Army Research Laboratory

Lethality/Vulnerability Server Functional Description and Interface Control Document for MATREX V0.5

by Geoffrey C. Sauerborn and Tranese S. Christy

ARL-MR-0582 **March 2004**

NOTICES

Disclaimers

Army Research Laboratory
Aberdeen Proving Ground, MD 21005-5066

ARL-MR-0582 **March 2004**

Lethality/Vulnerability Server Functional Description and Interface Control Document for MATREX V0.5

Geoffrey C. Sauerborn and Tranese S. Christy
Weapons and Materials Research Directorate, ARL

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* March 2004	**2. REPORT TYPE** Final	**3. DATES COVERED (From - To)** June 2003 to December 2003
4. TITLE AND SUBTITLE Lethality/Vulnerability Server Functional Description and Interface Control Document for MATREX V0.5		**5a. CONTRACT NUMBER** **5b. GRANT NUMBER** **5c. PROGRAM ELEMENT NUMBER**
6. AUTHOR(S) Geoffrey C. Sauerborn; Tranese S. Christy (both of ARL)		**5d. PROJECT NUMBER** 621618H80 **5e. TASK NUMBER** **5f. WORK UNIT NUMBER**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) U.S. Army Research Laboratory Weapons & Materials Research Directorate Aberdeen Proving Ground, MD 21005-5066		**8. PERFORMING ORGANIZATION REPORT NUMBER** ARL-MR-0582
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		**10. SPONSOR/MONITOR'S ACRONYM(S)** **11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This report describes the lethality/vulnerability server's interface control. That is it describes how other high level architecture (HLA) simulations can interact with the lethality/vulnerability server. The use of war game simulations to conduct weapon systems analysis has an established history. With the advancement of computer and network capabilities, it became practical and efficient to segment simulated systems across various computer platforms. This advancement in distributed simulation also brought with it new dilemmas such as how to ensure a "fair fight". For example, when a set of heterogeneous simulations is brought together, each simulation may treat the data (vulnerability data, terrain, or other synthetic environment representations) with subtle differences that create an unfair advantage for some simulated weapons. Distributing applications also increases the risk of having incorrect or stale data configurations on one or more of the systems. The lethality/vulnerability server is a tool that was designed to overcome some of these obstacles and help ensure a valid weapon system assessment. It allows diverse applications to draw from the same vulnerability description data set during a simulation run. The server can increase simulation preparation efficiency because configuring vulnerability damage is done once for all serviced applications. The lethality server currently delivers data descriptions in terms of standard fully damaged "mobility," "firepower," and "catastrophic" states. With relatively minor modifications, the server can be expanded to deliver partial (or degraded) damage and other types of data. This document is a functional description of the server with an emphasis on its HLA interface for the Modeling Architecture for Technology and Research EXperimentation (MATREX) V0.5 release, January 2004.

15. SUBJECT TERMS

distributed simulation; lethality; simulation interface; vulnerability

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Geoffrey C. Sauerborn
a. REPORT Unclassified	**b. ABSTRACT** Unclassified	**c. THIS PAGE** Unclassified	UL	20	**19b. TELEPHONE NUMBER** (*Include area code*) 410-278-8657

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18

Contents

List of Figures

List of Tables

1. Introduction

The lethality/vulnerability server (the lethality server) is a tool allowing diverse applications to draw from the same vulnerability description data set during a simulation run. Configuration of the vulnerability damage is done once and for all serviced applications. The lethality server currently delivers data descriptions in terms of standard fully damaged "mobility," "firepower," and "catastrophic" states. With relatively minor modifications, the server can be expanded to deliver partial (or degraded) damage and other types of data.

This document is a functional description of the server with an emphasis on its high level architecture (HLA) interface for the Modeling Architecture for Technology and Research EXperimentation (MATREX) V0.5 release (*1,2,3*).

2. MATREX Architecture

A short introduction to the MATREX architecture will help provide a better context for the lethality server. Distributed simulations have had great success within the Department of Defense (DoD) community, particularly in the area of training (for example, joining together flight simulators or armored vehicle simulators augmented by semi-automated simulated opposition forces). Among the many success stories are SIMNET (simulation network) developed in the early 1980's and the ensuing Close Combat Tactical Trainer (CCTT) delivered in the 1990's (*4,5*). Traditionally, these distributed simulation environments have been defined by joining simulators (or models) that are stand-alone applications in their own right. These simulations provide state revisions for the benefit of remote applications. MATREX builds upon object-oriented techniques and the foundation provided by the Joint Virtual Battlespace (JVB) (*6*). In such approaches, the synthetic environment segments the simulation space into a set of composible services that may be called upon when necessary (as portrayed in figure 1; figure 2 displays an overall logical architecture of the MATREX concept). Some advantages to the component approach shown in figure 1 have already been mentioned with regard to the lethality server, namely, streamlining data configuration and ensuring that all simulations are accessing the same validated source data, algorithms, or component.

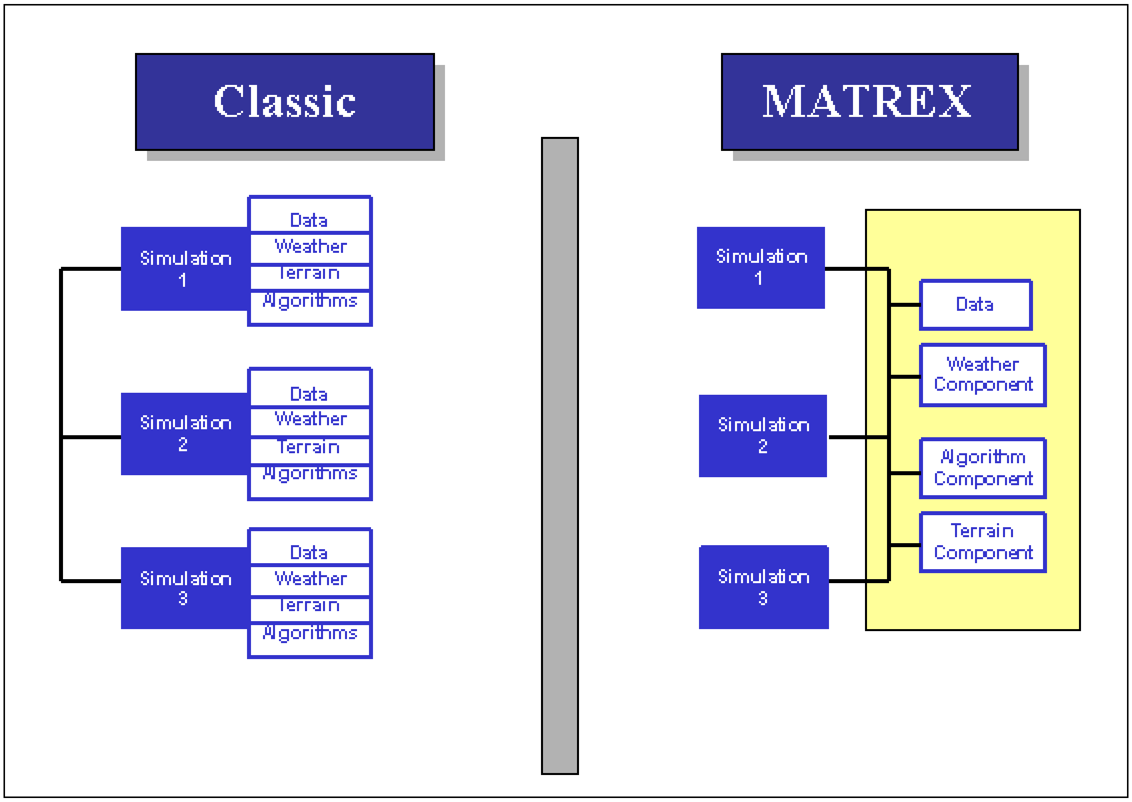

Figure 1. MATREX component architecture approach (*1, p 30*).

MATREX V0.5 was developed and delivered to the Future Combat System of Systems (FCS) lead systems integrator in December 2003. It incorporated a number of simulation services, data servers, and simulation components to include the lethality server. Some of these are shown in their logical organization within figure 2. This figure represents conceptual components and some potential interfaces (such as test and training enabling architecture (*7*), shared memory, etc.) that do not portray the actual set of delivered articles. This figure is shown merely to provide a better understanding of the MATREX approach; other sources can provide a deeper understanding (*1*). The delivered MATREX V0.5 run time components were orchestrated into an HLA federation object model and heavily tested. The lethality server's HLA object model interface to the MATREX environment is this report's main topic and is addressed now.

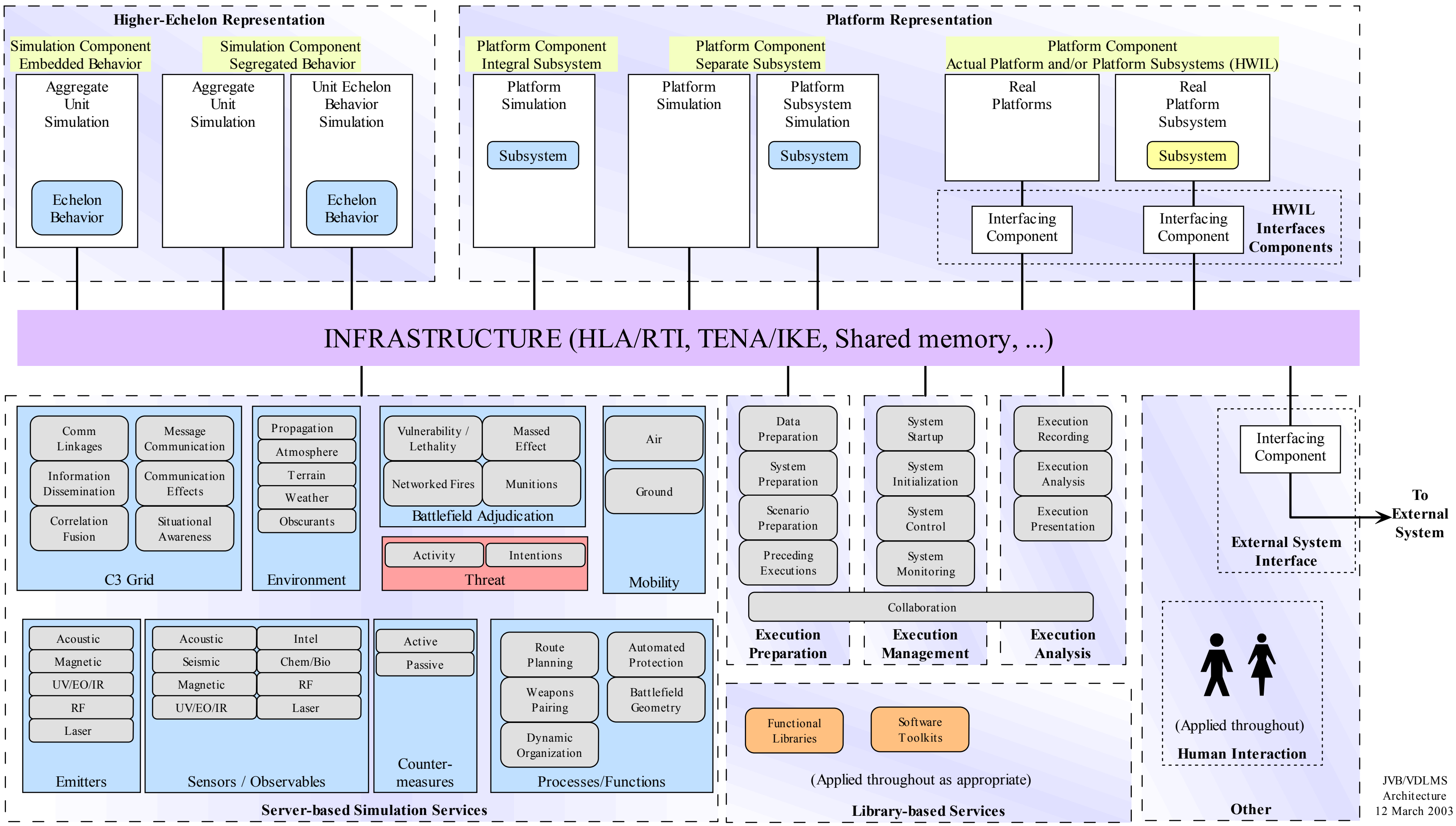

Figure 2. MATREX logical architecture (*1, p 36*).

3. HLA Interface

This section describes the HLA interface of the U.S. Army Research Laboratory's (ARL) table lookup lethality server integrated into the MATREX science and technology objective (STO). Descriptions of the lethality server's object model components are given. Additional MATREX HLA objects are referenced but are not explained in detail.

The simulation interactions in the MATREX federation object model (FOM) are grouped into a class structure. There is a "root service", SimulationService, from which lethality services and other service-related interactions can be derived. The lethality server's interaction class structure is shown in table 1.

Table 1. Lethality server interaction class structure.

Interaction1	Interaction2	Interaction3
SimulationService (IR)	Lethality (IR)	LethalityQuery (IR)
		LethalityResponse (IR)

This lethality class structure corresponds to a query-response mechanism which is depicted in figure 3.

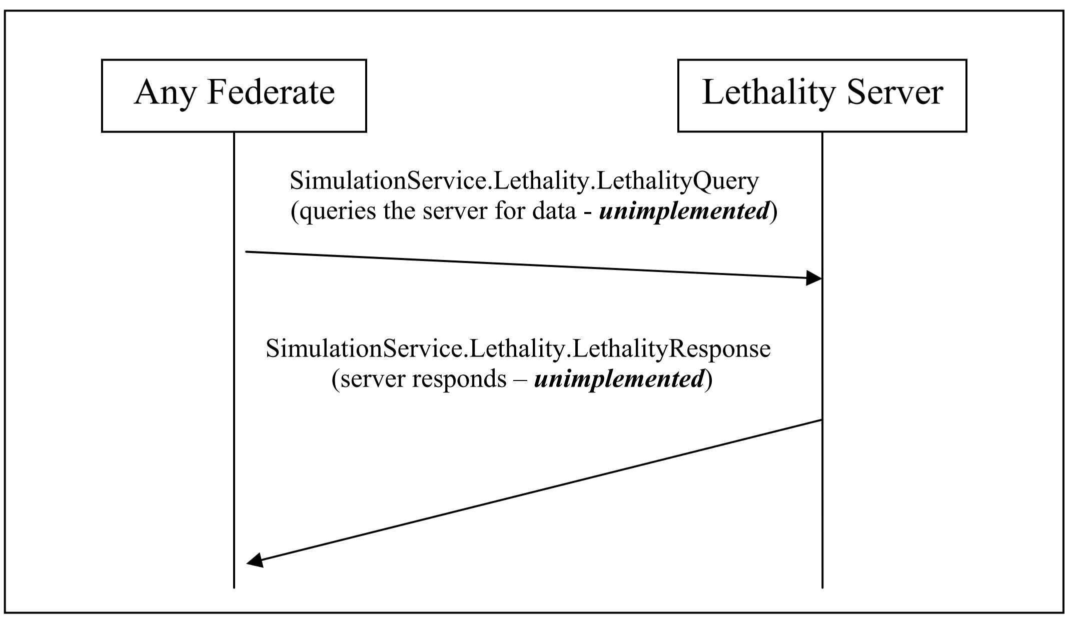

Figure 3. Query and response interaction.

A federate can query the lethality server for damage data. The query is made when a federate issues a SimulationService.Lethality.LethalityQuery interaction. The lethality server responds to the query by finding the results based on the passed parameters and issues a SimulationService. Lethality.LethalityResponse interaction. Tables 2 and 3 list and explain the parameters of the LethalityQuery and LethalityResponse interactions. ***This query-response designed feature is currently not implemented in the delivered MATREX V0.5 version of the server.***

Table 2. SimulationService.Lethality.LethalityQuery interaction.

Interaction Object	Parameter	Data Type	Parameter Description
Lethality Query	EntityID	String	The identifier of the entity whose damage status is being queried.
	Querying Initial Conditions	Boolean	If TRUE, the lethality server is to return initial conditions data used for damage calculations on the queried entity.
	Querying Probability Distribution	Boolean	The Boolean to indicate that the lethality server is to return the probability distribution used in the damage calculation.

Table 3. SimulationService.Lethality.LethalityResponse interaction.

Inter-action Object	Parameter	Data Type	Parameter Description
Lethality Response	Probability Distribution MFK	Double * 5	Five doubles that comprise the probability distribution used to compute damage. (This is in "thermometer" distribution (*8*) format.) *The server does not fill this field in the MATREX V0.5 delivery; however, for V&V purposes, these data are printed on the server's console (standard output) along with the file name of the vulnerability data table.*
	Initial Conditions	String	The initial conditions used to compute damage (a human-readable list of parameters and values).
	QueryID	Federate IDCDT	Normally, this would be the ServiceID of the LethalityQuery interaction associated with this response. *However: For MATREX V0.5 since queries are not implemented, this parameter references the MunitionDetonationServiceID of a single detonation event. The rest of the parameters all are in the context of this detonation event.*
	Instantaneous Damage	Damage StatusEDT	The damage state of the entity being queried.
	ErrorFlag	Boolean	The flag indicating whether the lethality calculation was successful. If TRUE, then some error occurred and the damage calculation was ***not*** successful. This will indicate that the value of the InstantaneousDamage parameter is ***not*** valid.
	ErrorMessage	String	If an error occurred ("ErrorFlag" == TRUE), this field may contain an error message.
	ResultsFlag	Lethality ErrorEDT	The enumerated value indicating the results of the damage calculation. This field will indicate either success or the reason for failure (see table 5).
	ResultsFlag Text	String	A short text explanation of the ResultsFlag parameter.

What ***does*** the server implement? In general, the lethality server broadcasts damage resulting from some event. In the current version of the lethality server, when a MunitionDetonation interaction occurs, the server broadcasts the result damage to the target by means of a DamageReport. This is depicted in figure 4. The DamageReport interaction contains the damage information of the targeted entity. Table 4 lists and explains the parameters within the DamageReport interaction objects.

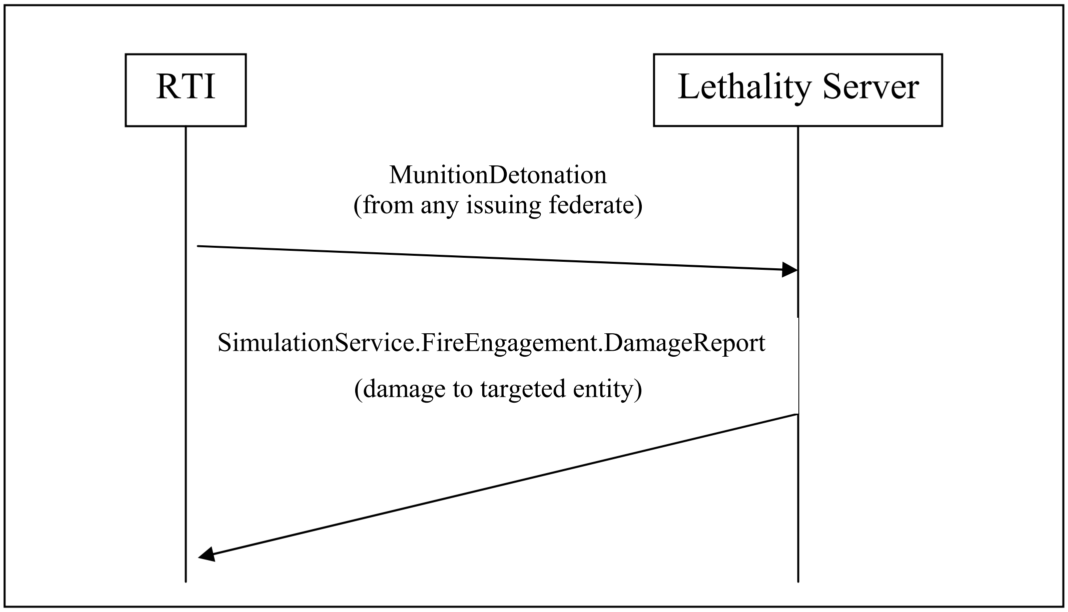

Figure 4. DamageReport interaction.

The server provides additional details for each DamageReport via the SimulationService. Lethality.LethalityResponse interaction class. This interaction ***is implemented*** and issued for every detonation event. The parameters returned by this interaction can be useful for validating the vulnerability process because they expose initial conditions and other items required to ensure that the server is functioning properly. You can turn this feature off by starting the server with the "-O AllwaysSendQueryResponse_OFF" option.

The lethality server uses munition-target interaction tables to calculate damage. The data in these tables are based on pristine targets (no previous damage). As a result, the damage data that are returned from the server are not cumulative and do not take into account the affected entity's current damage state. For instance, if as a result of a hit, a target suffered a mobility kill (M-Kill) and after a second hit, experienced a firepower kill (F-Kill), the final damage state would be an F-Kill. It is the responsibility of the entity to combine any new damage with its current damage state.

Table 4. Simulationservice.fireengagement.damagereport interaction. (Datatypes are defined in the MATREX V0.5 FOM.)

Interaction Object	Parameter	Data Type	Parameter Description
Damage Report	DamageState	DamageStatusEDT	The assessed damage state.
	Damage Location	LatLongAlt PositionCDT	The location at which the damage is assessed.
	FireDirective ServiceID	Federate IDCDT	The ServiceID of the FireDirective interaction that caused this DamageReport to occur. This is set to null if there was no preceding FireDirective interaction.
	TriggerPull ServiceID	Federate IDCDT	The ServiceID of the TriggerPull interaction that caused this DamageReport to occur. This is set to null if there was no preceding TriggerPull interaction.
	WeaponFire ServiceID	Federate IDCDT	The ServiceID of the WeaponFire interaction that caused this DamageReport to occur. This is set to null if there was no preceding WeaponFire interaction.
	Munition Detonation ServiceID	Federate IDCDT	The ServiceID of the MunitionDetonation interaction that caused this DamageReport to occur. This is set to null if there was no preceding MunitionDetonation interaction.
	ServiceID	Federate IDCDT	The unique identifier that tracks a simulation request, response, and event chain. This is inherited from the SimulationService class.
	Timestamp	Unsigned long long	The simulation time, in seconds and fractions of a second from midnight, 1 January 1900, at which the parameter values in this interaction are valid. This is inherited from the SimulationService class.
	FiringID	String	The EntityID of the object firing the munition. This is inherited from the FireEngagement class.
	TargetID	String	The EntityID of the object being fired at (if any). This is inherited from the FireEngagement class.
	MunitionID	String	The EntityID of the associated munition object (if any). This is inherited from the FireEngagement class.
	MunitionType	MunitionTypeEDT	The type of munition that is detonating. This is inherited from the FireEngagement class.
	WarheadType	WarheadTypeEDT	The type of warhead on the munition. This is inherited from the FireEngagement class.

The enumerated data type developed for the lethality server, LethalityErrorEDT is defined in table 5. LethalityErrorEDT is only used in the "ResultsFlag" parameter from the "LethalityResponse" interaction shown in table 3. The other enumerated data types listed in this document are defined in the MATREX V0.5 FOM: DamageStatusEDT, MunitionTypeEDT, and WarheadTypeEDT.

Table 5. Enumerations.

Identifier	Enumerator	Representation
LethalityErrorEDT	Success_NoError	0
	Error_Unknown	1
	Error_NoLookupTableFound	2
	Error_CorruptTable	3
	Error_MissingEnvironmentData	4

Table 6. Interactions (server subscribed).

Object	Usage
MunitionDetonation interaction	Triggers the server to broadcast the resulting damage
WeaponFire interaction	Some of this information may be used to calculate MunitionDetonation damage

4. Data Configuration

The server only issues DamageReport interactions for munition-target pairings that are specifically listed in its database.[1] This database is a flat file meta record currently situated in "$VLS_HOME/Data/Init/Matrex_V05_Meta.dat". However, this particular file name is configurable and defined in "$VLS_HOME/Data/Init/vls_db_init.ini" as described in section 5.

4.1 Configured Vehicles/Threats

The list of MATREX V0.5 targets and threats configured in the server is displayed in tables 7 and 8.

Table 7. List of vehicle types arbitrated by the lethality server (MATREX V0.5).

FOM Vehicle Type		
Platform FOM Enumeration	**OTB Platform Name**	**Platform Description**
RAH_66	vehicle_US_RAH66_AVCATT	RAH-66 Comanche
Shadow200	vehicle_US_SHADOW200_FD1	Unmanned Aerial Vehicle - Class IVa (UAV (CL IVa))
SoldierUGV	vehicle_US_LSTI_MULE	Multi-Function Utility/Logistics and Equipment (MULE) Vehicle
LSTI_HMMWV	vehicle_US_LSTI_HMMWV	HMMWV
LSTI_SUAV	vehicle_US_LSTI_SUAV	Unmanned Aerial Vehicle - Class III (UAV (CL III))
LSTI_OAV_LT	vehicle_US_LSTI_OAV_LT	Unmanned Aerial Vehicle - Class I (UAV (CL I))
LSTI_ARV_C1	vehicle_US_LSTI_ARV_C1	Armed Robotic Vehicle - Assault Variant, Light (ARV-A (L))
LSTI_BLOS	vehicle_US_LSTI_BLOS	Mounted Combat Systems (MCS)
LSTI_IC	vehicle_US_LSTI_IC	Infantry Carrier Vehicle (ICV)
LSTI_RSTA6	vehicle_US_LSTI_RSTA6	Reconnaissance and Surveillance Vehicle (R&SV)
LSTI_C2	vehicle_US_LSTI_C2	Command and Control Vehicle (C2V)
M1079	vehicle_US_LSTI_M1079	LMTV – Van
FCS_BOX_NETFIRE_FD1	vehicle_US_FCS_BOX_NETFIRES_FD1	Non-Line-of-Sight Launch System (NLOS LS)

[1] During certain circumstances, users may want the server to produce *invalid* damage assessments even if there are no vulnerability data for some munition and targets. To have the server issue a "DamageReport" even when lacking vulnerability data (i.e., produce fantastical DamageReports), start the server with the command line option: -O AllwaysSendDamage_ON If this option is turned on and there are no vulnerability data to describe an event, then the damage states issued are undefined.

LSTI_NLOS_CANNON	vehicle_US_LSTI_NLOS_CANNON	Non-Line-of-Sight Cannon (NLOS Cannon)
LSTI_NLOS_MORTAR	vehicle_US_LSTI_NLOS_MORTAR	Non-Line-of-Sight Mortar (NLOS Mortar)
BMP_2	vehicle_USSR_BMP2	BMP-2 Armored Fighting Vehicle
BRDM_2	vehicle_USSR_BRDM2	BRDM-2 Armored Reconnaissance Vehicle
BTR_80	vehicle_USSR_BTR80	BTR-80 APC
GAZ_66	vehicle_USSR_GAZ66	GAZ-66 Truck
MTLBU	vehicle_USSR_MTLBV_AVCATT	MT-Lbu (1V12) Artillery Command & Recon Vehicle
BMP_1	vehicle_USSR_BMP1	BMP-1 AFV surrogate for PT- 76
ZSU23_4	vehicle_USSR_ZSU23_4	Quad 23mm AAA Gun System
ZIL_131	vehicle_USSR_ZIL131_FDC	6x6 Truck
HOW_2S12	vehicle_USSR_2S19	152mm SP Howitzer
T_72M	Vehicle_USSR_T72M	T-72 Tank
BM_21	Vehicle_USSR_BM21	Multiple Rocket Launcher

Table 8. List of munition types arbitrated by the lethality server (MATREX V0.5).

FOM MunitionType	
Munition FOM Enumeration	**Munition Description**
munition_US_M829A2	KE M829E3 (72) Tank Main Gun
munition_US_M830A1	KE M830A1 (73) HEAT Tank Main Gun
munition_USSR_125HEAT	Tank Main Gun Round HEAT
munition_USSR_125SABOT	Tank Main Gun Round KE
munition_USSR_Sagger	AT Missile AT-3 ATGM
munition_USSR_Songster	AT Missile AT-8 ATGM
munition_USSR_Spandrel	Spandrel AT-5 ATGM
munition_US_Hellfire	Comanche AGM-114 Hellfire ATGM SAL
munition_US_Hydra70_M151	2.75in Rocket HE
munition_US_Hydra70_M261	2.75in Rocket HE Submunition
munition_US_Javelin	Javelin ATGM
munition_US_Stinger	FIM-92A Stinger AA Missile
munition_US_TOW	TOW Missile BGM-71D
munition_US_TOW2B	TOW Missile BGM-71F

4.2 Known Missing Data

All possible munition-vehicle permutations (from tables 7 and 8) are configured in the server's meta data record ("$VLS_HOME/Data/Init/Matrex_V05_Meta.dat") with the exception of the following munition-vehicle pairings shown in table 9. Most of these are "friendly" munitions attacking "friendly" targets or otherwise have a low likelihood of occurring in a normal battlefield scenario.

Table 9. Missing vulnerability data in the server's MATREX V0.5 delivery configuration. (Missing pairing marked by 'o'.)

Vehicle	Munition								
	munition_US_Hydra70_M151	munition_US_Hydra70_M261	munition_US_Hellfire_AGM_114	munition_US_TOW2B	munition_US_M829A2	munition_US_M830A1	munition_US_Javalin	munition_US_Stinger	munition_USSR_Sagger
vehicle_US_LSTI_NLOS_CANNON	o	o							
vehicle_US_LSTI_NLOS_MORTAR	o	o							
vehicle_US_LSTI_OAV_LT	o	o	o	o					
vehicle_US_LSTI_OAV_MED	o	o	o	o					
vehicle_US_LSTI_RSTA6	o	o							
vehicle_US_LSTI_SUAV			o	o	o	o	o	o	
vehicle_US_RAH66_AVCATT	o	o	o	o					
vehicle_US_SHADOW200_FD1			o	o	o	o	o	o	
vehicle_USSR_2S19									o
vehicle_USSR_BMP1									o
vehicle_USSR_BMP2									o
vehicle_USSR_BRDM2									o
vehicle_USSR_BTR80									o
vehicle_USSR_MTLBV_AVCATT									o
vehicle_USSR_T72M									o
vehicle_USSR_ZIL131_FDC									o
vehicle_USSR_ZSU23_4									o

5. Initializing the Server

This section explicitly notes server-starting options and references the formats and locations of initialization files and other preparatory information required to start the server.

5.1 Server Initialization Files

The environmental variable VLS_HOME must be set to point to the directory where the lethality server was installed. Primarily, this lets the server find its set of initialization and vulnerability data files. Initialization files are situated in the Data/Init subdirectory relative to VLS_HOME. That is, initialization data files are in the directory

```
${VLS_HOME}/Data/Init/
```

The main initialization file in this directory is **vls_db_init.ini**. This file tells the server where to find all the other initialization files. Only three initialization files are identified by **vls_db_init.ini**:

1. A Distributed Interactive Simulation (DIS) (*9*) enumeration file; these are the names and equivalent DIS numerical representation for entities, munitions, etc. More than 6,000 Institute of Electrical and Electronics Engineers (IEEE) standard enumerations are provided.

While the server is completely an HLA application, internally it retains DIS enumerations to identify vehicles and munitions. These DIS enumerations are mapped to the HLA federation's enumerations via two source code header files called "vehicles.h" and "munitions.h." (These are situated with the source code in $VLS_HOME/src/HLAMon.MATREX/ and include all vehicle and munition types from the MATREX V0.5 scenario. The vehicles and munitions shown in tables 7 and 8 are a subset of the MATREX V0.5 scenario entities.) Adding new munitions and vehicles (other than those delivered in the MATREX V0.5 scenario) will require editing of these header files and recompiling.

2. A DIS auxiliary enumeration file; intended for "additional" entities added for a particular exercise.

3. A lethality "***meta data***" file; this tells the server all it needs to know about the lethality data to be delivered upon demand. The meta data file contains meta data records.

A lethality meta data record identifies several items for the server. First, it specifies which type of V/L analysis method is used when a particular threat attacks a certain target. Then it identifies where the data are that describe the damage state outcomes (with respect to the type of vulnerability analysis method in question). Finally, the meta data record identifies which library functions are used to read and interpret the data source. (Identifying a library function allows flexibility in how data are stored and retrieved.) Vulnerability data need not be just static "look-up" tables.

Further details about vls_db_init.ini and the meta data format are presented in the document "VLS_DB_INIT(5)," Revision 0.5, March 1998 in the "$VLS_HOME/doc" directory or in appendix B from ARL report ARL-TR-1775 (*8*).

5.2 Server Command Line Options

The executable version of the server is generally in the directory:

```
${VLS_HOME}/bin/vlserver.exe
```

However, for MATREX V0.5, the execution (and main source code files) are in

```
${VLS_HOME}/src/HLAmon.MATREX/LethalityServer.exe
```

After setting the VLS_HOME environmental variable and starting the MATREX HLA run time infrastructure (RTI)[2], the server may be started. By default, the server will find the RTI and create or join a federation named "MATREXV0.5". The federation name may be changed in the initialization file "LethalityServer.HLAfc" situated in the HLAmon.MATREX starting directory. Table 10 displays other options that are selectable on the command line.

Table 10. Command line options.

Command Line Option	Result
-v	Verbose mode.
-L	Log HLA traffic. Bytes sent to and received by the HLA RTI[3] are printed in a log update file called "__updateLog.log#someNumber#." This will generate a large human-readable ASCII text file.
-O AllwaysSendDamage_ON	AllwaysSendDamage_ON: If set, the server will issue damage results, even if it has no data for the munition or target in question (i.e., it will invent a damage result and issue that). By default, this option is off.
-O AllwaysSendQueryResponse_OFF	AllwaysSendQueryResponse_OFF: If left unset, the server will issue (usually useful) debugging information in the "SimulationService.Lethality.LethalityResponse" FOM interaction class for all MunitionDetonation interactions (the associated MunitionDetonation.ServiceID will be referenced in the LethalityResponse's "QueryID" field).

5.3 Console Commands

As delivered in MATREX V0.5, the server does accept a limited number of commands that may be typed from the lethality server console after the server has been launched. Table 11 explains these commands that are activated when the indicated **key** is typed followed by the return (or enter) key.

The V0.5 executable was delivered with many internal "debug" options left activated; the result is an almost constant stream of text that scrolls across the console window. This can make it difficult to distinguish useful command results from the constant flow of debugging output and

---

[2]Instructions about starting and operating the RTI are not covered in this text. The HLA RTI used for MATREX V0.5 was RTI-NG version 1.3NGjvbV3b. You may obtain this RTI and its operating instructions by contacting the Joint Precision Strike Demonstration. Project Office, Ft Belvoir, VA or Virtual Technology Corporation, Alexandria, VA (http://www.virtc.com/).

[3]This RTI is the last version of the Defense Modeling and Simulation Office (DMSO) distributed HLA RTI (RTI-NG1.3v6) with additional revisions for the Joint Virtual Battlespace and MATREX projects.

thus limits the usefulness of console commands without recompiling a new executable version of the server with less text flow.

Table 11. Console commands.

Key	Command	Description
Q	Quit	Resign from the federation and cease execution.
S	Status	Shows the "status". Status in this case is simply the number of entities being monitored by the server. Because of internal accounting mechanisms, this count should always be two greater than the actual global entity count.
P	Print	Prints all data known about every entity being tracked as well as the most recently received or sent interactions. This command basically is a data "dump" of all HLA object model classes subscribed to and published.

6. Summary

The ARL table look-up server has undergone numerous changes since first introduced. Originally, it was a DIS-only application using transmission control protocol/internet protocol sockets for client connectivity (*8*). Later a hybrid HLA-DIS interface was added (*10*), then an all HLA interface (*11*) for the real-time platform-level reference HLA FOM (RPR-FOM) (*12*). Most recently, the server was integrated into a second HLA FOM (MATREX) requiring internal and external changes. This report described the server's MATREX objects and parameters that are the interface to the server. MATREX's composible architecture approach was also introduced.

References for initializing, starting, and executing some server console commands were provided.

7. References

1. The Joint Precision Strike Demonstration Office. *Modeling Architecture for Technology and Research EXperimentation (MATREX) System Architecture Description*, Coordination Draft, V0.8, Fort Belvoir, VA, unpublished.

2. U.S. Department of Defense. *High-Level Architecture Rules Version 1.3*, Washington, DC, 5 February 1998 (20 April 1998 Document Release). Also: "High-Level Architecture Object Model Template Version 1.3", and "High Level Architecture Interface Specification, Version 1.3".

3. Institute of Electrical & Electronics Engineers, Inc. *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules*; IEEE STD 1516-2000, 21 September 2000. Also: "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Federate Interface Specification", IEEE Std 1516.1-2000 and "IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Object Model Template (OMT) Specification" IEEE Std 1516.2-2000.

4. Lenoir, T.; Lowood, H. *Theaters of War: The Military-Entertainment Complex*, pp. 9; Stanford University, 2003. Web site http://www.stanford.edu/class/sts145/Library/Lenoir-Lowood_TheatersOfWar.pdf.

5. Fitzsimmons, E. "The Defense Modeling and Simulation Office: How It Started," *Simulation Technology Magazine*, published by the Simulation Interoperability Organization (SISO). Vol. 2 Issue 2a, June 29, 2000.

6. Mathis, T. "The Joint Virtual Battlespace (JVB) Science and Technology Objective (STO) transitions under the Modeling Architecture for Research, Technology, and Experimentation (MATREX) STO," *Simulation Technology Magazine*, Vol. 6 Issue 3, September 03, 2003. Web site http://www.sisostds.org/webletter/siso/iss_98/art_549.htm

7. U.S. Department of Defense, *TENA (Test and Training Enabling Architecture) Architecture Reference Document*, Version 2003 review edition, 4 November 2002, Foundation Initiative 2010. Web site http://www.fi2010.org/.

8. Sauerborn, G.C. *ARL Distributed Interactive Simulation (DIS) Lethality Communication Server, Volume II: User and Programmer's Manual*; ARL-TR-1775; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, February 1999.

9. Institute of Electrical & Electronics Engineers, Inc. *IEEE Standard for Distributed Interactive Simulation - Application Protocols*; IEEE STD 1278.1-1995; 21 September 1995.

10. Sauerborn, G.C. *Modifications of the Lethality Server for Initial RDEC Federation Integration*; ARL-MR-522; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, December 2001.

11. Sauerborn, G.C. *Vulnerability/Lethality Server High Level Architecture (HLA) Interface Control Document*; ARL-MR-575; U.S. Army Research Laboratory: Aberdeen Proving Ground, MD, February 2004.

12. Fischer, J.; Case, R.; Bertin, R., Eds. *Guidance, Rationale, and Interoperability Manual for the Real-time Platform Reference Federation Object Model (RPR FOM)*, Version 2.0D14v2, Simulation Interoperability Standards Organization, 11 March 2002.

NO. OF COPIES	ORGANIZATION
*	ADMINISTRATOR DEFENSE TECHNICAL INFO CTR ATTN DTIC OCA 8725 JOHN J KINGMAN RD STE 0944 FT BELVOIR VA 22060-6218 *pdf file only
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI IS R REC MGMT 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI OK TECH LIB 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL D D SMITH 2800 POWDER MILL RD ADELPHI MD 20783-1197
1	AMCOM MRDEC ATTN AMSMI RD W C MCCORKLE REDSTONE ARSENAL AL 35898-5240
1	HICKS AND ASSOCIATES ATTN G SINGLEY III 1710 GOODRICH DR STE 1300 MCLEAN VA 22102
1	CECOM SP & TERRESTRIAL COMMCTN DIV ATTN AMSEL RD ST MC M H SOICHER FT MONMOUTH NJ 07703-5203
1	US ARMY INFO SYS ENGRG CMND ATTN ASQB OTD F JENIA FT HUACHUCA AZ 85613-5300
1	US ARMY NATICK RDEC ACTING TECHNICAL DIR ATTN SSCNC T P BRANDLER NATICK MA 01760-5002
1	US ARMY RESEARCH OFC 4300 S MIAMI BLVD RSCH TRIANGLE PARK NC 27709

NO. OF COPIES	ORGANIZATION
1	US ARMY SIMULATION TRAIN & INSTRMNTN CMD ATTN J STAHL 12350 RESEARCH PARKWAY ORLANDO FL 32826-3726
1	US ARMY TANK-AUTOMOTIVE & ARMAMENTS CMD ATTN AMSTA AR TD M FISETTE BLDG 1 PICATINNY ARSENAL NJ 07806-5000
1	US ARMY TANK-AUTOMOTIVE CMD RD&E CTR ATTN AMSTA TA J CHAPIN WARREN MI 48397-5000
1	US ARMY TRAINING & DOCTRINE CMD BATTLE LAB INTEGRATION & TECH DIR ATTN ATCD B J A KLEVECZ FT MONROE VA 23651-5850
1	CDR US ARMY AVIATION RDEC CHIEF CREW ST R7D N BUCHER MS 243-4 AMES RESEARCH CENTER MOFFETT FIELD CA 94035
1	ITT INDUSTRIES ATTN M O'CONNOR 600 BLVD SOUTH STE 208 HUNTSVILLE AL 35802
1	DIR US ARL ATTN AMSRD ARL SL EP G MAREZ WHITE SANDS MISSILE RANGE NM 88002
1	DIR US ARMY TRAC ATTN ATRC WE L SOUTHARD WHITE SANDS MISSILE RANGE NM 88002
3	DIR US ARMY TRAC ATTN ATRC WEC J AGUILAR C DENNY D DURDA WHITE SANDS MISSILE RANGE NM 88002
3	CDR TARDEC ATTN AMSTA TR D M/S 207 FSCS R HALLE G SIMON WARREN MI 48397-5000
3	CDR ARDEC ATTN AMSTA AR FSS J CHU D MILLER B DAVIS PICATINNY ARSENAL NJ 07806-5000

NO. OF COPIES	ORGANIZATION
1	DEFENSE THREAT REDUCTION AGENCY ATTN SWE W ZIMMERS 6801 TELEGRAPH ROAD ALEXANDRIA VA 22310
2	JOINT VIRTUAL BATTLESPACE ATTN J McDONALD J GARCIA 10401 TOTTEN ROAD BLDG 399 SUITE 325 FT BELVOIR VA 22060-5823
1	US SBCCOM NATICK SOLDIER CTR ATTN AMSSB RSS MA (N) D TUCKER KANSAS STREET NATICK MA 01760-5020
1	HQ OPERATIONAL TEST CTR ATTN CSTE OTC MA S J HAMILL BLDG 91012 FT HOOD TX 76544-5068
1	SANDIA NATIONAL LABORATORIES ATTN: M J McDONALD PO BOX 5800 MS 1004 ALBUQUERQUE NM 87185-1004
	<u>ABERDEEN PROVING GROUND</u>
1	DIRECTOR US ARMY RSCH LABORATORY ATTN AMSRD ARL CI OK (TECH LIB) BLDG 4600
1	DIR AMSAA ATTN D JOHNSON BLDG 248
2	DIR AMSAA ATTN B BRADLEY A WONG BLDG 367
3	DIR AMSAA ATTN D HODGE P NORMAN K STEINER BLDG 392
5	US ARMY RESEARCH LABORATORY ATTN AMSRD ARL WM BF S WILKERSON G SAUERBORN (4) BLDG 390

NO. OF COPIES	ORGANIZATION
6	US ARMY RESEARCH LABORATORY ATTN AMSRD ARL SL BE L BUTLER R BOWERS C KENNEDY J ANDERSON T CHRISTY E GREENWALD BLDG 238
4	US ARMY RESEARCH LABORATORY ATTN AMSRD ARL SL BB R SANDMEYER P TANENBAUM B WARD W WINNER BLDG 328
1	US ARMY RESEARCH LABORATORY ATTN AMSRD ARL SL BE L ROACH BLDG 328
3	US ARMY RESEARCH LABORATORY ATTN AMSRD ARL CI CT G MOSS M THOMAS P JONES BLDG 321
1	US ARMY RESEARCH LABORATORY ATTN AMSRD ARL CI CT F BRUNDICK BLDG 1116A
1	DIR USARL AMSRD ARL WM W J SMITH BLDG 4600
1	DIR USARL AMSRD ARL WM BA D LYON BLDG 4600