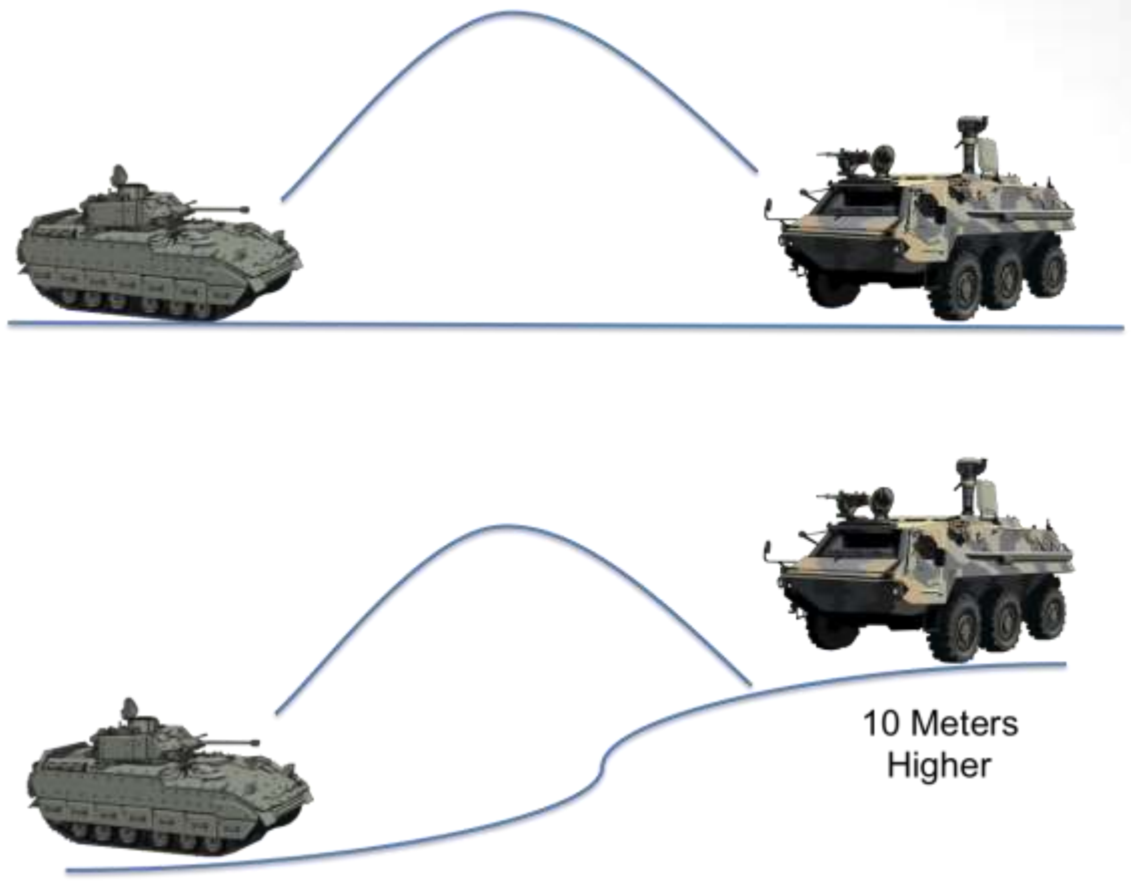


FIGURE 7: TERRAIN CORRELATION MISMATCH

While the simulation representing the firing entity may believe it has fired at the correct location and sends out an event describing a detonation at the appropriate location with an elevation of 0-meters, to the remote system the target entity is clamped to the ground at 10-meters and thus did not sustain any damage. The same could be true if there were objects in the way, such as a lifeform entity that to one system appears to be in the open, but to another system appears to be indoors. The entity could be shot through the walls.

Without a complete set of correlated terrain, a number of problems, both visual and otherwise can have a significant effect on ensuring that a fair-fight is possible and that the larger synthetic environment appears correctly to all simulations.

Intervisibility

Intervisibility refers to the relative ability to be seen under given conditions of distance, light, and atmosphere. Intervisibility can be a major issue between systems, especially in the virtual domain where the simulations typically render very high-detail first-person 3D environments.

A number of factors can influence Intervisibility:

- **3D models.** If the visual representations of platforms (e.g. Tanks) are different between systems the player may react differently. Differences may include the size, color, visual render and even weapon systems. This problem can persist even when the entity models are accurately mapped as the models are often specific to the system (e.g. VBS and Steel Beasts do not share common 3D model files).
- **Foliage:** The density and rendering of trees and other foliage can have a major impact, especially in military infantry systems. A good example is the rendering of trees, as one system may draw more branches and leaves than another, greatly reducing the visibility of the player on this system compared to his opposition who is playing with more sparse vegetation. Like entities, virtual systems do not share any common foliage rendering standards or common models.
- **Weather:** Just as the density of vegetation can impact a player's intervisibility so can the rendering of weather. For example, if one system can render clouds and rain while his opponent is left with blue skies, there is a clear mismatch. The same is true for day and night and the transition at dawn and dusk.

Intervisibility can undermine the ability to deliver a 'fair fight' and in turn undermine the value of the training environment. Careful attention must be paid to ensure that each system represents the visual environment in a consistent manner.

Conclusion

The task of setting up and deploying an LVC environment still remains a systems engineering problem. While the fundamental challenge of standards compliance may have been, for the most part, resolved there is still a significant engineering task to ensure that the connected systems work in a consistent manner without introducing any negative lessons or unfair advantages between the participating systems.

The following online resources will provide a greater understanding of the issues discussd in this paper:

- SISO Website - www.sisostds.org
- The Portico HLA project - www.porticoproject.org
- Free correlated terrain - www.calytrix.com/support/terrain/overview/
- LVC Game - www.calytrix.com

Take two simulations, add an experienced simulation engine, sprinkle with standards and stir.

FOR MORE INFORMATION ABOUT LVC SYSTEMS INTEGRATION:

Contact Calytrix:

Email:	info@calytrix.com
In the US:	+1 321 206-0628
In Australia:	+61 8 9226 4288
On line:	www.calytrix.com