# X3D Graphics and Distributed Interactive Simulation (DIS) Networking

*"All the world's a stage,*
*And all the men and women merely players;"*
- William Shakespeare, *As You Like It*

Don Brutzman
Naval Postgraduate School
brutzman@nps.edu

web|3D CONSORTIUM

# Contents

# Chapter Overview

# Overview

IEEE Distributed Interactive Simulation (DIS) protocol has been used for many years to build networked simulations that share state

X3D DIS component aligns these capabilities with X3D scenes to enable sharing of state data

- EspduTransform: protocol data units (PDUs) for EntityState, Collision, Fire, Detonation
- Signals: ReceiverPdu, SignalPdu, TransmitterPdu

Various open-source tools, codebases available

- e.g. Open-DIS open source in Java, C++, C#, etc.
- Currently only one X3D browser implementation ☹

# Overview: DIS Network Protocol

Distributed Interactive Simulation (DIS) network prototcol is an IEEE standard since 1995

- IEEE 1278 is a communications standard for physically based distributed simulations
- Standard defines the binary layout of a series of messages used to transmit simulation information
- Often used in military applications since IEEE 1278 covers a wide range of data, including entity location, velocity, and orientation, and more features such as signal and supply support
- Also used in civilian applications: air-traffic control, etc.

# PDU defined

Protocol data unit (PDU) generally considered to provide a unit block of information

- Delivered among peer entities on a network
- May contain control information, address information, or data

The IEEE specification for DIS defines a large number of PDU packets

- Standardizes byte format of packets "over the wire" for interoperability among participating simulations

# DIS PDU types

DIS specification defines 67 PDUs in 12 families

- *Entity information/interaction:* Entity State, Collision, Collision-Elastic, Entity State Update, Attribute
- *Warfare:* Fire, Detonation, Directed Energy Fire, Entity Damage Status
- *Radio communications:* Transmitter, Signal, Receiver, Intercom Signal, Intercom Control
- *Simulation management:* Start/Resume, Stop/Freeze, Acknowledge
- etc.

# Virtues of DIS

- DIS is a binary protocol; the exact layout of PDUs in binary format is specified, along with byte order
- Many standardized enumeration values. For example, the value "1" in the Entity Type field of a PDU specifies that the PDU is an Entity State PDU.
- Other fields and enumerated values can define what type of entity is being described, such as an M1A2 tank, an Arleigh Burke class destroyer, etc.
- DIS standard describes functional algorithms for handling PDUs, dead reckoning, and more.
- Thus both syntax and semantics for interoperability are well defined.

# Network communications

Two transport capabilities:

- Multicast (many-to-many on LAN) socket
- Unicast (point-to-point) socket

Entities and ownership

- Entity corresponds to some high-level vehicle that has control over its own motion
- Owners send packets, others listen for packets
- Multiple owners for one entity usually nonsensical

Thus non-symmetric configuration of readers and writers in each participants scene graph

# DIS Enumerations

- SISO-REF-010-00-2013: DIS Enumerations
- This document specifies the numerical values and associated definitions for those Distributed Interactive Simulation (DIS) Protocol Data Units (PDUs) fields that are identified as enumerations in IEEE 1278.1-1995 and IEEE 1278.1A-1998. Compliance with this document is mandatory for participation in a DIS exercise.
- 487 pages (!), steady evolution continues

# SIMNET and DIS

# IEEE, SISO and HLA

# SIMNET

- SIMNET was a wide area network with vehicle simulators and displays for real-time distributed combat simulation: tanks, helicopters and airplanes in a virtual battlefield. SIMNET was developed for and used by the United States military. SIMNET development began in the mid-1980s, was fielded starting in 1987, and was used for training until successor programs came online well into the 1990s.
- http://en.wikipedia.org/wiki/SIMNET

# DIS definition from Wikipedia

Distributed Interactive Simulation (DIS) is an open standard for conducting real-time platform-level wargaming across multiple host computers and is used worldwide, especially by military organizations but also by other agencies such as those involved in space exploration and medicine.

http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation

# DIS History 1

- The standard was developed over a series of "DIS Workshops" at the Interactive Networked Simulation for Training symposium, held by the University of Central Florida (UCF) Institute for Simulation and Training (IST). The standard itself is very closely patterned after the original SIMNET distributed interactive simulation protocol, developed by Bolt, Beranek and Newman (BBN) for Defense Advanced Research Project Agency (DARPA) in the early through late 1980s. BBN introduced the concept of dead reckoning to efficiently transmit the state of battle field entities.

# DIS History 2

- In the early 1990s, IST was contracted by the United States Defense Advanced Research Project Agency to undertake research in support of the US Army Simulator Network (SimNet) program. Funding and research interest for DIS standards development decreased following the proposal and promulgation of its successor, the High Level Architecture (simulation) in 1996. HLA was produced by the merger of the DIS protocol with the Aggregate Level Simulation Protocol (ALSP) designed by MITRE.

# DIS History 3

- There was a NATO standardisation agreement (STANAG 4482, Standardised Information Technology Protocols for Distributed Interactive Simulation (DIS), adopted in 1995) on DIS for modelling and simulation interoperability. This was retired in favour of HLA in 1998 and officially cancelled in 2010 by the NATO Standardisation Agency (NSA).
- Not clear why cancelled since DIS ≠ HLA...
- Credit: DIS site, history maintained by DIS PDG

http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation

# Simulation Interoperability Standards Organization (SISO)

International organization with broad agenda.

- "Simulation Interoperability & Reuse through Standards"

SISO (http://www.sisostds.org) is recognized as

- NATO Standards Development Organization (SDO)
- IEEE Standards Sponsor
- Category C Liaison Organization, ISO/IEC (JTC 1)

SISO maintains and develops the DIS standard via working-group efforts and meetings, especially at Simulation Interoperability Workshop (SIW) series in Orlando FL and Europe annually

# SISO Vision

- SISO will serve the global community of modeling and simulation professionals, providing an open forum for the collegial exchange of ideas, the examination and advancement of M&S-related technologies and practices, and the development of standards and other products that enable greater M&S capability, interoperability, credibility, reuse, and cost-effectiveness.

- Reference: SISO-ADM-004-2010, available via http://www.sisostds.org/AboutSISO/Overview.aspx

# DIS Product Development Group (PDG)

The SISO Standards Activity Committee (SAC) has a DIS Product Development Group (PDG) which updated the core DIS standard, 1278.1, and reintegrated content of 1278.1a, to form approved update standard 1278.1-2012.

DIS standards can be purchased from IEEE (often available free to members, universities) and also available online to SISO members.

# IEEE Standards Maintained by SISO

- IEEE 1278.1-2012 - IEEE Standard for Distributed Interactive Simulation - Application Protocols
- IEEE 1278.2 - IEEE Standard for Distributed Interactive Simulation - Communication Services and Profiles
- IEEE 1278.3 - IEEE Standard for Distributed Interactive Simulation Exercise Management & Feedback (EMF) - Recommended Practice
- IEEE 1278.4 - IEEE Standard for Distributed Interactive Simulation - Verification Validation & Accreditation

http://www.sisostds.org/ProductsPublications/Standards/IEEEStandards.aspx

# Getting copies of DIS standards

IEEE DIS standard documents are

- Available to SISO members without charge
- Available to IEEE members
- Often available to universities via site license


# SISO digital library:

# DIS Product Study Group (PSG)

References section includes several links

# DIS and High Level Architecture (HLA)

DIS protocol defines both wire format and semantics for consistent shared state

- Stateless, entities can join/leave any time
- Interoperability for all compliant implementations

HLA Run-Time Interface (RTI) is for codebases that implement HLA design principles

- Object model principles, no wire format, though DIS packets might be passed internally (RPR-FOM)
- Entities must be predeclared prior to start
- No interoperability guarantee for implementations
- Not an interoperability standard, usually proprietary

# X3D specification:  DIS component

X3D specification is componentized for extensibility, DIS  Component is one of many:

- http://www.web3d.org/standards
- Part 1: Architecture and base components
- Part 28 Distributed interactive simulation component

Provides satisfactory support for primary DIS nodes used in distributed virtual environments

# X3D specification: DIS support

DIS component includes following X3D nodes:

- EspduTransform
- ReceiverPdu, SignalPdu, TransmitterPdu
- DISEntityManager, DISEntityTypeMapping

DIS PDU message types

- Collision, Detonate, Entity State, Fire
  - (functionality bundled together in EspduTransform)
- Receiver, Signal, and Transmitter
- Numerous other DIS PDUs defined by DIS protocol, but corresponding X3D mappings are not defined.

back to Table of Contents

# History:

# DIS-Java-VRML

# History: DIS-Java-VRML

Shared state via distributed simulation has always been a goal of X3D practitioners

The core design and implementation efforts for X3D DIS networking were first performed by DIS-Java-VRML working group

- Virtual Reality Modeling Language (VRML97) standard is direct predecessor to X3D

This design has been carefully evolved over time to match practical experience gained by producing ever-larger X3D scenes

# DIS-Java-VRML home page

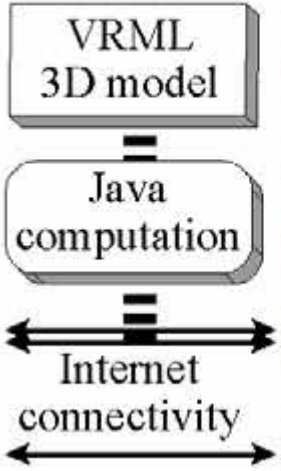
http://faculty.nps.edu/brutzman/vrtp/dis-java-vrml

**Distributed Interactive Simulation**
**DIS-Java-VRML Working Group**


**Contents**



**Project Overview**




The area of interest of this working group is the nexus of DIS, Java and VRML. The IEEE Distributed Interactive Simulation (DIS) Protocol is used to communicate state information (such as position, orientation, velocities and accelerations) among multiple entities participating in a shared network environment. Java is a portable networked programming language that can interoperate on any computer that includes a Web browser. The Virtual Reality Modeling Language (VRML) enables platform-independent interactive three-dimensional (3D) graphics across the Internet, and can be used to compose sophisticated 3D virtual environments.

The DIS-Java-VRML Working Group is developing a free software library, written in Java and interoperable with both DIS and VRML. There are a number of people contributing to the public-domain code archive. This project began in early 1997 and remains active. This software is protected under the terms of the GNU General Public License.

In the past, virtual world applications using DIS were deployed on expensive hardware and with custom software implementations. NPSNET was the first multicast DIS example of a large-scale virtual world. Like many such systems, NPSNET is written in C++ and uses the *Performer* application programming interface (API) libraries on Silicon Graphics Inc. (SGI) graphics workstations, in order to create a sophisticated and capable distributed system.

The International Standards Organization (ISO)-approved VRML 97 specification includes the rules for incorporating Java inside a VRML scene. This presents interesting possibilities for implementing virtual worlds on both high-cost and low-cost hardware in a portable way. Rather than be constrained to specific hardware, a DIS-compatible virtual world might potentially be viewed on any computer with a network connection and a web browser. The construction of large physics-based virtual worlds can now become inexpensive, pervasive and scalable.

DIS, Java and VRML can provide all of the pertinent capabilities needed to implement large-scale virtual environments (LSVEs). DIS is essentially a behavior protocol tuned for physics-based (i.e. "real world") many-to-many interactions. Java is the programming language used to implement the DIS protocol, perform math calculations, communicate with the network and communicate with the VRML scene. VRML 3D graphics are used to model and render both local and remote entities in shared virtual worlds.

# DIS-Java-VRML codebase

## Availability

- dis-java-vrml.tar.gz or dis-java-vrml.zip
- Last build 2003

## Provides perhaps-useful example code, remains well documented

**Software Reference**

A. Software Download and Installation
B. Javadoc Documentation
C. DIS Data Dictionary
D. auv (Autonomous Underwater Vehicle) Package
E. awt (Abstract Window Toolkit) Package
F. bridge Package
G. dis Package
H. disEnumerations Package
I. eaiDemoAUV & eaiDemoBoids Packages
J. helicopter Package
K. logger Package
L. math Package
M. relate Agent Architecture Package
N. testing Package
O. util (utilities) Package
P. org.web3d.vrtp.net (network) Package
Q. org.web3d.vrtp.security Package

# DIS-Java-VRML: Javadoc

All Classes

Packages
demo.auv
demo.helicopter
mil.navy.nps.awt
mil.navy.nps.bridge
mil.navy.nps.dis
mil.navy.nps.disEnumerations
mil.navy.nps.eaiDemoAUV
mil.navy.nps.eaiDemoBoids
mil.navy.nps.logger

**All Classes**

AcknowledgeFlagField
AcknowledgePdu
*Action*
ActionIDField
ActionRequestPdu
ActionResponsePdu
Agent
AgentHeloActionInterpreter
AgentHeloControlPanel
AgentTankActionInterpreter
AgentTankControlPanel
*AllPermissionsBadge*
AngularVelocity
Antenna
AntennaActionInterpreter
AntennaControlPanel
AntennaControlPanel_printBut
AntennaControlPanel_receive
AntennaControlPanel_signalP
AntennaControlPanel_transmi
AntennaPatternTypefield
AntennaStartPanel
ArticulationParameter
ArticulationParameterTest
AuvPduGenerator
AwtEspduSender
AwtEspduSenderFrame
AwtMulticastRelayClient
AwtMulticastRelayClientFrame

**Overview** Package Class **Tree** **Deprecated** **Index** **Help**

PREV NEXT FRAMES NO FRAMES

## DIS-Java-VRML Javadoc

| Packages | |
|---|---|
| **demo.auv** | NPS *Phoenix* Autonomous Underwater Vehicle (AUV) demonstration scenes and applications. |
| **demo.helicopter** | Multiplayer helicopter-tank battle demonstration, including a *Capture the Flag* game. |
| **mil.navy.nps.awt** | Protocol Data Unit (PDU) reader & writer applets, implemented using the Java Abstract Window Toolkit (AWT), that are useful for inspecting or setting the values of PDU fields. |
| **mil.navy.nps.bridge** | Includes several programs that enable multicast PDU streams to be redirected via unicast sockets. |
| **mil.navy.nps.dis** | Distributed Interactive Simulation (DIS) Protocol implementation, for standalone operation or integration with VRML scenes and entities. |
| **mil.navy.nps.disEnumerations** | An extensive class library providing predefined enumeration values, which are the special constants used to fill DIS protocol data unit (PDU) fields. |
| **mil.navy.nps.eaiDemoAUV** | A still-broken demo for the still-unstandardized External Authoring Interface (EAI). |
| **mil.navy.nps.eaiDemoBoids** | A still-broken demo for the still-unstandardized External Authoring Interface (EAI). |
| **mil.navy.nps.logger** | Protocol Data Unit (PDU) logger applets for recording and playing back PDUs. |
| **mil.navy.nps.math** | Contains several useful math-related classes. |
| **mil.navy.nps.relate** | The Relate agent relationship manager integrates Goals, Personalities, Relationships, Roles, Rules, SensedEnvironment and Sensor. |
| **mil.navy.nps.testing** | A variety of test programs. |
| **mil.navy.nps.util** | This utilities package is a class library that provides several useful extensions to Java which are of general use (and not specific to the DIS package). |
| **org.web3d.vrtp.net** | This networking package includes DatagramStreamBuffer, which, when interacting with the security package, provides platform-independent security to applications. |
| **org.web3d.vrtp.security** | The security package implements a way to do platform-independent code that breaks out of the Java sandbox to perform filesystem access, network access, or Java properties database access. |

# Current library:  Open-DIS

# Open-DIS motivation, design

Open-source implementation of DIS protocol

- Primary architect Don McGregor NPS
- Non-viral business-friendly BSD license
- Version control available on SourceForge repository http://open-dis.sourceforge.net/Open-DIS.html
- Contributions welcome

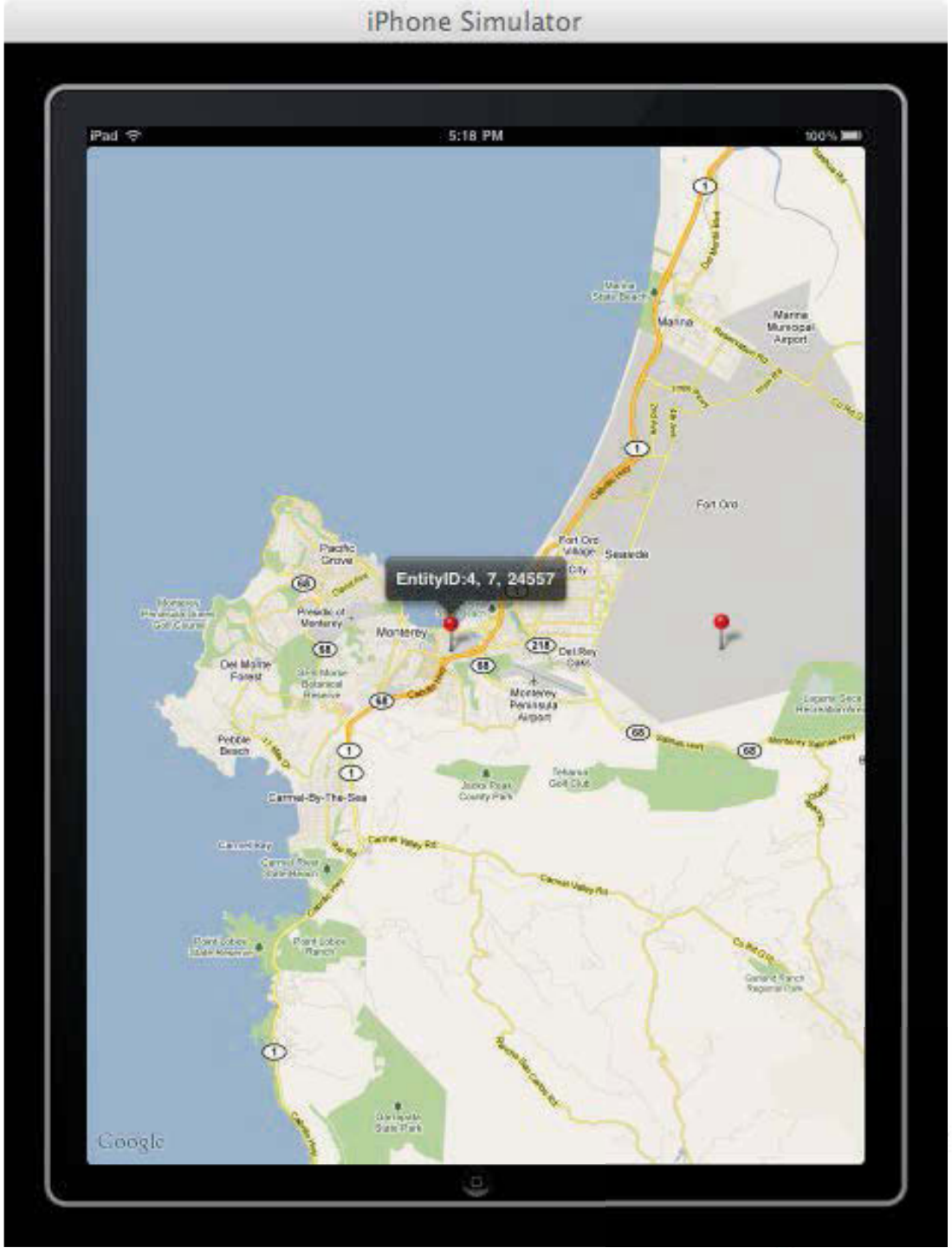
Multiple program languages

- Java, C++, C#, Objective C, Javascript
- Generated from object representation, thus adaptable for adding new PDU types
- Some semi-manual tuning is required

# Open-DIS on Mac

Objective C version of Open-DIS able to run on iPhone, iPad

Screen snapshot shows PDU tracks superimposed on Google Maps using iPhone Simulator




web 3D CONSORTIUM

# Open-DIS repository

http://open-dis.sourceforge.net/Open-DIS.html

**Open-DIS** DIS Documentation Downloads Collaboration Users
Examples Community Developers Propaganda MOVES

## Open-DIS

- Sourceforge Project Page
- Downloads
- Subversion Source Code Control

### Open-DIS: An Open Source Implementation of the Distributed Interactive Simulation Protocol

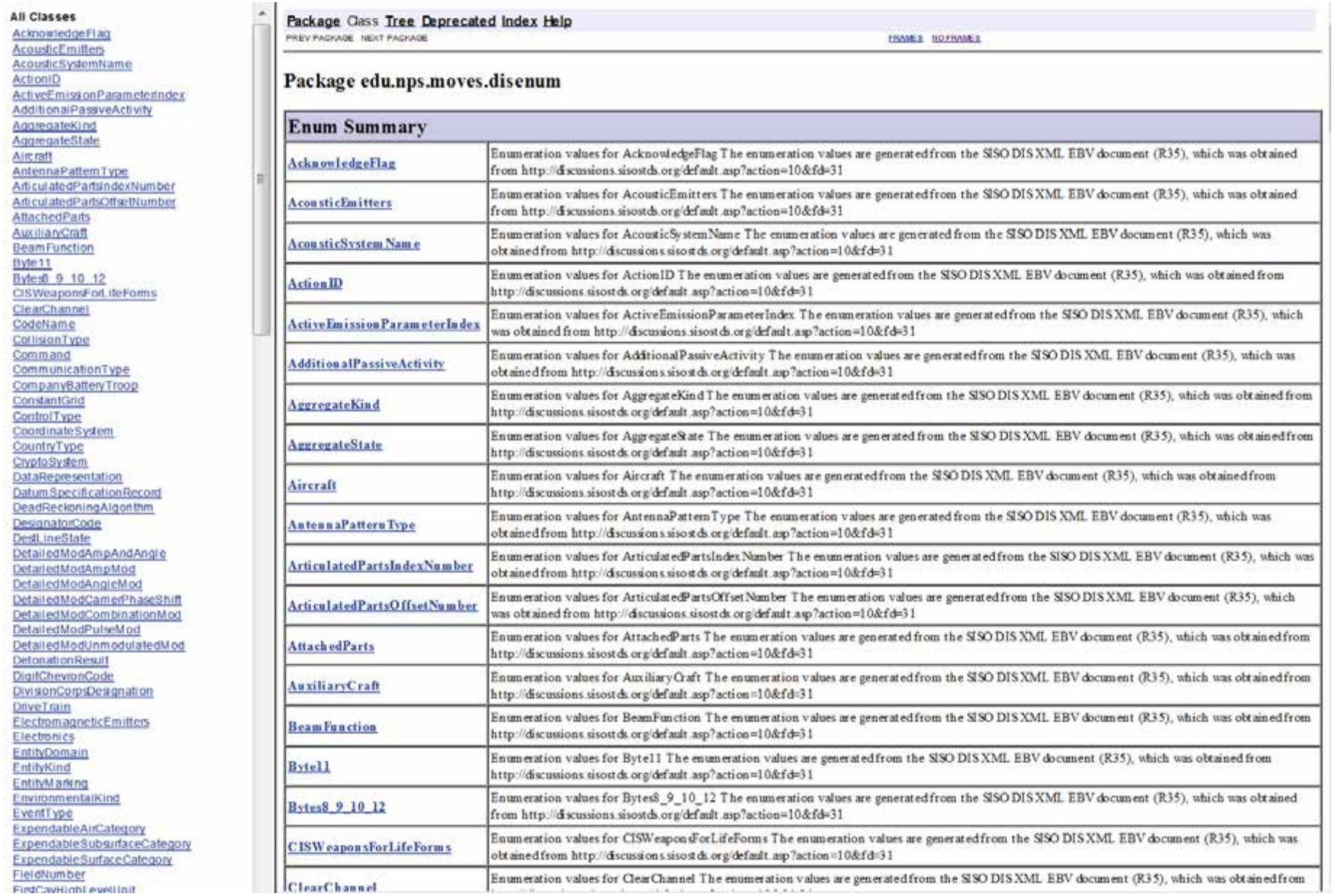
DIS is one of the most widely used protocols in Department of Defense, NATO, and allied nations real time/virtual world modeling and simulation. Open-DIS is a free, open source implementation of the standard in Java, C++, and C#. The project uses a BSD-style open source license, which is non-viral and business-friendly.

# Open-DIS Javadoc

http://open-dis.sourceforge.net/javadoc/open-dis/docs/index.html

# Open-DIS Enumerations Javadoc

http://open-dis.sourceforge.net/javadoc/disenum/docs



All Classes

AcknowledgeFlag
AcousticEmitters
AcousticSystemName
ActionID
ActiveEmissionParameterIndex
AdditionalPassiveActivity
AggregateKind
AggregateState
Aircraft
AntennaPatternType
ArticulatedPartsIndexNumber
ArticulatedPartsOffsetNumber
AttachedParts
AuxiliaryCraft
BeamFunction
Byte11
Bytes8_9_10_12
CISWeaponsForLifeForms
ClearChannel
CodeName
CollisionType
Command
CommunicationType
CompanyBatteryTroop
ConstantGrid
ControlType
CoordinateSystem
CountryType
CryptoSystem
DataRepresentation
DatumSpecificationRecord
DeadReckoningAlgorithm
DesignatorCode
DestLineState
DetailedModAmpAndAngle
DetailedModAmpMod
DetailedModAngleMod
DetailedModCarrierPhaseShift
DetailedModCombinationMod
DetailedModPulseMod
DetailedModUnmodulatedMod
DetonationResult
DigitChevronCode
DivisionCorpsDesignation
DriveTrain
ElectromagneticEmitters
Electronics
EntityDomain
EntityKind
EntityMarking
EnvironmentalKind
EventType
ExpendableAirCategory
ExpendableSubsurfaceCategory
ExpendableSurfaceCategory
FieldNumber
FirstCayHighLevelUnit

Package Class Tree Deprecated Index Help

PREV PACKAGE NEXT PACKAGE FRAMES NO FRAMES

## Package edu.nps.moves.disenum

| Enum Summary | |
|---|---|
| AcknowledgeFlag | Enumeration values for AcknowledgeFlag The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AcousticEmitters | Enumeration values for AcousticEmitters The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AcousticSystemName | Enumeration values for AcousticSystemName The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| ActionID | Enumeration values for ActionID The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| ActiveEmissionParameterIndex | Enumeration values for ActiveEmissionParameterIndex The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AdditionalPassiveActivity | Enumeration values for AdditionalPassiveActivity The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AggregateKind | Enumeration values for AggregateKind The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AggregateState | Enumeration values for AggregateState The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| Aircraft | Enumeration values for Aircraft The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AntennaPatternType | Enumeration values for AntennaPatternType The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| ArticulatedPartsIndexNumber | Enumeration values for ArticulatedPartsIndexNumber The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| ArticulatedPartsOffsetNumber | Enumeration values for ArticulatedPartsOffsetNumber The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AttachedParts | Enumeration values for AttachedParts The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| AuxiliaryCraft | Enumeration values for AuxiliaryCraft The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| BeamFunction | Enumeration values for BeamFunction The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| Byte11 | Enumeration values for Byte11 The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| Bytes8_9_10_12 | Enumeration values for Bytes8_9_10_12 The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| CISWeaponsForLifeForms | Enumeration values for CISWeaponsForLifeForms The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from http://discussions.sisostds.org/default.asp?action=10&fd=31 |
| ClearChannel | Enumeration values for ClearChannel The enumeration values are generated from the SISO DIS XML EBV document (R35), which was obtained from |

# Concepts:  DIS for X3D

## Common fields for X3D nodes

# Double precision requirements

Geospatial latitude, longitude position values require double precision accuracy

- Otherwise single-precision roundoff jitter equates to 3-10m of positional error

Graphics cards only support single precision

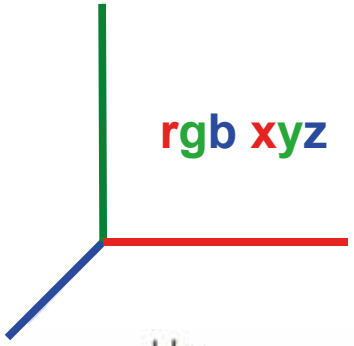
- Single precision 32 bit, double precision 64 bit

X3D Geospatial component reconciles this mismatch correctly and efficiently

Open-DIS uses double-precision satisfactorily

- However not yet integrated properly into X3D
- Use X-Y-Z local coordinate system instead

# Coordinate systems



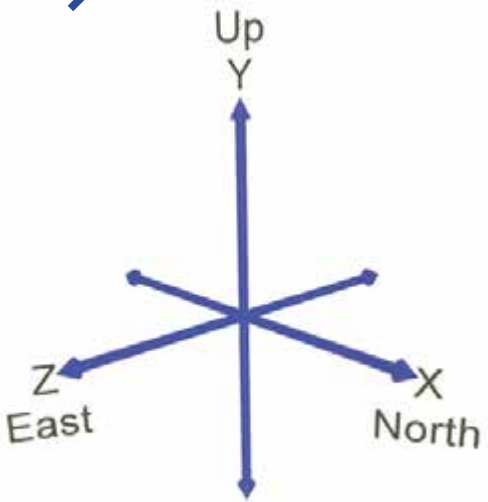

Right hand rule for X Y Z order

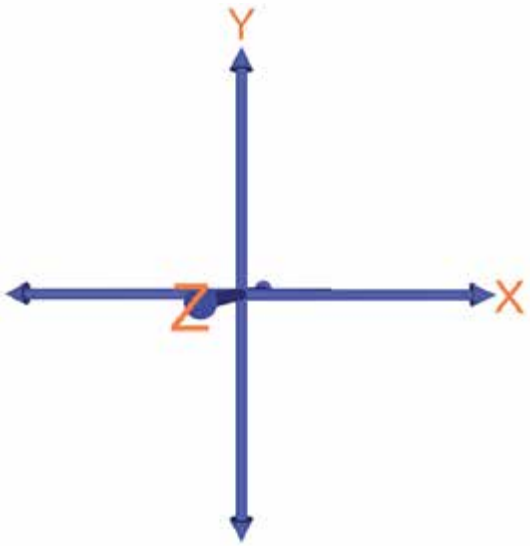
Y axis is up

Correspondence:  East, Up, South

Accept no substitutes!

- or at least realign them ☺



# Spatial reference frames

X3D is based on a right-handed Cartesion x,y,z coordinate system

- centered at arbitrary (0,0,0)

Geospatial data can be captured in a large variety of earth-oriented coordinate sytems

- It is important to keep these different coordinate systems straight, or else objects do not appear where they are expected
- Related to ellipsoid for actual Earth shape

# Common fields: *isActive*, *timestamp*

- *isActive* indicates if node has received a DIS packet (*isActive* true) or not (*isActive* false)

- *timestamp* field provides the time (SFTime) at which the DIS message arrived, referenced to local system time.

# Common field: *networkMode*

*networkMode* has 3 distinct possible values:

- `networkReader`: sender/owner writing updates,
- `networkWriter`: receiver/ghost reading updates,
- `standAlone`: independently from the network

Thus *networkMode* provides a way for an X3D scene to indicate whether a particular node is the owner or listener for a given entity.

- *networkMode*`='standAlone'` allows passive operation without opening network communications

# Common fields: network *address*, *port*

Typical transport mode is multicast, with specific *address* chosen by exercise

- Restricted to range 224.0.0.0 thru 239.255.255.255

Unicast address (or `localhost` value) allowed

- opens direct point-to-point socket

Similarly choose *port* value

- Restricted to range 1 thru 65535

Other networking fields:

- *multicastRelayHost, multicastRelayPort, rtpHeaderExpected, rtpHeaderHeard*

# Common fields for reading, writing

## *isReader, isWriter, isStandAlone*

- Boolean output events that can be ROUTEd within a scene to indicate whenever *networkMode* changes

## *readInterval*

- time in seconds between checking for receipt of DIS messages, intermediate PDUs are buffered
- Set to zero to disable reading

## *writeInterval*

- time in seconds between PDU transmissions
- Set to zero to disable writing

# Common fields for entity identification

Goal: build unique identifiers for each entity

- *siteID* identifies a given LAN or organization
- *applicationID* is unique for a given simulation
- *entityID* is unique to a given entity
- Background: *siteID* and *applicationID* fields are used to create DIS PDU Simulation Address record

Thus entity identification depends on a triplet of values, unique across all entities in that application and in the particular exercise.

# Common field: *metadata*

Each node can also contain Metadata nodes

- This is consistent throughout all X3D

Metadata nodes allow authors to add pairs of names and typed values to describe content

- Possible option for annotating, augmenting content in a valid machine-readable way
- MetadataSet, MetadataString, MetadataFloat, MetadataDouble, MetadataInteger, MetadataBoolean

Unlike comments, metadata value arrays are all available at run time

back to Table of Contents

# X3D Nodes and Examples



# EspduTransform:  shared state

Exposes DIS ESPDU values as an extended form of Transform node, matching semantics between DIS and X3D scene graph

- Also integrates Collision, Fire and Detonate PDUs since they are fundamentally integral to entity motion
- Distributed shared state among players
- Primary workhorse node of interest for DIS

TODO:  add, expose example scenes to show various ROUTE connections for animation

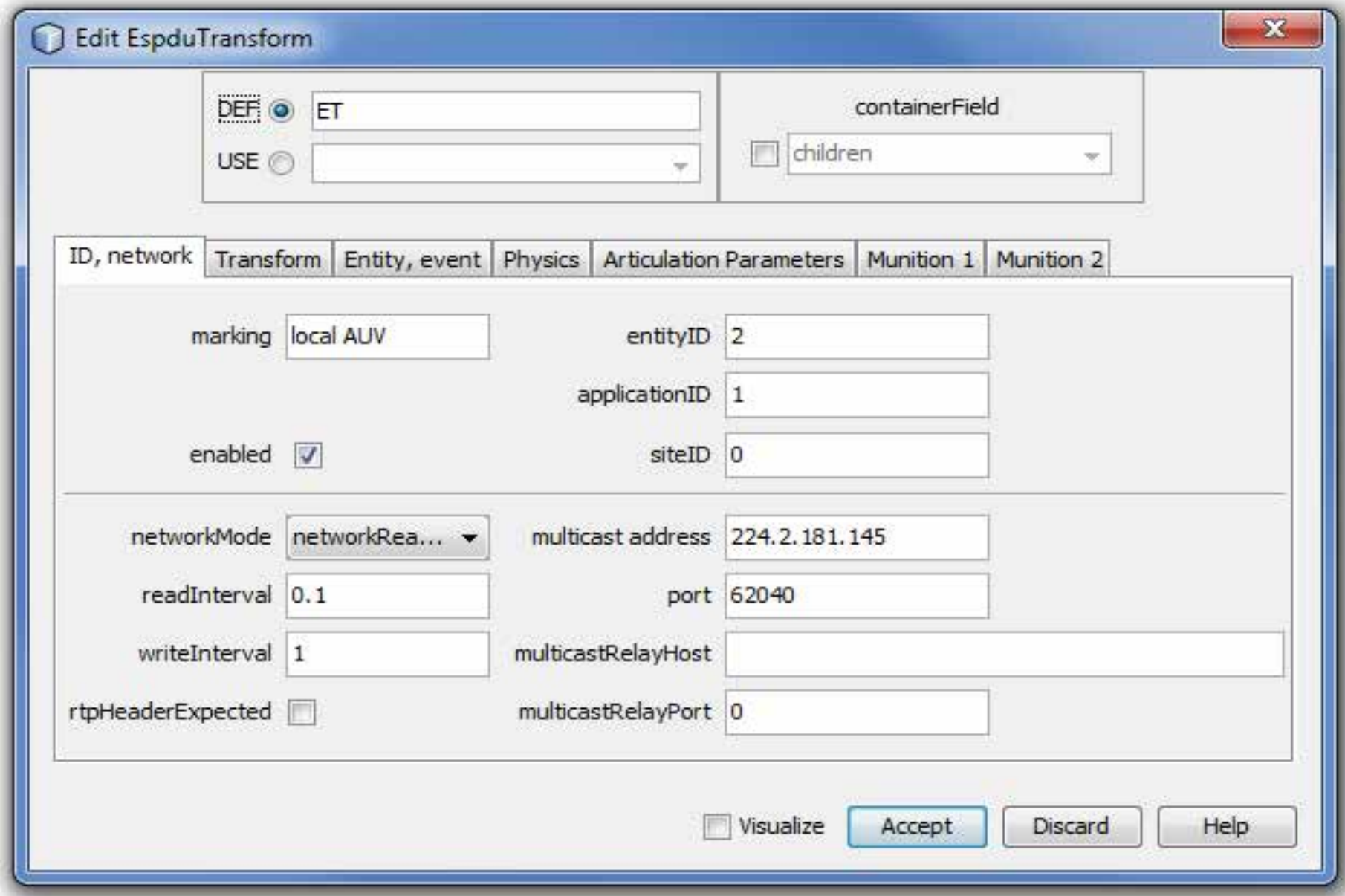
# EspduTransform:  shared state

Exposes DIS ESPDU values as an extended form of Transform node, matching semantics between DIS and X3D scene graph

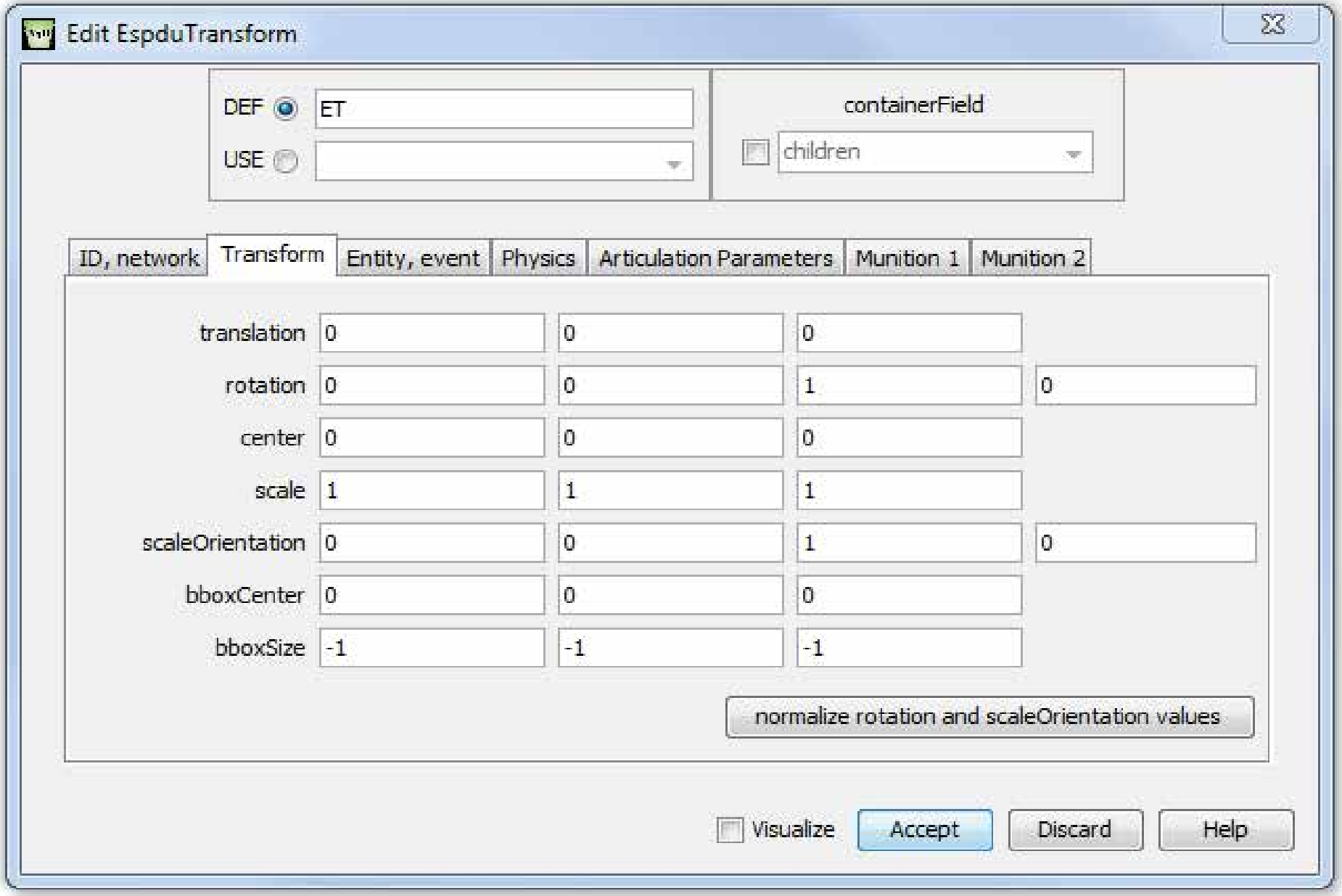
- Also integrates Collision, Fire and Detonate PDUs since they are fundamentally integral to entity motion
- Distributed shared state among players
- Primary workhorse node of interest for DIS

TODO:  add, expose example scenes to show various ROUTE connections for animation

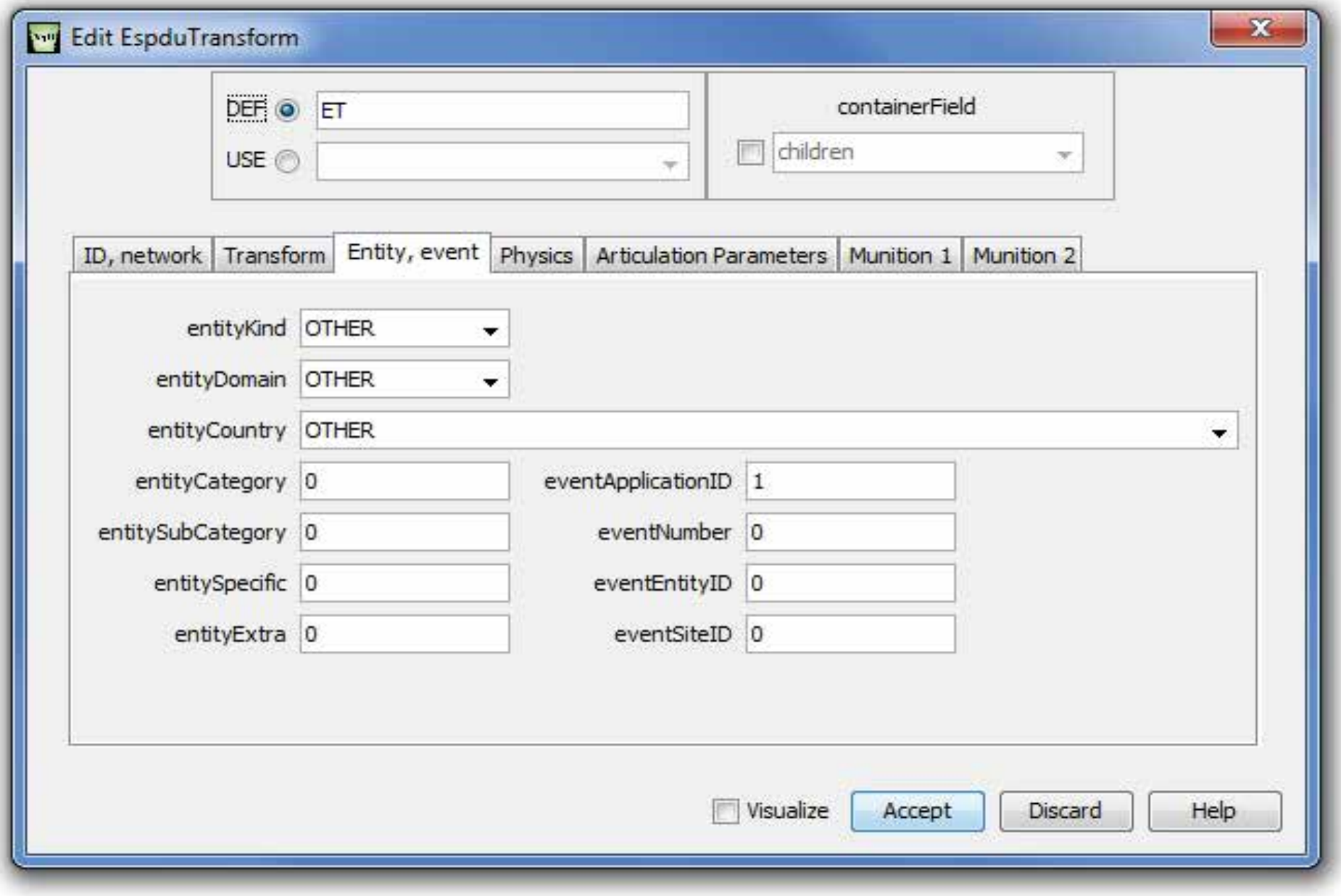
# EspduTransform: ID, network pane



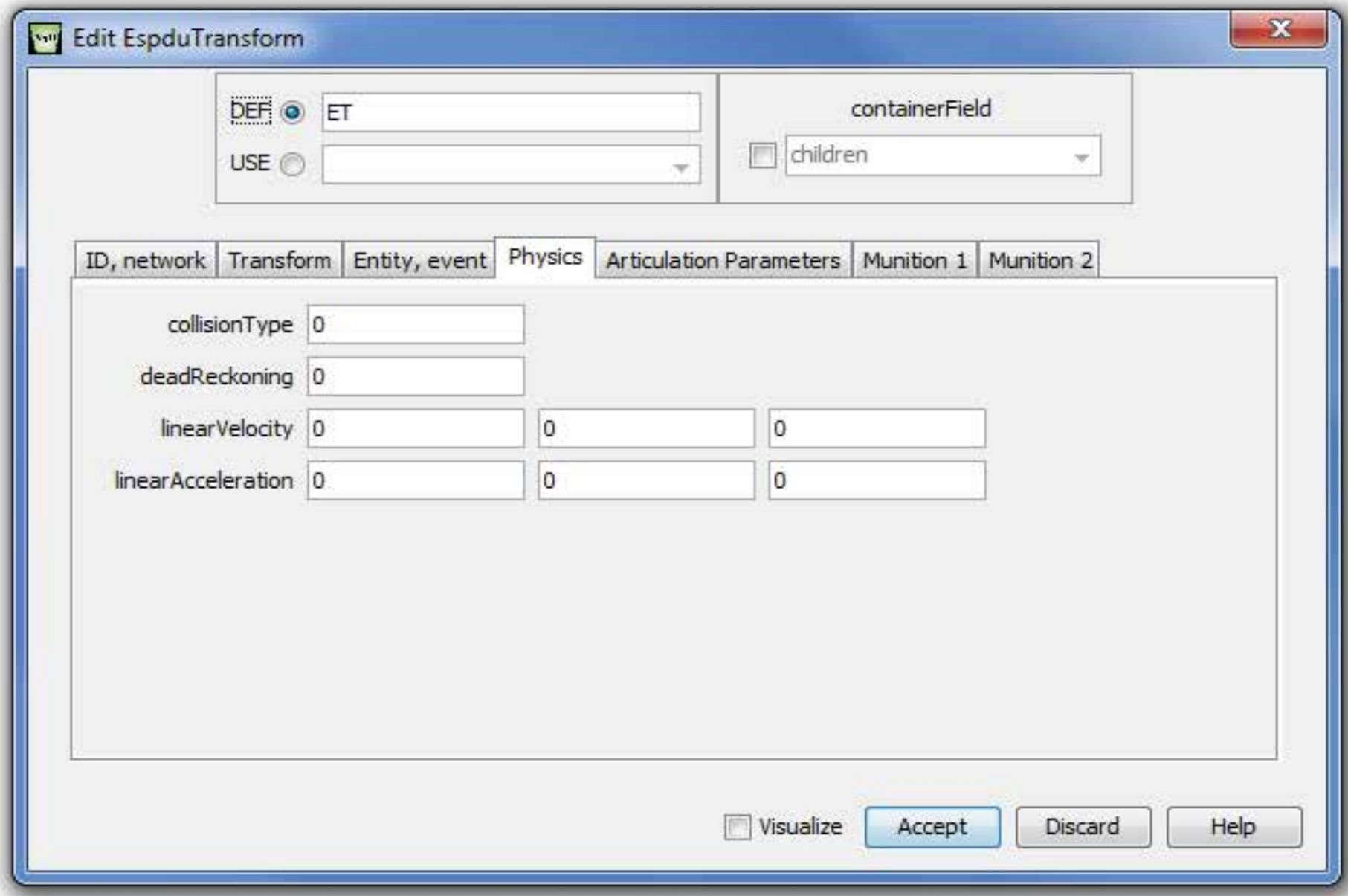
# EspduTransform: transform pane



# EspduTransform: entity, event pane

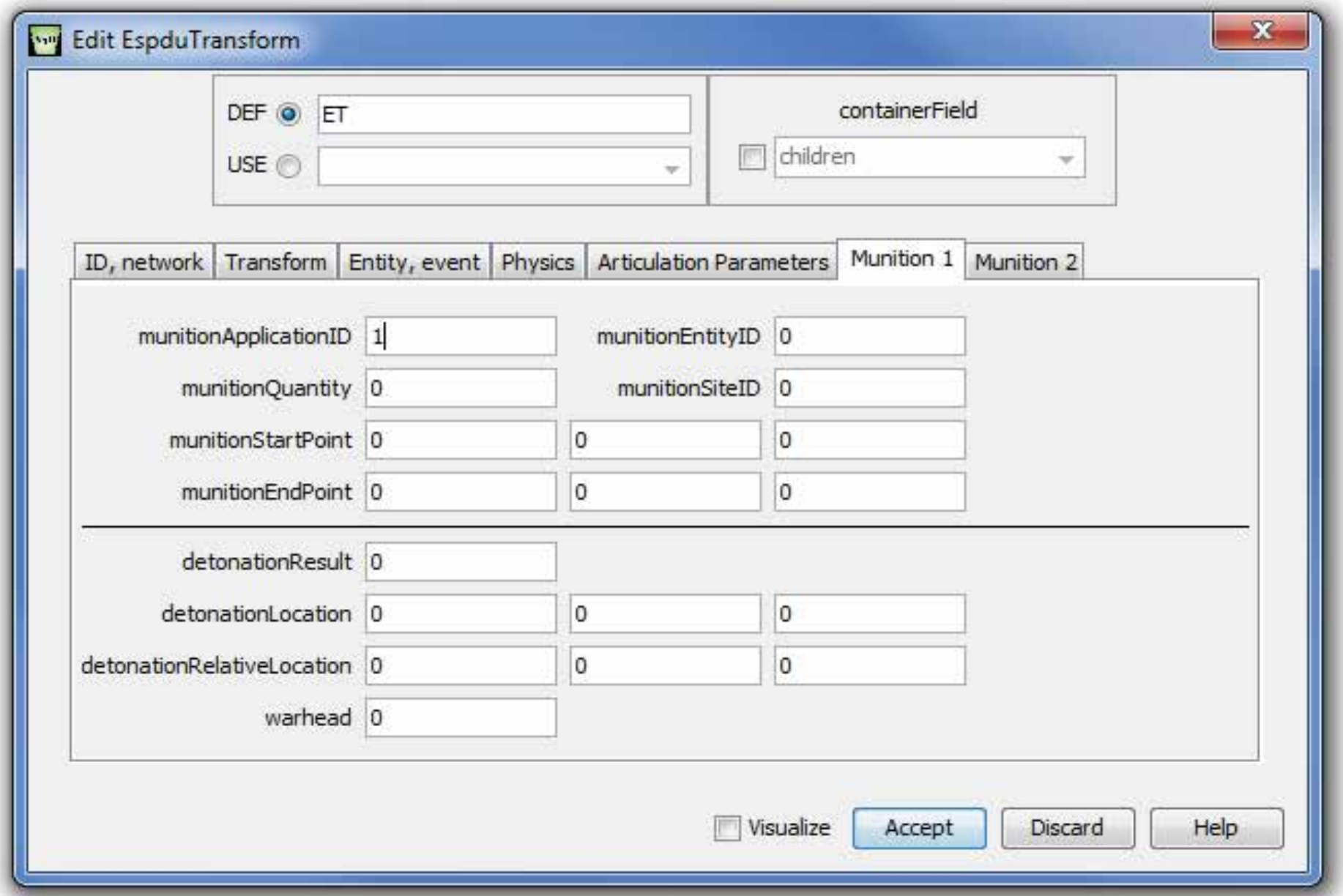


# EspduTransform: physics pane

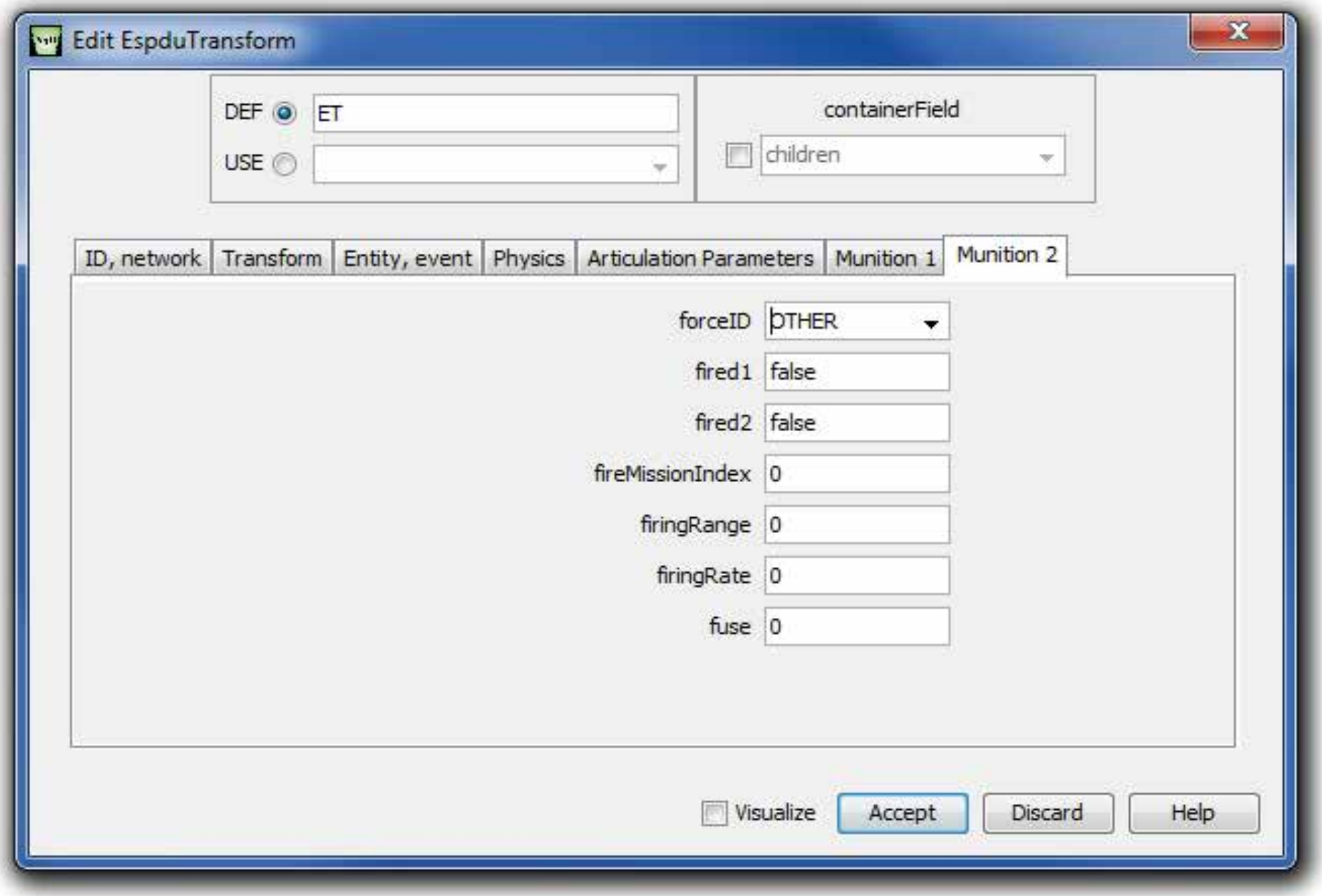


# EspduTransform: articulation parameters

# EspduTransform: munition1 pane



# EspduTransform: munition2 pane



| | |
|---|---|
| **EspduTransform** | **EspduTransform is a networked Transform node that can contain most nodes. EspduTransform integrates functionality for the following DIS PDUs: EntityStatePdu CollisionPdu DetonatePdu FirePdu CreateEntity RemoveEntity.**<br>**Hint: insert a Shape node before adding geometry or Appearance.** |
| **DEF** | **[DEF ID #IMPLIED]**<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>**Hint:** descriptive DEF names improve clarity and help document a model. |
| **USE** | **[USE IDREF #IMPLIED]**<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>**Hint:** USEing other geometry (instead of duplicating nodes) can improve performance.<br>**Warning:** do NOT include DEF (or any other attribute values) when using a USE attribute! |
| **enabled** | **[enabled accessType inputOutput, type SFBool (true\|false) "true"]**<br>Enables/disables the sensor node. |
| **marking** | **[marking accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Maximum of 11 characters for simple entity label. |
| **siteID** | **[siteID accessType inputOutput, type SFInt32 CDATA "0"]**<br>simulation/exercise siteID of the participating LAN or organization. |
| **applicationID** | **[applicationID accessType inputOutput, type SFInt32 CDATA "1"]**<br>simulation/exercise applicationID is unique for application at that site. |
| **entityID** | **[entityID accessType inputOutput, type SFInt32 CDATA "0"]**<br>simulation/exercise entityID is unique ID for entity within that application. |
| **forceID** | **[forceID accessType inputOutput, type SFInt32 CDATA "0"]** |
| **entityKind** | **[entityKind accessType inputOutput, type SFInt32 CDATA "0"]** |
| **entityDomain** | **[entityDomain accessType inputOutput, type SFInt32 CDATA "0"]** |
| **entityCountry** | **[entityCountry accessType inputOutput, type SFInt32 CDATA "0"]** |
| **entityCategory** | **[entityCategory accessType inputOutput, type SFInt32 CDATA "0"]** |
| **entitySubCategory** | **[entitySubCategory accessType inputOutput, type SFInt32 CDATA "0"]** |
| **entitySpecific** | **[entitySpecific accessType inputOutput, type SFInt32 CDATA "0"]** |

| | |
|---|---|
| entitySpecific | **[entitySpecific accessType inputOutput, type SFInt32 CDATA "0"]** |
| entityExtra | **[entityExtra accessType inputOutput, type SFInt32 CDATA "0"]** |
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" Example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port Example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally. Example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally. Example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |

| | |
|---|---|
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |
| isActive | **[isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Have we received a network update recently? |
| timestamp | **[timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]**<br>DIS timestamp in VRML time units from latest update. |
| translation | **[translation accessType inputOutput, type SFVec3f CDATA "0 0 0"]**<br>Position of children relative to local coordinate system, usually read from (or written to) remote, networked EspduTransform nodes. |
| rotation | **[rotation accessType inputOutput, type SFRotation CDATA "0 0 1 0"]**<br>Orientation of children relative to local coordinate system, usually read from (or written to) remote, networked EspduTransform nodes. |
| center | **[center accessType inputOutput, type SFVec3f CDATA "0 0 0"]**<br>Translation offset from origin of local coordinate system. |
| scale | **[scale accessType inputOutput, type SFVec3f CDATA "1 1 1"]**<br>Non-uniform x-y-z scale of child coordinate system, adjusted by center and scaleOrientation. |
| scaleOrientation | **[scaleOrientation accessType inputOutput, type SFRotation CDATA "0 0 1 0"]**<br>Preliminary rotation of coordinate system before scaling (to allow scaling around arbitrary orientations). |
| bboxCenter | **[bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]**<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | **[bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]**<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| linearVelocity | **[linearVelocity accessType inputOutput, type SFVec3f CDATA "0 0 0"]** |
| linearAcceleration | **[linearAcceleration accessType inputOutput, type SFVec3f CDATA "0 0 0"]** |
| deadReckoning | **[deadReckoning accessType inputOutput, type SFInt32 CDATA "0"]**<br>[0,65535] Dead reckoning algorithm being used to project position/orientation with velocities/accelerations. |
| isCollided | **[isCollided accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Has a matching CollisionPDU reported a collision? |

| | |
|---|---|
| collideTime | [collideTime accessType outputOnly, type SFTime CDATA #FIXED ""]<br>When were we collided with? |
| isDetonated | [isDetonated accessType outputOnly, type SFBool (true\|false) #FIXED ""]<br>Has a matching DetonationPDU reported a detonation? |
| detonateTime | [detonateTime accessType outputOnly, type SFTime CDATA #FIXED ""]<br>When were we detonated? |
| fired1 | [fired1 accessType inputOutput, type SFBool (true\|false) "false"]<br>Has the primary weapon (Fire PDU) been fired? |
| fired2 | [fired2 accessType inputOutput, type SFBool (true\|false) "false"]<br>Has the secondary weapon (Fire PDU) been fired? |
| firedTime | [firedTime accessType outputOnly, type SFTime CDATA #FIXED ""]<br>When did we shoot a weapon (Fire PDU)? |
| munitionStartPoint | [munitionStartPoint accessType inputOutput, type SFVec3f CDATA "0 0 0"]<br>eventout, uses exercise coordinates |
| munitionEndPoint | [munitionEndPoint accessType inputOutput, type SFVec3f CDATA "0 0 0"]<br>eventout, uses exercise coordinates |
| munitionSiteID | [munitionSiteID accessType inputOutput, type SFInt32 CDATA "0"]<br>Munition siteID |
| munitionApplicationID | [munitionApplicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>Munition applicationID, unique for application at that site |
| munitionEntityID | [munitionEntityID accessType inputOutput, type SFInt32 CDATA "0"]<br>Munition entityID is unique ID for entity firing munition within that application |
| fireMissionIndex | [fireMissionIndex accessType inputOutput, type SFInt32 CDATA #FIXED ""] |
| warhead | [warhead accessType inputOutput, type SFInt32 CDATA "0"] |
| fuse | [fuse accessType inputOutput, type SFInt32 CDATA "0"] |
| munitionQuantity | [munitionQuantity accessType inputOutput, type SFInt32 CDATA "0"] |

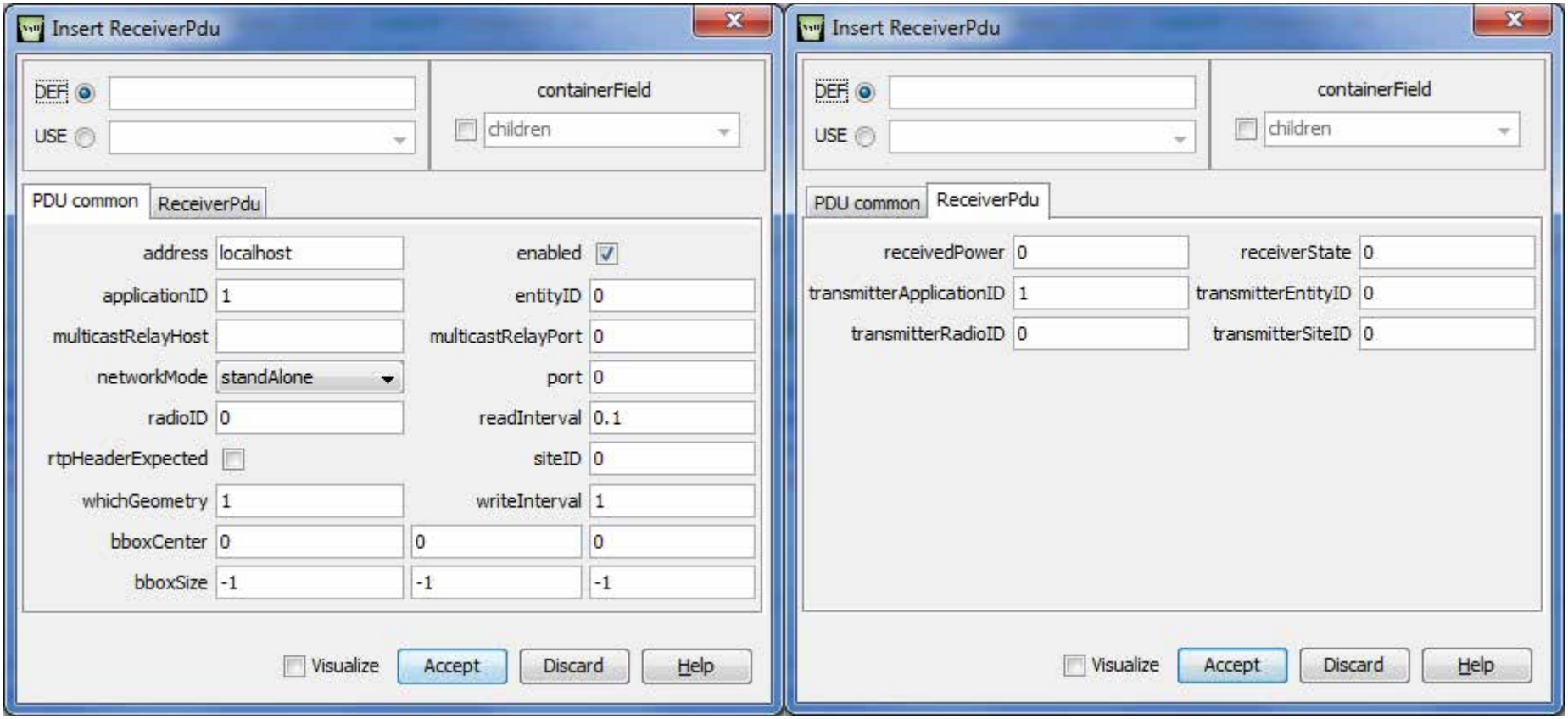
| | |
|---|---|
| firingRate | [firingRate accessType inputOutput, type SFInt32 CDATA "0"] |
| firingRange | [firingRange accessType inputOutput, type SFFloat CDATA "0"] |
| collisionType | [collisionType accessType inputOutput, type SFInt32 CDATA "0"] |
| detonationLocation | [detonationLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| detonationRelativeLocation | [detonationRelativeLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| detonationResult | [detonationResult accessType inputOutput, type SFInt32 CDATA "0"] |
| eventApplicationID | [eventApplicationID accessType inputOutput, type SFInt32 CDATA "1"] |
| eventEntityID | [eventEntityID accessType inputOutput, type SFInt32 CDATA "0"] |
| eventNumber | [eventNumber accessType inputOutput, type SFInt32 CDATA "0"] |
| eventSiteID | [eventSiteID accessType inputOutput, type SFInt32 CDATA "0"] |
| articulationParameterCount | [articulationParameterCount accessType inputOutput, type SFInt32 CDATA "0"]<br>First articulated parameter is articulationParameterValue0. |
| articulationParameterDesignatorArray | [articulationParameterDesignatorArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of designators for each articulated parameter. |
| articulationParameterChangeIndicatorArray | [articulationParameterChangeIndicatorArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of change counters, each incremented when an articulated parameter is updated #IMPLIED] . |
| articulationParameterIdPartAttachedToArray | [articulationParameterIdPartAttachedToArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of ID parts that each articulated parameter is attached to. |
| articulationParameterTypeArray | [articulationParameterTypeArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of type enumerations for each articulated parameter element. |
| articulationParameterArray | [articulationParameterArray accessType inputOutput, type MFFloat CDATA #IMPLIED] |
| set_articulationParameterValue0 | [set_articulationParameterValue0 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |

| | |
|---|---|
| set_articulationParameterValue1 | [set_articulationParameterValue1 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue2 | [set_articulationParameterValue2 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue3 | [set_articulationParameterValue3 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue4 | [set_articulationParameterValue4 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue5 | [set_articulationParameterValue5 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue6 | [set_articulationParameterValue6 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue7 | [set_articulationParameterValue7 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| articulationParameterValue0_changed | [articulationParameterValue0_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue1_changed | [articulationParameterValue1_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue2_changed | [articulationParameterValue2_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue3_changed | [articulationParameterValue3_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue4_changed | [articulationParameterValue4_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue5_changed | [articulationParameterValue5_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue6_changed | [articulationParameterValue6_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |

| | |
|---|---|
| articulationParameterValue7_changed | **[articulationParameterValue7_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]**<br>Get element of user-defined payload array. |
| containerField | **[containerField: NMTOKEN "children"]**<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | **[class CDATA #IMPLIED]**<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

# ReceiverPdu

- ReceiverPdu transmits state of radio frequency (RF) receivers modeled in the simulation.
- Exposes fields for ReceiverPdu node



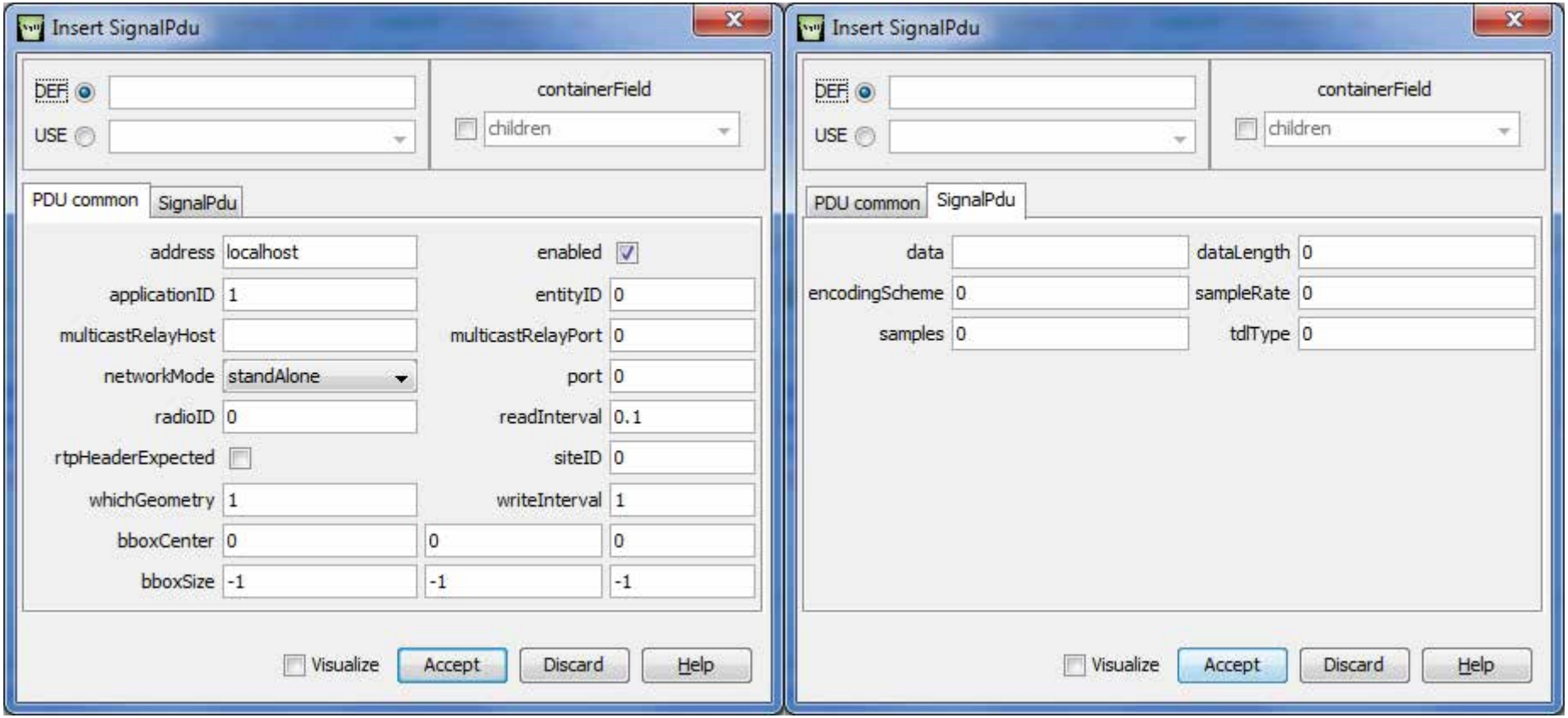
| ReceiverPdu | ReceiverPdu is a networked PDU information node. |
| --- | --- |
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| whichGeometry | [whichGeometry accessType inputOutput, type SFInt32 CDATA "1"]<br>Select geometry to render: -1 for no geometry, 0 for text trace, 1 for default geometry. |
| bboxCenter | [bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | [bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID unique ID for entity within that application. |

| | |
|---|---|
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" Example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |

| | |
|---|---|
| isActive | **[isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Have we had a network update recently?. |
| timestamp | **[timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]**<br>DIS timestamp in VRML units. |
| radioID | **[radioID accessType inputOutput, type SFInt32 CDATA "0"]** |
| receivedPower | **[receivedPower accessType inputOutput, type SFFloat CDATA "0"]** |
| receiverState | **[receiverState accessType inputOutput, type SFInt32 CDATA "0"]** |
| transmitterSiteID | **[transmitterSiteID accessType inputOutput, type SFInt32 CDATA "0"]** |
| transminerApplicationID | **[transmitterApplicationID accessType inputOutput, type SFInt32 CDATA "0"]** |
| transmitterEntityID | **[transmitterEntityID accessType inputOutput, type SFInt32 CDATA "0"]** |
| transmitterRadioID | **[transmitterRadioID accessType inputOutput, type SFInt32 CDATA "0"]** |
| containerField | **[containerField: NMTOKEN "children"]**<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | **[class CDATA #IMPLIED]**<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

# SignalPdu

- SignalPdu relays the transmission of voice, audio or other data modeled in a simulation
- Exposes fields for SignalPdu node



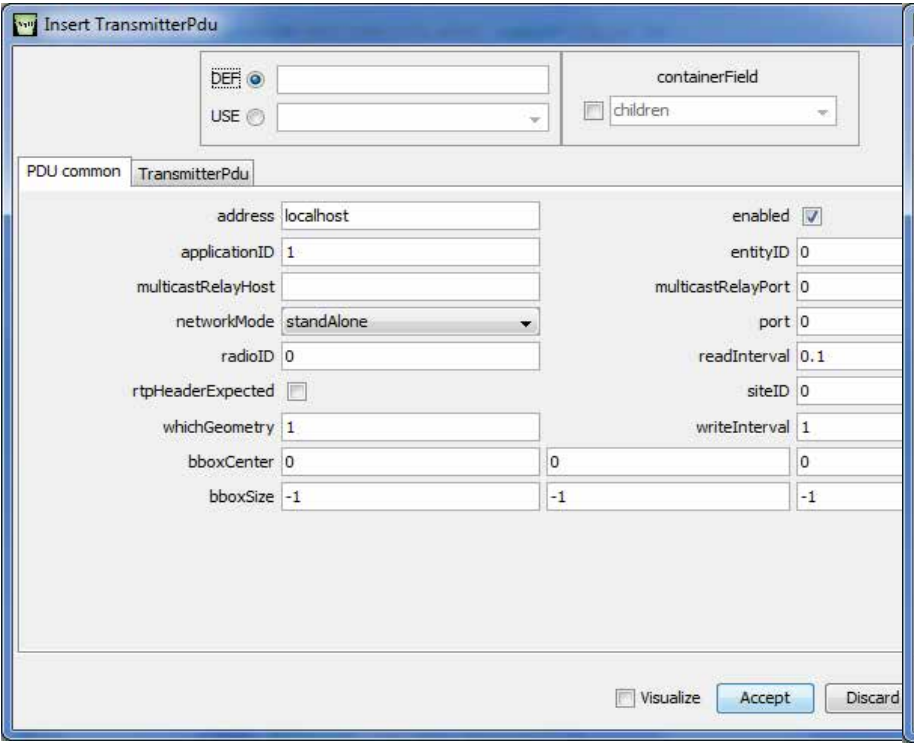
| SignalPdu | SignalPdu is a networked PDU information node. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| whichGeometry | [whichGeometry accessType inputOutput, type SFInt32 CDATA "1"]<br>Select geometry to render: -1 for no geometry, 0 for text trace, 1 for default geometry. |
| bboxCenter | [bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | [bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID unique ID for entity within that application. |

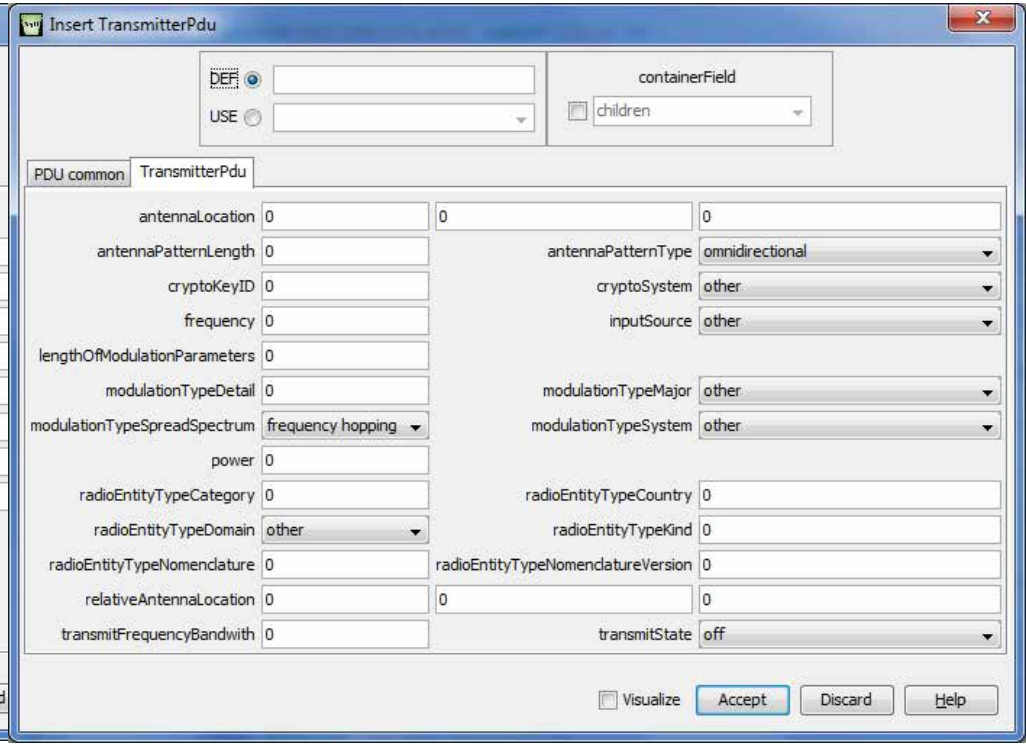
| | |
|---|---|
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |

| | |
|---|---|
| isActive | [isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]<br>Have we had a network update recently? |
| timestamp | [timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]<br>DIS timestamp in VRML units. |
| radioID | [radioID accessType inputOutput, type SFInt32 CDATA "0"] |
| encodingScheme | [encodingScheme accessType inputOutput, type SFInt32 CDATA "0"] |
| tdlType | [tdlType accessType inputOutput, type SFInt32 CDATA "0"] |
| sampleRate | [sampleRate accessType inputOutput, type SFInt32 CDATA "0"] |
| samples | [samples accessType inputOutput, type SFInt32 CDATA "0"] |
| dataLength | [dataLength accessType inputOutput, type SFInt32 CDATA "0"] |
| data | [data accessType inputOutput, type MFInt32 CDATA #IMPLIED] |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

# TransmitterPdu

- TransmitterPdu provides detailed info about a radio transmitter modeled in a simulation.
- Exposes fields for TransmitterPdu node



| TransmitterPdu | TransmitterPdu is a networked PDU information node. |
| --- | --- |
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| whichGeometry | [whichGeometry accessType inputOutput, type SFInt32 CDATA "1"]<br>Select geometry to render: -1 for no geometry, 0 for text trace, 1 for default geometry. |
| bboxCenter | [bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | [bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID unique ID for entity within that application. |

| | |
|---|---|
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |

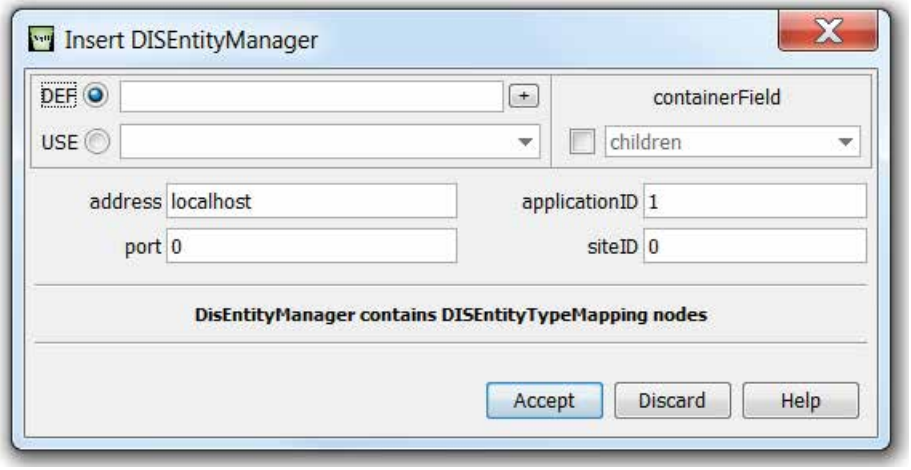
| | |
|---|---|
| isActive | [isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]<br>Have we had a network update recently?. |
| timestamp | [timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]<br>DIS timestamp in VRML units |
| radioID | [radioID accessType inputOutput, type SFInt32 CDATA "0"] |
| antennaLocation | [antennaLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| antennaPatternLength | [antennaPatternLength accessType inputOutput, type SFInt32 CDATA "0"] |
| antennaPatternType | [antennaPatternType accessType inputOutput, type SFInt32 CDATA "0"] |
| cryptoKeyID | [cryptoKeyID accessType inputOutput, type SFInt32 CDATA "0"] |
| cryptoSystem | [cryptoSystem accessType inputOutput, type SFInt32 CDATA "0"] |
| frequency | [frequency accessType inputOutput, type SFInt32 CDATA "0"] |
| inputSource | [inputSource accessType inputOutput, type SFInt32 CDATA "0"] |
| lengthOfModulationParameters | [lengthOfModulationParameters accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeDetail | [modulationTypeDetail accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeMajor | [modulationTypeMajor accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeSpreadSpectrum | [modulationTypeSpreadSpectrum accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeSystem | [modulationTypeSystem accessType inputOutput, type SFInt32 CDATA "0"] |
| power | [power accessType inputOutput, type SFFloat CDATA "0"] |

| | |
|---|---|
| radioEntityTypeCategory | [radioEntityTypeCategory accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeCountry | [radioEntityTypeCountry accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeDomain | [radioEntityTypeDomain accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeKind | [radioEntityTypeKind accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeNomenclature | [radioEntityTypeNomenclature accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeNomenclatureVersion | [radioEntityTypeNomenclatureVersion accessType inputOutput, type SFInt32 CDATA "0"] |
| relativeAntennaLocation | [relativeAntennaLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| transmitFrequencyBandwidth | [transmitFrequencyBandwidth accessType inputOutput, type SFFloat CDATA "0.0"] |
| transmitState | [transmitState accessType inputOutput, type SFInt32 CDATA "0"] |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

# DISEntityManager

DISEntityManager node notifies content when new entities arrive or current entities leave

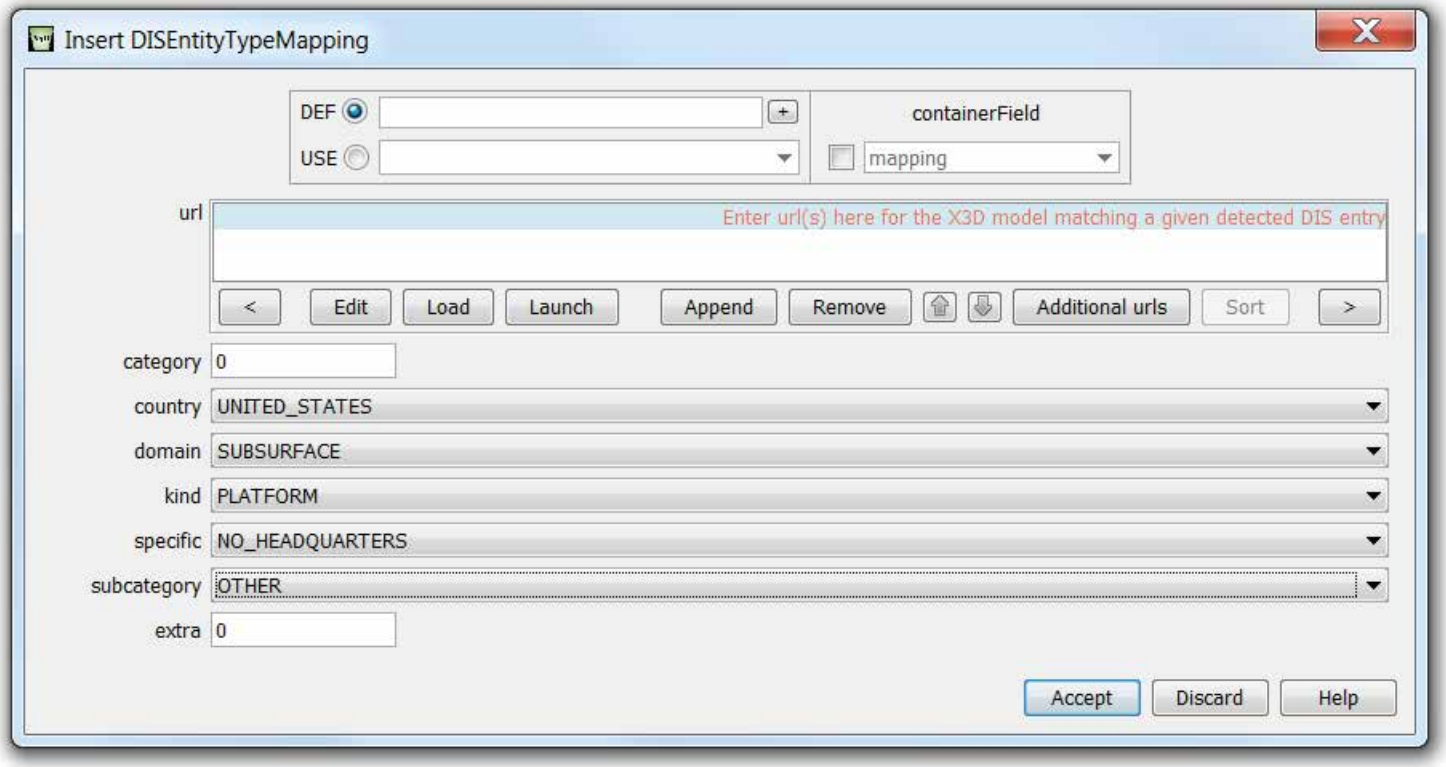
- Identifies multicast *address, port* as well as session identification *applicationID, siteID*
- Contains list of DISEntityTypeMapping nodes which define entity filters, correspondences for each uniquely defined exercise



| DISEntityManager | DISEntityManager node notifies content when new entities arrive or current entities leave. DISEntityManager may contain any number of DISEntityTypeMapping nodes. Incoming matches produce EspduTransform nodes containing the corresponding x3d model. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| address | [address accessType inputOutput, type SFString CDATA "localhost"]<br>Multicast address, or else "localhost" example: 224.2.181.145. |
| port | [port accessType inputOutput, type SFInt32 CDATA "0"]<br>Multicast port example: 62040. |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

# DISEntityTypeMapping

- Exposes fields for DISEntityTypeMapping
- Provides correspondence between detected entity identification fields and X3D models



| | |
|---|---|
| DISEntityTypeMapping | **DISEntityTypeMapping maps received DIS Entity type information to an X3D model, thus providing visual and behavioral representations matching received packets. Fields are processed in order: kind, domain, country, category, subcategory, specific, extra.**<br>Hint: **0 values are wildcards. All values in the ordered list must be 0 after the first 0 is defined.** |
| DEF | **[DEF ID #IMPLIED]**<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | **[USE IDREF #IMPLIED]**<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| url | **[url accessType inputOutput, type MFString CDATA #IMPLIED]**<br><br>Hint: Strings can have multiple values, so separate each string by quote marks [ "http://www.url1.org" "http://www.url2.org" "etc." ].<br>Hint: XML encoding for " is " (a character entity).<br>Warning: strictly match directory and filename capitalization for http links!<br>Hint: can replace embedded blank(s) in url queries with %20 for each blank character. |
| kind | **[kind accessType inputOutput, type SFInt32 CDATA "0"]** |
| domain | **[domain accessType inputOutput, type SFInt32 CDATA "0"]** |
| country | **[country accessType inputOutput, type SFInt32 CDATA "0"]** |
| category | **[category accessType inputOutput, type SFInt32 CDATA "0"]** |
| subCategory | **[subCategory accessType inputOutput, type SFInt32 CDATA "0"]** |
| specific | **[specific accessType inputOutput, type SFInt32 CDATA "0"]** |
| extra | **[extra accessType inputOutput, type SFInt32 CDATA "0"]** |
| containerField | **[containerField: NMTOKEN "children"]**<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | **[class CDATA #IMPLIED]**<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

# Applications and Examples

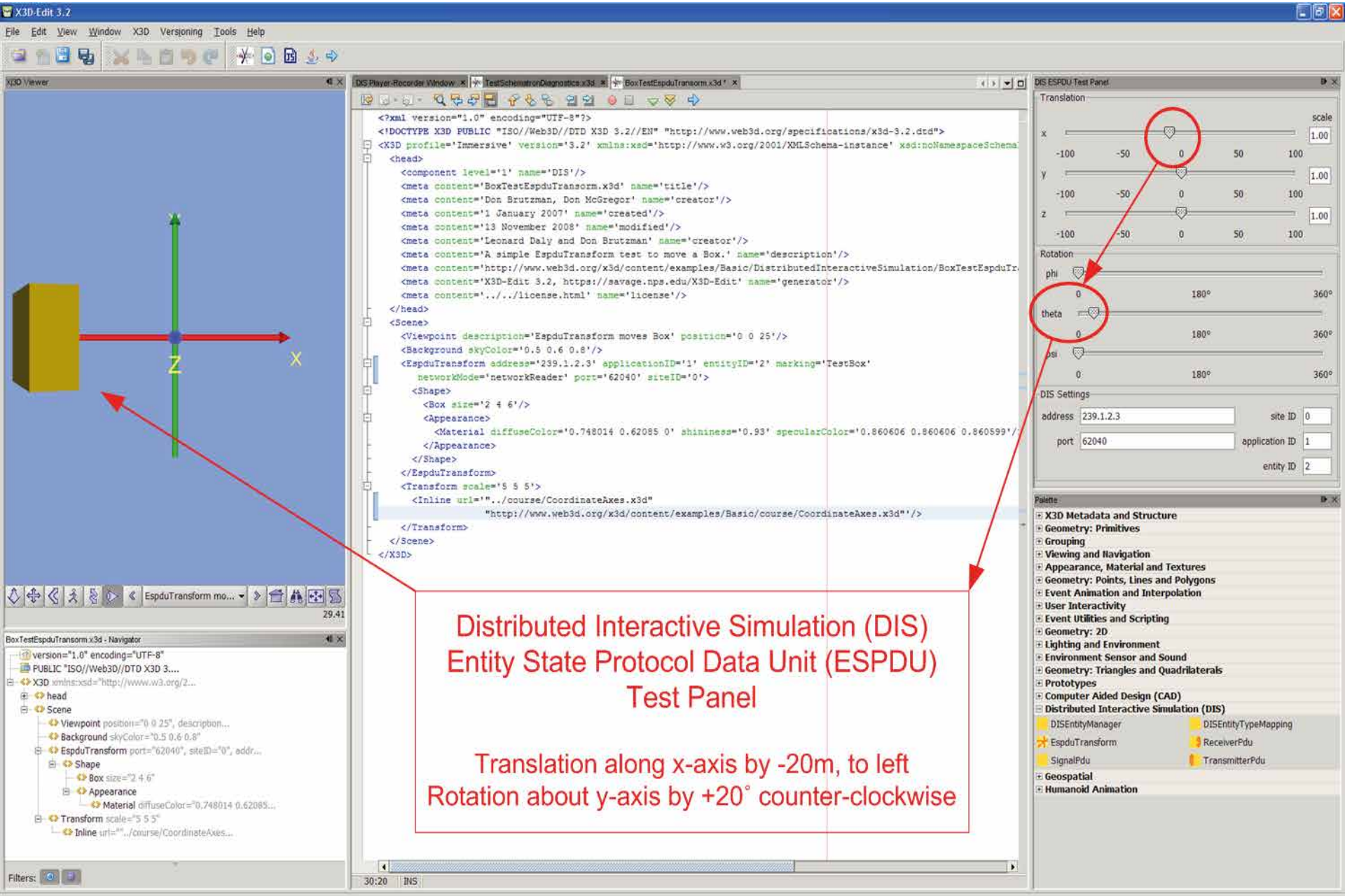
# Setup: Microsoft loopback adapter

- Unlike Unix, Microsoft Windows does not have multicast loopback turned on by default
- Special setup thus needed for solitary testing
  - Conflicts can emerge when also using Cisco VPN
- Help page provided by AUV Workbench

# Network interoperability

- X3D Anchor node functionality matches HTML anchor element: jump or bookmark
  - Planning new capability: refresh/interval, similar to HTML refresh, to improve server-side interaction
- Support slowly emerging for DIS protocol
  - X3D-Edit: PDU player/recorder, PDU generators
  - Open source OpenDIS codebase Java, C++, C#, Objective C (for iPhone) and JavaScript
  - Integration with Sun's *Darkstar* massive multiplayer online game (MMOG) Java server – NPS thesis
  - Active work: JavaScript, Websockets, HTML5

# DIS Networking Test Panel




Distributed Interactive Simulation (DIS)
Entity State Protocol Data Unit (ESPDU)
Test Panel

Translation along x-axis by -20m, to left
Rotation about y-axis by +20° counter-clockwise

# X3D DIS Implementations

More work needed!

X3DOM: work in progress

Xj3D: open source Java

- www.xj3D.org
- Embedded in (and launchable by) X3D-Edit

Other players to follow?

- BS Contact, InstantReality, FreeWrl, OpenVRML

Feature comparison available:

- Player support for X3D components wiki

# Obtaining example scenes

Primary:  X3D Basic archives, DistributedInteractiveSimulation directory

- http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation
- Under version control at Sourceforge

Also some in X3D Savage archives, located in various directories (such as Scenarios)

- https://savage.nps.edu/Savage
- Maintained under version control at NPS

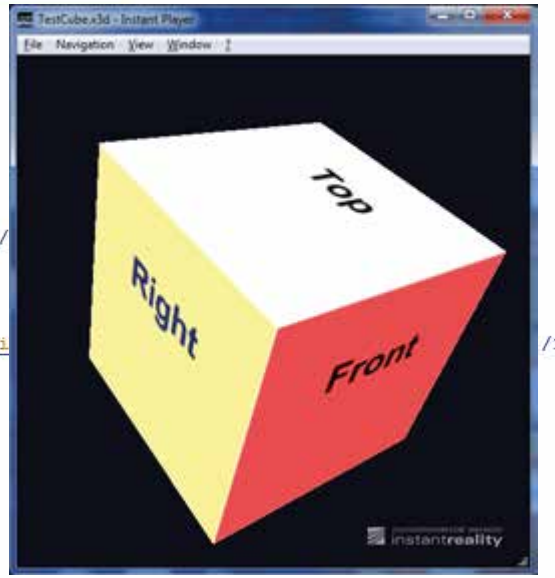
# Test shape: TestCube.x3d

Simple shape with labeled ImageTexture images on each side, no DIS included

TestCube.x3d - Editor

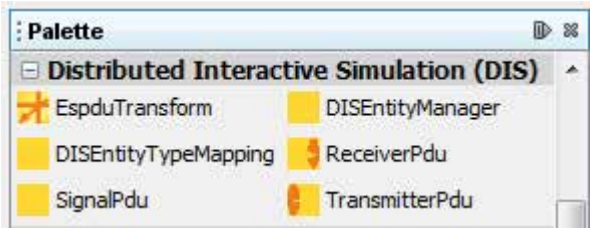
TestCube.x3d

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3.2.dtd">
3  <X3D profile='Immersive' version='3.2' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3d.org/speci
4    <head>
5      <meta content='TestCube.x3d' name='title'/>
6      <meta content='Don Brutzman' name='creator'/>
7      <meta content='1 November 2008' name='created'/>
8      <meta content='16 October 2010' name='modified'/>
9      <meta content='Provides a cube-shaped test model with each face individually labeled.' name='description'/>
10     <meta content='experiment with indices to remove TextureTransform rotations' name='TODO'/>
11     <meta content='http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/TestCube.x3d' name='identifier'/>
12     <meta content='X3D-Edit 3.2, https://savage.nps.edu/X3D-Edit' name='generator'/>
13     <meta content='../license.html' name='license'/>
14   </head>
15   <Scene>
16     <Background skyColor='0.05 0.05 0.1'/>
17     <Group>
18       <Viewpoint description='Front view' orientation='0 1 0 1.57' position='8 0 0'/>
19       <Shape DEF='Front'>
20         <IndexedFaceSet coordIndex='0 1 2 3' texCoordIndex='0 1 2 3'>
21           <Coordinate DEF='Points' point='1 1 1 1 -1 1 1 -1 -1 1 1 -1 -1 1 1 -1 -1 1 -1 -1 -1 -1 1 -1'/
22           <TextureCoordinate DEF='DefaultTextureCoordinate' point='0 0 1 0 1 1 0 1'/>
23         </IndexedFaceSet>
24         <Appearance>
25           <ImageTexture url='"images/front.png" "http://www.web3d.org/x3d/content/examples/Basic/Distri                    />
26           <TextureTransform DEF='RotateRight' rotation='-1.57'/>
27         </Appearance>
28       </Shape>
29       <Viewpoint description='Back view' orientation='0 1 0 -1.57' position='-8 0 0'/>
30       <Shape>
40       <Viewpoint description='Left view' orientation='0 1 0 3.1416' position='0 0 -8'/>
41       <Shape>
51       <Viewpoint description='Right view' position='0 0 8'/>
52       <Shape DEF='Right'>
```




1 | 1 INS

# X3D-Edit DIS support




X3D-Edit authoring tool includes multiple support capabilities for DIS usage

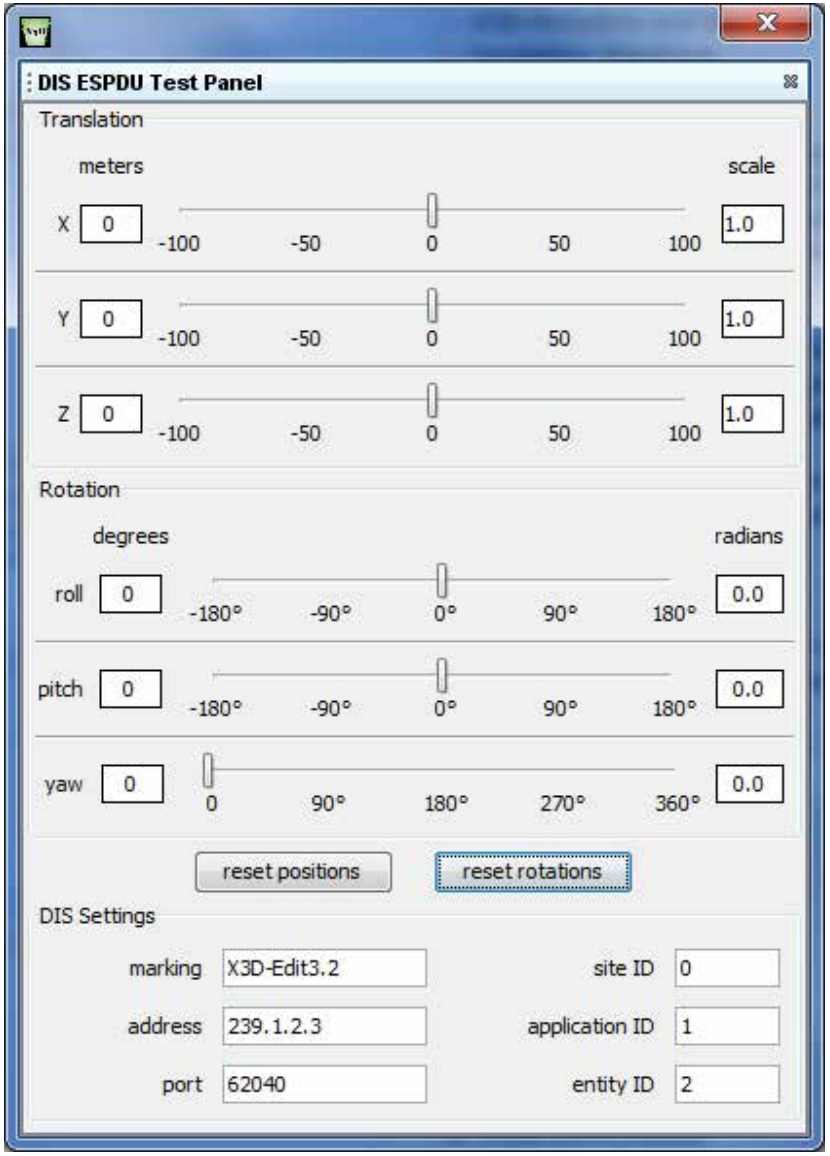
- Edit panels for X3D nodes: EspduTransform, ReceiverPdu, SignalPdu, TransmitterPdu, DISEntityManager, DISEntityTypeMapping
- DIS ESPDU Test panel for sending values
- DIS Player-Recorder for recording, playing back, loading or saving packets
- DIS PDU Sender Test menu selection for sending one of each PDU type available

# X3D-Edit DIS ESPDU packet tester

Select values to send

- *x y z* in meters multiplied by scale
- *roll pitch yaw* shown in degrees, also radians
- Type in *marking* and network parameters

Each change sends a new ESPDU packet



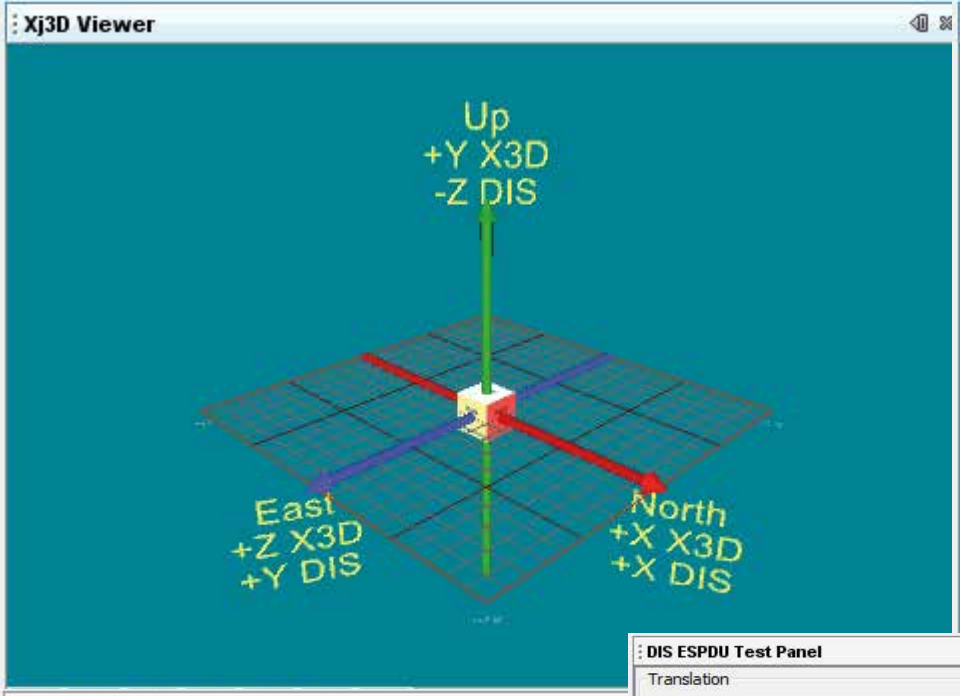
# Example: TestCubeEspdu.x3d

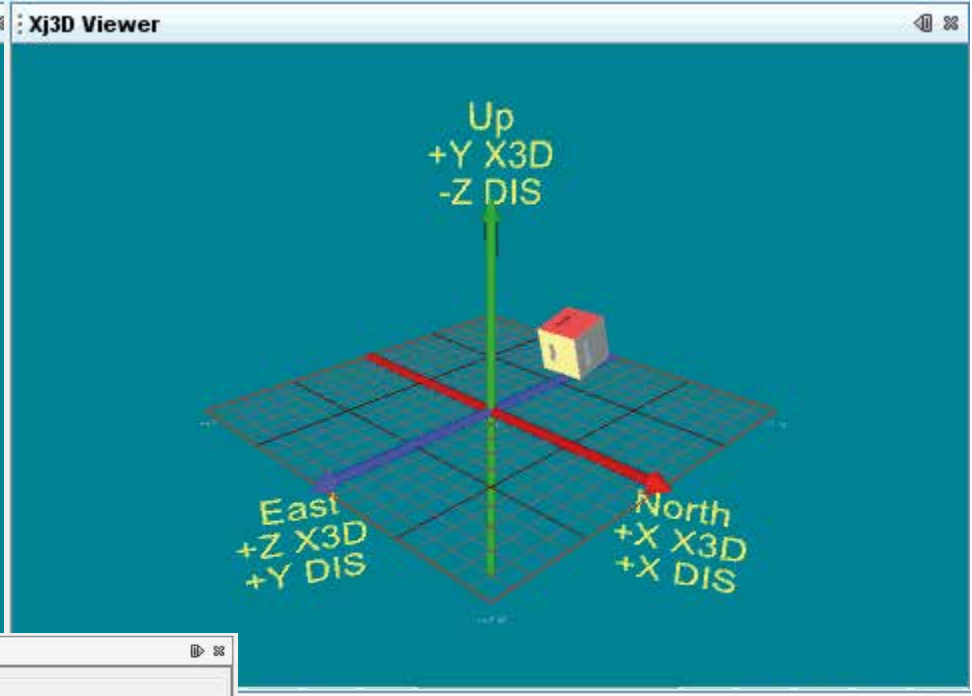
TestCubeEspdu.x3d - Editor

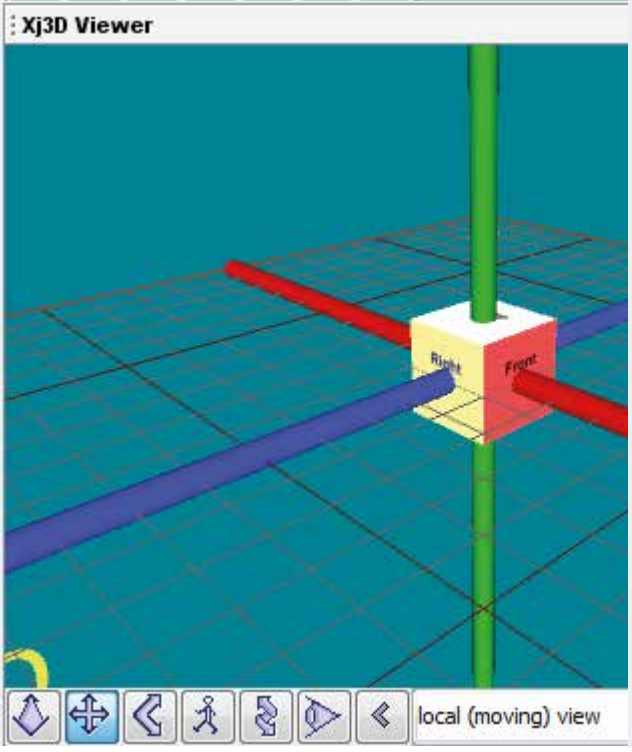
TestCubeEspdu.x3d

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN" "http://www.web3d.org/specifications/x3d-3.2.dtd">
<X3D profile='Immersive' version='3.2' xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance' xsd:noNamespaceSchemaLocation='http://www.web3
  <head>
    <component level='1' name='DIS'/>
    <meta content='TestCubeEspdu.x3d' name='title'/>
    <meta content='Don Brutzman' name='creator'/>
    <meta content='19 January 2009' name='created'/>
    <meta content='17 October 2010' name='modified'/>
    <meta content='DIS-enabled test shape for DIS ESPDU test panel.' name='description'/>
    <meta content='http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/TestCubeEspdu.x3d' name='identifier'/>
    <meta content='X3D-Edit 3.2, https://savage.nps.edu/X3D-Edit' name='generator'/>
    <meta content='../license.html' name='license'/>
  </head>
  <Scene>
    <Background skyColor='0 0.509804 0.580392'/>
    <Viewpoint description='ESPDU test cube' orientation='-0.4335 0.8831 0.1796 0.8772' position='25 15 25'/>
    <EspduTransform DEF='TestCubeTransform' address='239.1.2.3' applicationID='1' entityID='2'
      marking='TestCube' networkMode='networkReader' port='62040' siteID='0'>
      <Viewpoint description='local (moving) view' orientation='-0.4335 0.8831 0.1796 0.8772' position='10 6 10'/>
      <Switch whichChoice='0'>
        <Inline url='"TestCube.x3d"
                 "http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/TestCube.x3d"'/>
        <!-- alternate object to view -->
        <Transform DEF='ShrinkHeloToToySize' scale='0.25 0.25 0.25'>
          <Inline url='"../StudentProjects/StealthHelo.x3d" "http://www.web3d.org/x3d/content/examples/Basic/StudentProjects/StealthHelo.x3
        </Transform>
      </Switch>
    </EspduTransform>
    <!-- provide visual context with axes and grid -->
    <Transform scale='10 10 10'>
      <Inline url='"CoordinateAxesX3dDIS.x3d" "http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/CoordinateA
               "CoordinateAxesX3dDIS.wrl" "http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/CoordinateA
    </Transform>
    <Inline DEF='GridXY_20x20Fixed' url='"../../Savage/Tools/Authoring/GridXZ_20x20Fixed.x3d" "https://savage.nps.edu/Savage/Tools/Authorir
                                         "../../Savage/Tools/Authoring/GridXZ_20x20Fixed.wrl" "https://savage.nps.edu/Savage/Tools/Authorir
  </Scene>
</X3D>
```

36 | 42 INS




DIS ESPDU Test Panel

Translation

| | meters | -100 | -50 | 0 | 50 | 100 | scale |
|---|---|---|---|---|---|---|---|
| X | 5 | | | | | | 1.0 |
| Y | 5 | | | | | | 1.0 |
| Z | 0 | | | | | | 1.0 |

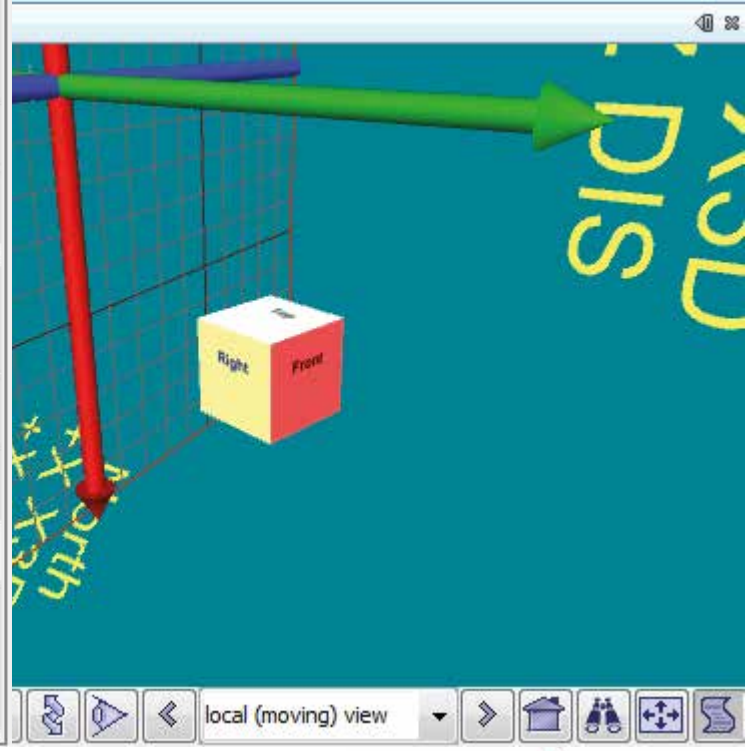
Rotation

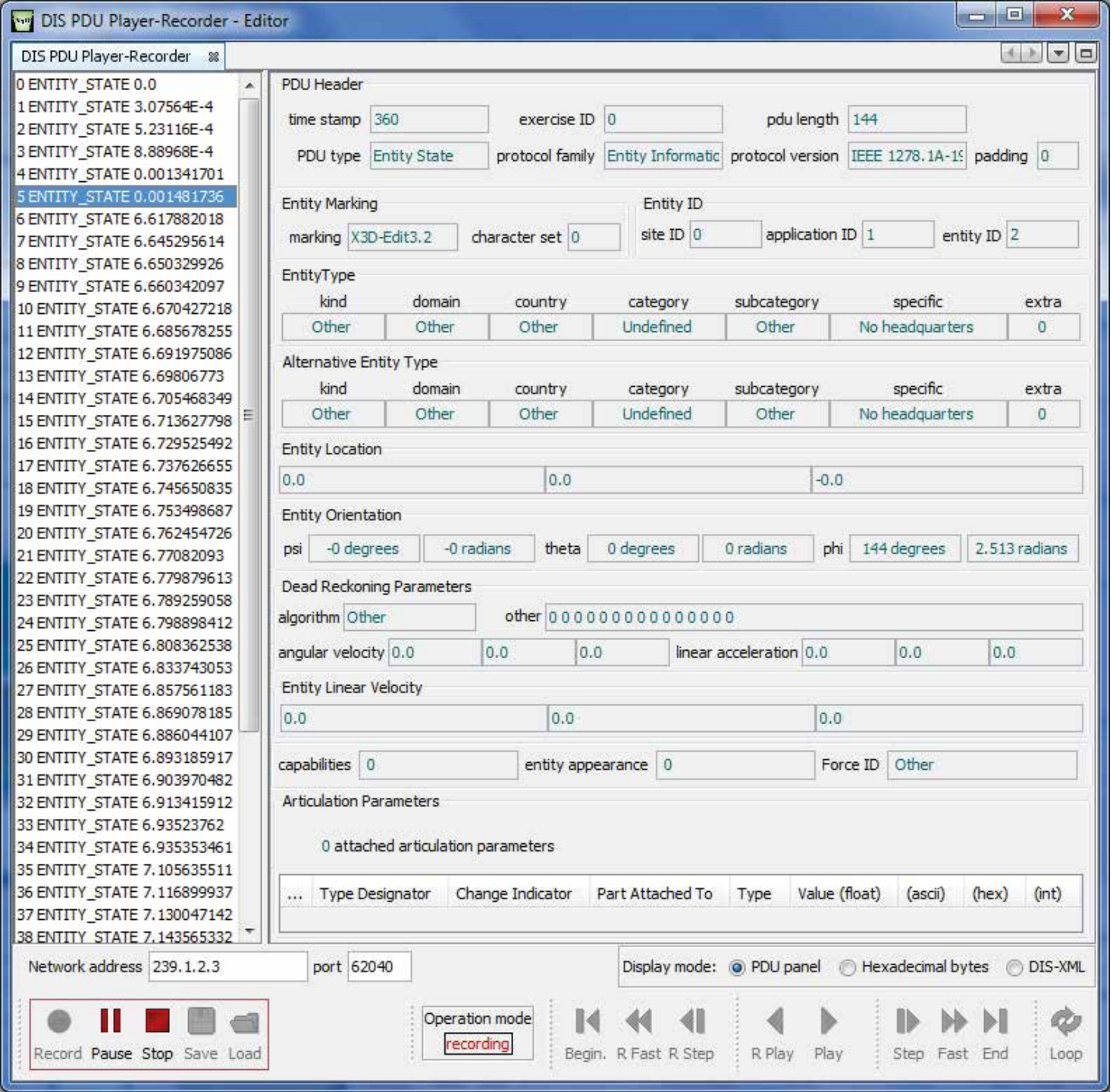
| | degrees | -180° | -90° | 0° | 90° | 180° | radians |
|---|---|---|---|---|---|---|---|
| roll | 13 | | | | | | 0.227 |
| pitch | 91 | | | | | | 1.588 |
| yaw | -2 | | | | | | -0.035 |

reset positions | reset rotations

DIS Settings

| | | | |
|---|---|---|---|
| marking | X3D-Edit3.2 | site ID | 0 |
| address | 239.1.2.3 | application ID | 1 |
| port | 62040 | entity ID | 2 |






DIS player-recorder

embedded in X3D-Edit

# DIS PDU Sender Test menu selection

Menu item:

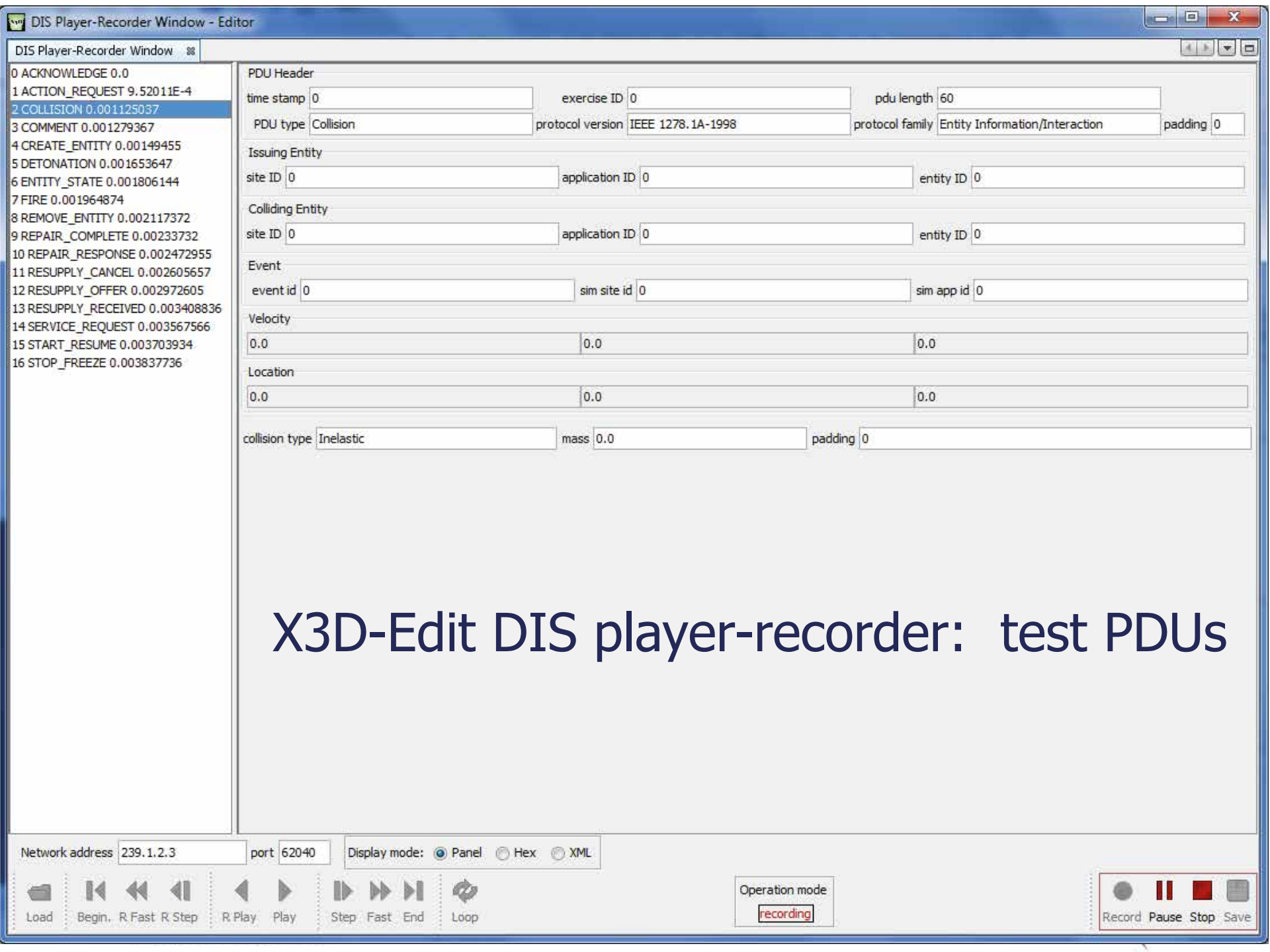
- *X3D* > *Networking* > *DIS PDU Sender Test*

Sends one PDU of each type supported

Useful for testing application connectivity

- Console of logged results: *View* > *IDE Log*
- Example excerpt:

```
Sent PDU of type edu.nps.moves.dis.CollisionPdu
Sent PDU of type edu.nps.moves.dis.CommentPdu
Sent PDU of type edu.nps.moves.dis.CreateEntityPdu
Sent PDU of type edu.nps.moves.dis.DetonationPdu
Sent PDU of type edu.nps.moves.dis.EntityStatePdu
Sent PDU of type edu.nps.moves.dis.FirePdu
Sent PDU of type edu.nps.moves.dis.RemoveEntityPdu
[...]
```




# X3D-Edit DIS player-recorder: test PDUs

# Also in NPS Savage archives: specific scenarios available

***Scenarios***

| | | |
|---|---|---|
| Amphibious Raid Camp Pendleton | Capture The Flag | Collision Uss Greeneville Mv Ehime Maru |
| Jfcom Dcee Exercise July 2003 | Limited Objective Experiment Port Hueneme | Remus Mission 10 MAR 2003 |
| Tank Maneuver | Uss Cole Terrorist Attack | UW 3303 Minefield Search |

These scenes typically have

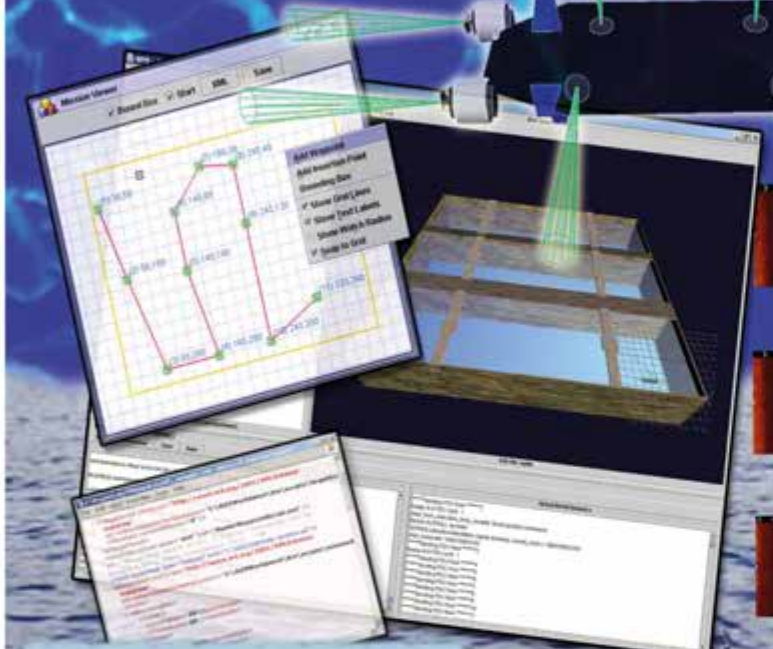
- EspduTransform nodes with session info, as parent nodes translating/rotating child models
- Inline nodes that retrieve non-networked platform models for rendering in the scene

# Autonomous Unmanned Vehicle (AUV) Workbench

The NPS Autonomous Unmanned Vehicle Workbench (AUVW) supports physics-based mission rehearsal, real-time task-level control of robot missions, and replay of recorded results in support of autonomous unmanned underwater, surface and air vehicles.

- Includes embedded DIS networking to connect physics-based modeling of robot missions to Xj3D visualization pane using Savage models
- https://savage.nps.edu/AuvWorkbench
- https://savage.nps.edu/Savage/AuvWorkbench/OperatingAreas

Robot Mission Planning and 3D Visualization



# Autonomous Unmanned Vehicle Workbench

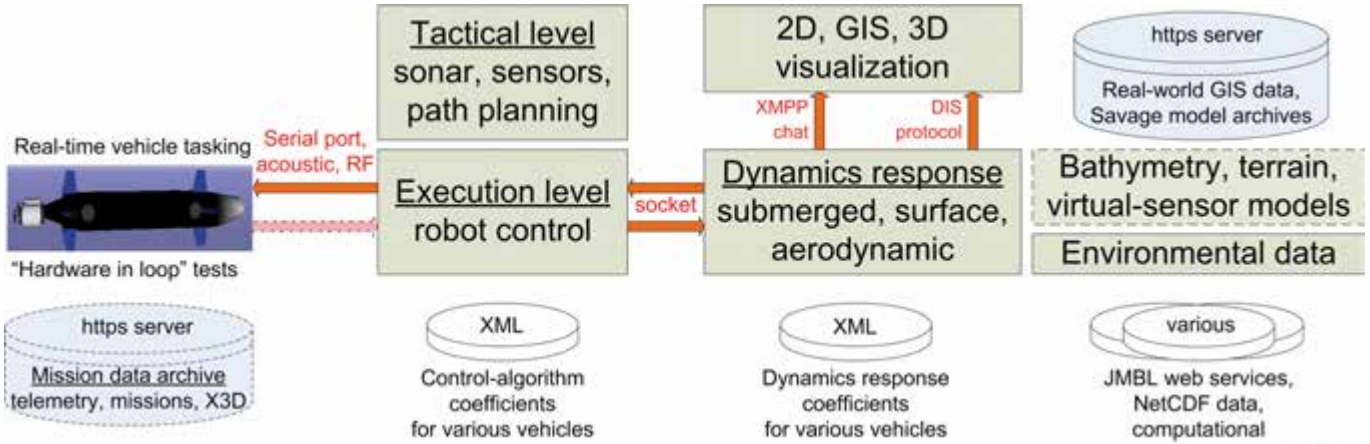
*Rehearsal*
*Reality*
*Replay*

THE MOVES INSTITUTE
NAVAL POSTGRADUATE SCHOOL
contact: brutzman@nps.navy.mil

"Effects-Based Thinking"

# AUV Workbench DIS Output

- Robot mission definitions are executed by robot controller, driving the vehicle
- Dynamics engine computes motion response, sent to network and Xj3D viewer via DIS
- DIS packets are DIS-XML chat or native PDUs



# DIS bridging

Some applications can be found in Open-DIS codebase to bridge DIS packets LAN-to-LAN

- Accomplished by opening unicast socket between LANs which passes packets, relaying them to/from multicast at endpoints
- This is needed since routing defaults do not allow multicast to escape an individual LAN
- Macedonia, Michael R. and Brutzman, Donald P., "MBone Provides Audio and Video Across the Internet," *IEEE COMPUTER*, vol. 27 no. 4, April 1994, pp. 30-36.

# Selectively Reliable Multicast Protocol (SRMP) DIS-HLA Gateway

*GMUGateway* is a research-oriented protocol translator developed in Java for distributed simulations.

- Provides interoperability among different types of simulation architectures
- Converts DIS protocol packets to High Level Architecture (HLA) Run-Time Infrastructure (RTI) service calls, and vice versa.
- Uses NPS DIS-Java-VRML implementation (for DIS) and RPR-FOM (for HLA)
- http://netlab.gmu.edu/SRMP/gatewayDoc.php

# DIS networking thesis: Steve Zeswitz

Zeswitz, Steven, *NPSNET: Integration of Distributed Interactive Simulation (DIS) Protocol for Communication Architecture and Information Interchange*, Masters Thesis, Naval Postgraduate School, Monterey California, September 1993.

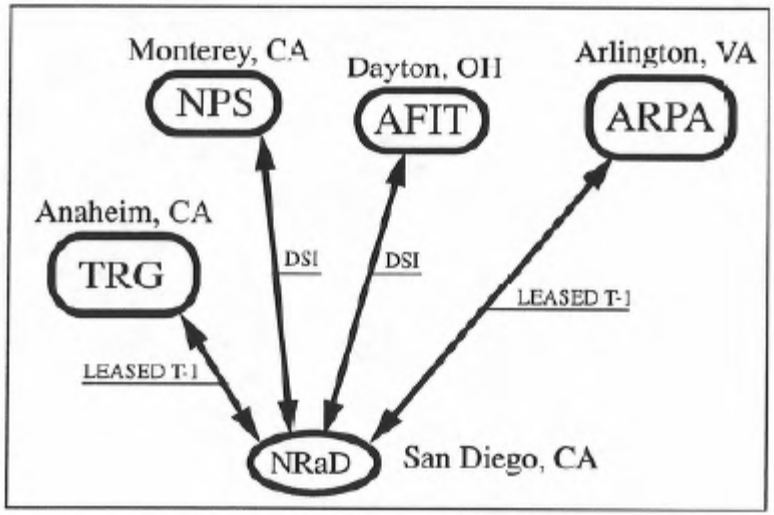


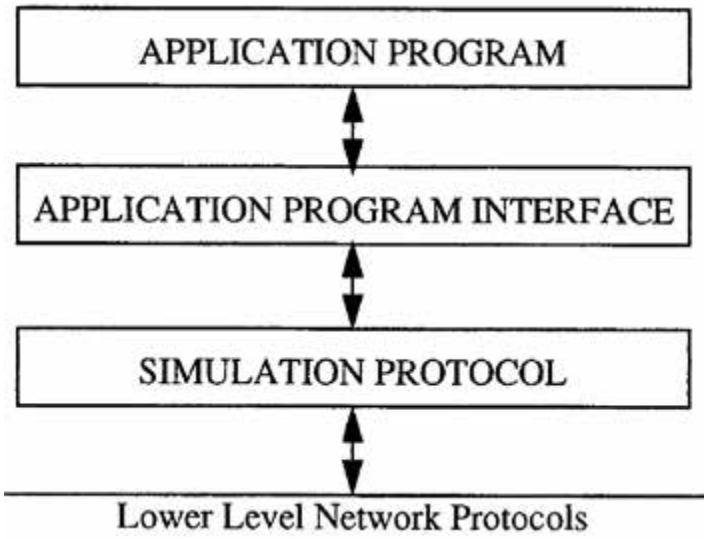
Figure 7 Wide Area Network Configuration 2

Early work.

# DIS networking thesis: Steve Stone

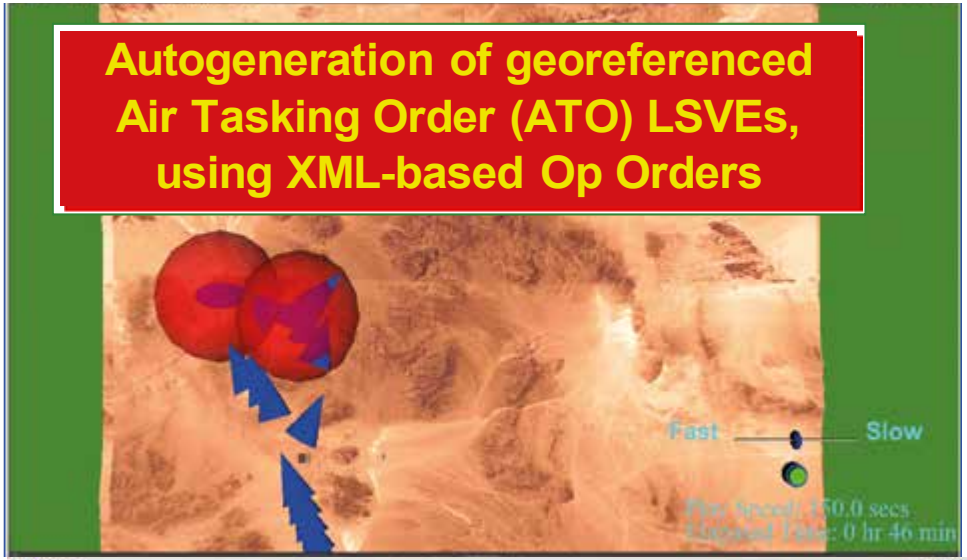
Stone, Steven W.,

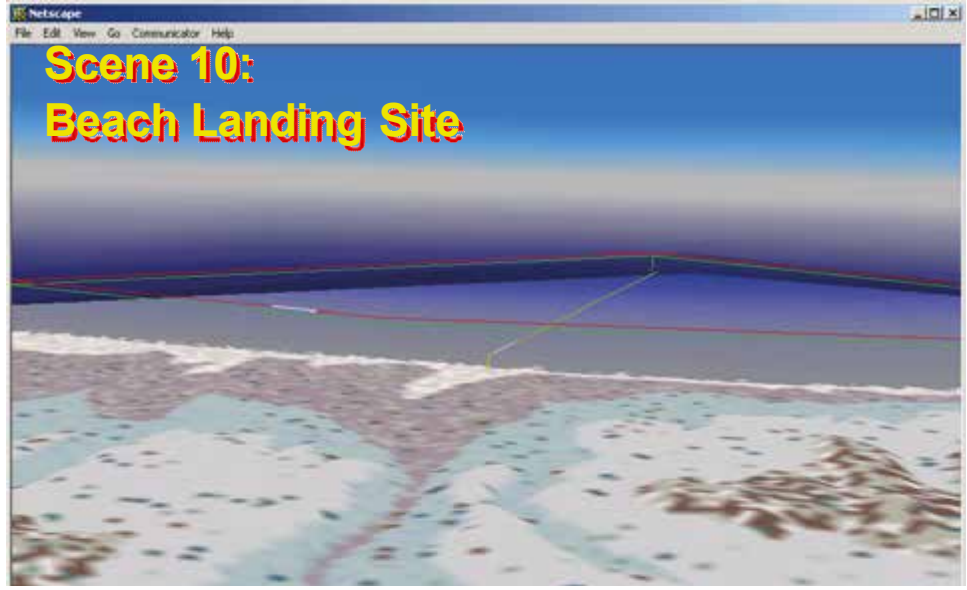
*A rapidly reconfigurable, application layer, virtual environment network protocol,* Masters Thesis, Naval Postgraduate School (NPS), Monterey California, June 1996. Built an improved NPSNET implementation.



# Savage thesis: Shane Nicklaus

Nicklaus, Shane D., *Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE)*, Master's Thesis, Naval Postgraduate School, Monterey California, September 2001. Information Systems Technology curriculum. Co-advisors Curtis L. Blais and Dan Boger.



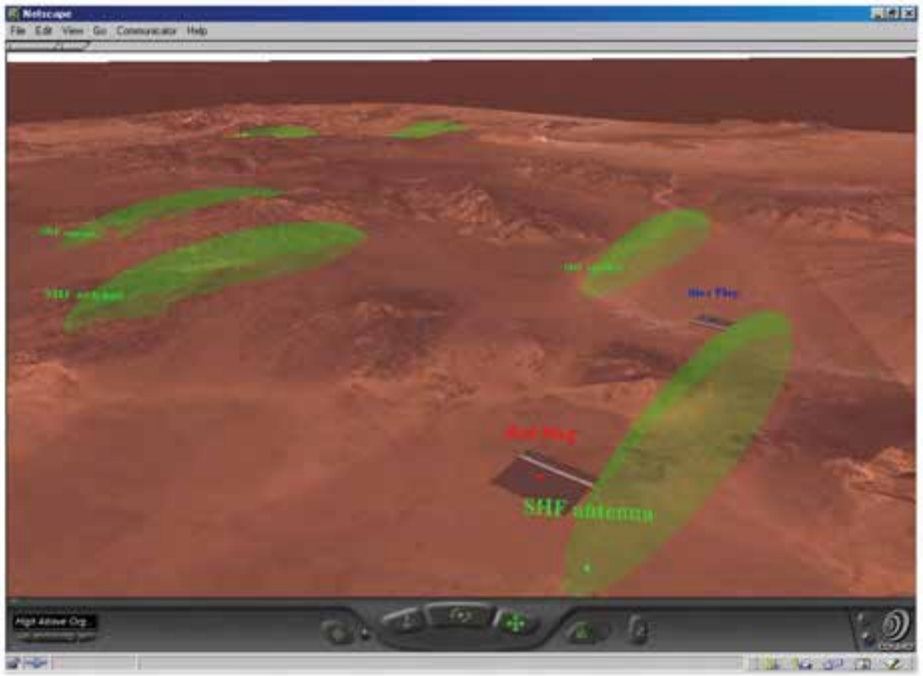
# H-Anim thesis:  Tom Miller

Miller, Thomas E., *Integrating Realistic Human Group Behaviors Into A Networked 3D Virtual Environment*, Master's Thesis, Naval Postgraduate School, Monterey California, September 2000.  MOVES curriculum.  Received NPS Outstanding Academic Achievement Award for highest academic honors among Department of Defense (DoD) students.


Human team preparing to
enter helicopter

# Signals thesis:  Dave Laflam

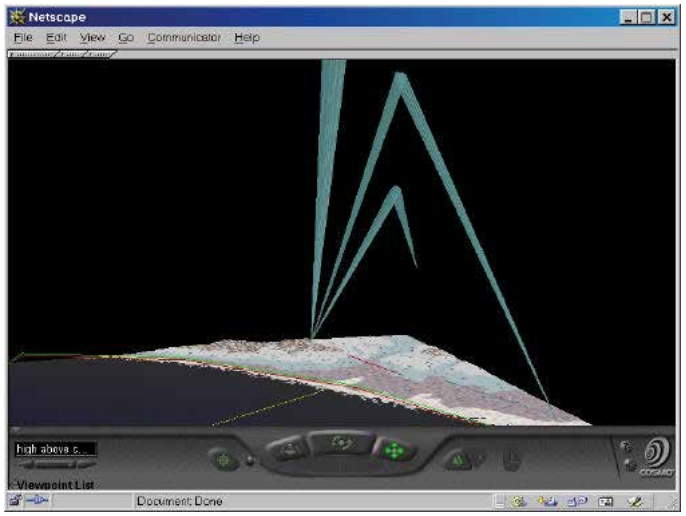
Laflam, David W.,
*3D Visualization of Theater-Level Radio Communications Using a Networked Virtual Environment*,
Master's Thesis, Naval Postgraduate School, Monterey California, September 2000.
MOVES curriculum.




web 3D CONSORTIUM

X3D

# Visualization thesis: Mike Hunsberger



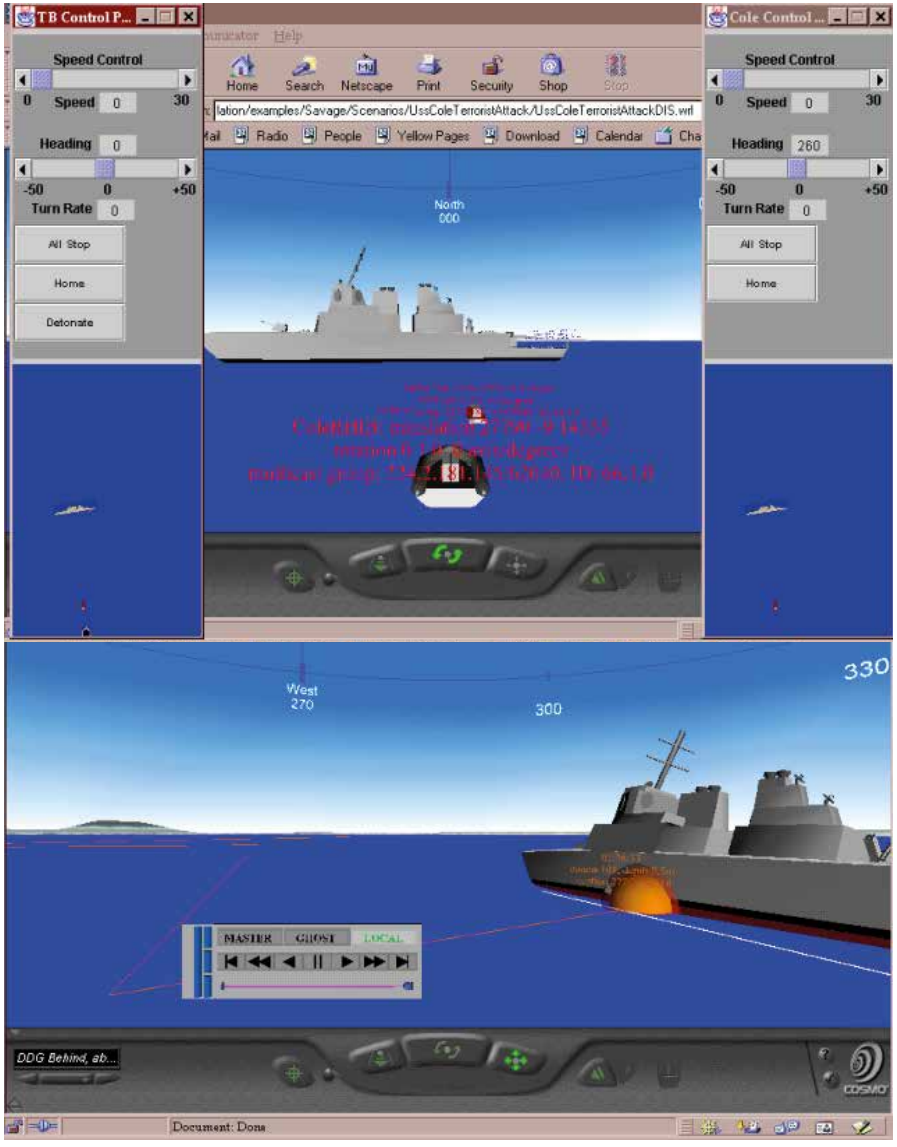

Hunsberger, Michael G., *3D Visualization Of Tactical Communications for Planning and Operations Using Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D)*, Master's Thesis, Naval Postgraduate School, Monterey California, June 2001. Awarded Air Force Association Award for Outstanding Air Force Student. Dual Master of Science degrees: Systems Technology for Joint $C^3$ Systems and Computer Science.

web|3D CONSORTIUM

<X3D>

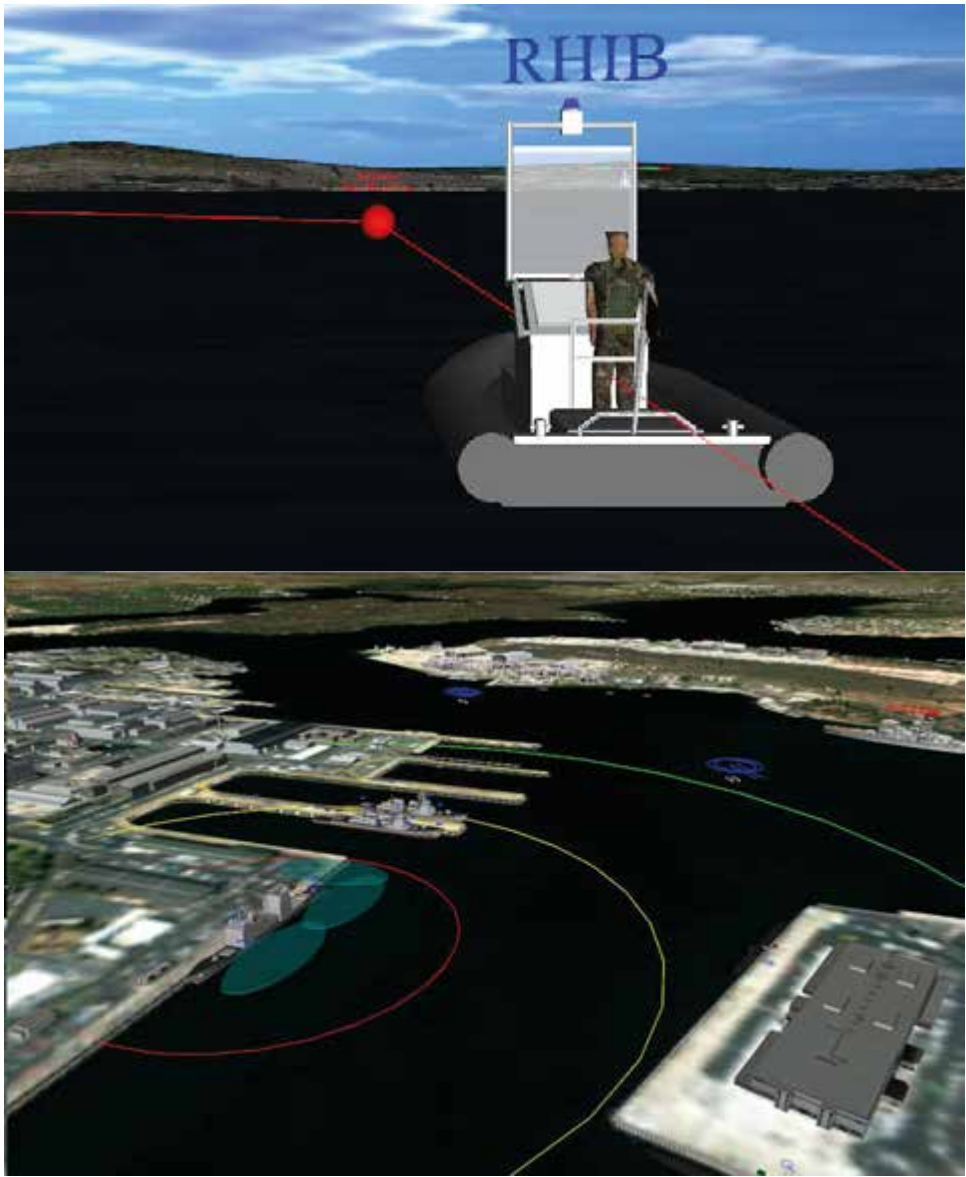
# Scenario thesis: James Harney

Harney, James W., *Analyzing Anti-Terrorist Tactical Effectiveness of Picket Boats for Force Protection of Navy Ships Using X3d Graphics and Agent-Based Simulation,* Masters Thesis, Naval Postgraduate School, Monterey California, March 2003. Co advisors Curtis L. Blais, Gordon Schacher, and John Hiles.



# Scenario thesis: Pat Sullivan

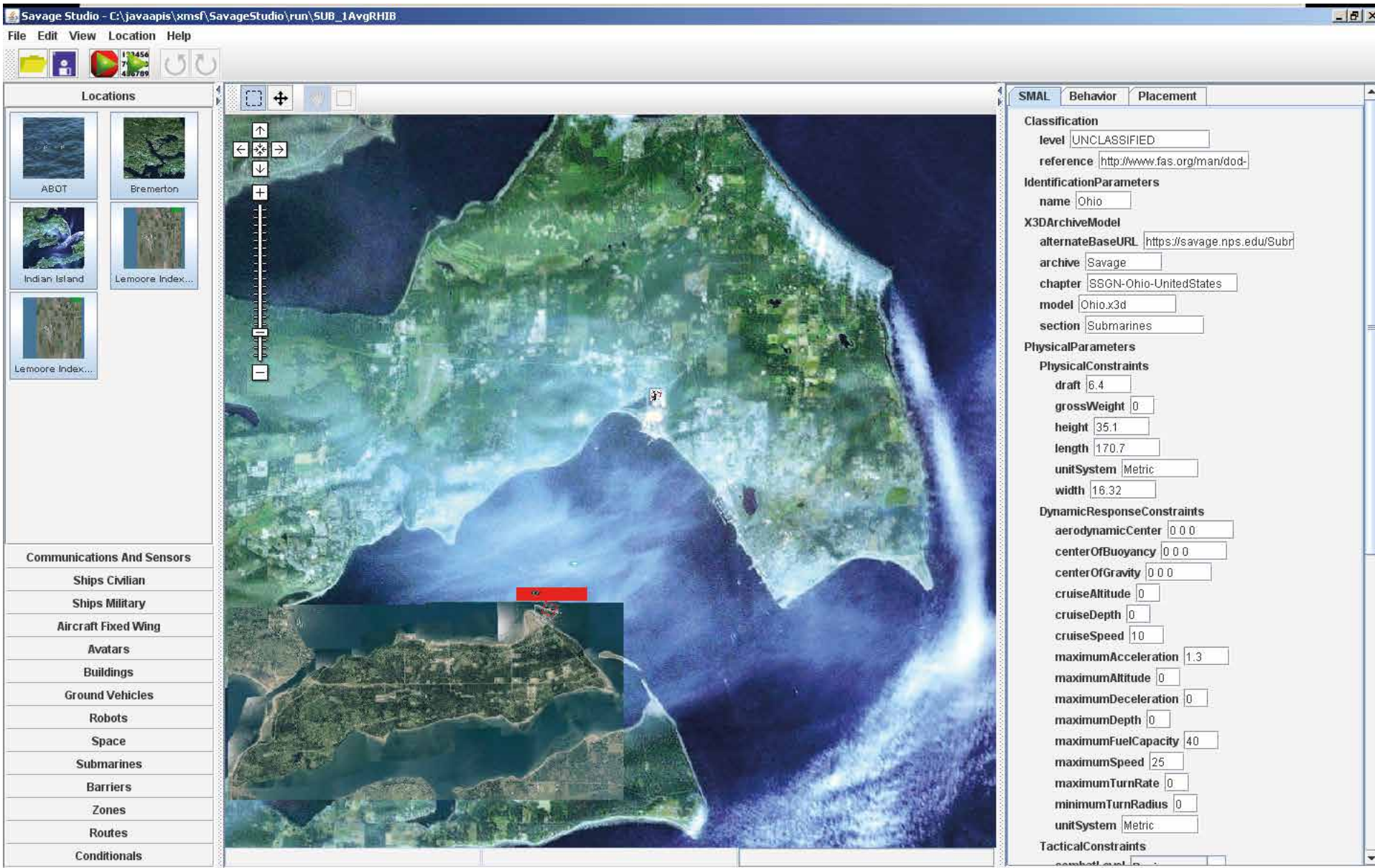
Sullivan, Patrick J., *Evaluating the Effectiveness of Waterside Security Alternatives for Force Protection of Navy Ships and Installations using X3D Graphics and Agent-Based Simulation,* Masters Thesis, Naval Postgraduate School, Monterey California, September 2006. Co-advisor Curt Blais.



# SMAL Metadata Thesis: Travis Rauch

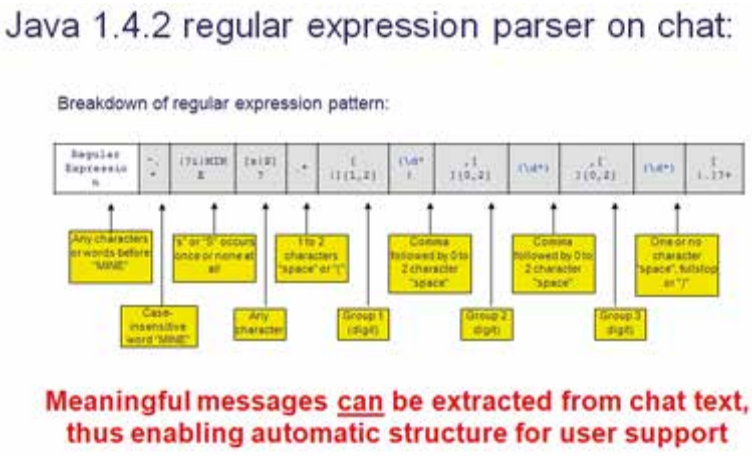
- Rauch, Travis, *Savage Modeling Analysis Language (SMAL): Metadata for Tactical Simulations and X3D Visualizations,* Masters Thesis, Naval Postgraduate School, Monterey California, March 2006.
- SAVAGE Modeling and Analysis Language (SMAL) is the NPS MOVES strategy for identifying tactical, physical and simulation-oriented information regarding models for vehicles, terrain and entities in virtual environments. Equivalent XML and X3D representations for SMAL are defined.
- https://savage.nps.edu/Savage/Tools/SMAL/SMAL.html

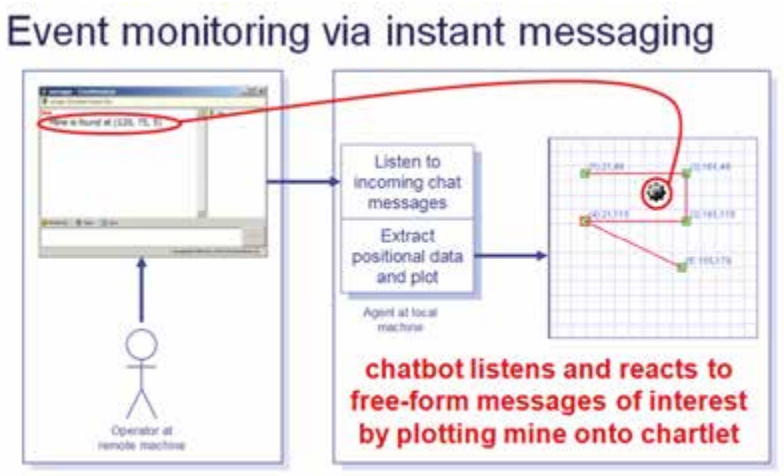
# SavageStudio scenario using SMAL metadata



# DIS, XMPP thesis: LEE, Daryl Chin-Siong

LEE, Daryl Chin Siong, *NPS AUV Workbench: Collaborative Environment for Autonomous Underwater Vehicle (AUV) Mission Planning and 3D Visualization*, Master's Thesis, Naval Postgraduate School, Monterey California, March 2004. Computer Science curriculum. Co-advisor Curtis Blais, second readers John Hiles and Duane Davis.



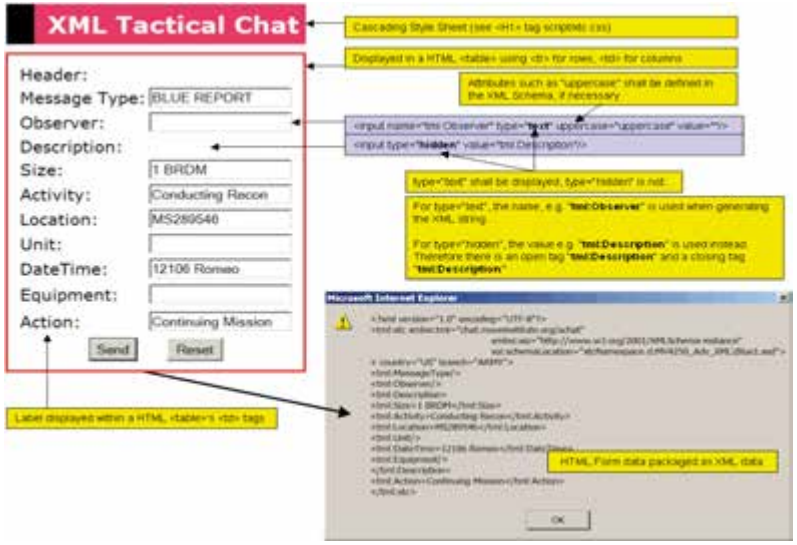



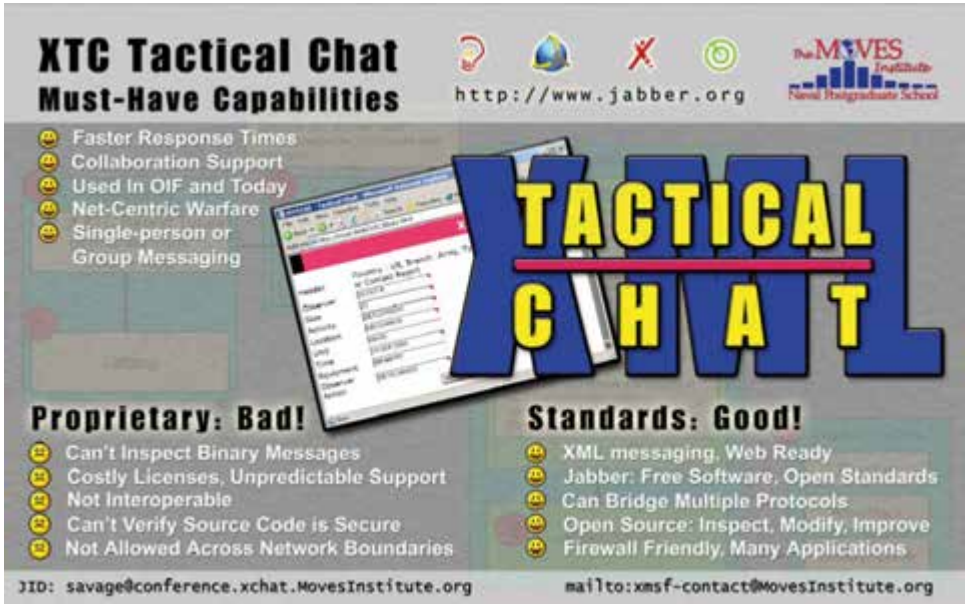
# XMPP Chat Thesis: Dan DeVos

DeVos, Daniel A., *XML Tactical Chat (XTC): The Way Ahead for Navy Chat*,

Masters Thesis, Naval Postgraduate School, Monterey California, September 2007. Second reader Don McGregor.

Set stage for DIS-XML.



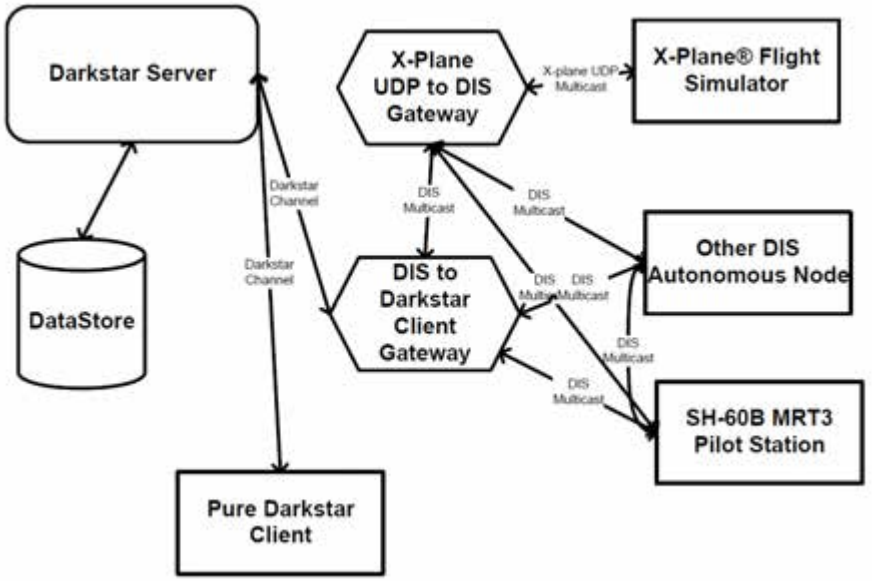



# Simulator thesis: Chris Fitzpatrick

Fitzpatrick, Christopher, *Integration of Robotic Technology, X3D Computer Graphics and Digital Imaging to Modernize the Expeditionary Warfare Demonstrator (EWD),* Masters Thesis, Naval Postgraduate School, Monterey California, September 2009. Second reader Amela Sadagic. Awarded SPAWAR Student Research Fellowship September 2008.

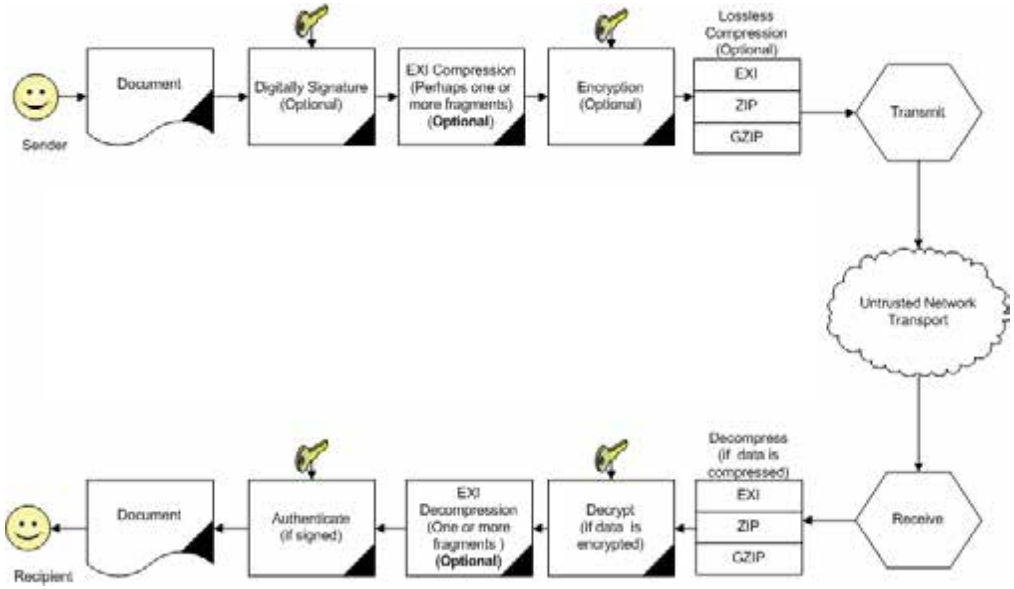
# MMOG thesis: Tariq Rashid

Rashid, Tariq, *Integrating Distributed Interactive Simulations with the Project Darkstar Open-Source Massively Multiplayer Online Game (MMOG) Middleware*, Masters Thesis, Naval Postgraduate School, Monterey California, September 2009.
Co-advisor Don McGregor, second reader Amela Sadagic.




# XML Security Thesis: Jeff Williams

Williams, Jeffrey S., *Document-Centric XML Encryption and Authentication for Coalition Messaging*, Masters Thesis, Naval Postgraduate School, Monterey California, September 2009.




Potentially usable with DIS-XML, EXI compression

# Discrete Event Simulation (DES)

# DES: Simkit

# DES: DisKit

# DES:  Viskit

# DES: SavageStudio

# Additional Resources

# Wireshark


Wireshark is an open-source network-protocol analyzer and debugging tool

Wireshark has a built-in decoder for DIS

- Analyze->Enabled Protocols to make sure DIS is turned on
- Select a packet
- Analyze->Decode As to view as DIS; can also set all packets sent to or from a port to be automatically decoded as DIS

Available at

- http://www.wireshark.org

# Jenkins continuous build server

NPS has stood up an online continuous integration server for regular builds

- Uses open-source Jenkins under Apache.

for model and software developers

Performs nightly compilation builds of code, error-checking validation of model content

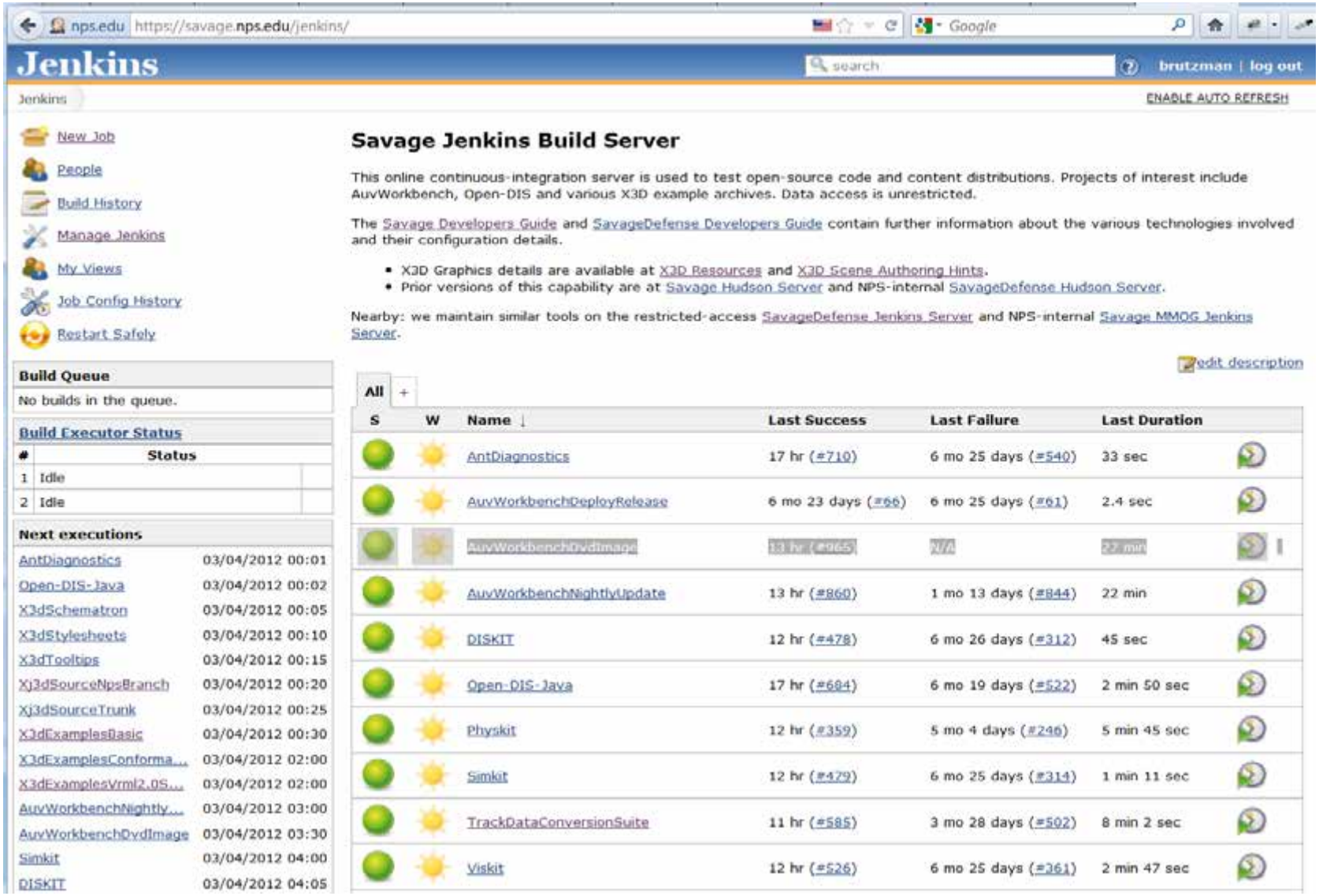
- Xj3D open-source player and other tools
- X3D example archives, all on SourceForge

Provide further quality assurance (QA) for X3D

- https://savage.nps.edu/Savage/developers.html#Jenkins
- https://savage.nps.edu/jenkins

web3D CONSORTIUM

# Jenkins continuous-build testing




https://savage.nps.edu/jenkins/job/Open-DIS-Java

# Chapter Summary

# Chapter Summary

X3D DIS component allows networking of shared entity state for positioning objects, entity management, collision detection, fire/detonate projectiles, and propagation/receipt of signals

Multiple technical challenges are steadily being addressed, simple scalability is the primary goal

Ongoing work is building repeatable, royalty-free results available for broad use on the Web

# Suggested exercises

Test, adapt example scenes provided in archive

Create (or adapt) a simple Java, C++ or Javascript program to send PDUs

Monitor PDUs being sent from an existing DIS simulation program using X3D model, X3D-Edit

(we're getting closer...) join a virtual environment

# Future work: lots!

# TODO #1: Planned projects

- Geospatial coordinates in X3D Specification
- DisEntityManager scenario: working example
- Upgrade Open-DIS to DIS 2012 changes
  - Status?
- Recording, playing back DIS PDU packets to/from a MySQL database
- Provide updated support for VRML/X3D examples in Mark Pullen's online course Networked Virtual Environments (NVEs)

# TODO #2

- Open-DIS ports to Javascript: WebSockets and WebRTC available, improve documentation
- Add Open-DIS to X3DOM: testing in progress
  - Breakthrough moment after years of work
- Upgrade MV3500 Networked Simulation as an online distributed-learning course

# TODO #3: X3D-Edit

- Open-DIS server stream-relay capabilities
  - Simplify, automate server-to-server (s2s) bridging
  - Embedded in X3D-Edit for local server creation
  - Bundle over XMPP chat for broader routing
- Autogenerate Java, Javascript enumerations using Enumeration Byte Value (EBV) .xml
  - Publish classes in Open-DIS archive (check current)
  - Bundle in X3D-Edit panes, online documentation
- DIS data capture, distillation as smoothed interpolators for offline/archived playback
  - Track recording and playback for any entity

# TODO #3: other NPS tools

- X3DOM interoperability
  - X3dToX3dom.xslt stylesheet support
  - Tooltips and quality assurance (QA) testing
  - Publish series of examples
- Integrate, document visualization tools use
  - AUV Workbench mission publication, replay
  - Viskit playback control
  - SavageStudio scenario authoring
- Update past work to meet current research
  - Dave Laflam thesis on signals visualization
  - Tom Miller thesis on grouped humanoid animation

# TODO #4: other tools

- Codebase repeatability and interoperability
  - Wireshark usage and examples with DIS
  - AMIE virtual-world bridge connections
  - Test and Training Enabling Architecture (TENA) interoperability
  - Add DIS support to major X3D players: BS Contact, InstantReality, perhaps other codebases
  - X-Plane usage and examples with DIS
- Revisit scalable MMOG game server concepts
  - compare/contrast to SISO WebLVC work
  - Is another MMOG codebase really needed, or might peer-to-peer (p2p) approaches prove sufficient?

# TODO #5: and more, here we go!

- Important thesis work now available
  - Compare compression techniques using XML-based Efficient XML Interchange (EXI)
  - Encryption and signature of streams, PDUs
  - Security considerations of Web-based DIS

# Partnership, sponsor opportunities

- DIS-XML encoding proposed and tested as an addition to the DIS standard
- DIS-HLA bridge to an open-source RTI
- Consider integration with Test and Training Enabling Architecture (TENA)
- Your project here?

# References

# References 1

*X3D: Extensible 3D Graphics for Web Authors*
by Don Brutzman and Leonard Daly, Morgan Kaufmann Publishers, April 2007, 468 pages.


- http://x3dGraphics.com

X3D Resources and X3D Basic Examples Archive

- http://www.web3d.org/x3d/content/examples/X3dResources.html
- http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation

web|3D CONSORTIUM

<X3D>

# References 2

X3D-Edit Authoring Tool

- https://savage.nps.edu/X3D-Edit

X3D Scene Authoring Hints

- http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html

X3D Graphics Specification

- http://www.web3d.org/x3d/specifications
- Also available as help pages within X3D-Edit

# References 3

## SISO Digital Library for DIS

- SISO-REF-020-2008: DIS Plain and Simple Guide
- The Complete DIS PDU Guide (also database)
- Variable Parameter Record Guide (VPRG)
- DIS Version Difference Guide
- SISO-REF-010-2010-RC1 Enumeration and Bit Encoded Values for DIS (500 pages)

# References 4

## SISO Digital Library for DIS

- DIS Product Development Group
- DIS Find it Fast Guide
- DIS XML and Databases

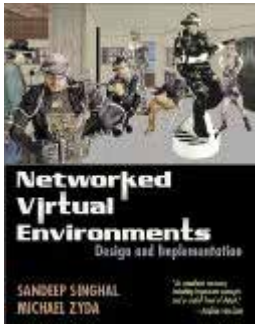
# References 5


- Ralph Toms and Cameron Kellough, *Guidelines for the Development of Efficient Algorithms for Spatial Operations*, SRI International, 20 February 2006
- Ralph Toms and Paul Birkel, *Choosing a Coordinate Framework for Simulations*, 1999
- Paul Birkel and Ralph Toms, SEDRIS *Spatial Reference Model (SRM) tutorial*, 2004

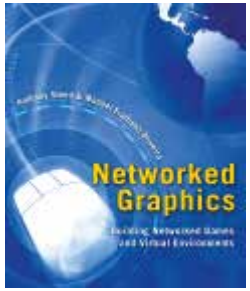
# References 6


- David L. Neyland, *Virtual Combat: A Guide To Distributed Interactive Simulation*, Stackpole Books, 1997.


- Sandeep Singhal and Michael Zyda, *Networked virtual environments: design and implementation*, ACM Press/Addison-Wesley, 1999. Online course available.




- Anthony Steed and Manuel Fradinho Oliveira, *Building Networked Games and Virtual Environments*, Morgan Kaufman, 2009.



# Contact

**Don Brutzman**

*brutzman@nps.edu*

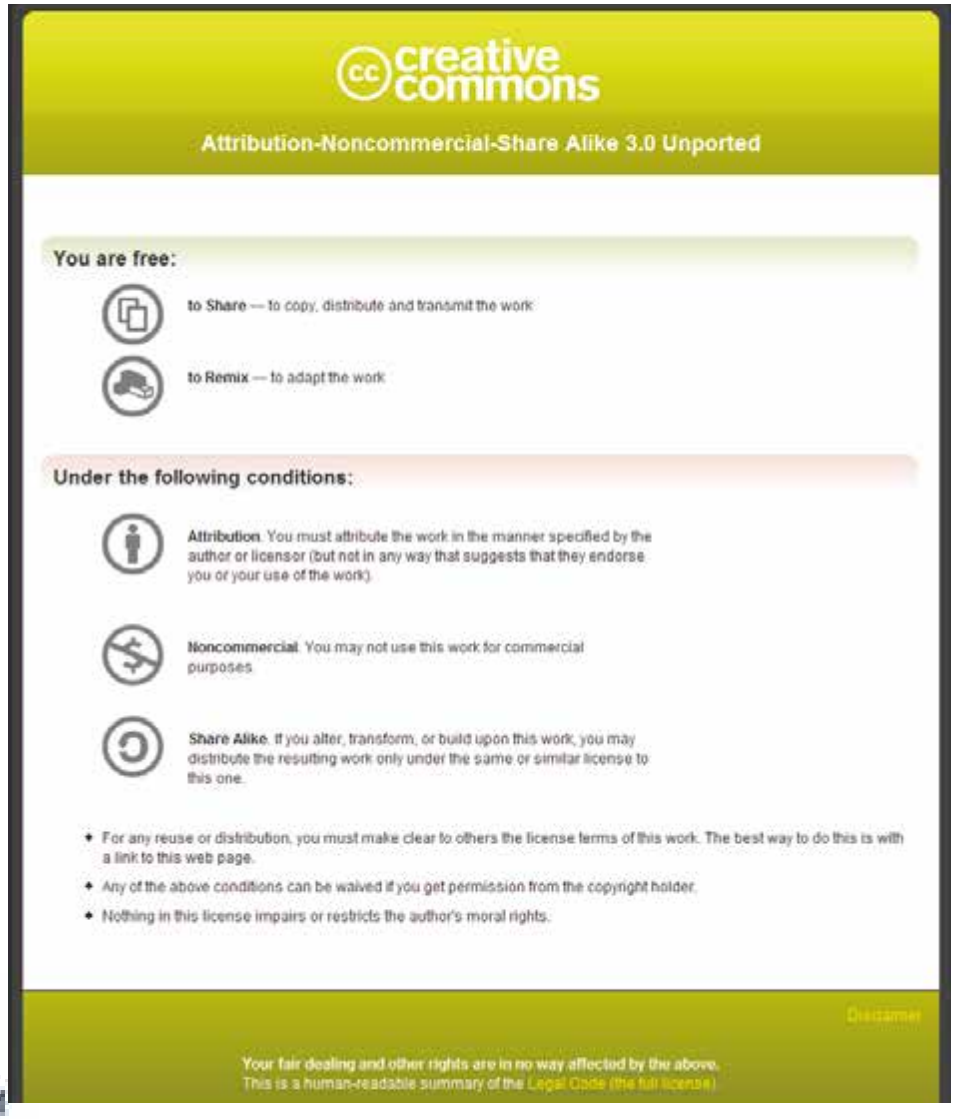
*http://faculty.nps.edu/brutzman*

Code USW/Br, Naval Postgraduate School
Monterey California 93943-5000 USA
1.831.656.2149 voice

web|3D CONSORTIUM

# Creative Commons open-source license

http://creativecommons.org/licenses/by-nc-sa/3.0




web 3D CONSORTIUM

X3D

# Open-source license
# for X3D-Edit software and X3D example scenes

http://www.web3d.org/x3d/content/examples/license.html

# X3D Graphics and Distributed Interactive Simulation (DIS) Networking

*"All the world's a stage,*
*And all the men and women merely players;"*
- William Shakespeare, *As You Like It*

Don Brutzman
Naval Postgraduate School
brutzman@nps.edu

web3D CONSORTIUM


"All the worlds a stage"

# Contents




# Chapter Overview


# Overview

IEEE Distributed Interactive Simulation (DIS) protocol has been used for many years to build networked simulations that share state

X3D DIS component aligns these capabilities with X3D scenes to enable sharing of state data

- EspduTransform: protocol data units (PDUs) for EntityState, Collision, Fire, Detonation
- Signals: ReceiverPdu, SignalPdu, TransmitterPdu

Various open-source tools, codebases available

- e.g. Open-DIS open source in Java, C++, C#, etc.
- Currently only one X3D browser implementation ☹

http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation

Distributed Interactive Simulation (DIS) is an open standard for conducting real-time platform-level wargaming across multiple host computers and is used worldwide, especially by military organizations but also by other agencies such as those involved in space exploration and medicine.

**Application protocol**

Simulation state information is encoded in formatted messages, known as protocol data units (PDUs) and exchanged between hosts using existing transport layer protocols, including multicast, though broadcast User Datagram Protocol is also supported.

# Overview: DIS Network Protocol

Distributed Interactive Simulation (DIS) network prototcol is an IEEE standard since 1995

- IEEE 1278 is a communications standard for physically based distributed simulations
- Standard defines the binary layout of a series of messages used to transmit simulation information
- Often used in military applications since IEEE 1278 covers a wide range of data, including entity location, velocity, and orientation, and more features such as signal and supply support
- Also used in civilian applications: air-traffic control, etc.


# PDU defined

Protocol data unit (PDU) generally considered to provide a unit block of information

- Delivered among peer entities on a network
- May contain control information, address information, or data

The IEEE specification for DIS defines a large number of PDU packets

- Standardizes byte format of packets "over the wire" for interoperability among participating simulations

web3D CONSORTIUM


http://en.wikipedia.org/wiki/Protocol_data_unit

PDUs are relevant in relation to each of the first 4 layers of the OSI model as follows:

1. The Layer 1 (Physical Layer) PDU is the bit
2. The Layer 2 (Data Link Layer) PDU is the frame
3. The Layer 3 (Network Layer) PDU is the packet
4. The Layer 4 (Transport Layer) PDU is the segment (e.g. TCP segment)

(Layer 5 and above are referred to as data.)

# DIS PDU types

DIS specification defines 67 PDUs in 12 families

- *Entity information/interaction:* Entity State, Collision, Collision-Elastic, Entity State Update, Attribute
- *Warfare:* Fire, Detonation, Directed Energy Fire, Entity Damage Status
- *Radio communications:* Transmitter, Signal, Receiver, Intercom Signal, Intercom Control
- *Simulation management:* Start/Resume, Stop/Freeze, Acknowledge
- etc.

web3D CONSORTIUM


*http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation#Protocol_data_units_.5B3.5D*

Family list continued:

•*Logistics:* Service Request, Resupply Offer, Resupply Received, Resupply Cancel, Repair Complete, Repair Response

•*Distributed emission regeneration:* Designator, Electromagnetic Emission, IFF/ATC/NAVAIDS, Underwater Acoustic, Supplemental Emission/Entity State (SEES)

•*Entity management, Minefield, Synthetic environment*

•*Simulation management with reliability*

•Live entity, Non-real time families

•*Information Operations*: Information Operations Action, Information Operations Report

# Virtues of DIS

- DIS is a binary protocol; the exact layout of PDUs in binary format is specified, along with byte order
- Many standardized enumeration values. For example, the value "1" in the Entity Type field of a PDU specifies that the PDU is an Entity State PDU.
- Other fields and enumerated values can define what type of entity is being described, such as an M1A2 tank, an Arleigh Burke class destroyer, etc.
- DIS standard describes functional algorithms for handling PDUs, dead reckoning, and more.
- Thus both syntax and semantics for interoperability are well defined.


# Network communications

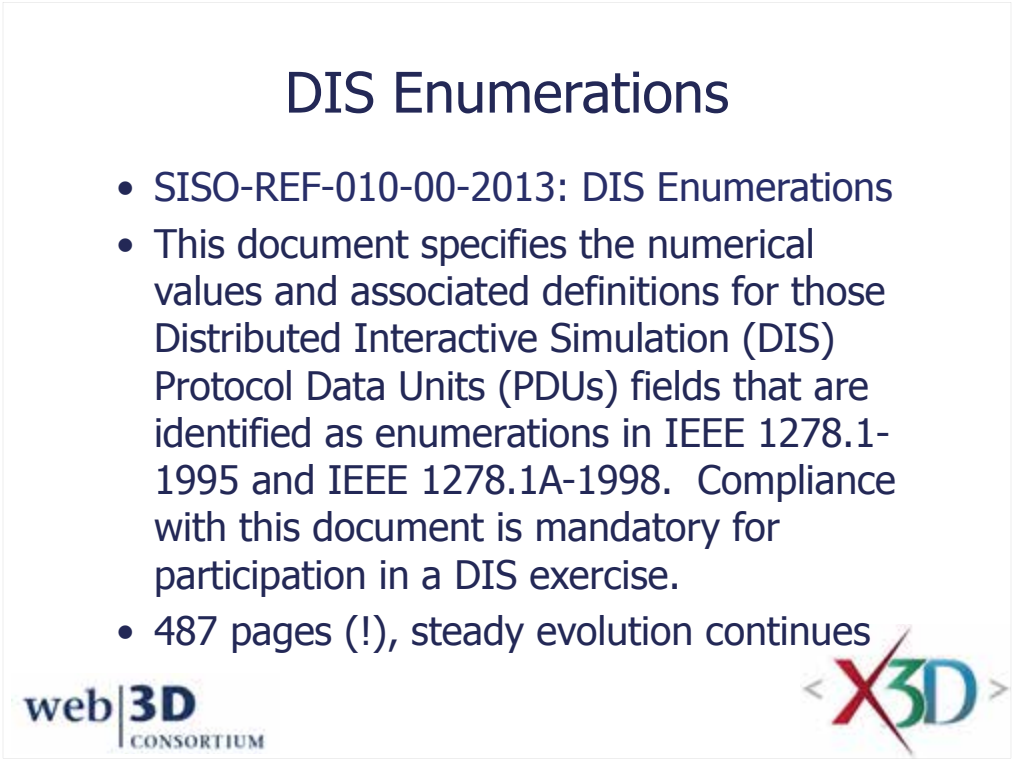
Two transport capabilities:

- Multicast (many-to-many on LAN) socket
- Unicast (point-to-point) socket

Entities and ownership

- Entity corresponds to some high-level vehicle that has control over its own motion
- Owners send packets, others listen for packets
- Multiple owners for one entity usually nonsensical

Thus non-symmetric configuration of readers and writers in each participants scene graph




"Steady evolution" indicated by multiple posts on weekly basis to add new entity types.

back to Table of Contents

# SIMNET and DIS

# IEEE, SISO and HLA


# SIMNET

- SIMNET was a wide area network with vehicle simulators and displays for real-time distributed combat simulation: tanks, helicopters and airplanes in a virtual battlefield. SIMNET was developed for and used by the United States military. SIMNET development began in the mid-1980s, was fielded starting in 1987, and was used for training until successor programs came online well into the 1990s.
- http://en.wikipedia.org/wiki/SIMNET


# DIS definition from Wikipedia

Distributed Interactive Simulation (DIS) is an open standard for conducting real-time platform-level wargaming across multiple host computers and is used worldwide, especially by military organizations but also by other agencies such as those involved in space exploration and medicine.

http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation


# DIS History 1

- The standard was developed over a series of "DIS Workshops" at the Interactive Networked Simulation for Training symposium, held by the University of Central Florida (UCF) Institute for Simulation and Training (IST). The standard itself is very closely patterned after the original SIMNET distributed interactive simulation protocol, developed by Bolt, Beranek and Newman (BBN) for Defense Advanced Research Project Agency (DARPA) in the early through late 1980s. BBN introduced the concept of dead reckoning to efficiently transmit the state of battle field entities.


# DIS History 2

- In the early 1990s, IST was contracted by the United States Defense Advanced Research Project Agency to undertake research in support of the US Army Simulator Network (SimNet) program. Funding and research interest for DIS standards development decreased following the proposal and promulgation of its successor, the High Level Architecture (simulation) in 1996. HLA was produced by the merger of the DIS protocol with the Aggregate Level Simulation Protocol (ALSP) designed by MITRE.


# DIS History 3

- There was a NATO standardisation agreement (STANAG 4482, Standardised Information Technology Protocols for Distributed Interactive Simulation (DIS), adopted in 1995) on DIS for modelling and simulation interoperability. This was retired in favour of HLA in 1998 and officially cancelled in 2010 by the NATO Standardisation Agency (NSA).
- Not clear why cancelled since DIS ≠ HLA...
- Credit: DIS site, history maintained by DIS PDG

http://en.wikipedia.org/wiki/Distributed_Interactive_Simulation


# Simulation Interoperability Standards Organization (SISO)

International organization with broad agenda.

- "Simulation Interoperability & Reuse through Standards"

SISO (http://www.sisostds.org) is recognized as

- NATO Standards Development Organization (SDO)
- IEEE Standards Sponsor
- Category C Liaison Organization, ISO/IEC (JTC 1)

SISO maintains and develops the DIS standard via working-group efforts and meetings, especially at Simulation Interoperability Workshop (SIW) series in Orlando FL and Europe annually


# SISO Vision

- SISO will serve the global community of modeling and simulation professionals, providing an open forum for the collegial exchange of ideas, the examination and advancement of M&S-related technologies and practices, and the development of standards and other products that enable greater M&S capability, interoperability, credibility, reuse, and cost-effectiveness.

- Reference: SISO-ADM-004-2010, available via http://www.sisostds.org/AboutSISO/Overview.aspx

## DIS Product Development Group (PDG)

The SISO Standards Activity Committee (SAC) has a DIS Product Development Group (PDG) which updated the core DIS standard, 1278.1, and reintegrated content of 1278.1a, to form approved update standard 1278.1-2012.

DIS standards can be purchased from IEEE (often available free to members, universities) and also available online to SISO members.


## IEEE Standards Maintained by SISO

- IEEE 1278.1-2012 - IEEE Standard for Distributed Interactive Simulation - Application Protocols
- IEEE 1278.2 - IEEE Standard for Distributed Interactive Simulation - Communication Services and Profiles
- IEEE 1278.3 - IEEE Standard for Distributed Interactive Simulation Exercise Management & Feedback (EMF) - Recommended Practice
- IEEE 1278.4 - IEEE Standard for Distributed Interactive Simulation - Verification Validation & Accreditation

http://www.sisostds.org/ProductsPublications/Standards/IEEEStandards.aspx

web3D CONSORTIUM


http://www.sisostds.org/ProductsPublications/Standards/IEEEStandards.aspx

# Getting copies of DIS standards

IEEE DIS standard documents are

- Available to SISO members without charge
- Available to IEEE members
- Often available to universities via site license



From: "Duncan Miller" (dmiller@sisostds.org) 21 OCT 2010

We are pleased to announced that SISO is now able to provide free access to SISO-developed IEEE Standards, as well as standards developed and promulgated directly by SISO as an accredited Standards Development Organization.

Access to these standards via the SISO website is restricted to SISO members only. Information on how to become a SISO member and/or maintain your membership may be found on the SISO web site, http://www.sisostds.org under

"Membership > Become a Member."

The initial standards available under this agreement include [...HLS details...]

The DSEEP (1730) and the new version of DIS (1278.1) standards will be added in the next few months as they are completed and published, ensuring that SISO members have access to the newest versions of the standards.

To login: http://discussions.sisostds.org/default.asp?boardid=2&action=9&read=49142&fid=11#69589

SISO: http://www.sisostds.org


# SISO digital library:

# DIS Product Study Group (PSG)

References section includes several links

## DIS and High Level Architecture (HLA)

DIS protocol defines both wire format and semantics for consistent shared state

- Stateless, entities can join/leave any time
- Interoperability for all compliant implementations

HLA Run-Time Interface (RTI) is for codebases that implement HLA design principles

- Object model principles, no wire format, though DIS packets might be passed internally (RPR-FOM)
- Entities must be predeclared prior to start
- No interoperability guarantee for implementations
- Not an interoperability standard, usually proprietary

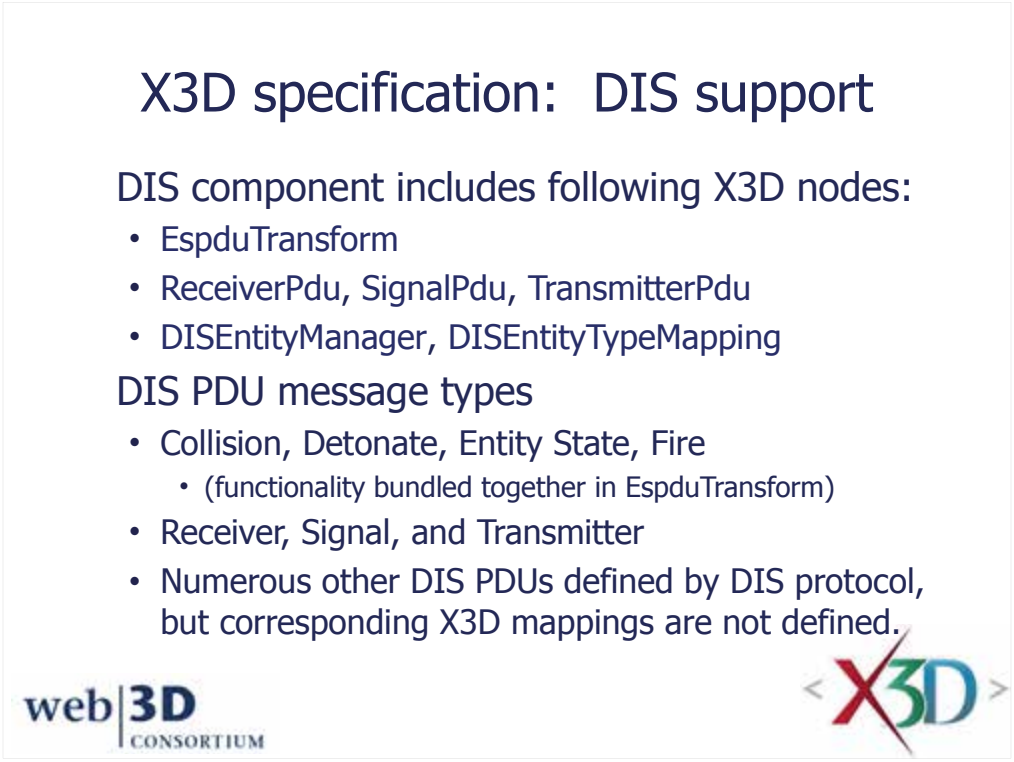
Not clear why Simulation Interoperability Standards Organization (or IEEE for that matter) approved a standard that does not include interoperability. Further hard to understand what the point is to have a standard that does not include interoperablity.

# X3D specification: DIS component

X3D specification is componentized for extensibility, DIS Component is one of many:

- http://www.web3d.org/standards
- Part 1: Architecture and base components
- Part 28 Distributed interactive simulation component

Provides satisfactory support for primary DIS nodes used in distributed virtual environments


DIS PDUs for Collision, Detonate, Entity State, and Fire have their functionality bundled together in X3D EspduTransform since the physics of motion are closely coupled together. This allows for coherent X3D rendering implementations.

back to Table of Contents

# History:

# DIS-Java-VRML


# History: DIS-Java-VRML

Shared state via distributed simulation has always been a goal of X3D practitioners

The core design and implementation efforts for X3D DIS networking were first performed by DIS-Java-VRML working group

- Virtual Reality Modeling Language (VRML97) standard is direct predecessor to X3D

This design has been carefully evolved over time to match practical experience gained by producing ever-larger X3D scenes


# DIS-Java-VRML home page

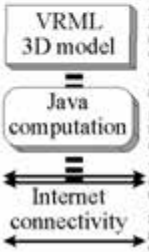
http://faculty.nps.edu/brutzman/vrtp/dis-java-vrml

**Distributed Interactive Simulation**
**DIS-Java-VRML Working Group**


**Contents**



**Project Overview**




The area of interest of this working group is the nexus of DIS, Java and VRML. The IEEE Distributed Interactive Simulation (DIS) Protocol is used to communicate state information (such as position, orientation, velocities and acceleration) among multiple entities participating in a shared network environment. Java is a portable networked programming language that can interoperate on any computer that includes a Web browser. The Virtual Reality Modeling Language (VRML) enables platform-independent interactive three-dimensional (3D) graphics across the Internet, and can be used to compose sophisticated 3D virtual environments.

The DIS-Java-VRML Working Group is developing a free software library, written in Java and interoperable with both DIS and VRML. There are a number of people contributing to the public-domain code archive. This project began in early 1997 and remains active. This software is protected under the terms of the GNU General Public License.

In the past, virtual world applications using DIS were deployed on expensive hardware and with custom software implementations. NPSNET was the first multicast DIS example of a large-scale virtual world. Like many such systems, NPSNET is written in C++ and uses the *Performer* application programming interface (API) libraries on Silicon Graphics Inc. (SGI) graphics workstations, in order to create a sophisticated and capable distributed system.

The International Standards Organization (ISO)-approved VRML 97 specification includes the rules for incorporating Java inside a VRML scene. This presents interesting possibilities for implementing virtual worlds on both high-cost and low-cost hardware in a portable way. Rather than be constrained to specific hardware, a DIS-compatible virtual world might potentially be viewed on any computer with a network connection and a web browser. The construction of large physics-based virtual worlds can now become inexpensive, pervasive and scalable.

DIS, Java and VRML can provide all of the pertinent capabilities needed to implement large-scale virtual environments (LSVEs). DIS is essentially a behavior protocol based on physics-based (i.e. "real world") many-to-many interactions. Java is the programming language used to implement the DIS protocol, perform math calculations, communicate with the network and communicate with the VRML scene. VRML 3D graphics are used to model and render both local and remote entities in shared virtual worlds.

# DIS-Java-VRML codebase

## Availability

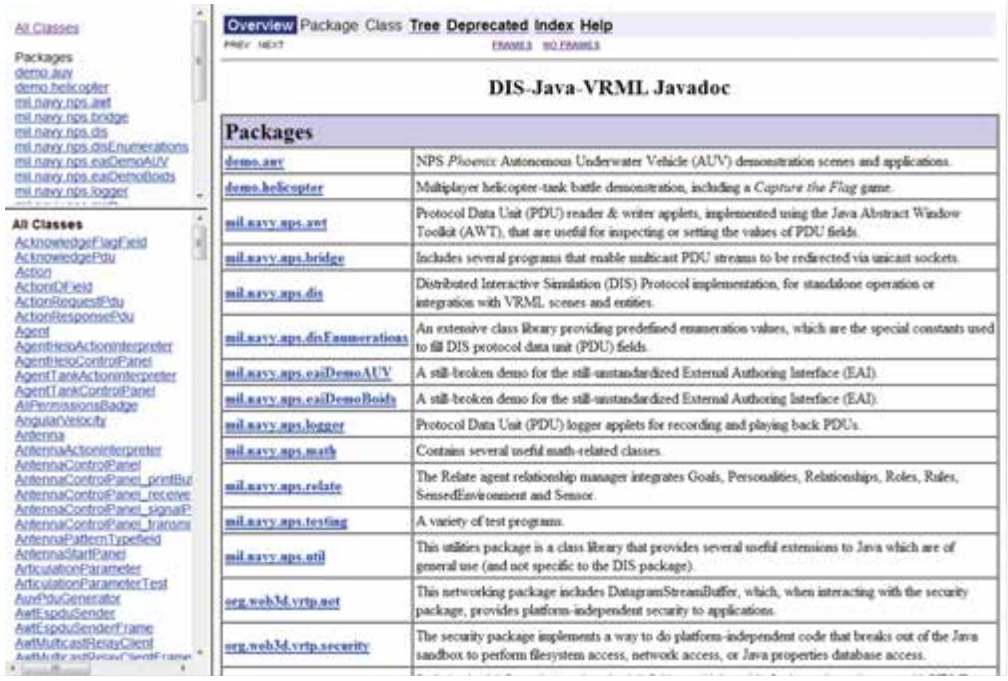
- dis-java-vrml.tar.gz or dis-java-vrml.zip
- Last build 2003

## Provides perhaps-useful example code, remains well documented

**Software Reference**

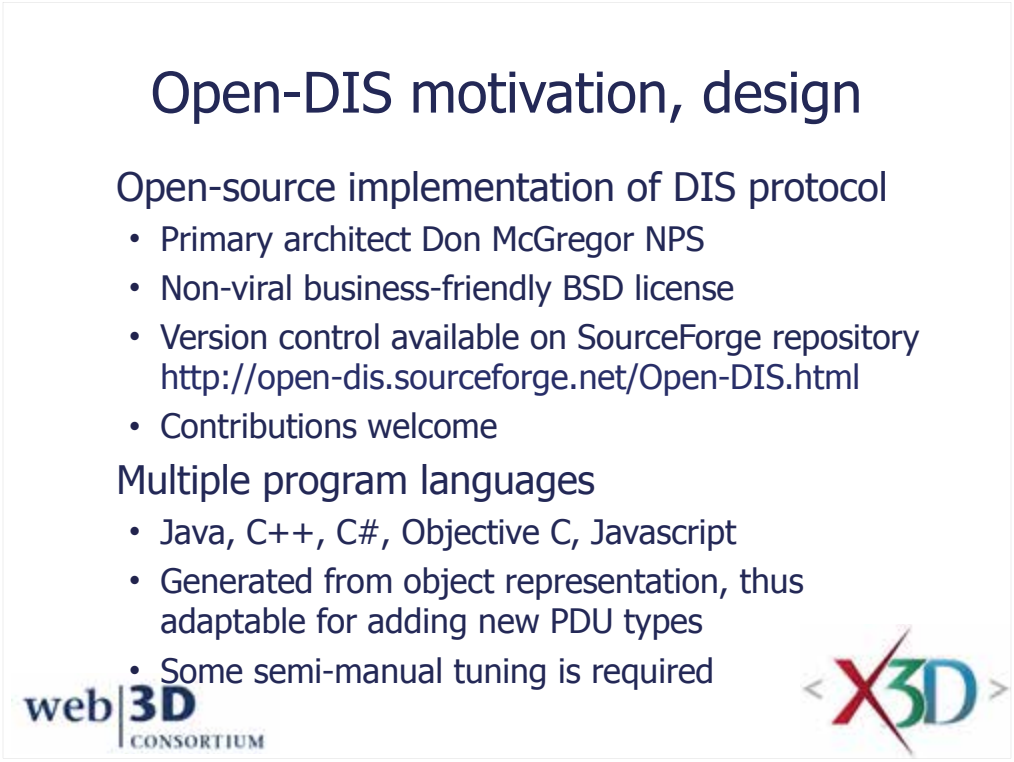
A. Software Download and Installation
B. Javadoc Documentation
C. DIS Data Dictionary
D. auv (Autonomous Underwater Vehicle) Package
E. awt (Abstract Window Toolkit) Package
F. bridge Package
G. dis Package
H. disEnumerations Package
I. eaiDemoAUV & eaiDemoBoids Packages
J. helicopter Package
K. logger Package
L. math Package
M. relate Agent Architecture Package
N. testing Package
O. util (utilities) Package
P. org.web3d.vrtp.net (network) Package
Q. org.web3d.vrtp.security Package

# DIS-Java-VRML: Javadoc



back to Table of Contents

# Current library: Open-DIS


Don McGregor writes

In Open-DIS I had an existing binary protocol specified in the IEEE document, but no machine-readable format. I wound up describing it in terms of XML. The reason I did that was that it could carry a bit more semantic knowledge of the protocol. For example, some fields in DIS describe the length of a variable-length list elsewhere in the PDU. It's not just a 4-byte integer, it's a 4-byte integer that's describing something about another field in the PDU. I could exploit that information when generating the source code for the various languages. Also, getting the ASN.1 to match the existing binary encoding would have been somewhat difficult. I suspect the ASN.1 implementations would have had to have been tweaked some, which kind of defeats the purpose--you're giving up using an off the shelf standard implementation. If SISO had specified the protocol in therms of ASN.1 to begin with that wouldn't have been a problem.

Anyway, once I had the abstract description of the protocol in XML, I could generate a language implementation of that protocol by writing about 1-2K lines of code to translate the XML into an implementation. The implementation has, for each PDU,

* Getters and setters for attributes

* methods for marshalling and unmarshalling the data

* A few other utility methods

So far the XMLPG code generates Java, C#, C++, and Objective-C.

It's imperfect--there are some problems when you start mixing and matching units, such as bytes vs. bits, or doing things that are not on byte boundaries. But it gets me about 80% of the way there, and the idea is to modify the code after the generator creates most of it.

# Open-DIS on Mac

Objective C version of Open-DIS able to run on iPhone, iPad

Screen snapshot shows PDU tracks superimposed on Google Maps using iPhone Simulator

web3D CONSORTIUM

# Open-DIS repository

http://open-dis.sourceforge.net/Open-DIS.html

# Open-DIS Javadoc

http://open-dis.sourceforge.net/javadoc/open-dis/docs/index.html


http://open-dis.sourceforge.net/javadoc/open-dis/docs/index.html

# Open-DIS Enumerations Javadoc

http://open-dis.sourceforge.net/javadoc/disenum/docs


http://open-dis.sourceforge.net/javadoc/disenum/docs

back to Table of Contents

# Concepts: DIS for X3D

Common fields for X3D nodes


# Double precision requirements

Geospatial latitude, longitude position values require double precision accuracy

- Otherwise single-precision roundoff jitter equates to 3-10m of positional error

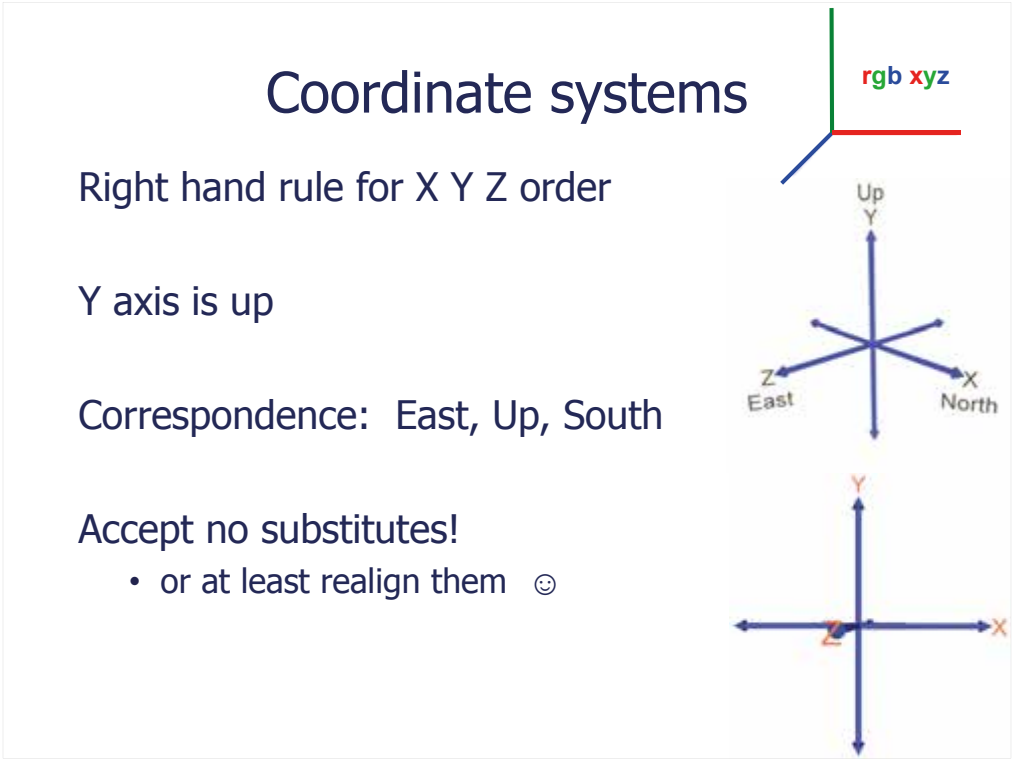
Graphics cards only support single precision

- Single precision 32 bit, double precision 64 bit

X3D Geospatial component reconciles this mismatch correctly and efficiently

Open-DIS uses double-precision satisfactorily

- However not yet integrated properly into X3D
- Use X-Y-Z local coordinate system instead




See Figures 3.1 and 3.1, page 68, *X3D for Web Authors*

There are a total of eight different Euler angle systems, each with different relative orientations for the X, Y and Z axes.

Half of these follow a left-hand rule, rather than a right-hand rule. Occasionally a graphics book comes out that presents mathematical equations using a left-hand rule. Immediately throw such books in the fire so that further pain and suffering is prevented!

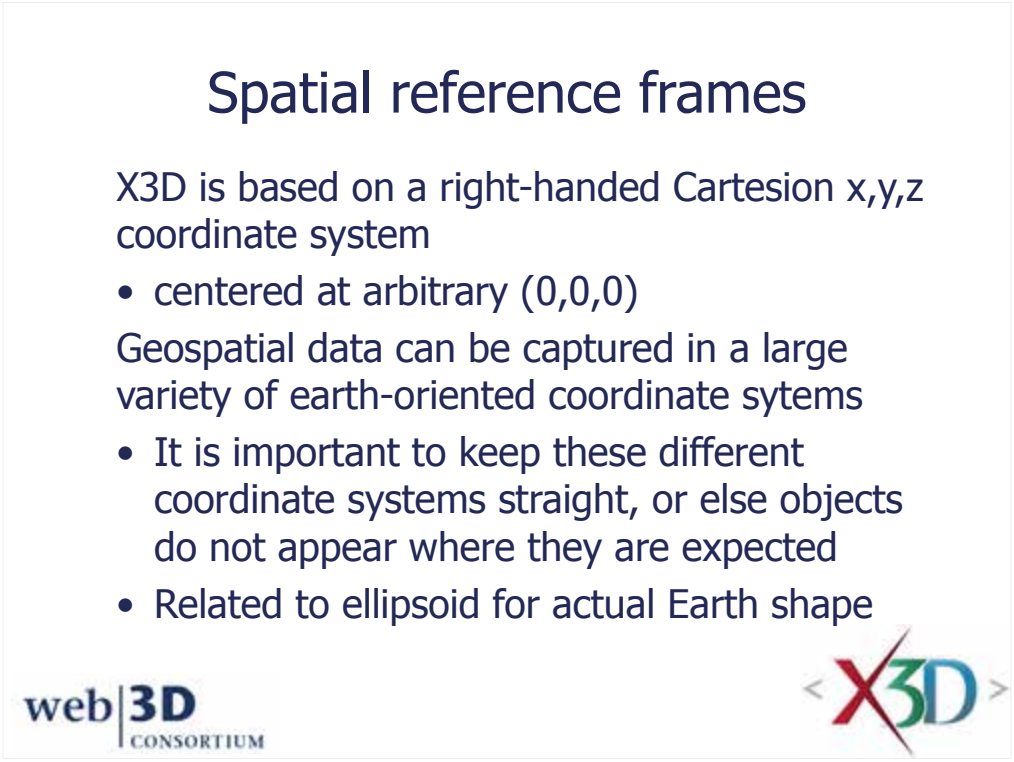
The second and third displayed examples are

http://www.x3dbook.com/examples/X3dForWebAuthors/Chapter03-Grouping/CoordinateAxesNSEW.x3d

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter03-Grouping/CoordinateAxes.x3d

http://X3dGraphics.com/examples/X3dForWebAuthors/Chapter03-Grouping/CoordinateAxesInlineExample.x3d

Ordinarily we ignore correspondences with geographic North, South, East and West, since regular X3D coordinates are single-precision floating point, while the Geospatial nodes use double-precision floating-point values in order to capture latitude and longitude coordinates with sufficient accuracy.




See the references by Raph Toms and Paul Birkel for further details.

## Common fields: *isActive*, *timestamp*

- *isActive* indicates if node has received a DIS packet (*isActive* true) or not (*isActive* false)
- *timestamp* field provides the time (SFTime) at which the DIS message arrived, referenced to local system time.


## Common field: *networkMode*

*networkMode* has 3 distinct possible values:

- `networkReader`: sender/owner writing updates,
- `networkWriter`: receiver/ghost reading updates,
- `standAlone`: independently from the network

Thus *networkMode* provides a way for an X3D scene to indicate whether a particular node is the owner or listener for a given entity.

- *networkMode*=`'standAlone'` allows passive operation without opening network communications

web3D CONSORTIUM


Keeping networking turned off by default may be a useful security option or configuration presence for some scenes.

## Common fields: network *address*, *port*

Typical transport mode is multicast, with specific *address* chosen by exercise

- Restricted to range 224.0.0.0 thru 239.255.255.255

Unicast address (or `localhost` value) allowed

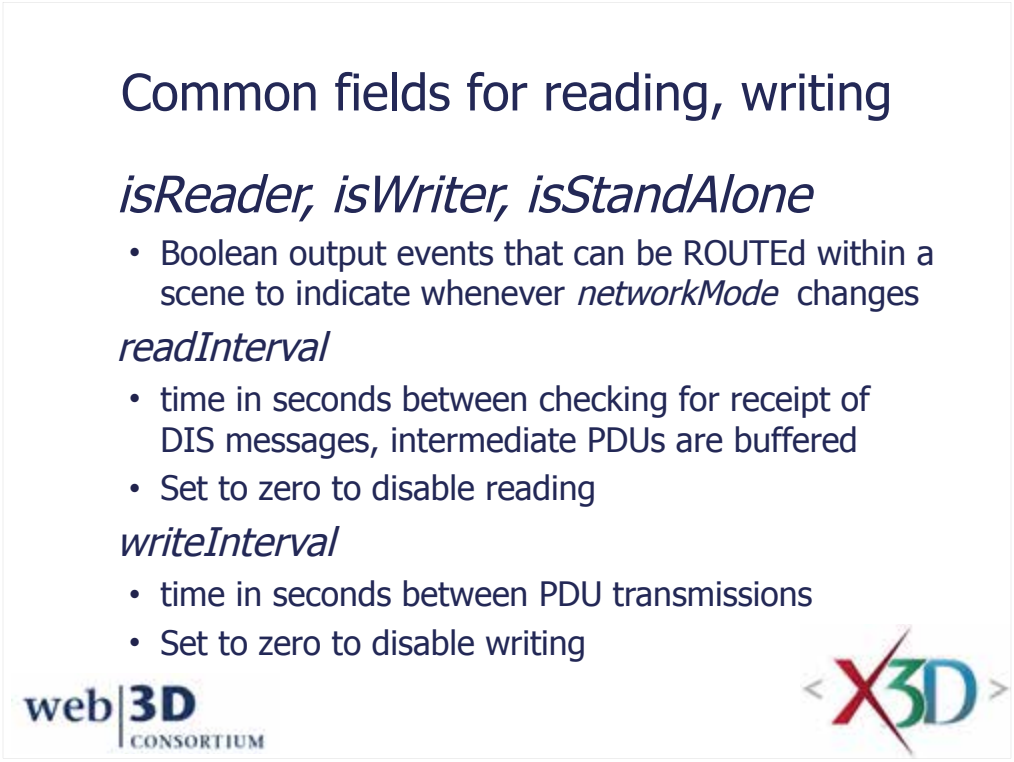
- opens direct point-to-point socket

Similarly choose *port* value

- Restricted to range 1 thru 65535

Other networking fields:

- *multicastRelayHost, multicastRelayPort, rtpHeaderExpected, rtpHeaderHeard*

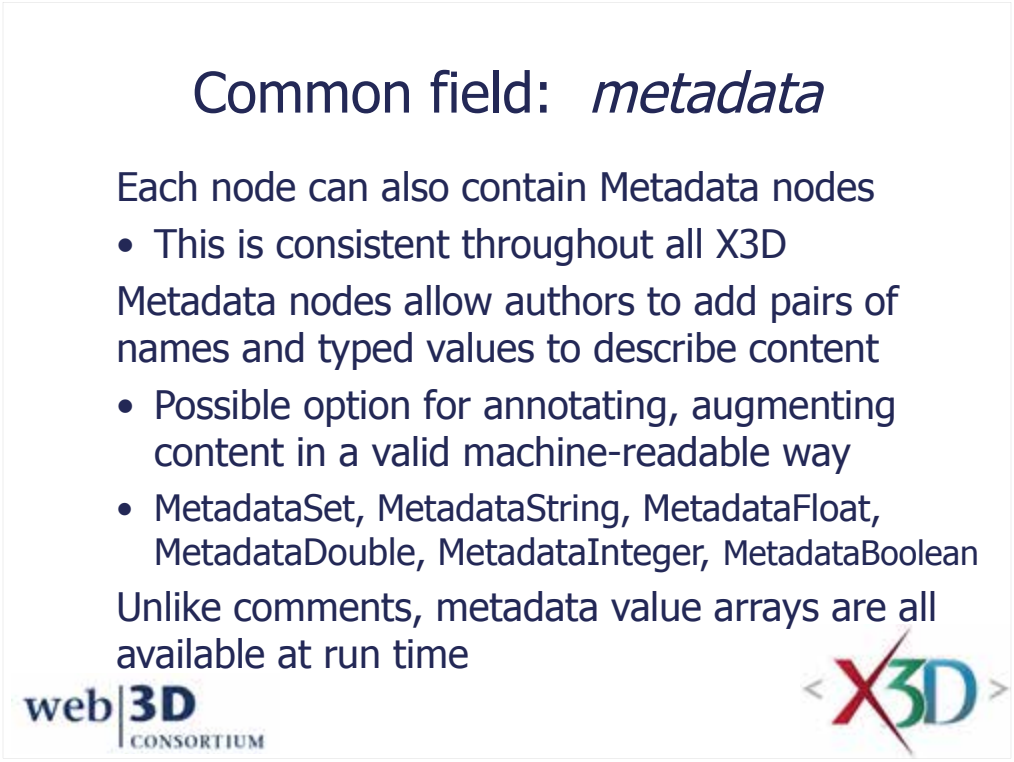

*isReader, isWriter, isStandAlone* provide run-time events of activity corresponding ton *networkMode*="networkReader", *networkMode*="networkWriter", or *networkMode*="standAlone"

# Common fields for entity identification

Goal: build unique identifiers for each entity

- *siteID* identifies a given LAN or organization
- *applicationID* is unique for a given simulation
- *entityID* is unique to a given entity
- Background: *siteID* and *applicationID* fields are used to create DIS PDU Simulation Address record

Thus entity identification depends on a triplet of values, unique across all entities in that application and in the particular exercise.




See X3D Abstract Specification Core Componentfor Metadata node definitions

X3D for Web Authors textbook includes a free online Metadata chapter

back to Table of Contents

# X3D Nodes and Examples


# EspduTransform: shared state

Exposes DIS ESPDU values as an extended form of Transform node, matching semantics between DIS and X3D scene graph

- Also integrates Collision, Fire and Detonate PDUs since they are fundamentally integral to entity motion
- Distributed shared state among players
- Primary workhorse node of interest for DIS

TODO: add, expose example scenes to show various ROUTE connections for animation

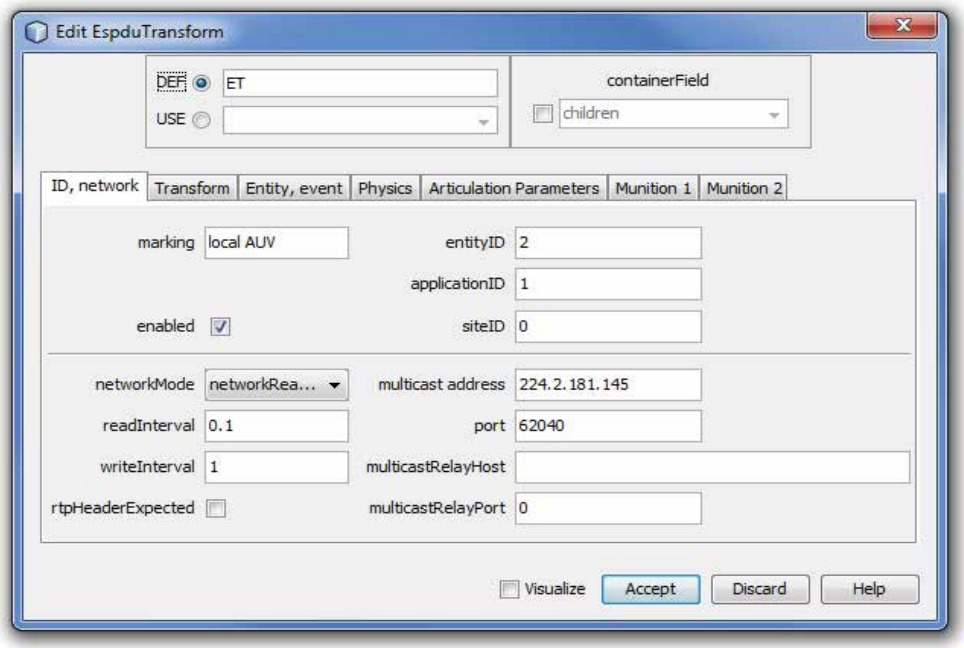

# EspduTransform: shared state

Exposes DIS ESPDU values as an extended form of Transform node, matching semantics between DIS and X3D scene graph

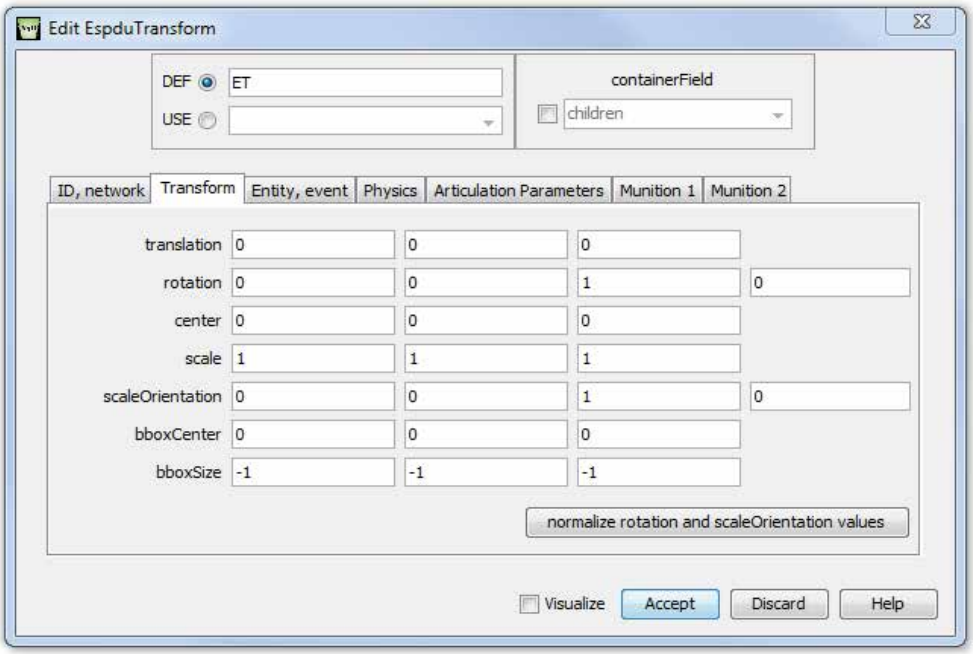
- Also integrates Collision, Fire and Detonate PDUs since they are fundamentally integral to entity motion
- Distributed shared state among players
- Primary workhorse node of interest for DIS

TODO: add, expose example scenes to show various ROUTE connections for animation

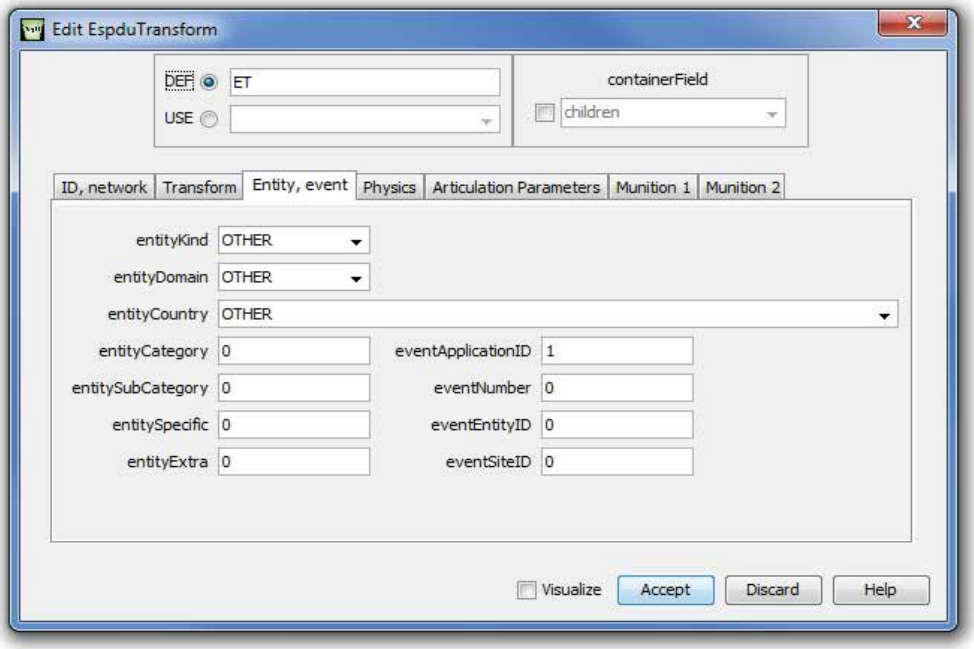

# EspduTransform: ID, network pane



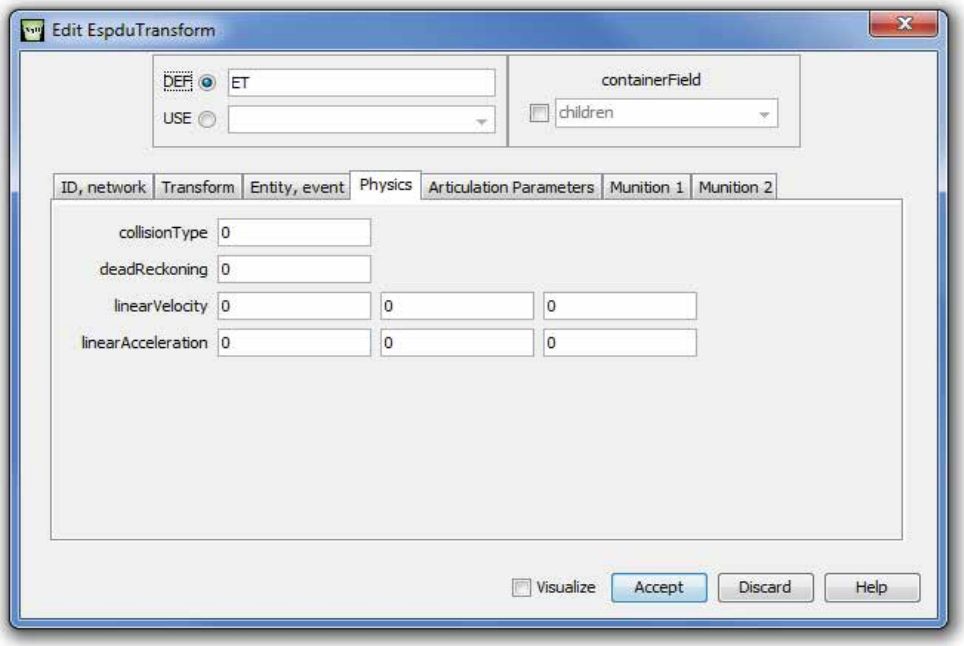
# EspduTransform: transform pane



# EspduTransform: entity, event pane

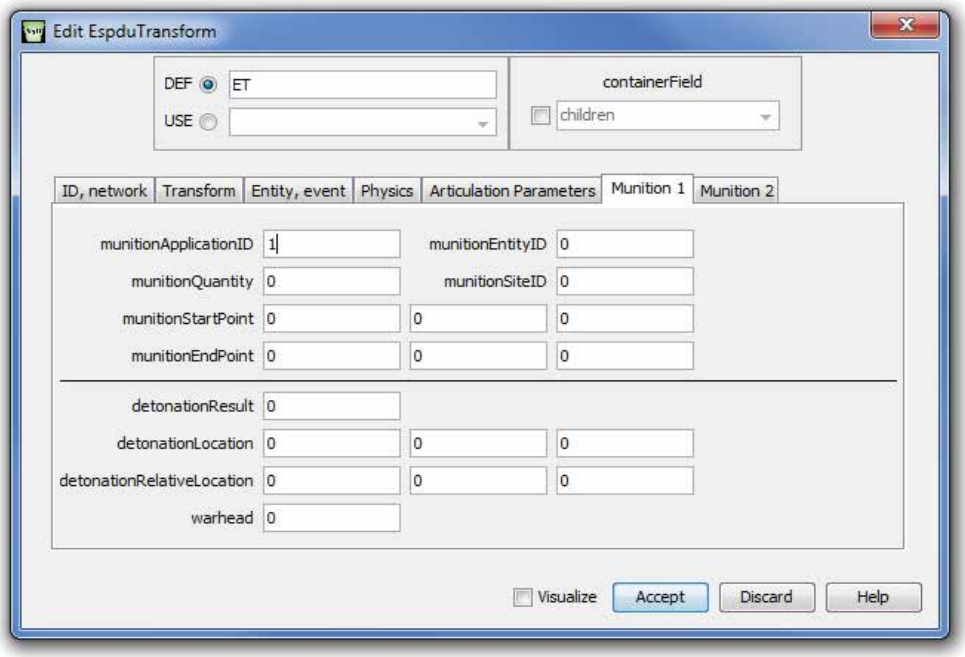


# EspduTransform: physics pane

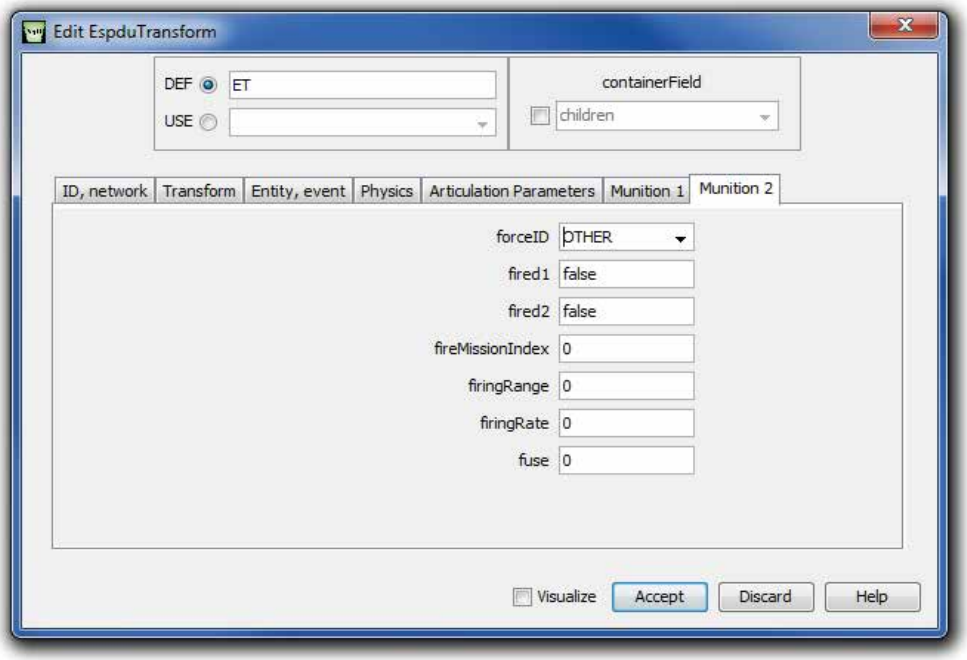


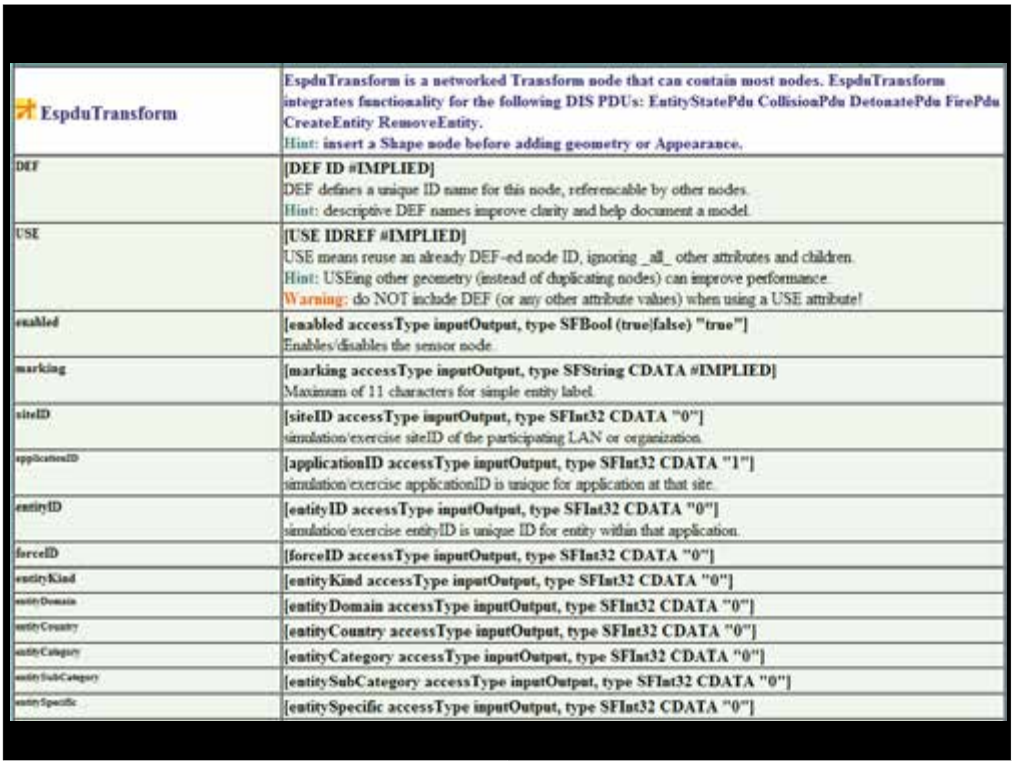
# EspduTransform: articulation parameters

# EspduTransform: munition1 pane



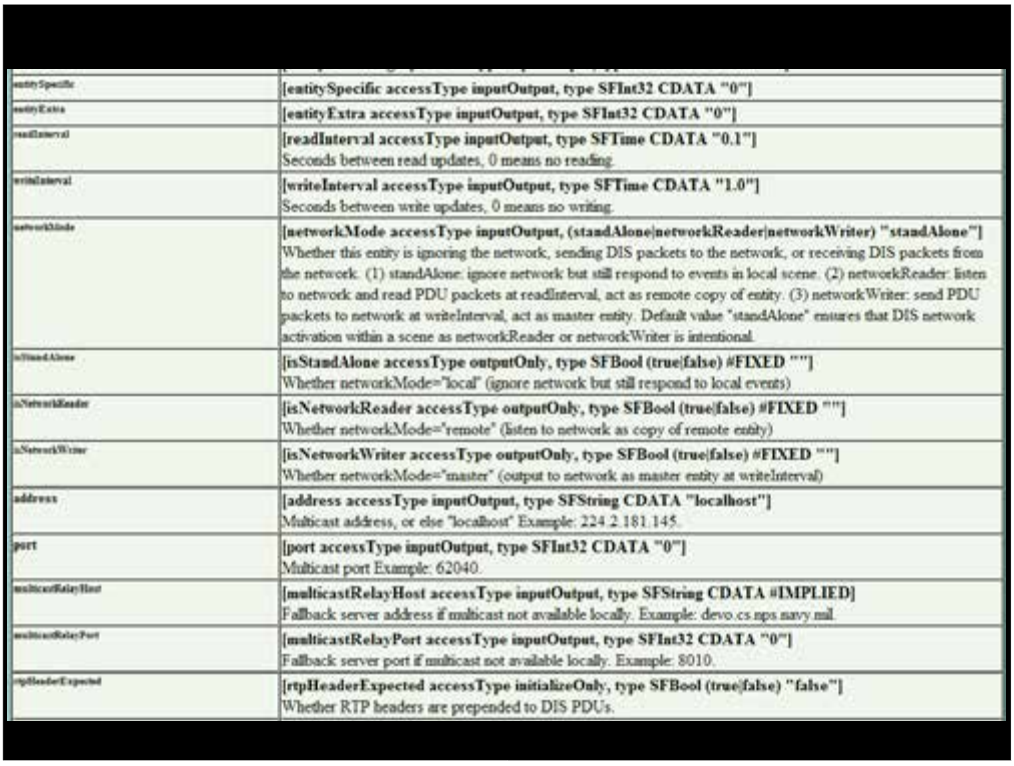
# EspduTransform: munition2 pane





| EspduTransform | EspduTransform is a networked Transform node that can contain most nodes. EspduTransform integrates functionality for the following DIS PDUs: EntityStatePdu CollisionPdu DetonatePdu FirePdu CreateEntity RemoveEntity.<br>Hint: insert a Shape node before adding geometry or Appearance. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| marking | [marking accessType inputOutput, type SFString CDATA #IMPLIED]<br>Maximum of 11 characters for simple entity label. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>simulation/exercise siteID of the participating LAN or organization. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>simulation/exercise applicationID is unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>simulation/exercise entityID is unique ID for entity within that application. |
| forceID | [forceID accessType inputOutput, type SFInt32 CDATA "0"] |
| entityKind | [entityKind accessType inputOutput, type SFInt32 CDATA "0"] |
| entityDomain | [entityDomain accessType inputOutput, type SFInt32 CDATA "0"] |
| entityCountry | [entityCountry accessType inputOutput, type SFInt32 CDATA "0"] |
| entityCategory | [entityCategory accessType inputOutput, type SFInt32 CDATA "0"] |
| entitySubCategory | [entitySubCategory accessType inputOutput, type SFInt32 CDATA "0"] |
| entitySpecific | [entitySpecific accessType inputOutput, type SFInt32 CDATA "0"] |

http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform



| | |
|---|---|
| entitySpecific | **[entitySpecific accessType inputOutput, type SFInt32 CDATA "0"]** |
| entityExtra | **[entityExtra accessType inputOutput, type SFInt32 CDATA "0"]** |
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" Example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port Example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally. Example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally. Example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |

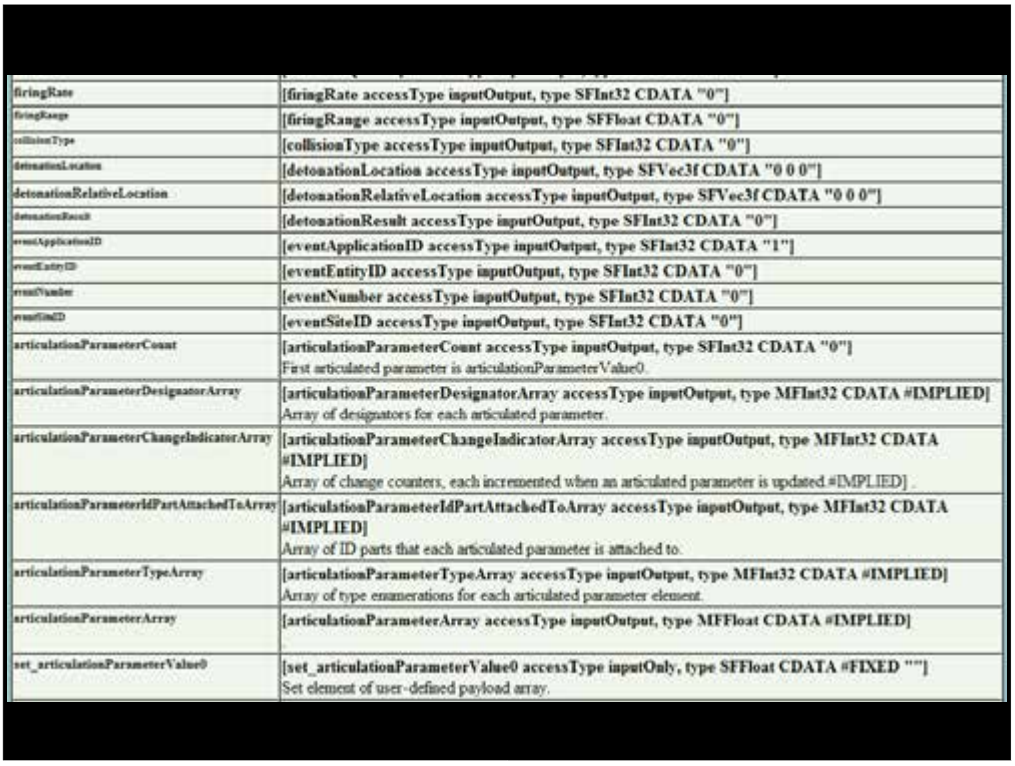
http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform

| | |
|---|---|
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |
| isActive | **[isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Have we received a network update recently? |
| timestamp | **[timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]**<br>DIS timestamp in VRML time units from latest update. |
| translation | **[translation accessType inputOutput, type SFVec3f CDATA "0 0 0"]**<br>Position of children relative to local coordinate system, usually read from (or written to) remote, networked EspduTransform nodes. |
| rotation | **[rotation accessType inputOutput, type SFRotation CDATA "0 0 1 0"]**<br>Orientation of children relative to local coordinate system, usually read from (or written to) remote, networked EspduTransform nodes. |
| center | **[center accessType inputOutput, type SFVec3f CDATA "0 0 0"]**<br>Translation offset from origin of local coordinate system. |
| scale | **[scale accessType inputOutput, type SFVec3f CDATA "1 1 1"]**<br>Non-uniform x-y-z scale of child coordinate system, adjusted by center and scaleOrientation. |
| scaleOrientation | **[scaleOrientation accessType inputOutput, type SFRotation CDATA "0 0 1 0"]**<br>Preliminary rotation of coordinate system before scaling (to allow scaling around arbitrary orientations). |
| bboxCenter | **[bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]**<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | **[bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]**<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| linearVelocity | **[linearVelocity accessType inputOutput, type SFVec3f CDATA "0 0 0"]** |
| linearAcceleration | **[linearAcceleration accessType inputOutput, type SFVec3f CDATA "0 0 0"]** |
| deadReckoning | **[deadReckoning accessType inputOutput, type SFInt32 CDATA "0"]**<br>[0,65535] Dead reckoning algorithm being used to project position/orientation with velocities/accelerations. |
| isCollided | **[isCollided accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Has a matching CollisionPDU reported a collision? |

http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform

| | |
|---|---|
| collideTime | **[collideTime accessType outputOnly, type SFTime CDATA #FIXED ""]**<br>When were we collided with? |
| isDetonated | **[isDetonated accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Has a matching DetonationPDU reported a detonation? |
| detonateTime | **[detonateTime accessType outputOnly, type SFTime CDATA #FIXED ""]**<br>When were we detonated? |
| fired1 | **[fired1 accessType inputOutput, type SFBool (true\|false) "false"]**<br>Has the primary weapon (Fire PDU) been fired? |
| fired2 | **[fired2 accessType inputOutput, type SFBool (true\|false) "false"]**<br>Has the secondary weapon (Fire PDU) been fired? |
| firedTime | **[firedTime accessType outputOnly, type SFTime CDATA #FIXED ""]**<br>When did we shoot a weapon (Fire PDU)? |
| munitionStartPoint | **[munitionStartPoint accessType inputOutput, type SFVec3f CDATA "0 0 0"]**<br>eventout, uses exercise coordinates. |
| munitionEndPoint | **[munitionEndPoint accessType inputOutput, type SFVec3f CDATA "0 0 0"]**<br>eventout, uses exercise coordinates. |
| munitionSiteID | **[munitionSiteID accessType inputOutput, type SFInt32 CDATA "0"]**<br>Munition siteID. |
| munitionApplicationID | **[munitionApplicationID accessType inputOutput, type SFInt32 CDATA "1"]**<br>Munition applicationID, unique for application at that site. |
| munitionEntityID | **[munitionEntityID accessType inputOutput, type SFInt32 CDATA "0"]**<br>Munition entityID is unique ID for entity firing munition within that application. |
| fireMissionIndex | **[fireMissionIndex accessType inputOutput, type SFInt32 CDATA #FIXED ""]** |
| warhead | **[warhead accessType inputOutput, type SFInt32 CDATA "0"]** |
| fuse | **[fuse accessType inputOutput, type SFInt32 CDATA "0"]** |
| munitionQuantity | **[munitionQuantity accessType inputOutput, type SFInt32 CDATA "0"]** |

http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform

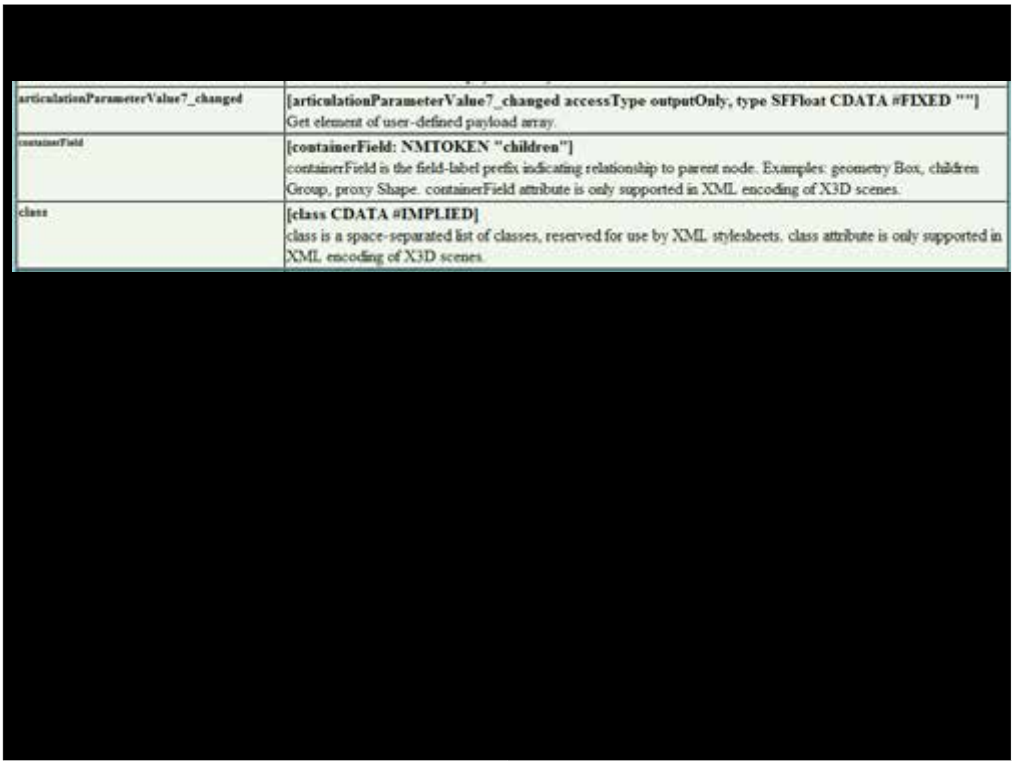


| | |
|---|---|
| firingRate | [firingRate accessType inputOutput, type SFInt32 CDATA "0"] |
| firingRange | [firingRange accessType inputOutput, type SFFloat CDATA "0"] |
| collisionType | [collisionType accessType inputOutput, type SFInt32 CDATA "0"] |
| detonationLocation | [detonationLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| detonationRelativeLocation | [detonationRelativeLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| detonationResult | [detonationResult accessType inputOutput, type SFInt32 CDATA "0"] |
| eventApplicationID | [eventApplicationID accessType inputOutput, type SFInt32 CDATA "1"] |
| eventEntityID | [eventEntityID accessType inputOutput, type SFInt32 CDATA "0"] |
| eventNumber | [eventNumber accessType inputOutput, type SFInt32 CDATA "0"] |
| eventSiteID | [eventSiteID accessType inputOutput, type SFInt32 CDATA "0"] |
| articulationParameterCount | [articulationParameterCount accessType inputOutput, type SFInt32 CDATA "0"]<br>First articulated parameter is articulationParameterValue0. |
| articulationParameterDesignatorArray | [articulationParameterDesignatorArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of designators for each articulated parameter. |
| articulationParameterChangeIndicatorArray | [articulationParameterChangeIndicatorArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of change counters, each incremented when an articulated parameter is updated.#IMPLIED] . |
| articulationParameterIdPartAttachedToArray | [articulationParameterIdPartAttachedToArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of ID parts that each articulated parameter is attached to. |
| articulationParameterTypeArray | [articulationParameterTypeArray accessType inputOutput, type MFInt32 CDATA #IMPLIED]<br>Array of type enumerations for each articulated parameter element. |
| articulationParameterArray | [articulationParameterArray accessType inputOutput, type MFFloat CDATA #IMPLIED] |
| set_articulationParameterValue0 | [set_articulationParameterValue0 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |

http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform

| | |
|---|---|
| set_articulationParameterValue1 | [set_articulationParameterValue1 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue2 | [set_articulationParameterValue2 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue3 | [set_articulationParameterValue3 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue4 | [set_articulationParameterValue4 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue5 | [set_articulationParameterValue5 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue6 | [set_articulationParameterValue6 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| set_articulationParameterValue7 | [set_articulationParameterValue7 accessType inputOnly, type SFFloat CDATA #FIXED ""]<br>Set element of user-defined payload array. |
| articulationParameterValue0_changed | [articulationParameterValue0_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue1_changed | [articulationParameterValue1_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue2_changed | [articulationParameterValue2_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue3_changed | [articulationParameterValue3_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue4_changed | [articulationParameterValue4_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue5_changed | [articulationParameterValue5_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |
| articulationParameterValue6_changed | [articulationParameterValue6_changed accessType outputOnly, type SFFloat CDATA #FIXED ""]<br>Get element of user-defined payload array. |

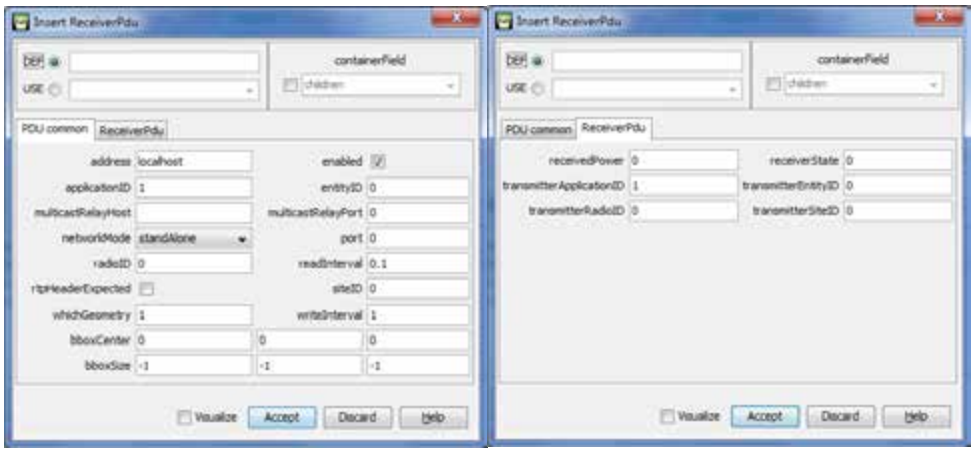
http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform




http://www.web3d.org/x3d/content/X3dTooltips.html#EspduTransform

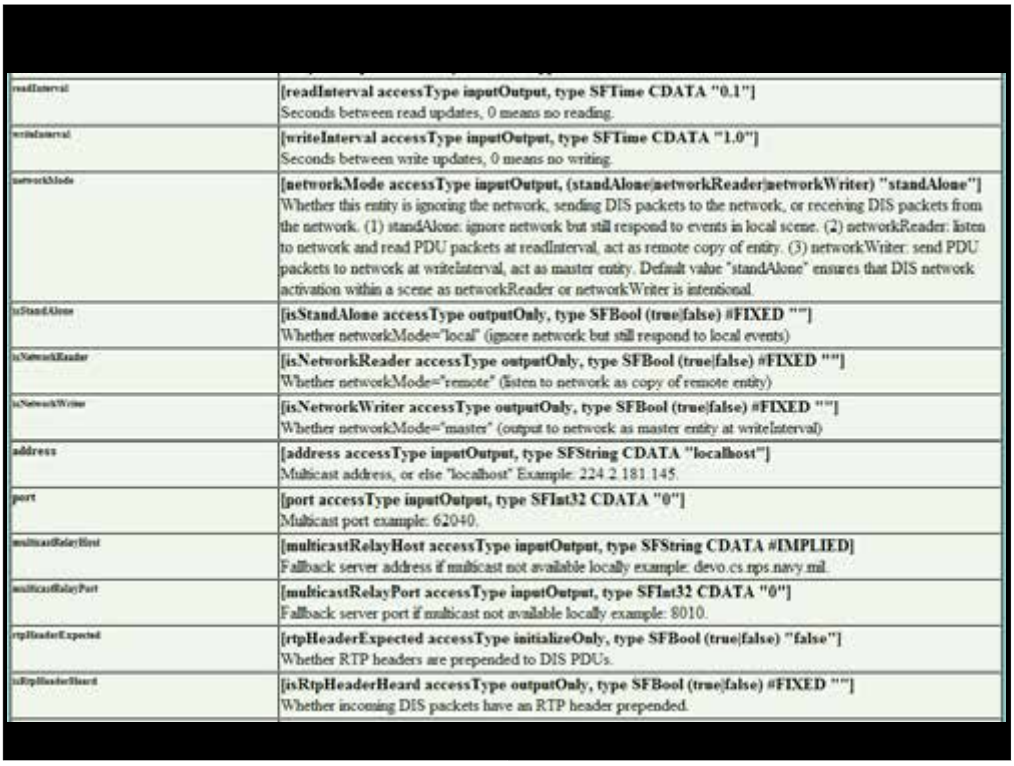
# ReceiverPdu

- ReceiverPdu transmits state of radio frequency (RF) receivers modeled in the simulation.
- Exposes fields for ReceiverPdu node



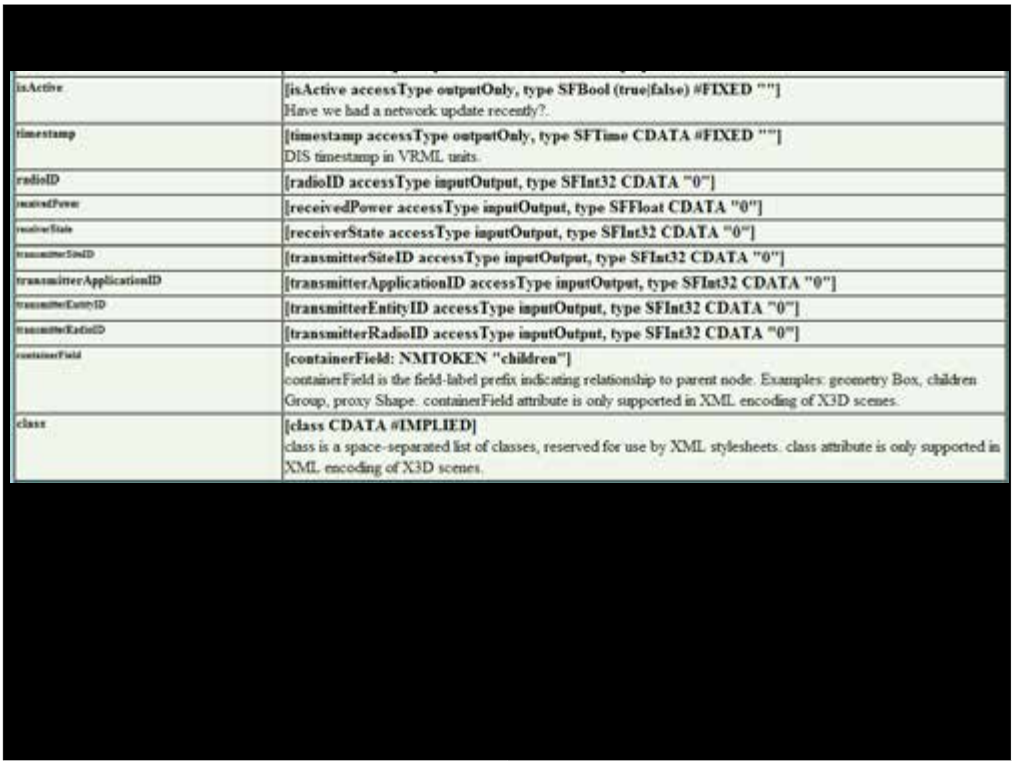
| ReceiverPdu | ReceiverPdu is a networked PDU information node. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| whichGeometry | [whichGeometry accessType inputOutput, type SFInt32 CDATA "1"]<br>Select geometry to render: -1 for no geometry, 0 for text trace, 1 for default geometry. |
| bboxCenter | [bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | [bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID unique ID for entity within that application. |

http://www.web3d.org/x3d/content/X3dTooltips.html#ReceiverPdu



| | |
|---|---|
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" Example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |

http://www.web3d.org/x3d/content/X3dTooltips.html#ReceiverPdu

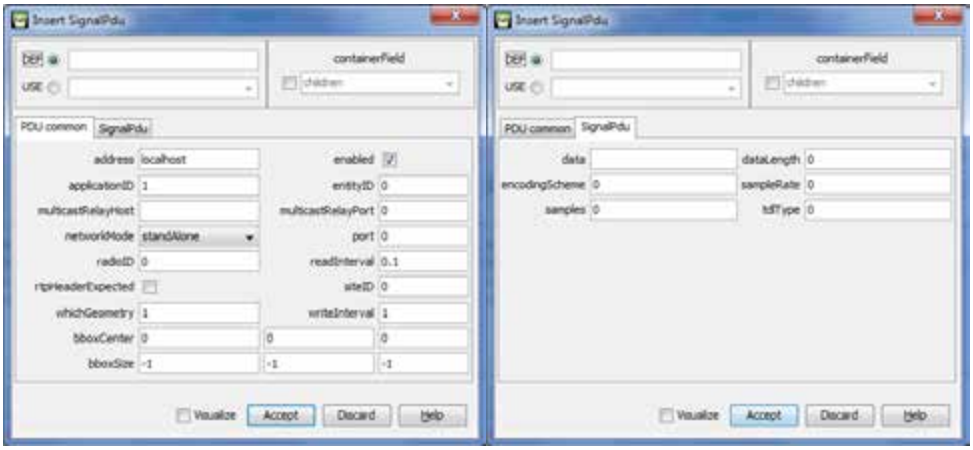


| | |
|---|---|
| isActive | [isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]<br>Have we had a network update recently?. |
| timestamp | [timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]<br>DIS timestamp in VRML units. |
| radioID | [radioID accessType inputOutput, type SFInt32 CDATA "0"] |
| receivedPower | [receivedPower accessType inputOutput, type SFFloat CDATA "0"] |
| receiverState | [receiverState accessType inputOutput, type SFInt32 CDATA "0"] |
| transmitterSiteID | [transmitterSiteID accessType inputOutput, type SFInt32 CDATA "0"] |
| transmitterApplicationID | [transmitterApplicationID accessType inputOutput, type SFInt32 CDATA "0"] |
| transmitterEntityID | [transmitterEntityID accessType inputOutput, type SFInt32 CDATA "0"] |
| transmitterRadioID | [transmitterRadioID accessType inputOutput, type SFInt32 CDATA "0"] |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

http://www.web3d.org/x3d/content/X3dTooltips.html#ReceiverPdu

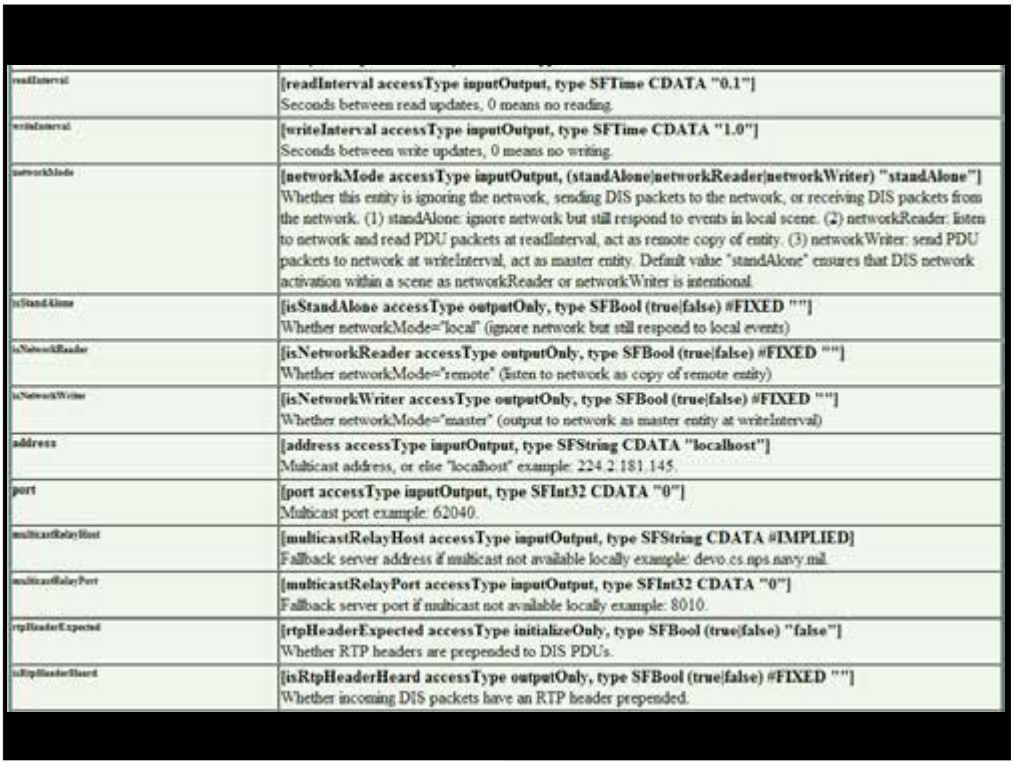
# SignalPdu

- SignalPdu relays the transmission of voice, audio or other data modeled in a simulation
- Exposes fields for SignalPdu node



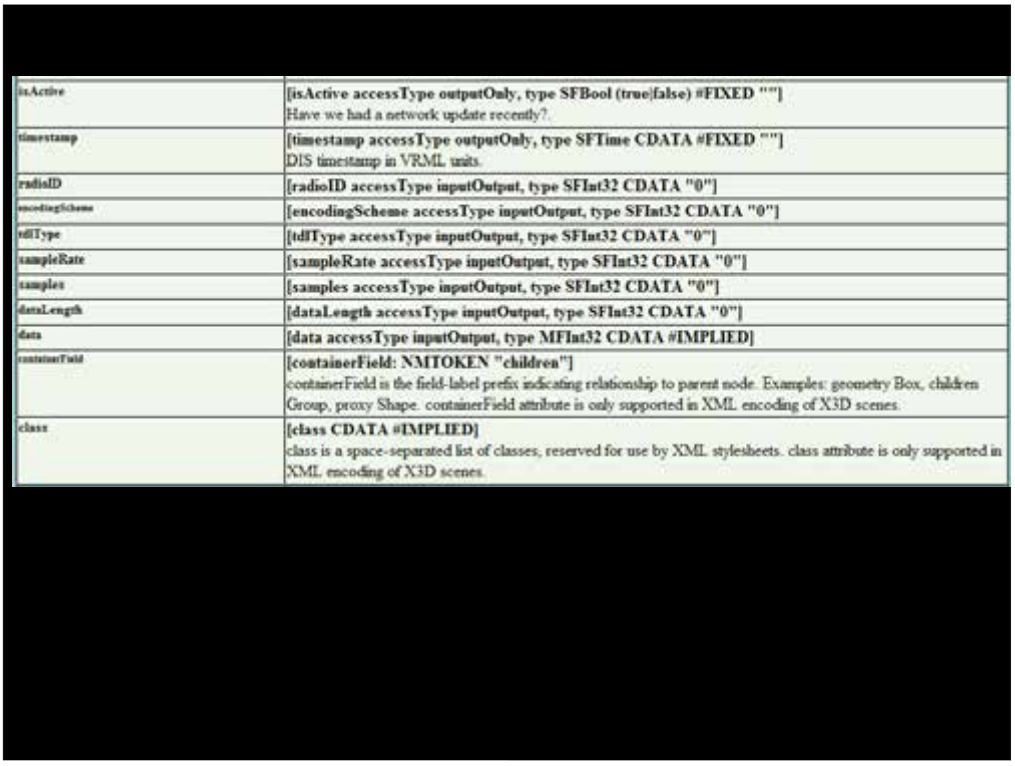
| SignalPdu | SignalPdu is a networked PDU information node. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| whichGeometry | [whichGeometry accessType inputOutput, type SFInt32 CDATA "1"]<br>Select geometry to render: -1 for no geometry, 0 for text trace, 1 for default geometry. |
| bboxCenter | [bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | [bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID unique ID for entity within that application. |

http://www.web3d.org/x3d/content/X3dTooltips.html#SignalPdu



| | |
|---|---|
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |

http://www.web3d.org/x3d/content/X3dTooltips.html#SignalPdu

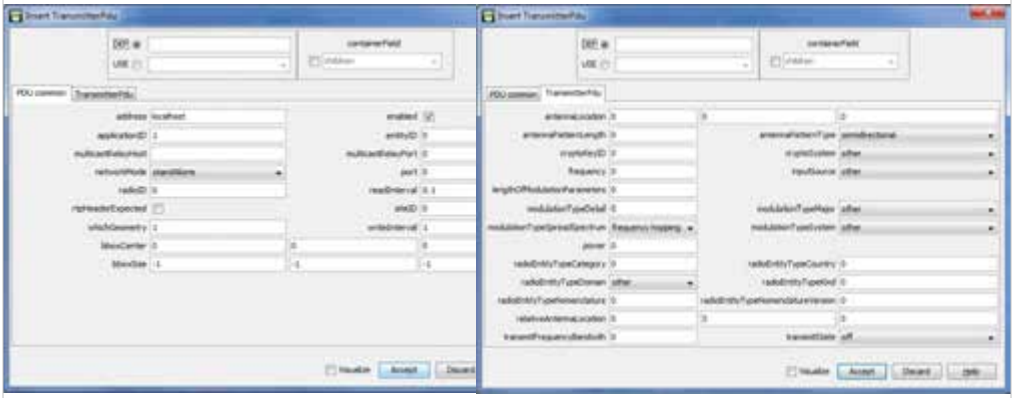


| | |
|---|---|
| isActive | [isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]<br>Have we had a network update recently? |
| timestamp | [timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]<br>DIS timestamp in VRML units. |
| radioID | [radioID accessType inputOutput, type SFInt32 CDATA "0"] |
| encodingScheme | [encodingScheme accessType inputOutput, type SFInt32 CDATA "0"] |
| tdlType | [tdlType accessType inputOutput, type SFInt32 CDATA "0"] |
| sampleRate | [sampleRate accessType inputOutput, type SFInt32 CDATA "0"] |
| samples | [samples accessType inputOutput, type SFInt32 CDATA "0"] |
| dataLength | [dataLength accessType inputOutput, type SFInt32 CDATA "0"] |
| data | [data accessType inputOutput, type MFInt32 CDATA #IMPLIED] |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

http://www.web3d.org/x3d/content/X3dTooltips.html#SignalPdu

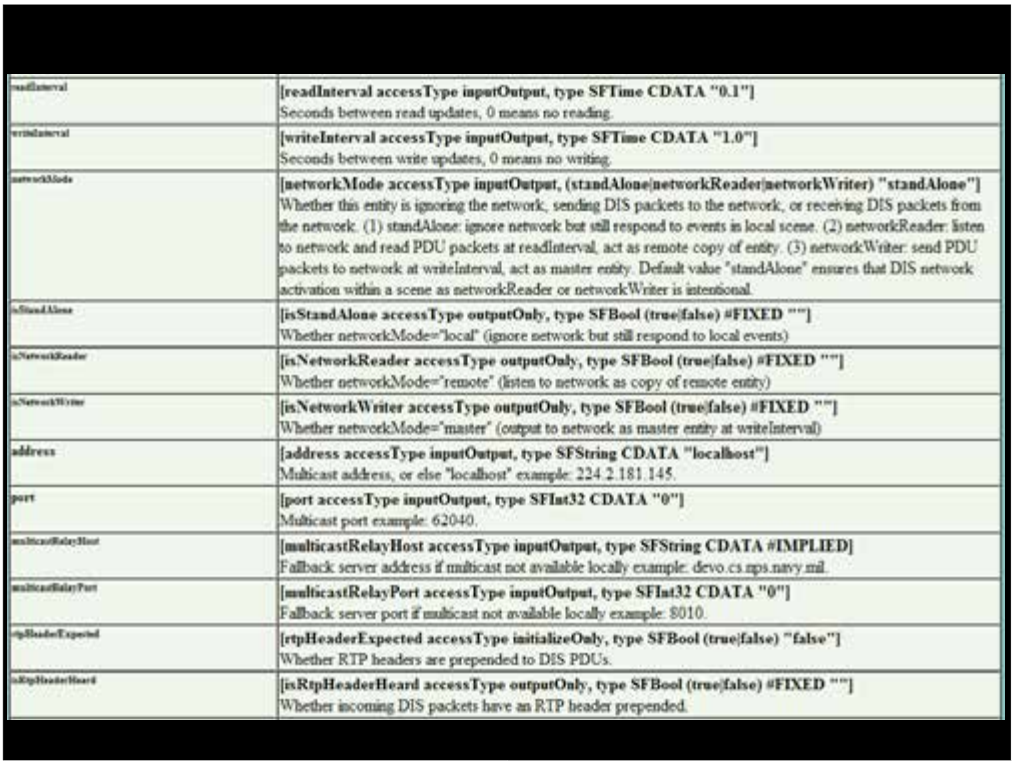
# TransmitterPdu

- TransmitterPdu provides detailed info about a radio transmitter modeled in a simulation.
- Exposes fields for TransmitterPdu node



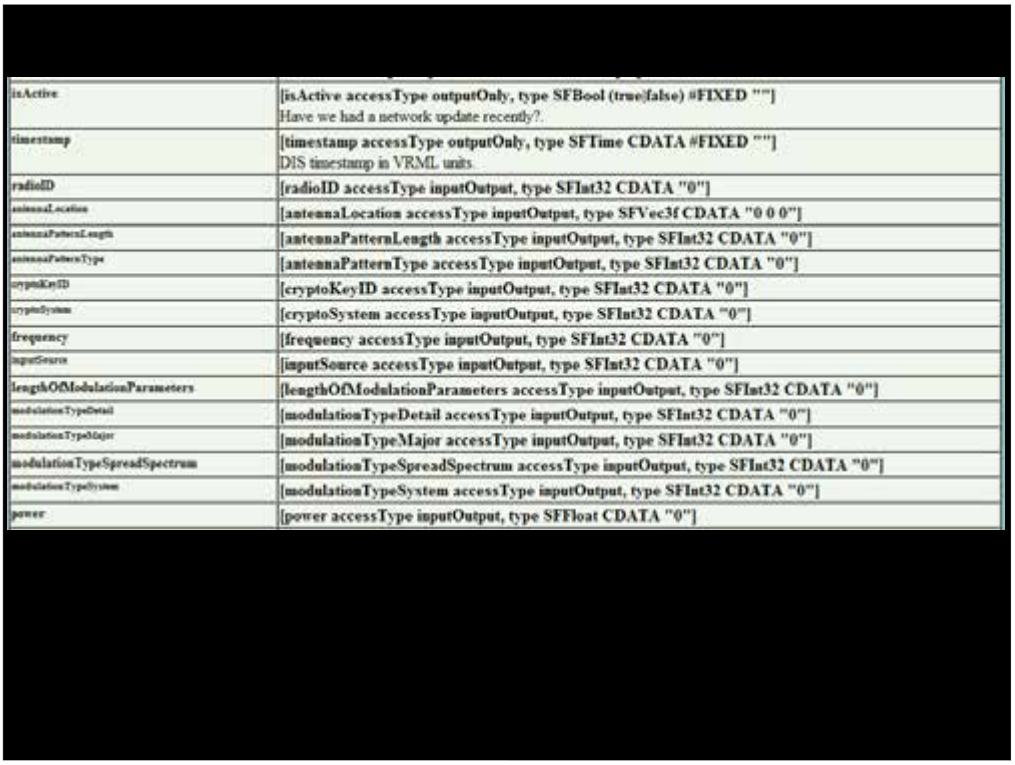
| TransmitterPdu | TransmitterPdu is a networked PDU information node. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| enabled | [enabled accessType inputOutput, type SFBool (true\|false) "true"]<br>Enables/disables the sensor node. |
| whichGeometry | [whichGeometry accessType inputOutput, type SFInt32 CDATA "1"]<br>Select geometry to render: -1 for no geometry, 0 for text trace, 1 for default geometry. |
| bboxCenter | [bboxCenter accessType initializeOnly, type SFVec3f CDATA "0 0 0"]<br>Bounding box center: position offset from origin of local coordinate system. |
| bboxSize | [bboxSize accessType initializeOnly, type SFVec3f CDATA "-1 -1 -1"]<br>Bounding box size: automatically calculated, can be specified as an optimization or constraint. |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| entityID | [entityID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID unique ID for entity within that application. |

http://www.web3d.org/x3d/content/X3dTooltips.html#TransmitterPdu



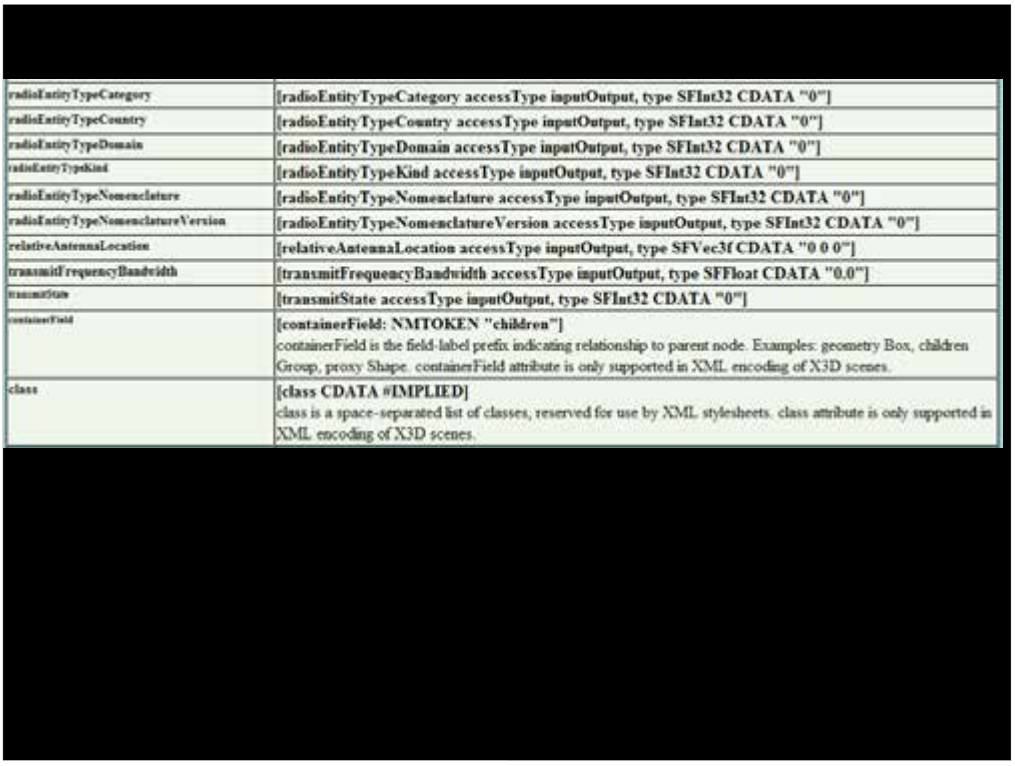
| | |
|---|---|
| readInterval | **[readInterval accessType inputOutput, type SFTime CDATA "0.1"]**<br>Seconds between read updates, 0 means no reading. |
| writeInterval | **[writeInterval accessType inputOutput, type SFTime CDATA "1.0"]**<br>Seconds between write updates, 0 means no writing. |
| networkMode | **[networkMode accessType inputOutput, (standAlone\|networkReader\|networkWriter) "standAlone"]**<br>Whether this entity is ignoring the network, sending DIS packets to the network, or receiving DIS packets from the network. (1) standAlone: ignore network but still respond to events in local scene. (2) networkReader: listen to network and read PDU packets at readInterval, act as remote copy of entity. (3) networkWriter: send PDU packets to network at writeInterval, act as master entity. Default value "standAlone" ensures that DIS network activation within a scene as networkReader or networkWriter is intentional. |
| isStandAlone | **[isStandAlone accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="local" (ignore network but still respond to local events) |
| isNetworkReader | **[isNetworkReader accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="remote" (listen to network as copy of remote entity) |
| isNetworkWriter | **[isNetworkWriter accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether networkMode="master" (output to network as master entity at writeInterval) |
| address | **[address accessType inputOutput, type SFString CDATA "localhost"]**<br>Multicast address, or else "localhost" example: 224.2.181.145. |
| port | **[port accessType inputOutput, type SFInt32 CDATA "0"]**<br>Multicast port example: 62040. |
| multicastRelayHost | **[multicastRelayHost accessType inputOutput, type SFString CDATA #IMPLIED]**<br>Fallback server address if multicast not available locally example: devo.cs.nps.navy.mil. |
| multicastRelayPort | **[multicastRelayPort accessType inputOutput, type SFInt32 CDATA "0"]**<br>Fallback server port if multicast not available locally example: 8010. |
| rtpHeaderExpected | **[rtpHeaderExpected accessType initializeOnly, type SFBool (true\|false) "false"]**<br>Whether RTP headers are prepended to DIS PDUs. |
| isRtpHeaderHeard | **[isRtpHeaderHeard accessType outputOnly, type SFBool (true\|false) #FIXED ""]**<br>Whether incoming DIS packets have an RTP header prepended. |

http://www.web3d.org/x3d/content/X3dTooltips.html#TransmitterPdu



| | |
|---|---|
| isActive | [isActive accessType outputOnly, type SFBool (true\|false) #FIXED ""]<br>Have we had a network update recently?. |
| timestamp | [timestamp accessType outputOnly, type SFTime CDATA #FIXED ""]<br>DIS timestamp in VRML units. |
| radioID | [radioID accessType inputOutput, type SFInt32 CDATA "0"] |
| antennaLocation | [antennaLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| antennaPatternLength | [antennaPatternLength accessType inputOutput, type SFInt32 CDATA "0"] |
| antennaPatternType | [antennaPatternType accessType inputOutput, type SFInt32 CDATA "0"] |
| cryptoKeyID | [cryptoKeyID accessType inputOutput, type SFInt32 CDATA "0"] |
| cryptoSystem | [cryptoSystem accessType inputOutput, type SFInt32 CDATA "0"] |
| frequency | [frequency accessType inputOutput, type SFInt32 CDATA "0"] |
| inputSource | [inputSource accessType inputOutput, type SFInt32 CDATA "0"] |
| lengthOfModulationParameters | [lengthOfModulationParameters accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeDetail | [modulationTypeDetail accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeMajor | [modulationTypeMajor accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeSpreadSpectrum | [modulationTypeSpreadSpectrum accessType inputOutput, type SFInt32 CDATA "0"] |
| modulationTypeSystem | [modulationTypeSystem accessType inputOutput, type SFInt32 CDATA "0"] |
| power | [power accessType inputOutput, type SFFloat CDATA "0"] |

http://www.web3d.org/x3d/content/X3dTooltips.html#TransmitterPdu



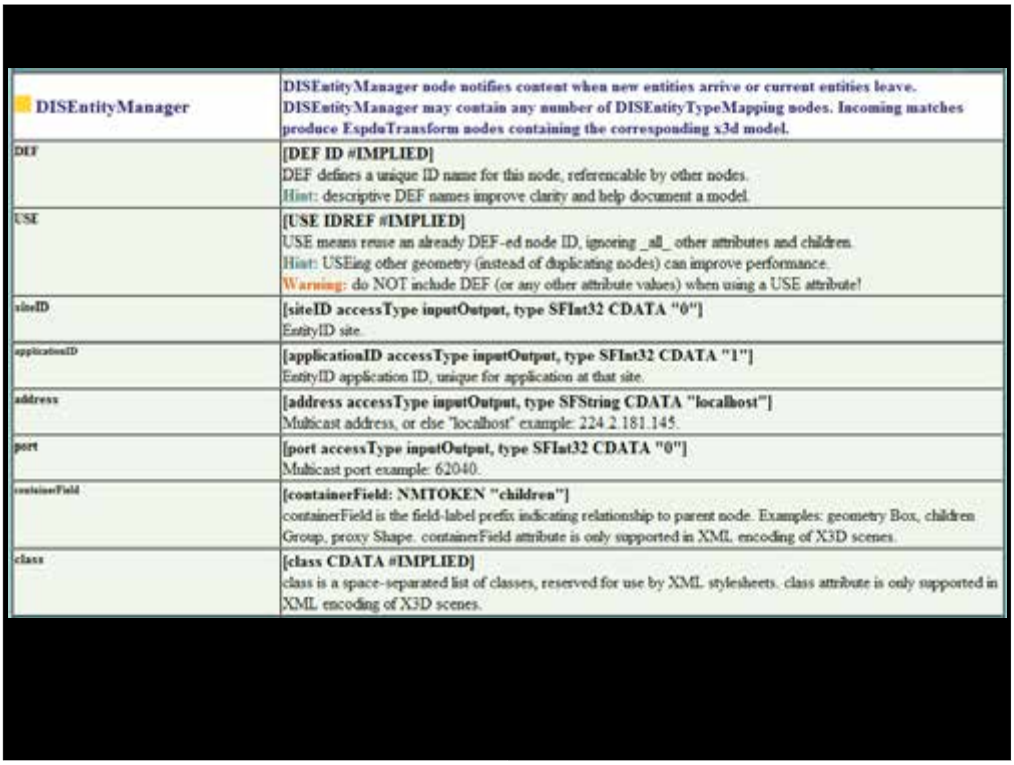
| | |
|---|---|
| radioEntityTypeCategory | [radioEntityTypeCategory accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeCountry | [radioEntityTypeCountry accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeDomain | [radioEntityTypeDomain accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeKind | [radioEntityTypeKind accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeNomenclature | [radioEntityTypeNomenclature accessType inputOutput, type SFInt32 CDATA "0"] |
| radioEntityTypeNomenclatureVersion | [radioEntityTypeNomenclatureVersion accessType inputOutput, type SFInt32 CDATA "0"] |
| relativeAntennaLocation | [relativeAntennaLocation accessType inputOutput, type SFVec3f CDATA "0 0 0"] |
| transmitFrequencyBandwidth | [transmitFrequencyBandwidth accessType inputOutput, type SFFloat CDATA "0.0"] |
| transmitState | [transmitState accessType inputOutput, type SFInt32 CDATA "0"] |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

http://www.web3d.org/x3d/content/X3dTooltips.html#TransmitterPdu

# DISEntityManager

DISEntityManager node notifies content when new entities arrive or current entities leave

- Identifies multicast *address, port* as well as session identification *applicationID, siteID*
- Contains list of DISEntityTypeMapping nodes which define entity filters, correspondences for each uniquely defined exercise

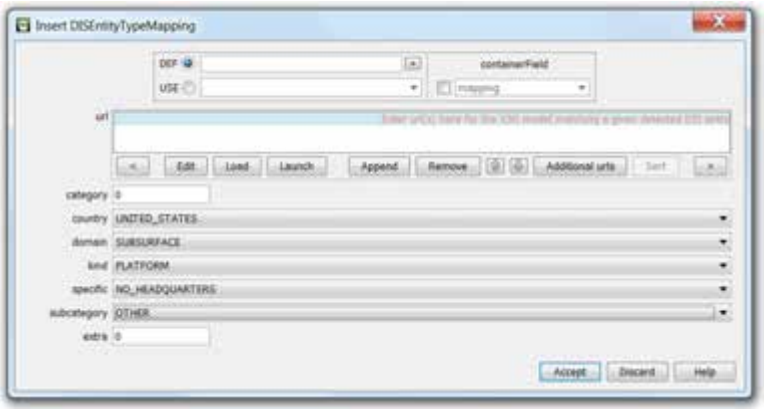


| DISEntityManager | DISEntityManager node notifies content when new entities arrive or current entities leave. DISEntityManager may contain any number of DISEntityTypeMapping nodes. Incoming matches produce EspduTransform nodes containing the corresponding x3d model. |
|---|---|
| DEF | [DEF ID #IMPLIED]<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | [USE IDREF #IMPLIED]<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| siteID | [siteID accessType inputOutput, type SFInt32 CDATA "0"]<br>EntityID site. |
| applicationID | [applicationID accessType inputOutput, type SFInt32 CDATA "1"]<br>EntityID application ID, unique for application at that site. |
| address | [address accessType inputOutput, type SFString CDATA "localhost"]<br>Multicast address, or else "localhost" example: 224.2.181.145. |
| port | [port accessType inputOutput, type SFInt32 CDATA "0"]<br>Multicast port example: 62040. |
| containerField | [containerField: NMTOKEN "children"]<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | [class CDATA #IMPLIED]<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

http://www.web3d.org/x3d/content/X3dTooltips.html#TextureCoordinateGenerator

# DISEntityTypeMapping

- Exposes fields for DISEntityTypeMapping
- Provides correspondence between detected entity identification fields and X3D models



| | |
|---|---|
| DISEntityTypeMapping | **DISEntityTypeMapping maps received DIS Entity type information to an X3D model, thus providing visual and behavioral representations matching received packets. Fields are processed in order: kind, domain, country, category, subcategory, specific, extra.**<br>Hint: **0 values are wildcards. All values in the ordered list must be 0 after the first 0 is defined.** |
| DEF | **[DEF ID #IMPLIED]**<br>DEF defines a unique ID name for this node, referencable by other nodes.<br>Hint: descriptive DEF names improve clarity and help document a model. |
| USE | **[USE IDREF #IMPLIED]**<br>USE means reuse an already DEF-ed node ID, ignoring _all_ other attributes and children.<br>Hint: USEing other geometry (instead of duplicating nodes) can improve performance.<br>Warning: do NOT include DEF (or any other attribute values) when using a USE attribute! |
| url | **[url accessType inputOutput, type MFString CDATA #IMPLIED]**<br><br>Hint: Strings can have multiple values, so separate each string by quote marks [ "http://www.url1.org" "http://www.url2.org" "etc." ].<br>Hint: XML encoding for " is " (a character entity).<br>Warning: strictly match directory and filename capitalization for http links!<br>Hint: can replace embedded blank(s) in url queries with %20 for each blank character. |
| kind | **[kind accessType inputOutput, type SFInt32 CDATA "0"]** |
| domain | **[domain accessType inputOutput, type SFInt32 CDATA "0"]** |
| country | **[country accessType inputOutput, type SFInt32 CDATA "0"]** |
| category | **[category accessType inputOutput, type SFInt32 CDATA "0"]** |
| subCategory | **[subCategory accessType inputOutput, type SFInt32 CDATA "0"]** |
| specific | **[specific accessType inputOutput, type SFInt32 CDATA "0"]** |
| extra | **[extra accessType inputOutput, type SFInt32 CDATA "0"]** |
| containerField | **[containerField: NMTOKEN "children"]**<br>containerField is the field-label prefix indicating relationship to parent node. Examples: geometry Box, children Group, proxy Shape. containerField attribute is only supported in XML encoding of X3D scenes. |
| class | **[class CDATA #IMPLIED]**<br>class is a space-separated list of classes, reserved for use by XML stylesheets. class attribute is only supported in XML encoding of X3D scenes. |

http://www.web3d.org/x3d/content/X3dTooltips.html#TextureCoordinateGenerator

back to Table of Contents

# Applications and Examples


## Setup: Microsoft loopback adapter

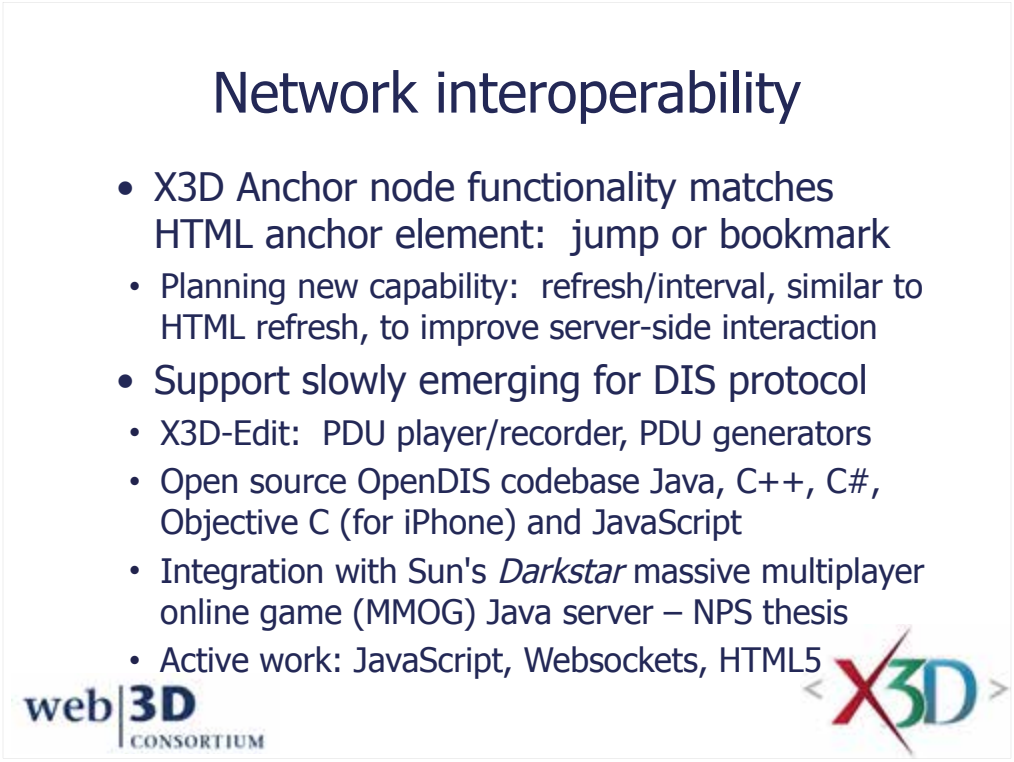
- Unlike Unix, Microsoft Windows does not have multicast loopback turned on by default
- Special setup thus needed for solitary testing
  - Conflicts can emerge when also using Cisco VPN
- Help page provided by AUV Workbench


Feedback thread link

TODO: consider allowing 127.0.0.1 loopback address as a supported option to multicast channel, sidestepping multicast impediments on Windows when performing simple testing on localhost only. Potential issue: handling multiple readers/writers. Example implementation already exists as part of Xj3D.

**Does the windows 7 loopback adapter support multicast?**

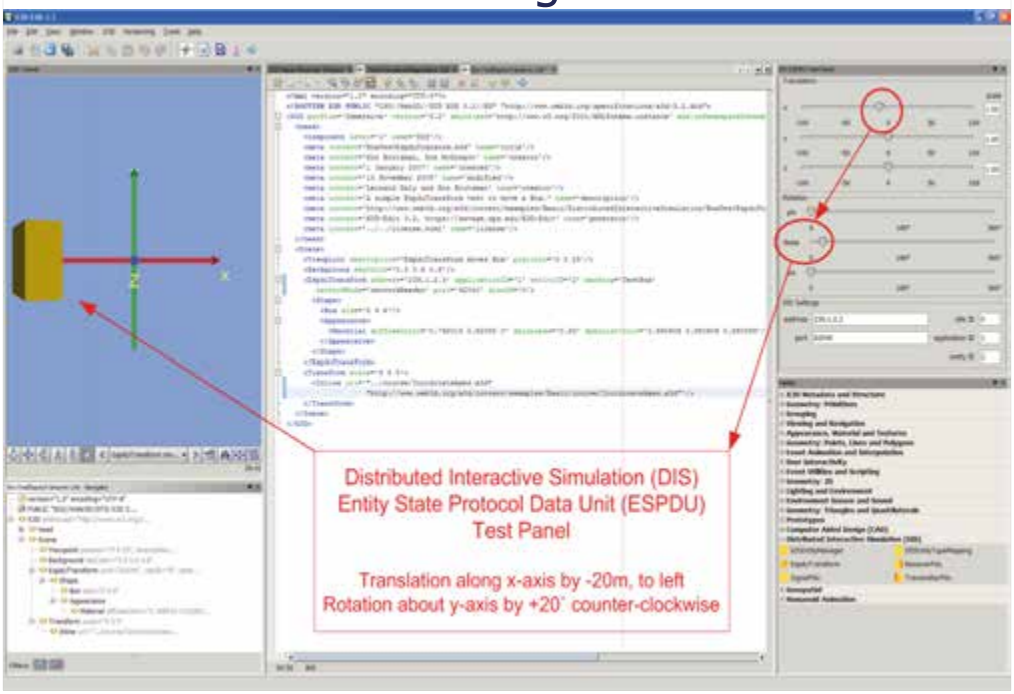
http://stackoverflow.com/questions/7162288/does-the-windows-7-loopback-adapter-support-multicast




Anchor refresh capability proposed for X3D v3.3:

http://www.web3d.org/membership/login/memberwiki/index.php/X3D_v3.3_Specification_Changes#X3DUrlObject_abstract_interface:_refresh_for_scene_reload

# DIS Networking Test Panel



# X3D DIS Implementations

More work needed!

X3DOM: work in progress

Xj3D: open source Java

- www.xj3D.org
- Embedded in (and launchable by) X3D-Edit

Other players to follow?

- BS Contact, InstantReality, FreeWrl, OpenVRML

Feature comparison available:

- Player support for X3D components wiki


# Obtaining example scenes

Primary: X3D Basic archives, DistributedInteractiveSimulation directory

- http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation
- Under version control at Sourceforge

Also some in X3D Savage archives, located in various directories (such as Scenarios)

- https://savage.nps.edu/Savage
- Maintained under version control at NPS

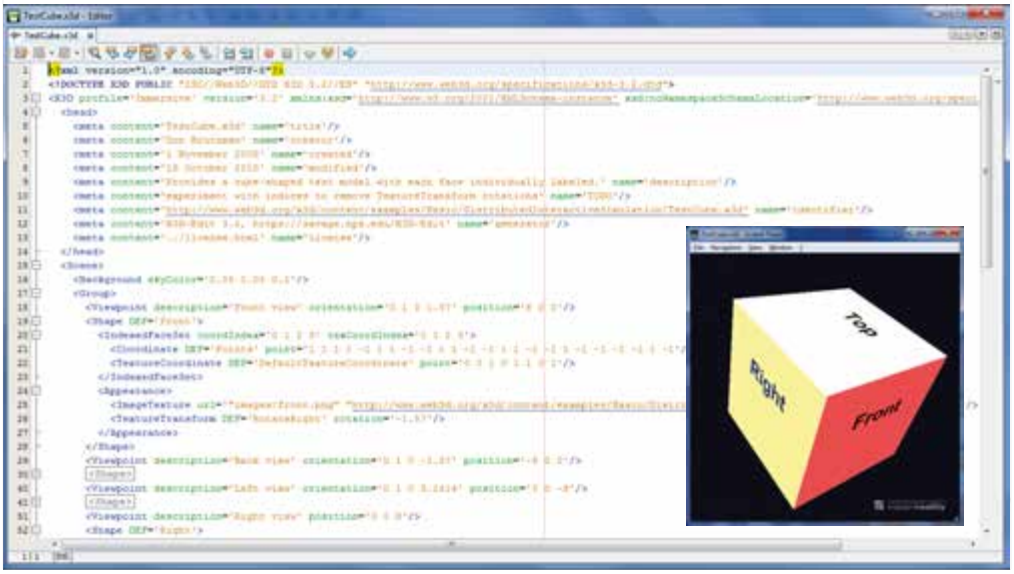
web3D CONSORTIUM


For NPS students, staff and partners, some additional DIS-capable X3D scenes and scenarios are available on the SavageDefense X3D archive

- https://savagedefense.nps.navy.mil/SavageDefense

# Test shape: TestCube.x3d

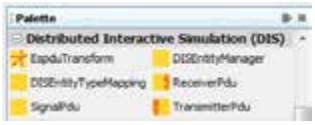
Simple shape with labeled ImageTexture images on each side, no DIS included




Available at

http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/TestCube.x3d

## X3D-Edit DIS support




X3D-Edit authoring tool includes multiple support capabilities for DIS usage

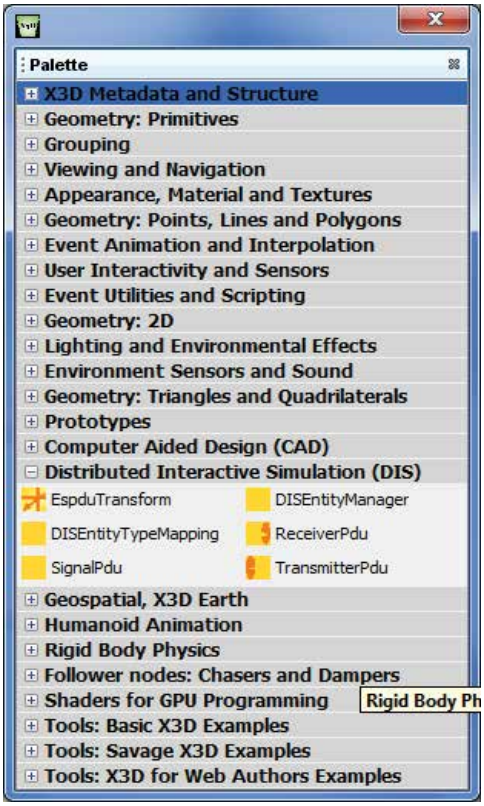
- Edit panels for X3D nodes: EspduTransform, ReceiverPdu, SignalPdu, TransmitterPdu, DISEntityManager, DISEntityTypeMapping
- DIS ESPDU Test panel for sending values
- DIS Player-Recorder for recording, playing back, loading or saving packets
- DIS PDU Sender Test menu selection for sending one of each PDU type available

web3D CONSORTIUM


X3D-Edit networking functionality menu:

• *X3D => Networking =>*
*DIS PDU Player-Recorder*, *ESPDU Sender Panel*, and *PDU Sender Test*

X3D-Edit drag-and-drop node editing palette:

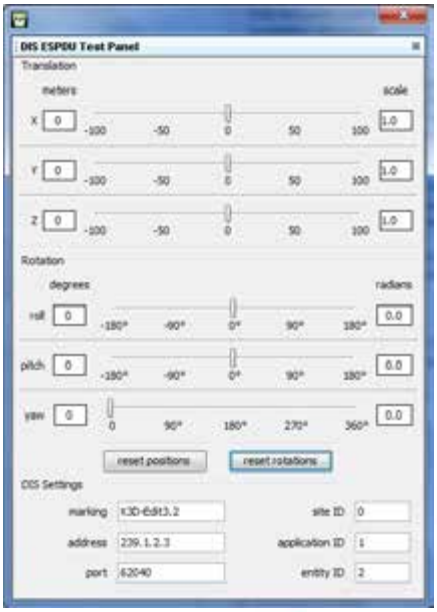


# X3D-Edit DIS ESPDU packet tester

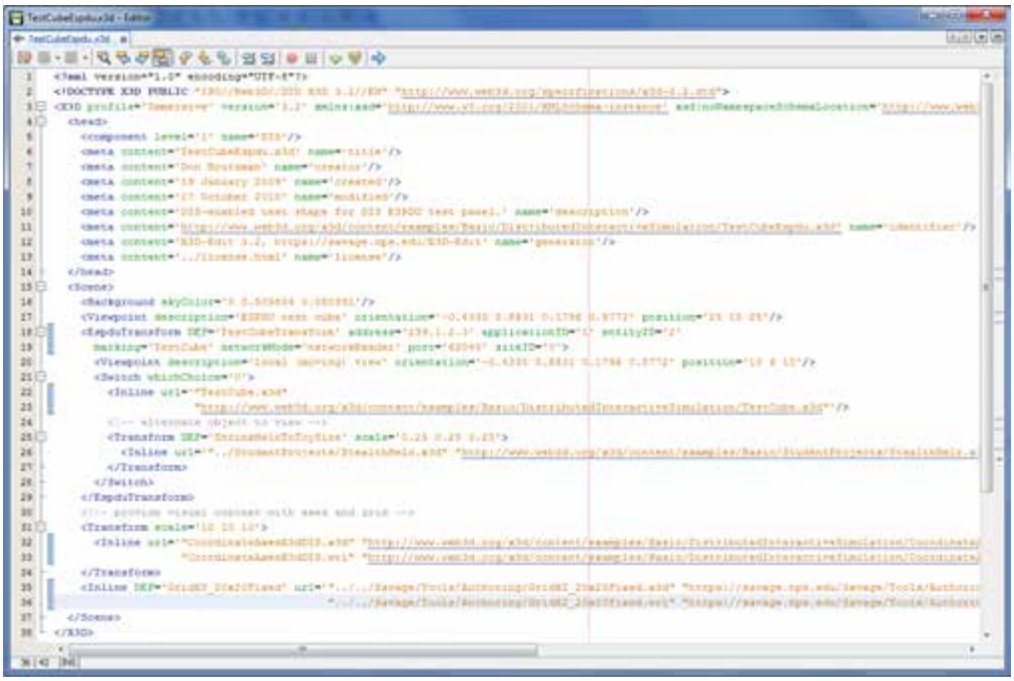
Select values to send

- *x y z* in meters multiplied by scale
- *roll pitch yaw* shown in degrees, also radians
- Type in *marking* and network parameters

Each change sends a new ESPDU packet

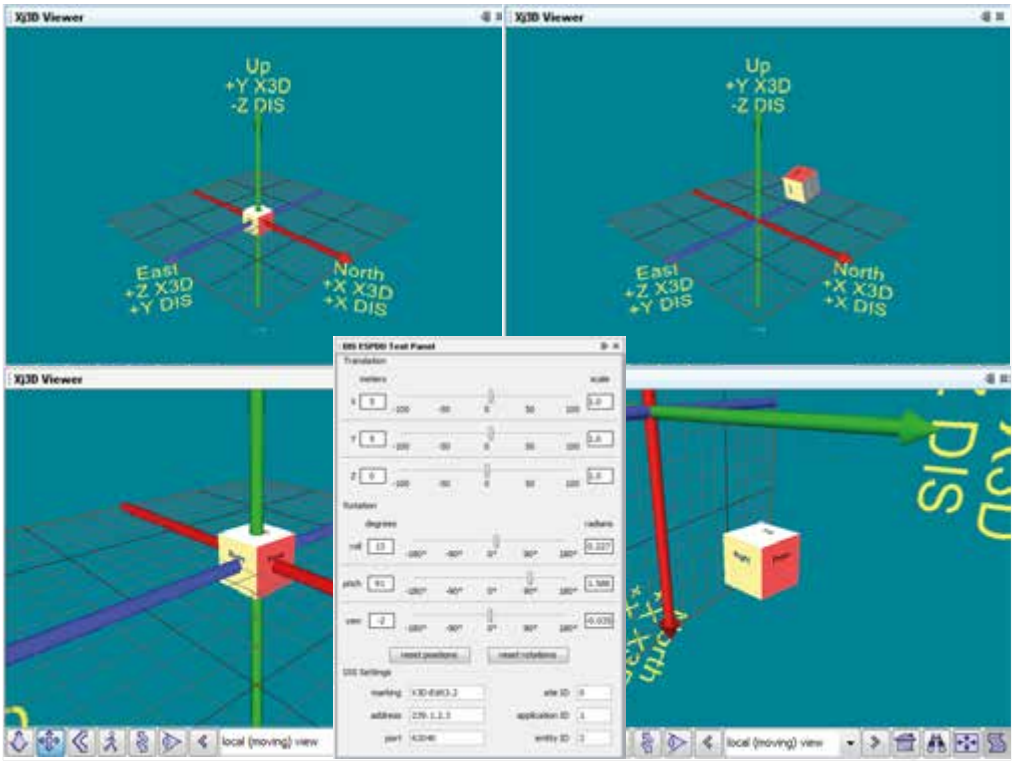


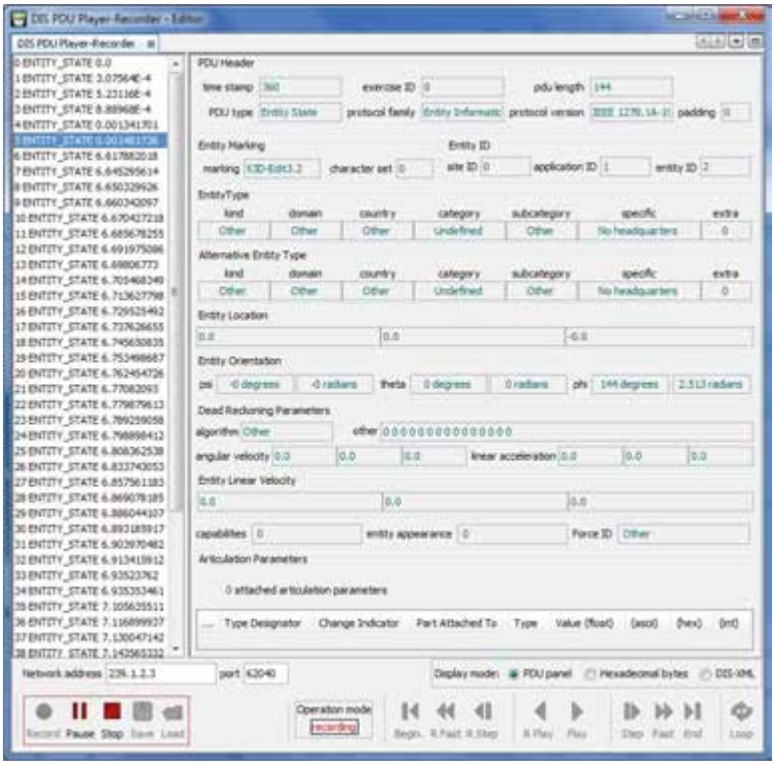
# Example: TestCubeEspdu.x3d



Available at

http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation/TestCubeEspdu.x3d


X3D Viewer
Up
+Y X3D
-Z DIS
East
+Z X3D
+Y DIS
North
+X X3D
+X DIS
X3D Viewer
Up
+Y X3D
-Z DIS
East
+Z X3D
+Y DIS
North
+X X3D
+X DIS
X3D Viewer
local (moving) view
local (moving) view




DIS player-recorder

embedded in X3D-Edit

# DIS PDU Sender Test menu selection

Menu item:

- *X3D > Networking > DIS PDU Sender Test*

Sends one PDU of each type supported

Useful for testing application connectivity

- Console of logged results: *View > IDE Log*
- Example excerpt:

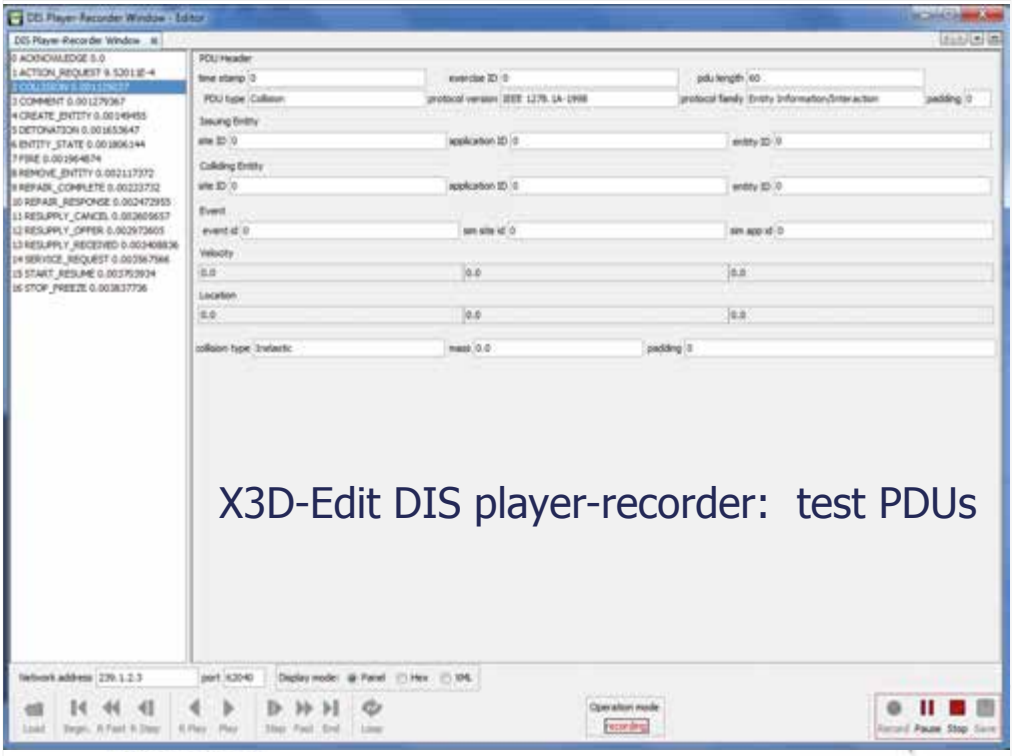
```
Sent PDU of type edu.nps.moves.dis.CollisionPdu
Sent PDU of type edu.nps.moves.dis.CommentPdu
Sent PDU of type edu.nps.moves.dis.CreateEntityPdu
Sent PDU of type edu.nps.moves.dis.DetonationPdu
Sent PDU of type edu.nps.moves.dis.EntityStatePdu
Sent PDU of type edu.nps.moves.dis.FirePdu
Sent PDU of type edu.nps.moves.dis.RemoveEntityPdu
[...]
```

web3D CONSORTIUM


Example IDE Log excerpt:

[…]
Sent PDU of type edu.nps.moves.dis.AcknowledgePdu
Sent PDU of type edu.nps.moves.dis.ActionRequestPdu
Sent PDU of type edu.nps.moves.dis.CollisionPdu
Sent PDU of type edu.nps.moves.dis.CommentPdu
Sent PDU of type edu.nps.moves.dis.CreateEntityPdu
Sent PDU of type edu.nps.moves.dis.DetonationPdu
Sent PDU of type edu.nps.moves.dis.EntityStatePdu
Sent PDU of type edu.nps.moves.dis.FirePdu
Sent PDU of type edu.nps.moves.dis.RemoveEntityPdu
Sent PDU of type edu.nps.moves.dis.RepairCompletePdu
Sent PDU of type edu.nps.moves.dis.RepairResponsePdu
Sent PDU of type edu.nps.moves.dis.ResupplyCancelPdu
Sent PDU of type edu.nps.moves.dis.ResupplyOfferPdu
Sent PDU of type edu.nps.moves.dis.ResupplyReceivedPdu
Sent PDU of type edu.nps.moves.dis.ServiceRequestPdu
Sent PDU of type edu.nps.moves.dis.StartResumePdu
Sent PDU of type edu.nps.moves.dis.StopFreezePdu
[...]
PDU of type TRANSMITTER not created or sent
PDU of type SIGNAL not created or sent
PDU of type RECEIVER not created or sent
[...]

[...]
Sent PDU of type edu.nps.moves.dis.AcknowledgePdu
Sent PDU of type edu.nps.moves.dis.ActionRequestPdu
Sent PDU of type edu.nps.moves.dis.CollisionPdu
Sent PDU of type edu.nps.moves.dis.CommentPdu
Sent PDU of type edu.nps.moves.dis.CreateEntityPdu
Sent PDU of type edu.nps.moves.dis.DetonationPdu
Sent PDU of type edu.nps.moves.dis.EntityStatePdu
Sent PDU of type edu.nps.moves.dis.FirePdu
Sent PDU of type edu.nps.moves.dis.RemoveEntityPdu
Sent PDU of type edu.nps.moves.dis.RepairCompletePdu
Sent PDU of type edu.nps.moves.dis.RepairResponsePdu
Sent PDU of type edu.nps.moves.dis.ResupplyCancelPdu
Sent PDU of type edu.nps.moves.dis.ResupplyOfferPdu
Sent PDU of type edu.nps.moves.dis.ResupplyReceivedPdu
Sent PDU of type edu.nps.moves.dis.ServiceRequestPdu
Sent PDU of type edu.nps.moves.dis.StartResumePdu
Sent PDU of type edu.nps.moves.dis.StopFreezePdu


X3D-Edit DIS player-recorder: test PDUs

# Also in NPS Savage archives: specific scenarios available


These scenes typically have

- EspduTransform nodes with session info, as parent nodes translating/rotating child models
- Inline nodes that retrieve non-networked platform models for rendering in the scene

web3D CONSORTIUM

# Autonomous Unmanned Vehicle (AUV) Workbench

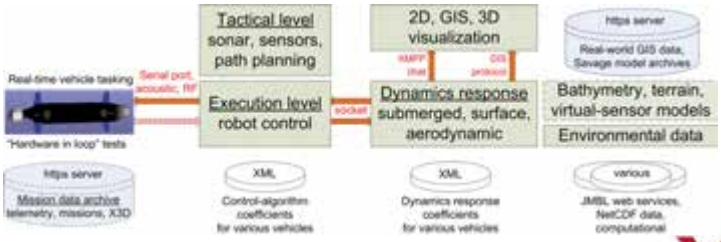
The NPS Autonomous Unmanned Vehicle Workbench (AUVW) supports physics-based mission rehearsal, real-time task-level control of robot missions, and replay of recorded results in support of autonomous unmanned underwater, surface and air vehicles.

- Includes embedded DIS networking to connect physics-based modeling of robot missions to Xj3D visualization pane using Savage models
- https://savage.nps.edu/AuvWorkbench
- https://savage.nps.edu/Savage/AuvWorkbench/OperatingAreas


Robot Mission Planning and 3D Visualization
Autonomous
Unmanned
Vehicle
Workbench
Rehearsal
Reality
Replay
THE MOVES INSTITUTE
"Effects-Based Thinking"

# AUV Workbench DIS Output

- Robot mission definitions are executed by robot controller, driving the vehicle
- Dynamics engine computes motion response, sent to network and Xj3D viewer via DIS
- DIS packets are DIS-XML chat or native PDUs




web3D CONSORTIUM

# DIS bridging

Some applications can be found in Open-DIS codebase to bridge DIS packets LAN-to-LAN

- Accomplished by opening unicast socket between LANs which passes packets, relaying them to/from multicast at endpoints
- This is needed since routing defaults do not allow multicast to escape an individual LAN
- Macedonia, Michael R. and Brutzman, Donald P., "MBone Provides Audio and Video Across the Internet," *IEEE COMPUTER*, vol. 27 no. 4, April 1994, pp. 30-36.

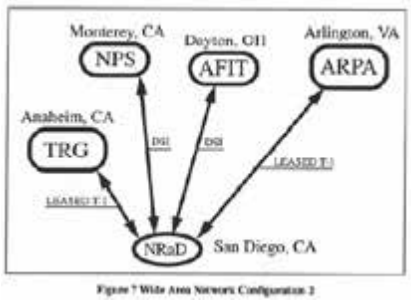
## Selectively Reliable Multicast Protocol (SRMP) DIS-HLA Gateway

*GMUGateway* is a research-oriented protocol translator developed in Java for distributed simulations.

- Provides interoperability among different types of simulation architectures
- Converts DIS protocol packets to High Level Architecture (HLA) Run-Time Infrastructure (RTI) service calls, and vice versa.
- Uses NPS DIS-Java-VRML implementation (for DIS) and RPR-FOM (for HLA)
- http://netlab.gmu.edu/SRMP/gatewayDoc.php

# DIS networking thesis: Steve Zeswitz

Zeswitz, Steven, *NPSNET: Integration of Distributed Interactive Simulation (DIS) Protocol for Communication Architecture and Information Interchange*, Masters Thesis, Naval Postgraduate School, Monterey California, September 1993.




Early work.

https://savage.nps.edu/AuvWorkbench/website/documentation/theses/ZeswitzDistributedInteractiveSimulationProtocolThesis.September1993.pdf

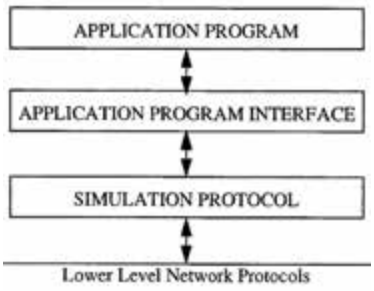
Abstract.

The Computer Science Department at the Naval Postgraduate School in Monterey, California has developed a low-cost real-time interactive network based simulation system, known as NPSNET, that uses Silicon Graphics workstations. NPSNET has used non-standard protocols which constrains its participation in distributed simulation. DIS specifies standard protocols and is emerging as the international standard for distributed simulation. This research focused on the development of a robust, high-performance implementation of the DIS Version 2.0.3 protocol to support graphic simulation systems (e.g. NPSNET). The challenge was to comply with the standard and minimize network latency thereby maintaining the time and space coherence of distributed simulations. The resulting DIS Network Library consists of an application program interface (API) to low level network routines, a host of network utilities, and a network harness that takes advantage of multiprocessor workstations. The library was successfully tested on our local network and two configurations of a T-I based internet, the Defense Simulation Internet (DSI), with the Air Force Institute of Technology and Advanced Research Projects Agency. The testing confirmed that the semantics and syntax of the DIS protocol is properly implemented and the latency incurred by the network does not adversely effect the simulation application.

# DIS networking thesis: Steve Stone

Stone, Steven W.,

*A rapidly reconfigurable, application layer, virtual environment network protocol,* Masters Thesis, Naval Postgraduate School (NPS), Monterey California, June 1996. Built an improved NPSNET implementation.




http://edocs.nps.edu/npspubs/scholarly/theses/1996/Jun/96Jun_Stone.pdf

Abstract.

The current Distributed Interactive Simulations (DIS) Protocol has a limited ability to support real time, simulated engagements of more than 1000 entities because of its excessive use of network resources. It also lacks the extensibility to add new protocol data units to support new simulation requirements. To solve these problems it is necessary to design an implement a rapidly reconfigurable network protocol that can be easily changed and distributed to all entities in a large scale simulation. This protocol must be highly flexible and allow for the optimization of data content during execution. The approach used was to design and build a rapidly reconfigurable network protocol and the tools necessary to use it. This was accomplished in four phases. First, a protocol using the concepts of Self-defined Messages with Multiple Presentations was developed. Second, a formal grammar to describe the protocol was designed. Third, an existing protocol development tool, the DIS protocol Support Utility, was modified to use the new protocol and grammar. Fourth, the protocol was tested to determine its effect on network resource utilization. As a result of this effort, a network protocol for distributed simulations that can be optimized at run-time and easily modified has been developed. Testing shows that the protocol can reduce the network bandwidth necessary for a large-scale distributed simulation by up to 70%.

# Savage thesis: Shane Nicklaus

Nicklaus, Shane D., *Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE)*, Master's Thesis, Naval Postgraduate School, Monterey California, September 2001. Information Systems Technology curriculum. Co-advisors Curtis L. Blais and Dan Boger.

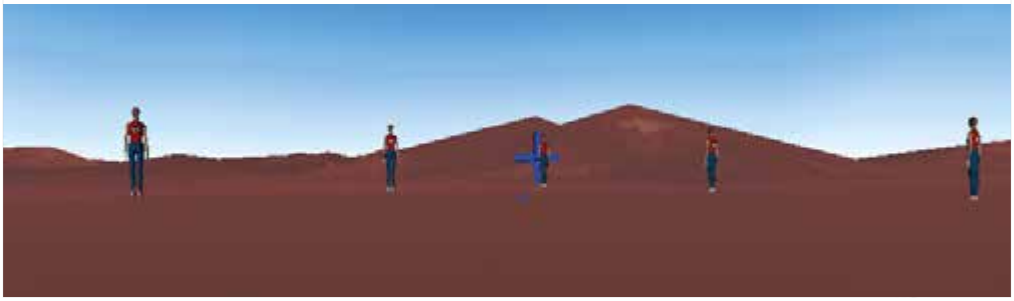

http://edocs.nps.edu/npspubs/scholarly/theses/2001/Sep/01Sep_Nicklaus.pdf

Abstract.

Today's planning and modeling systems use two-dimensional (2D) representations of the three-dimensional (3D) battlespace. This presents a challenge for planners, commanders, and troops to understand the true nature of the battlespace. This thesis shows how 3D visualization can give both operation planners and executors a better understanding of the battlespace that can augment today*s 2D systems. Automatic creation of a 3D model for an amphibious operation allows the planner to view an operation order as a whole, from different perspectives. Recommended changes can be made and their effects immediately known. Warfighters can use the same tools for mission preparation and review. The United States and NATO nations use the Land C2 Information Exchange Data Model (LC2IEDM), formally known as the Generic Hub, as a common method for exchanging data between independent systems. As part of the Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE) project, this research presents an integrated Web access and 3D visualization strategy for Department of Defense (DOD) tactical messaging and operation orders using the Generic Hub data model and the Extensible Markup Language (XML). A number of alternative yet consistent ways to represent an amphibious operation scenario demonstrate the power, flexibility and scalability of the SAVAGE approach

# H-Anim thesis: Tom Miller

Miller, Thomas E., *Integrating Realistic Human Group Behaviors Into A Networked 3D Virtual Environment*, Master's Thesis, Naval Postgraduate School, Monterey California, September 2000. MOVES curriculum. Received NPS Outstanding Academic Achievement Award for highest academic honors among Department of Defense (DoD) students.



https://savage.nps.edu/Savage/documents/DisJavaVrml-HumanoidTeamsThesis-MillerSeptember2000.pdf
https://savage.nps.edu/Savage/documents/DisJavaVrml-HumanoidTeamsBrief-MillerSeptember2000.ppt

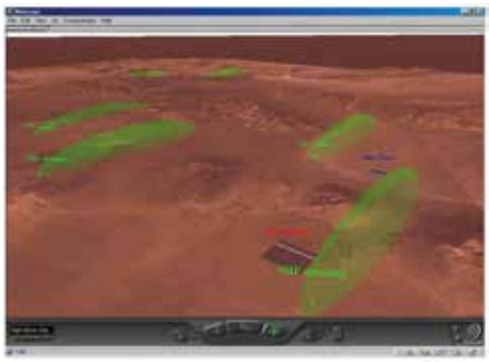
Abstract.

Virtual humans operating inside large-scale virtual environments (VE) are typically controlled as single entities. Coordination of group activity and movement is usually the responsibility of their "real world" human controllers. Georeferencing coordinate systems, single-precision versus double-precision number representation and network delay requirements make group operations difficult. Mounting multiple humans inside shared or single vehicles, (i.e. air-assault operations, mechanized infantry operations, or small boat/riverine operations) with high fidelity is often impossible. The approach taken in this thesis is to reengineer the DIS-Java-VRML Capture the Flag game geolocated at Fort Irwin, California to allow the inclusion of human entities. Human operators are given the capability of aggregating or mounting nonhuman entities for coordinated actions. Additionally, rapid content creation of human entities is addressed through the development of a native tag set for the Humanoid Animation (H-Anim) 1.1 Specification in Extensible 3D (X3D). Conventions are demonstrated for integrating the DIS-Java-VRML and H-Anim draft standards using either VRML97 or X3D encodings. The result of this work is an interface to aggregate and control articulated humans using an existing model with a standardized motion library in a networked virtual environment. Virtual human avatars can be mounted and unmounted from aggregation entities. Simple demonstration examples show coordinated tactical maneuver among multiple humans with and without vehicles. Live 3D visualization of animated humanoids on realistic terrain is then portrayed inside freely available web browsers.


This is another snapshot from the Miller thesis.

## Signals thesis: Dave Laflam

Laflam, David W., *3D Visualization of Theater-Level Radio Communications Using a Networked Virtual Environment*, Master's Thesis, Naval Postgraduate School, Monterey California, September 2000. MOVES curriculum.

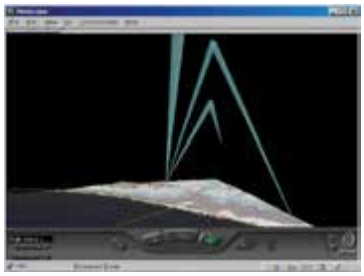


https://savage.nps.edu/Savage/documents/DisJavaVrml-SignalsVisualizationThesis-LaflamSeptember2000.pdf

*Abstract*.

The military is heavily reliant on the transfer of information among various networks in day-to-day operations. Radio-based communications networks that support this volume of information are complex, difficult to manage, and change frequently. Communications network planners need a way to clearly visualize and communicate mobile operational network capabilities, particularly to network users. By using the DJS-Java-VRML simulation and modeling toolkit, visualizations of radio-frequency energy and radio path- profiling data can be quickly generated as 3D models. These animated 3D visualizations can be loaded into a networked virtual environment, so that communications planners can detect a variety of problems such as radio frequency interference and gaps in coverage. Planners can also brief senior staff, plan within their own staff, and collaborate with communications staff planners in distant locations using such virtual environments. DIS-Java-VRML visualization tools can provide a clear picture of the battle space with respect to the deployed communications architecture. The prototypes presented in this thesis demonstrate the ability to generate a shared visualization that can show a radio communications network in 3D. Such dynamic visualizations increase communications planning information bandwidth and yield more intuitive ways of presenting information to users. Higher information density in a more intuitive format enables better understanding with quicker reaction times. This thesis and the visualization tool discussed provide the foundation for fundamental improvements in visualizing radio communications environments.

## Visualization thesis: Mike Hunsberger



Hunsberger, Michael G., *3D Visualization Of Tactical Communications for Planning and Operations Using Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D)*, Master's Thesis, Naval Postgraduate School, Monterey California, June 2001. Awarded Air Force Association Award for Outstanding Air Force Student. Dual Master of Science degrees: Systems Technology for Joint C3 Systems and Computer Science.

https://savage.nps.edu/Savage/documents/DisJavaVrml-SignalsVisualizationThesis-HunsbergerJune2001.pdf

Abstract.

The military is increasingly reliant on communication networks for day-to-day tasks as well as large-scale military operations. Tactical communications networks are growing progressively more complex as the amount of information required on the battlefield increases. Communication planners require more advanced tools to perform and manage signal-planning activities. This work examines the use of 3D visualizations to assist in tactical signal planning. These visualizations are developed using Virtual Reality Modeling Language (VRML), Extensible 3D (X3D) graphics, and Distributed Information Simulation (DIS) for network connectivity. These visualizations and the connectivity provide signal planners the ability to generate 3D scenarios quickly identifying problems such as frequency interference, connectivity problems, and marginal-coverage areas. Network connectivity also provides a collaborative planning environment for geographically dispersed units. The NATO Global Hub Land C2 Information Exchange Data Model (LC2IEDM) is a semantic model designed for information passing between systems. This work also examines LC2IEDM for its ability to represent tactical communication plans and facilitate the autogeneration of 3D scenarios.

# Scenario thesis: James Harney

Harney, James W., *Analyzing Anti-Terrorist Tactical Effectiveness of Picket Boats for Force Protection of Navy Ships Using X3d Graphics and Agent-Based Simulation,* Masters Thesis, Naval Postgraduate School, Monterey California, March 2003. Co advisors Curtis L. Blais, Gordon Schacher, and John Hiles.

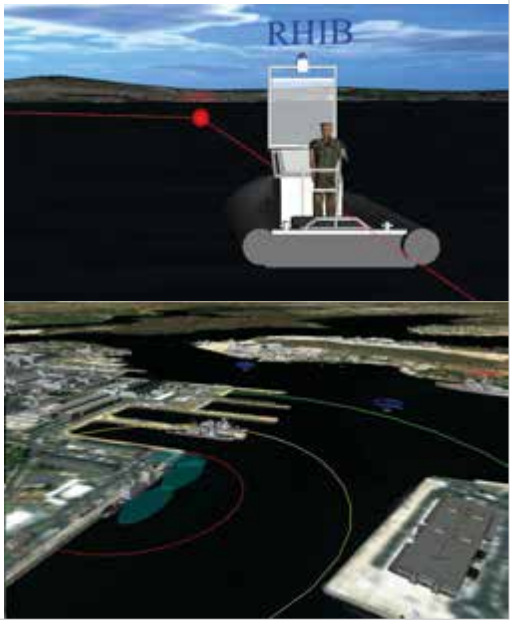
http://edocs.nps.edu/npspubs/scholarly/theses/2003/Mar/03Mar_Harney.pdf

Abstract.

Despite the many advances achieved within both Modeling and Simulation and Information Technology over the past several decades, practical application of such technology remains under-utilized by operational units in the United States Navy. Furthermore, when such technology has been deployed in the last decade it has been to exercise operator proficiency or increase C4I battlespace awareness. Few tools have allowed operational warfighters to run 'what-if' simulation scenarios to aid in development of tactical plans for executing published doctrine. The approach taken in this thesis is to select an exemplar warfare area, in this case Anti-Terrorism and Force Protection for Navy ships, and through research and development to identify, develop, and deploy the necessary modeling and simulation (M & S) technologies to demonstrate a prototypical planning tool that can be used by today's deployed warfighter. All research and work is conducted in a web-based, 'user-centric' fashion utilizing a combination of user-driven and agentbased control of entities for simulation iterations, along with various open source technologies which include Extensible 3D Graphics (X3D), Scalable Vector Graphics (SVG), and Extensible Markup Language (XML). Conventions are demonstrated for the integration of the many academic disciplines utilized during this research to achieve automatic generation of tactically significant scenarios. In order to give the end-user the greatest insight towards potential drawbacks in the tactical planning against surface-borne terrorist threats, various 2D and 3D media provide both real-time and non-real time scenario playback. The result of this work is a fully integrated, prototypical, Java-based application that demonstrates how various Open-Source, web-based technologies can be applied in order to provide the tactical operator with tools to aid in Force Protection planning. Scenarios can be auto generated, viewed, analyzed, and manipulated by end users with little to no computer experience necessary beyond requirements for operation of a desktop personal computer (PC) in the Information Technology for the 21st Century (IT-21) environment at sea. This approach has broad applicability to improve the tactical awareness and defensive posture of ships defending against terrorist attacks in port.

# Scenario thesis: Pat Sullivan

Sullivan, Patrick J., *Evaluating the Effectiveness of Waterside Security Alternatives for Force Protection of Navy Ships and Installations using X3D Graphics and Agent-Based Simulation,* Masters Thesis, Naval Postgraduate School, Monterey California, September 2006.
Co-advisor Curt Blais.




http://handle.dtic.mil/100.2/ADA457197

http://edocs.nps.edu/npspubs/scholarly/theses/2006/Sep/06Sep_Sullivan.pdf

Abstract.

The individuals charged with the task of planning, developing and implementing force protection measures both at the unit and installation level must consider numerous factors in formulating the best defensive posture. Currently, force protection professionals utilize multiple sources of information regarding capabilities of systems that are available, and combine that knowledge with the requirements of their installation to create an overall plan. A crucial element missing from this process is the ability to determine, prior to system procurement, the most effective combination of systems and employment for a wide range of possible terrorist attack scenarios. This thesis is inspired by the work done by James Harney, LT, USN (2003). The thesis will expand the Anti-Terrorism Force Protection Tool developed during the original thesis by including the capability of testing force protection measures in multiple scenarios by utilizing models of force protection equipment and forces, virtual worlds of existing naval facilities, and terrorist agents that exhibit intent and behavioral characteristics which can test the effectiveness of the force protection equipment used. The result of this work is a scalable and repeatable methodology for generating large-scale, agent-based simulations for AT/FP problem domains providing 3D visualization, report generation, and statistical analysis.

## SMAL Metadata Thesis: Travis Rauch

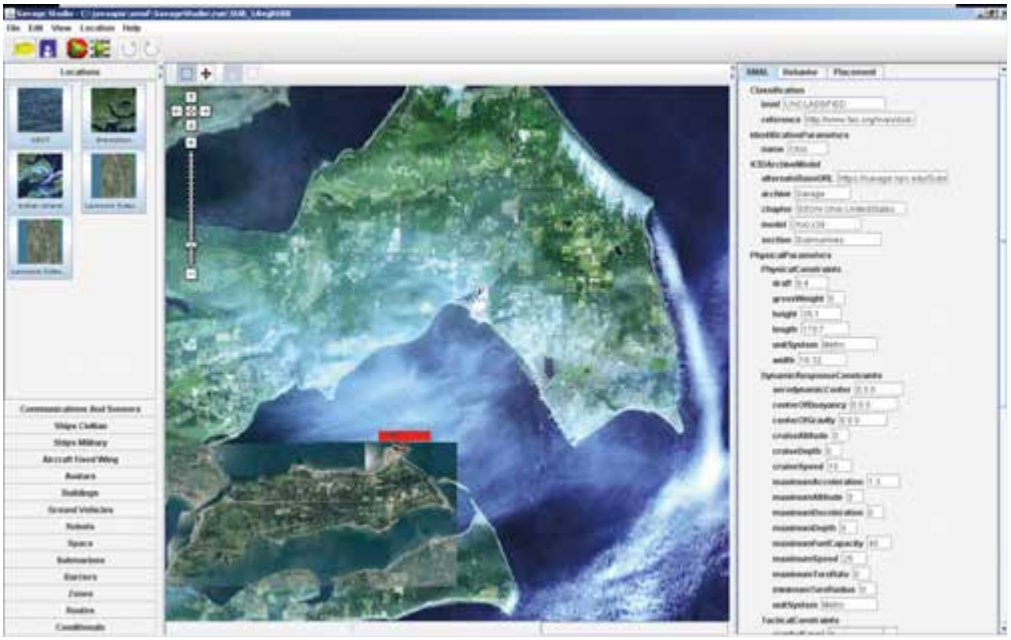
- Rauch, Travis, *Savage Modeling Analysis Language (SMAL): Metadata for Tactical Simulations and X3D Visualizations,* Masters Thesis, Naval Postgraduate School, Monterey California, March 2006.
- SAVAGE Modeling and Analysis Language (SMAL) is the NPS MOVES strategy for identifying tactical, physical and simulation-oriented information regarding models for vehicles, terrain and entities in virtual environments. Equivalent XML and X3D representations for SMAL are defined.
- https://savage.nps.edu/Savage/Tools/SMAL/SMAL.html

https://savage.nps.edu/Savage/Tools/SMAL/docs/TravisRauchThesisSmalMetadata.2006March.pdf

Abstract.

Visualizing operations environments in three dimensions (3D) supports the warfighters' ability to make rapid, well-informed decisions by presenting complex systems in a naturalistic, integrated display format. Unfortunately, constructing these environments is a time-consuming task requiring specific expertise not typically available in the command center. The future use of 3D visualization in military operations depends on the ability of personnel with minimal graphics experience to create virtual environments quickly and accurately by leveraging data-driven customization of content from model archives with the data available in the command center. Practical 3D visualization depends on standardized scene autogeneration. The Extensible 3D (X3D) Graphics family of specifications is approved by the International Standards Organization (ISO) as the Web-based format for the interchange and rendering of 3D scenes. Previous work has demonstrated that an archive of X3D scenes, such as the Scenario Authoring and Visualization for Advanced Graphical Environments (SAVAGE) library, can be used to autogenerate sophisticated 3D tactical environments. Assembling and making sense of the data necessary to autogenerate a 3D environment requires context and good documentation, best accomplished through metadata. Metadata also supports data-centric, component-based design; key philosophies in promoting interoperability of networked applications. Coupled with recent developments in X3D, enhanced features of the Savage X3D Model archives are now sufficiently mature to support rapid generation of tactical environments. This thesis proposes an XML metadata standard to collect and organize the information necessary to create and populate a tactical 3D virtual environment: the Savage Modeling and Analysis Language (SMAL). The logical extension of a well designed standard is the ability to cross the boundaries of usage, allowing simulators to share data with command and control (C2) suites and mission planning tools based on the construction of a virtual scene. SMAL provides the informational "glue" necessary to perform tactical modeling, simulation, and analysis using networked, physics-based X3D virtual environments.

## SavageStudio scenario using SMAL metadata



SMAL website:

https://savage.nps.edu/Savage/Tools/SMAL/SMAL.html

Savage Studio:

https://savage.nps.edu/Savage/developers.html#DES

## DIS, XMPP thesis: LEE, Daryl Chin-Siong

LEE, Daryl Chin Siong, *NPS AUV Workbench: Collaborative Environment for Autonomous Underwater Vehicle (AUV) Mission Planning and 3D Visualization*, Master's Thesis, Naval Postgraduate School, Monterey California, March 2004. Computer Science curriculum. Co-advisor Curtis Blais, second readers John Hiles and Duane Davis.



http://edocs.nps.edu/npspubs/scholarly/theses/2004/Mar/04Mar_Lee.pdf

Abstract:

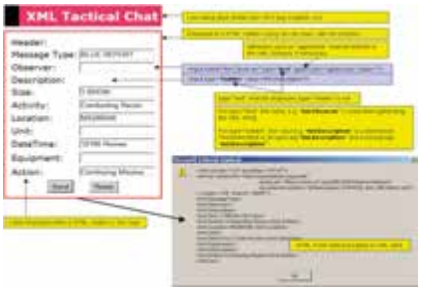
The absence of common software platforms for Autonomous Underwater Vehicle (AUV) mission planning and analysis is an ongoing impediment to collaborative work between research institutions, their partners, and end users. This thesis details the design and implementation of a distributable application to facilitate AUV mission planning and analysis. Java-based open-source libraries and a component-based framework provide diverse functionalities. The extensible Markup Language (XML) is used for data storage and message exchange, Extensible 3D (X3D) Graphics for visualization and XML Schema-based Binary Compression (XSBC) for data compression. The AUV Workbench provides an intuitive cross-platform-capable tool with extensibility to provide for future enhancements such as agent-based control, asynchronous reporting and communication, loss-free message compression and built-in support for mission data archiving. This thesis also investigates the Jabber instant messaging protocol, showing its suitability for text and file messaging in a tactical environment. Exemplars show that the XML backbone of this open-source technology can be leveraged to enable both human and agent messaging with improvements over current systems. Integrated Jabber instant messaging support makes the NPS AUV Workbench the first custom application supporting XML Tactical Chat (XTC). Results demonstrate that the AUV Workbench provides a capable testbed for diverse AUV technologies, assisting in the development of traditional single-vehicle operations and agent-based multiple-vehicle methodologies. The flexible design of the Workbench further encourages integration of new extensions to serve operational needs. Exemplars demonstrate how in-mission and post-mission event monitoring by human operators can be achieved via simple web page, standard clients or custom instant messaging client. Finally, the AUV Workbench's potential as a tool in the development of multiple-AUV tactics and doctrine is discussed.

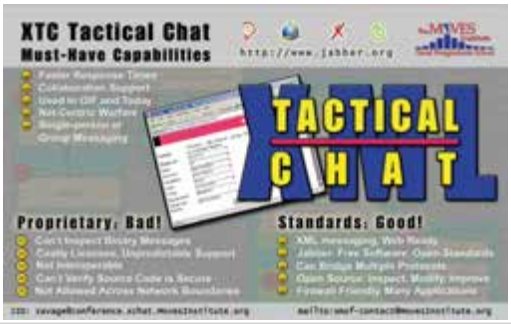
# XMPP Chat Thesis: Dan DeVos

DeVos, Daniel A., *XML Tactical Chat (XTC): The Way Ahead for Navy Chat*,

Masters Thesis, Naval Postgraduate School, Monterey California, September 2007. Second reader Don McGregor.

Set stage for DIS-XML.




http://edocs.nps.edu/npspubs/scholarly/theses/2007/Sep/07Sep_DeVos.pdf

Abstract.

The motivation for pursuing XML-based tactical chat includes the great potential of this technology and fixing limitations of current chat programs. XTC capabilities have the potential to completely upgrade and restructure all tactical military communications. The current tools for military chat include IRC, Yahoo, MSN, AIM, ICQ, and NKO. None of these provides the full functionality or interoperability needed in a joint environment. Moreover, if a nonproprietary chat protocol is developed, it can lead to a decision-support environment in which data, text, audio, and video can be logged, evaluated and managed, all in a Web environment where no additional specialized software or hardware is needed. Chat technology challenges for the military fit into three areas: tactical, technical, and administrative. Tactically, there are many ways chat can be used, but effective practices are not yet defined in procedures or doctrine. Joint forces use a myriad of chat programs that don't interoperate and are usually proprietary. Technically, many chat programs are barred by firewalls and lack a robust interface to allow logging and searching past chats. From an administrative prospective, plain-text chat has no structure. Scheduling and controlling who attends or converses remains undefined. Within DoD there is no standard for how, when, and by whom chats ought to be conducted. Possible approaches to these problems include adopting a proprietary chat system or customizing an open-source implementation. Proprietary solutions are costly, do not interoperate well, and are too inflexible for a technology that is evolving rapidly. Open-source software can provide a solution that is adaptable, extensible, quick to implement, straightforward to maintain, and relatively inexpensive. This thesis provides a preliminary assessment of XML-based tactical chat (XTC) using an open source, open-standards solution. Promising initial results demonstrate that an XML document can be sent from a XHTML page in a Web browser to an off-the-shelf Jabber client via a Web server. Further, available server and client implementations can enable a research and development plan for rapid development.

# Simulator thesis: Chris Fitzpatrick

Fitzpatrick, Christopher, *Integration of Robotic Technology, X3D Computer Graphics and Digital Imaging to Modernize the Expeditionary Warfare Demonstrator (EWD),* Masters Thesis, Naval Postgraduate School, Monterey California, September 2009. Second reader Amela Sadagic. Awarded SPAWAR Student Research Fellowship September 2008.

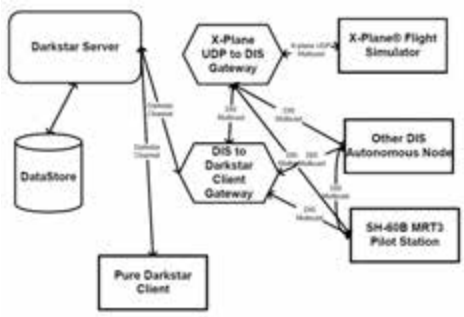
http://edocs.nps.edu/npspubs/scholarly/theses/2009/Sep/09Sep_Fitzpatrick.pdf

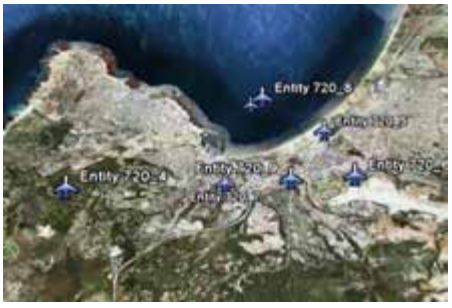
Abstract.

In the summer of 2008, the Commandant of the Marine Corps (CMC) released a message to all Marines and Sailors detailing plans to revitalize U.S. naval amphibious competency. Current responsibilities in Iraq and Afghanistan have significantly reduced available training time causing overall amphibious readiness to suffer. In response, this thesis evaluates 3D visualization techniques and other virtual environment technologies available to support these mission-critical training goals. The focus of this research is to modernize the Expeditionary Warfare Demonstrator (EWD) located aboard Naval Amphibious Base (NAB) Little Creek, Virginia. The EWD has been used to demonstrate doctrine, tactics, and procedures for all phases of amphibious operations to large groups of Navy, Marine Corps, Joint, Coalition and civilian personnel for the last 55 years. However, it no longer reflects current doctrine and is therefore losing credibility and effectiveness. In its current configuration, the EWD is limited to a single training scenario since the display's ship models rely on a static pulley system to show movement and the terrain display ashore is fixed. To address these shortfalls, this thesis first recommends the usage of the wireless communication capability within Sun's Small Programmable Object Technology (SunSPOT) to create robotic vehicles to replace the current ship models. This enables large-group visualization and situational awareness of the numerous coordinated surface maneuvers needed to support Marines as they move from ship to shore. The second recommendation is to improve visualization ashore through the creation of Extensible 3D Graphics (X3D) scenes depicting high-fidelity 3D models and enhanced 3D terrain displays for any location. This thesis shows how to create these scenes and project them from overhead in order to modernize the gymnasium-sized EWD into an amphibious wargaming table suitable for both amphibious staff training and operational planning. Complimentary use of BASE-IT projection tables and digital 3D holography can further provide smallgroup, close-up views of key battlespace locations. It is now possible to upgrade an aging training tool by implementing the technologies recommended in this thesis to support the critical training and tactical needs of the integrated Navy and Marine Corps amphibious fighting force.

# MMOG thesis: Tariq Rashid

Rashid, Tariq, *Integrating Distributed Interactive Simulations with the Project Darkstar Open-Source Massively Multiplayer Online Game (MMOG) Middleware*, Masters Thesis, Naval Postgraduate School, Monterey California, September 2009.
Co-advisor Don McGregor, second reader Amela Sadagic.



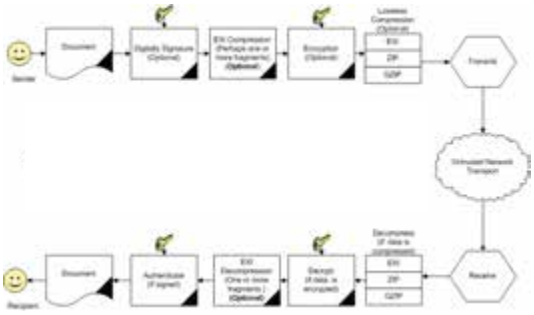

http://edocs.nps.edu/npspubs/scholarly/theses/2009/Sep/09Sep_Rashid.pdf

Abstract.

Recently, a great deal of attention has been given to the use of Massively Multiplayer Online Games (MMOGS) for both gaming and military applications. The revenue generated by MMOGs and the effect that they have on the network infrastructure has resulted in significantly more developmental resources being applied to commercial MMOG technology than for military distributed virtual (DVE) development. All DVEs share a common set of characteristics, and additional requirements exist for the interoperability of military DVEs. It is possible to exploit these similarities to take advantage of developments in the supporting technologies of commercial MMOGs. Specific capabilities of interest include scalability for large numbers of players, capacity for large amounts of network traffic, portability across operating systems, and adaptability to connect diverse codebases, network protocols, and data formats. Project Darkstar is a Sun Labs research project, which has developed an open-source middleware for MMOGs. This thesis has produced and tests a MMOG server, which interconnects heterogeneous simulators in a DVE using the Project Darkstar middleware and locally developed network gateways. The performance of the system and the character of the network traffic it generates are analyzed. Initial test results warrant further development and eventual deployment.

# XML Security Thesis: Jeff Williams

Williams, Jeffrey S., *Document-Centric XML Encryption and Authentication for Coalition Messaging*, Masters Thesis, Naval Postgraduate School, Monterey California, September 2009.



Potentially usable with DIS-XML, EXI compression

http://edocs.nps.edu/npspubs/scholarly/theses/2009/Sep/09Sep%5FWilliams.pdf

Abstract.

Different agencies and different nations are not able to securely communicate and share structured information due to differences in security policies and data formats. The current evolution of security and data policies is not solving this fundamental problem. Document-based message-centric XML security can provide satisfactory security within a diversified communications framework between traditional and nontraditional partners by utilizing existing Web standards for XML canonicalization, XML digital signature, XML compression and XML encryption. Vulnerabilities related to the exchange of cryptographic technologies are minimized by strictly adhering to open-standards technology. This approach thus resolves multi-partner trust challenges in regards to using another entity's equipment, software, or policy requirements through the proper adoption of standards-based structured data and alternative cryptographic algorithms. Exemplar results demonstrated in this thesis show that XML Security is a feasible approach for operations that include multiple agencies and coalition partners. Alternative solutions are also available using proprietary technologies, but such approaches lock participants into commercial contracts, prohibit distribution and provide suspect capabilities. Therefore, they cannot attain interagency or international acceptance. Such methods involve the use of unique or proprietary message formats with customized encryption and compression algorithms that are not available for broad scrutiny by open source communities. Closed approaches cannot gain group trust. This thesis specifically investigates XML standardization methods for various categories of unclassified data to provide secure information exchange among a wide audience, e.g. multi-agency task force or multinational coalition partners. Using an XML document-centric approach is a helpful organizing principle for this problem that provides levels of security consistent with common business practices achieved, within the constraints of the respective organizational security policies of each participant. The resulting design patterns for XML document development enhance confidentiality, integrity, and authentication commensurate with the nature of the unclassified document generated, while maintaining information objects at an appropriate level of security and acceptable level of risk.

# Discrete Event Simulation (DES)


# DES: Simkit


# DES: DisKit


# DES: Viskit


# DES: SavageStudio


back to Table of Contents

# Additional Resources


# Wireshark


Wireshark is an open-source network-protocol analyzer and debugging tool

Wireshark has a built-in decoder for DIS

- Analyze->Enabled Protocols to make sure DIS is turned on
- Select a packet
- Analyze->Decode As to view as DIS; can also set all packets sent to or from a port to be automatically decoded as DIS

Available at

- http://www.wireshark.org

web3D CONSORTIUM


TODO screenshots

## Jenkins continuous build server

NPS has stood up an online continuous integration server for regular builds

for model and software developers

- Uses open-source Jenkins under Apache.

Performs nightly compilation builds of code, error-checking validation of model content

- Xj3D open-source player and other tools
- X3D example archives, all on SourceForge

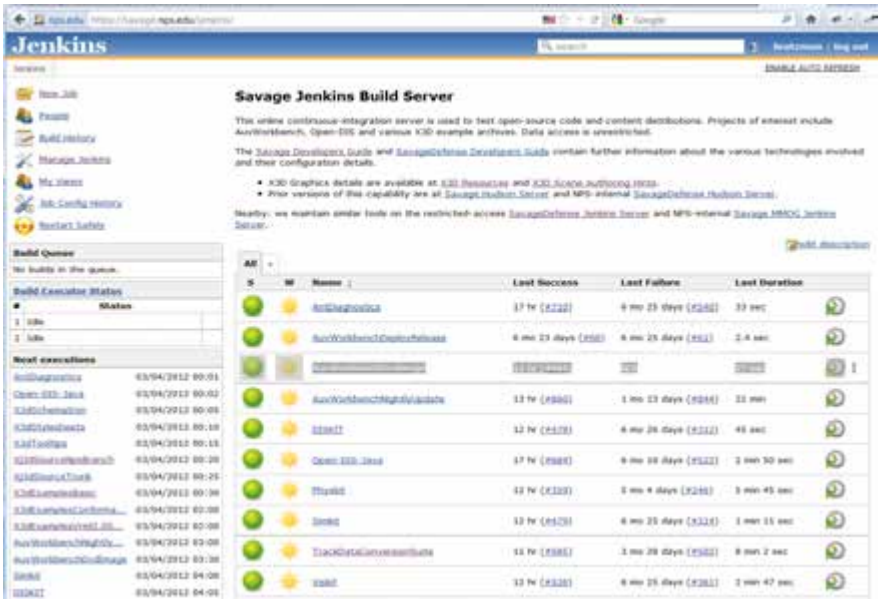
Provide further quality assurance (QA) for X3D

- https://savage.nps.edu/Savage/developers.html#Jenkins
- https://savage.nps.edu/jenkins

web3D CONSORTIUM

<X3D>

# Jenkins continuous-build testing




https://savage.nps.edu/jenkins/job/Open-DIS-Java

back to Table of Contents

# Chapter Summary


# Chapter Summary

X3D DIS component allows networking of shared entity state for positioning objects, entity management, collision detection, fire/detonate projectiles, and propagation/receipt of signals

Multiple technical challenges are steadily being addressed, simple scalability is the primary goal

Ongoing work is building repeatable, royalty-free results available for broad use on the Web


# Suggested exercises

Test, adapt example scenes provided in archive

Create (or adapt) a simple Java, C++ or Javascript program to send PDUs

Monitor PDUs being sent from an existing DIS simulation program using X3D model, X3D-Edit

(we're getting closer...) join a virtual environment


# Future work: lots!


# TODO #1: Planned projects

- Geospatial coordinates in X3D Specification
- DisEntityManager scenario: working example
- Upgrade Open-DIS to DIS 2012 changes
  - Status?
- Recording, playing back DIS PDU packets to/from a MySQL database
- Provide updated support for VRML/X3D examples in Mark Pullen's online course Networked Virtual Environments (NVEs)


# TODO #2

- Open-DIS ports to Javascript: WebSockets and WebRTC available, improve documentation
- Add Open-DIS to X3DOM: testing in progress
  - Breakthrough moment after years of work
- Upgrade MV3500 Networked Simulation as an online distributed-learning course


# TODO #3: X3D-Edit

- Open-DIS server stream-relay capabilities
  - Simplify, automate server-to-server (s2s) bridging
  - Embedded in X3D-Edit for local server creation
  - Bundle over XMPP chat for broader routing
- Autogenerate Java, Javascript enumerations using Enumeration Byte Value (EBV) .xml
  - Publish classes in Open-DIS archive (check current)
  - Bundle in X3D-Edit panes, online documentation
- DIS data capture, distillation as smoothed interpolators for offline/archived playback
  - Track recording and playback for any entity

# TODO #3: other NPS tools

- X3DOM interoperability
  - X3dToX3dom.xslt stylesheet support
  - Tooltips and quality assurance (QA) testing
  - Publish series of examples
- Integrate, document visualization tools use
  - AUV Workbench mission publication, replay
  - Viskit playback control
  - SavageStudio scenario authoring
- Update past work to meet current research
  - Dave Laflam thesis on signals visualization
  - Tom Miller thesis on grouped humanoid animation

# TODO #4: other tools

- Codebase repeatability and interoperability
  - Wireshark usage and examples with DIS
  - AMIE virtual-world bridge connections
  - Test and Training Enabling Architecture (TENA) interoperability
  - Add DIS support to major X3D players: BS Contact, InstantReality, perhaps other codebases
  - X-Plane usage and examples with DIS
- Revisit scalable MMOG game server concepts
  - compare/contrast to SISO WebLVC work
  - Is another MMOG codebase really needed, or might peer-to-peer (p2p) approaches prove sufficient?

# TODO #5: and more, here we go!

- Important thesis work now available
  - Compare compression techniques using XML-based Efficient XML Interchange (EXI)
  - Encryption and signature of streams, PDUs
  - Security considerations of Web-based DIS

# Partnership, sponsor opportunities

- DIS-XML encoding proposed and tested as an addition to the DIS standard
- DIS-HLA bridge to an open-source RTI
- Consider integration with Test and Training Enabling Architecture (TENA)
- Your project here?


back to Table of Contents

# References


# References 1

*X3D: Extensible 3D Graphics for Web Authors* by Don Brutzman and Leonard Daly, Morgan Kaufmann Publishers, April 2007, 468 pages.


- http://x3dGraphics.com

X3D Resources and X3D Basic Examples Archive

- http://www.web3d.org/x3d/content/examples/X3dResources.html
- http://www.web3d.org/x3d/content/examples/Basic/DistributedInteractiveSimulation

web|3D CONSORTIUM

<X3D>

# References 2

X3D-Edit Authoring Tool

- https://savage.nps.edu/X3D-Edit

X3D Scene Authoring Hints

- http://x3dgraphics.com/examples/X3dSceneAuthoringHints.html

X3D Graphics Specification

- http://www.web3d.org/x3d/specifications
- Also available as help pages within X3D-Edit


# References 3

## SISO Digital Library for DIS

- SISO-REF-020-2008: DIS Plain and Simple Guide
- The Complete DIS PDU Guide (also database)
- Variable Parameter Record Guide (VPRG)
- DIS Version Difference Guide
- SISO-REF-010-2010-RC1 Enumeration and Bit Encoded Values for DIS (500 pages)

# References 4

## SISO Digital Library for DIS

- DIS Product Development Group
- DIS Find it Fast Guide
- DIS XML and Databases

# References 5


- Ralph Toms and Cameron Kellough, *Guidelines for the Development of Efficient Algorithms for Spatial Operations*, SRI International, 20 February 2006
- Ralph Toms and Paul Birkel, *Choosing a Coordinate Framework for Simulations*, 1999
- Paul Birkel and Ralph Toms, SEDRIS *Spatial Reference Model (SRM) tutorial*, 2004



# References 6


- David L. Neyland, *Virtual Combat: A Guide To Distributed Interactive Simulation*, Stackpole Books, 1997.


- Sandeep Singhal and Michael Zyda, *Networked virtual environments: design and implementation*, ACM Press/Addison-Wesley, 1999. Online course available.


- Anthony Steed and Manuel Fradinho Oliveira, *Building Networked Games and Virtual Environments*, Morgan Kaufman, 2009.

# Contact

**Don Brutzman**

*brutzman@nps.edu*

*http://faculty.nps.edu/brutzman*

Code USW/Br, Naval Postgraduate School
Monterey California 93943-5000 USA
1.831.656.2149 voice


## Creative Commons open-source license

http://creativecommons.org/licenses/by-nc-sa/3.0


web|3D CONSORTIUM

<X3D>

Attribution-Noncommercial-Share Alike 3.0 Unported

You are free:

* to Share — to copy, distribute and transmit the work

* to Remix — to adapt the work

Under the following conditions:

* Attribution. You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).

Attribute this work: What does "Attribute this work" mean?

The page you came from contained embedded licensing metadata, including how the creator wishes to be attributed for re-use. You can use the HTML here to cite the work. Doing so will also include metadata on your page so that others can find the original work as well.

* Noncommercial. You may not use this work for commercial purposes.

* Share Alike. If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.

* For any reuse or distribution, you must make clear to others the license terms of this work. The best way to do this is with a link to this web page.

* Any of the above conditions can be waived if you get permission from the copyright holder.

* Nothing in this license impairs or restricts the author's moral rights.

# Open-source license
## for X3D-Edit software and X3D example scenes

http://www.web3d.org/x3d/content/examples/license.html



Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the names of the Naval Postgraduate School (NPS) Modeling Virtual Environments and Simulation (MOVES) Institute nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

License available at

http://www.web3d.org/x3d/content/examples/license.txt

http://www.web3d.org/x3d/content/examples/license.html

Good references on open source:


Andrew M. St. Laurent, *Understanding Open Source and Free Software Licensing*, O'Reilly Publishing, Sebastopol California, August 2004. http://oreilly.com/catalog/9780596005818/index.html


Herz, J. C., Mark Lucas, John Scott, *Open Technology Development: Roadmap Plan*, Deputy Under Secretary of Defense for Advanced Systems and Concepts, Washington DC, April 2006. http://handle.dtic.mil/100.2/ADA450769