JFT-145-1-OS.SRM-V1

30-September-94

SOFTWARE REFERENCE MANUAL

FOR THE

ORDNANCE SERVER (OS) CSCI 3

OF THE

ADA DISTRIBUTED INTERACTIVE SIMULATION (ADIS) SUPPORT SYSTEM

CONTRACT NO. N00421-92-D-0028

CDRL SEQUENCE NO: A009

Prepared for:

Naval Air Warfare Center, Aircraft Division (NAWCAD)
Systems Engineering Test Directorate (SETD)
Manned Flight Simulator (MFS)

Prepared by:

J. F. Taylor, Inc.
Rt. 235 and Maple Rd.
Lexington Park, MD 20653

Authenticated by:	Approved by:
(Contracting Agency)	(Contractor)
(Date)	(Date)

TABLE OF CONTENTS

µWHAT IS THE ORDNANCE SERVER?....................1
What is needed to use the Ordnance Server?....................1
What do you need to know to use the Ordnance Server?....................1
HOW DOES THE ORDNANCE SERVER WORK?....................1
An Overview....................1
More Specific Points....................1
Special Features....................2
Additional Fly-Out Models....................2
Additional Guidance Methods....................2
Articulated Parameters....................2
Configuration Files....................2
Continual Updates....................2
Variable Timeslice....................3
HOW DO YOU USE THE ORDNANCE SERVER?....................3
How do you bring up the Ordnance Server?....................3
How do you configure the Ordnance Server?....................3
How do you control the Ordnance Server?....................3
HOW DO YOU MODIFY THE ORDNANCE SERVER?....................4
How do you add status codes and error messages?....................4
How do you add a fly-out model?....................4
How do you add a guidance method?....................4
HOW DO YOU TROUBLESHOOT THE ORDNANCE SERVER?....................5
What's wrong when . . . ?....................5
APPENDIX A....................11
WHAT DOES EACH UNIT IN THE ORDNANCE SERVER DO?....................11
APPENDIX B....................17
APPENDIX C....................18

TABLES

µTable 1 Functions, Units and Packages and their Abbreviations and Correlations....................16

WHAT IS THE ORDNANCE SERVER?tc "WHAT IS THE ORDNANCE SERVER?"\L 1§

The Ada Distributed Interactive Simulation system contains three components: DIS Gateway (DG), DIS Library (DL), and the Ordnance Server (OS). The Ordnance Server operates as a client of the DIS Gateway, the DIS Interface. The DIS Library provides the Ordnance Server with routines to convert from one coordinate system to another as well as filter and sort lists of entities. All routines required for manipulating these entity lists are also located in the DIS Library.

The OS is a software package, implementing Distributed Interactive Simulation (DIS) protocols (Version 2.0.3), which provides multiple fly-out models for munitions fired from a single site or entity (referred to as the parent). The user specifies the types of munitions, various inputs regarding flight performance, the specific fly-out model for each munition, and representations within the DIS simulated world. The OS initiates a fly-out model when a Fire PDU (Protocol Data Unit) is received from a parent site or entity. The OS offers multiple fly-out models including the dead reckoning FPW model from the DIS Standard, a kinematic model for bombs and rockets, and a generic model for guided munitions which includes multiple guidance methods. These models were implemented and tested for air munitions. Although they may successfully model underwater munitions, they have not been tested with this type of munition in mind.

What is needed to use the Ordnance Server?tc "What is needed to use the Ordnance Server?"\l 2§

The Ordnance Server was developed on a Silicon Graphics Indigo R4000 running IRIX 5.2. The OS code compiles using Verdix Ada Compiler 6.2.1 and X Window System Version X11R5 and Motif 1.2. Therefore, this documentation assumes you have a working knowledge of Ada. Two versions of the Ordnance Server exist depending on the terrain database interface used; one implements the WGS 84 model and the other calls the CTDB library from ModSAF.

What do you need to know to use the Ordnance Server?tc "What do you need to know to use the Ordnance Server?"\l 2§

A working knowledge of the DIS 2.0.3 Standard and Ada is an absolute must! You should also have at least a cursory knowledge of the DIS Gateway. You should know basic flight parameters for your particular munition, but default data for a generic munition is provided in a configuration file to allow you to get up and running rather quickly.

HOW DOES THE ORDNANCE SERVER WORK?tc "HOW DOES THE ORDNANCE SERVER WORK?"\L 1§

An OverviewTC "An Overview"\l 2§

The user must initialize the Ordnance Server through the Graphical User Interface (GUI). The GUI allows the user to select which munitions are available for a particular simulation and how they will be represented, to input flight data, to select fly-out models, and to control various aspects of the simulation. This data may be saved in and loaded from configuration files to ease the initialization process.

The Ordnance Server receives event PDUs from the DIS Gateway. (Event PDUs include Simulation Management, Fire, Collision, and Detonation PDUs.) When a Fire PDU from a parent site or entity is received, the OS initializes all data for this particular munition and places the munition on a list of active munitions. After processing event PDUs, the OS updates all munitions on the active list.

For each munition on the active list, the OS updates the target position and velocity (including finding a new target when required), moves the munition forward one timeslice, generates an Entity State PDU for tracked munitions[1] and an Emission PDU for self-guided munitions, checks for collisions and finally detonates the munition when appropriate. Target data is updated only when the specified fly-out model depends on a target. For instance, the trajectory of rockets and dropped bombs are best modeled by the kinematic fly-out model. When the kinematic fly-out model is specified, units which maintain target data are simply never called. Similar logic applies to Emission PDUs which only guided munitions require.

For a better understanding of each of the units in the OS code, refer to Appendix A.

[1] Entity state PDUs are created by the OS every timeslice and sent to the DIS Gateway. The DIS Gateway puts the Entity State PDU on the network according to thresholds configured into the DG by the user.

More Specific Pointstc "More Specific Points"\l 2§

The active and frozen lists are separate doubly linked lists. Munitions may be moved from one list to the other based on Simulation Management PDUs. Placing a munition on the frozen list is a way of preserving the most recent data about the munition in the event that the munition would be resumed in the future. Munitions are placed on the active list when fired and removed when detonated.

All commonly accessed or computationally intensive parameters are stored in the munition hash table. Each entry in the hash table corresponds to a single munition and is composed of several records. These records allow similar parameters to be grouped together; for example, all the parameters related to the termination of a flight, regardless of the type of detonation that occurs, are grouped together in a record called Termination_Parameters. The hashing index is based on the Entity_ID field within the Entity_ID record. The size of the hash table and the increment used when collisions occur are user specified. The size of the hash table should be a prime number to avoid the prospect of an infinite loop during collision resolution.

Special Featurestc "Special Features"\l 2§

The Ordnance Server has a couple of features to make life a little easier. Most of them are built in so that you'll never even realize they exist, but others are here for special things. These features include variable timeslice, continual updates, configuration files, and hooks for both articulated parameters and additional fly-out models.

Additional Fly-Out Modelstc "Additional Fly-Out Models"\l 3§

The following criteria must be met by a fly-out model before it can be added to the Ordnance Server:

a) data required by the Ordnance Server must correlate with data from your fly-out model,
b) data available through the Ordnance Server (either through user inputs, internal calculations, or PDUs) must provide all the data for your model, and
c) the model must be able to return status codes.

See *How do you add a fly-out model?* for detailed instructions for utilizing this feature.

Additional Guidance Methodstc "Additional Guidance Methods"\l 3§

Adding a guidance method is even simpler than adding a fly-out model because the guidance models only need to provide two outputs: required azimuth and required elevation. The generic fly-out model uses these azimuth and elevation heading angles of the target in the munition's coordinate system to calculate the correction required to move the munition toward the target. To calculate these heading angles, the new guidance model may use any of the data stored in the munition hash table.

See *How do you add a guidance method?* for detailed instructions for utilizing this feature.

Articulated Parameterstc "Articulated Parameters"\l 3§

Currently, articulated parameters may be included in Entity State and Detonation PDUs when they are issued. However, since the Ordnance Server does not utilize articulated parameters for any purpose, articulated parameters are not read from incoming Entity State PDUs and are not included in outgoing PDUs. By modifying the GUI to allow user input of articulated parameters, they could be incorporated into the OS for future use. In addition, if you need to use articulated parameters data from incoming PDUs, the data is already available.

Configuration Filestc "Configuration Files"\l 3§

To make setting up the Ordnance Server easier, configuration files have been incorporated. These files have an easy-to-read format and can be created through the GUI by simply saving to a configuration file after typing in the parameter values. The configuration files may be loaded at a later time for other exercises and modified as needed.

See *How do you configure the Ordnance Server?* for detailed instructions for utilizing this feature.

Continual Updatestc "Continual Updates"\l 3§

Although the user must enter data prior to starting the Ordnance Server, most of the data may be modified at any time during the simulation. For example, if part of the way through an exercise you realize that the maximum range for a particular munition should have been 15000 meters rather than 10000 meters, simply change the number at the GUI screen. The new maximum range will be used for any munition fired after the change is complete. Some parameters must not be changed after the initial Run state is entered. These parameters include the Hash Table Size and the Hash Table Increment.

Variable Timeslicetc "Variable Timeslice"\l 3§

The Ordnance Server accepts a desired cycle time from the user and attempts to operate at that rate. During one cycle the OS updates all the munitions and, in the time remaining, processes as many events as possible. However, for more munitions and event activity, longer timeslices will be needed. To prevent problems with the munition fly-outs, the OS allows itself to exceed the specified timeslice in order to complete the necessary munition updates by running over into the next timeslice. Each munition is updated based on the amount of time that has elapsed since the last update. This method assures that each munition is updated with some regularity and doesn't lose time relative to the simulation. For instance, if a munition flies out for 5 seconds, only 5 seconds of simulation time should pass during the fly-out.

HOW DO YOU USE THE ORDNANCE SERVER?tc "HOW DO YOU USE THE ORDNANCE SERVER?"\L 1§

How do you bring up the Ordnance Server?tc "How do you bring up the Ordnance Server?"\l 2§

The Ordnance Server and its GUI screen can be brought up very simply. At the prompt in the directory where the executable for the OS exists (OS*), type OS <ENTER>. A window will appear with the title "ADIS X-based Ordnance Server Interface" and two selections: File and XOS. At this point the Ordnance Server will be in the Freeze mode where it is waiting for a run command or user inputs.

How do you configure the Ordnance Server?tc "How do you configure the Ordnance Server?"\l 2§

Before giving a run command to the Ordnance Server, you should input the types of munitions which are anticipated during the exercise. A default configuration file exists which can be easily modified. At the "ADIS X-based Ordnance Server Interface," click "XOS" to reveal the menu of parameters to be defined by the user. "Set Simulation Parameters" contains all parameters that affect the simulation and only need to be set once. "Set Ordnance Parameters" and "Set General Parameters" contain parameters which need to be defined for each munition to be included in the simulation. These munition specific parameters may be saved in a separate configuration file for each munition to allow ease of reuse in future simulations. Then, for each simulation, only the munitions which are needed could be loaded into the OS in a few steps.

After entering parameters at any screen, the parameters are not passed from the GUI to the OS until "Apply" is selected. Then, to change any selection, modify the value and reselect "Apply." Once the parameters have been applied, then you may save them to the configuration file by selecting "Save Current Data in Configuration File" under "File." Only after "Apply" is selected may a configuration file be created because the configuration file is created from within the OS, not the GUI. Therefore, the values displayed by the GUI may not be the values within the OS.

To reuse a configuration file, select "Open Configuration File" under "File." Enter or select the name of the configuration file. If necessary, make any changes, then select "Apply" to pass these values from the GUI to the OS.

At any point during the simulation, you may change any munition parameter by modifying the value and selecting "Apply." Munition parameters are determined at launch; therefore, the new value will be assigned to any munitions fired after the change is made but will not affect any munitions already in flight.

How do you control the Ordnance Server?tc "How do you control the Ordnance Server?"\l 2§

At the OS's GUI screen under Set Simulation Parameters, the Simulation State selections appear along with the other simulation parameters. The five choices allow the user to start or resume, to freeze (pause), to halt, to reset the OS, or to process a single step. After selecting a simulation state, the user must press “Apply.” Any state may be selected at any time. The simulation state affects the entire simulation; however, in response to Simulation Management PDUs, the OS will freeze or resume single entities as requested.

To begin running an exercise, you must select “Run” followed by “Apply.” If you need to pause the exercise, select “Freeze” and “Apply.” Freeze allows Simulation Management PDUs to be processed but ignores other events and does not update any munitions. To resume running the exercise, select “Run” and “Apply” again. If you need to start over for any reason, you may select “Reset” and “Apply” at any point. All munitions will be eliminated from the Ordnance Server and the user inputs will maintain their current values.

The Single Step mode, designed for testing, allows you to initiate one cycle of processing. When the cycle is complete, the OS freezes processing until the user makes another selection. When you wish to continue, you may continue processing one cycle at a time by selecting “Single Step” and “Apply” again.

By selecting “Halt,” the user causes the OS to shut down all client/server interactions with the DG. The OS GUI will close itself and the OS will exit.

HOW DO YOU MODIFY THE ORDNANCE SERVER?TC "HOW DO YOU MODIFY THE ORDNANCE SERVER?"\L 1§

Some of areas of the Ordnance Server which you may find yourself wanting to modify include adding status codes and error messages for any code you may which to incorporate, adding fly-out models and/or guidance methods, and modifying configuration files. Below are detailed instructions on how to make these modifications and therefore make the Ordnance Server even better for your purposes.

How do you add status codes and error messages?tc "How do you add status codes and error messages?"\l 2§

Your model is expected to return a status code so calling routines may determine whether any returned parameters are valid or whether the requested operation was performed successfully. To maintain the current error handling, add status codes (enumerations) as needed to your model and to OS_Status_.ada, and a suitable error message for each enumeration into OS_Error_Messages_.ada. Copy the format already in place for the other error messages; each error message should appear in quotes and be immediately preceded by a "+" with no space between them. The "+" has been overloaded to account for the variable length of the message.

"Status" should be an out-only passed parameter for each of your procedures. Initialize "Status" immediately after the "begin" statement to be equal to "OS_Status.SUCCESS." In the event of an error, set "Status" to the new status code enumeration and perform any other error handling as necessary.

The error handling in the calling routine may be more important than the error handling within your routine. Remember, any section of code which calls a routine needs to check the returning status and report and handle any errors that exist. The error may be reported by calling Report_Errors (ERR_Report_Errors.ada) within the Errors package. With this in mind, if your routine calls any other routines, your routine should complete all these steps.

How do you add a fly-out model?tc "How do you add a fly-out model?"\l 2§

Before attempting to add a fly-out model, make certain the information available from the user will be sufficient for your model. See Appendix B for a listing of data available within the Ordnance Server. Then make certain your fly-out model can provide all the parameters required to issue the necessary PDUs. Your model will be expected to update certain other parameters as well. See Appendix C for a listing of all of these output parameters.

Now you are ready to make the actual modifications to the Ordnance Server. First, add an enumeration to the FLY_OUT_MODEL_IDENTIFIER in OS_Data_Types_.ada. The name should begin with "FOM_" like the enumerations which already exist. The enumeration will automatically appear with the other choices on the GUI screen after everything has been recompiled.

If your model requires target data, the Ordnance Server has to know to update the target location. In Update_Munition (MUN_Update_Munition.ada), a case statement separates the fly-out models requiring target data from those that don't. Simply search for the phrase "require target data" in the file. The first branch is the "Fly-out models which require target data"; the second branch is the "Fly-out models which do not require target data." Add your fly-out model identifier to the appropriate case statement branch.

Finally, the Ordnance Server needs to make a call to your fly-out model, or all these modifications will be useless. The fly-out model calls are made in Move_Munition (FOM_Move_Munition.ada). Simply add a "when" branch for your fly-out model identifier which includes the call to your model and status checking upon return from your model. If you do not add a branch to the case statement for your fly-out model, the kinematic model will be called because the "when OTHERS" branch will be utilized. As part of the status checking, remember to add an exception for your fly-out model and either provide your own error handling or add your exception to some of the exception handling which already exists. The latter will be easier and will make certain all steps are taken to completely handle the exception.

At this point, recompile the Ordnance Server and the OS GUI and give it a try.

How do you add a guidance method?tc "How do you add a guidance method?"\l 2§

A fly-out model uses a guidance method for target tracking. The only fly-out model originally provided with the Ordnance Server is the Generic_Fly_Out_Model (FOM_Generic_Fly_Out_Model.ada); therefore, these instructions will tell you how to add a guidance method to the Generic_Fly_Out_Model. Only you will know the best way to implement a guidance method into any fly-out models you may add.

The guidance method really only needs to provide the Generic_Fly_Out_Model with the azimuth and elevation heading angles of the target. To determine these angles, the guidance method may use any of the data available with the Ordnance Server (See Appendix B).

Once the guidance method is ready to be incorporated into the Ordnance Server, the enumeration in OS_Data_Types_.ada for GUIDANCE_MODEL_IDENTIFIER must be modified to include your new model. Once this enumeration is modified and the OS and the GUI are recompiled, the new model will appear on the GUI screen as a possible selection.

Then the Generic_Fly_Out_Model unit must be modified to allow a call to the new model. Simply add a branch in the case statement for your guidance model enumeration immediately followed by the call to your guidance model. For error handling, copy the error handling which immediately follows one of the original guidance methods. If an error occurs in your new model, this error handling will report the error and still allow the fly-out to continue.

At this point, recompile the Ordnance Server and the OS GUI and give it a try.

HOW DO YOU TROUBLESHOOT THE ORDNANCE SERVER?tc "HOW DO YOU TROUBLESHOOT THE ORDNANCE SERVER?"\L 1§

In the following section, "What's wrong when . . . ?," the status codes for each unit are listed in alphabetical order immediately followed by the error message that is printed to the screen and log file. Every status code begins with the abbreviation of the unit where the error occurred; every error message includes the name of the unit in which the error occurred. Following the status code and error message is a brief explanation of the error and possible causes. This section is certainly not all inclusive since operations such as assignments, if-then blocks, and case statements are usually considered error-free if they compile, but at least you have a good place to start. For units which contain primarily error-free operations, the explanation will contain "No reasonable errors likely."

Remember, errors are propagated out in many cases. If the same error appears in the listing several times with timestamps which are relatively similar, the error probably propagated out and was reported by each unit that propagated the error. For example, if an undefined error occurs in Get_Entity_State_Data (GESD_ERROR), the error is reported by Beam_Rider_Guidance (as GESD_ERROR) and propagated to Generic_Fly_Out_Model where it is also reported (as GESD_ERROR). Therefore, try to fix the first error in a series and then test to see how many of the errors were related to the first one before getting aggravated by the thought of having three errors when you really only have one.

What's wrong when . . . ?TC "What's wrong when . . . ?"\l 2§

Activate_Munition

AM_ERROR

An undefined error occurred in Activate_Munition

The most likely cause would be something wrong with allocating memory for the Munition_Data_Pointer.

Add_Related_Entity_Data

ARED_ERROR

An undefined error occurred in Add_Related_Entity_Data

The most likely cause would be something wrong with allocating memory for the new General Parameters record.

Beam_Rider_Guidance

BRG_ERROR

An undefined error occurred in Beam_Rider_Guidance

If the velocity of the munition is too slow to approach the target and stay within the radar beam, then the desired heading of the munition begins to exceed 90 degrees and the calculations blow up. In this case, when the error is detected and reported, an alternate method of guidance is used and the Beam Rider method is attempted again the next timeslice. The Beam Rider method is continually tried because if the munition is still being propelled, it could gain enough speed to begin closing on the target.

Check_for_Detonation

CFD_ERROR

An undefined error occurred in Check_for_Detonation

No reasonable errors likely.

Check_for_Parent_Illumination

CFPI_ERROR

An undefined error occurred in Check_for_Parent_Illumination

No reasonable errors likely.

Collision_Guidance

CG_ERROR

An undefined error occurred in Collision_Guidance

If the velocity of the munition is slower than the velocity of the target, then the desired heading of the munition begins to exceed 90 degrees and the calculations blow up. In this case, when the error is detected and reported, an alternate method of guidance is used and the Collision method is attempted again the next timeslice. The Collision method is continually tried because if the munition is still being propelled, it could gain enough speed to begin closing on the target.

Clear_List

CL_ERROR

An undefined error occurred in Clear_List

The most likely cause would be something wrong with deallocating memory for the Munition_Data_Pointer or the entire linked list entry.

Cancel_Timer

CT_ERROR

An undefined error occurred in Cancel_Timer

No reasonable errors likely.

CT_SETITIMER_FAILED_ERROR

An error occurred when Cancel_Timer called System_ITimer.SetITimer

An error occurred within the system call to SetITimer.

Determine_Detonation_Result

DDR_ERROR

An undefined error occurred in Determine_Detonation_Result

No reasonable errors likely.

Detonate_Due_to_Error

DDTE_ERROR

An undefined error occurred in Detonate_Due_to_Error

No reasonable errors likely.

DRA_FPW_Fly_Out_Model

DFFOM_ERROR

An undefined error occurred in DRA_FPW_Fly_Out_Model

No reasonable errors likely.

DIS Gateway

DG_ERROR

An error occurred in a DG routine called by the OS

Most calls to DG routines would return an error only if something catastrophic occurred within the DG or to the client connection to the DG. However, if Get_Entity_State_Data attempts to get information about an entity that does not exist, this error would be raised.

DIS Library

DL_ERROR

An error occurred in a DL routine called by the OS

Most likely something failed inside a list manipulation routine.

Deactivate_Munition

DM_ERROR

An undefined error occurred in Deactivate_Munition

The most likely cause would be something wrong with deallocating memory for the Munition_Data_Pointer.

Find_Closest_Entities

FCE_ERROR

An undefined error occurred in Find_Closest_Entities

No reasonable errors likely.

Find_Entity_Data_By_ID

FEDBID_ERROR

An undefined error occurred in Find_Entity_Data_By_ID

No reasonable errors likely.

Find_G

FG_ERROR

An undefined error occurred in Find_G

No reasonable errors likely.

Freeze_Munition

FM_ERROR

An undefined error occurred in Freeze_Munition

No reasonable errors likely.

Find_Related_Entity_Data

FRED_ERROR

An undefined error occurred in Find_Related_Entity_Data

No reasonable errors likely.

FRED_TYPE_DNE_ERROR

Entity type does not exist in list searched by Find_Related_Entity_Data

Neither the entity type specified nor a more generic type was found among the entity types defined by the user.

Get_Events

GE_ERROR

An undefined error occurred in Get_Events

The most likely cause would be something wrong with deallocating memory for the Event PDU_Pointer.

Get_Entity_State_Data

GESD_ERROR

An undefined error occurred in Get_Entity_State_Data

No reasonable errors likely.

Generic_Fly_Out_Model

GFOM_ERROR

An undefined error occurred in Generic_Fly_Out_Model

No reasonable errors likely.

Get_Height_Above_Terrain

GHAT_ERROR

An undefined error occurred in Get_Height_Above_Terrain

Unless you are running the ModSAF version and the ModSAF call fails, no reasonable errors are likely.

Issue_Acknowledge_PDU

IAPDU_ERROR

An undefined error occurred in Issue_Acknowledge_PDU

No reasonable errors likely.

Issue_Collision_PDU

ICPDU_ERROR

An undefined error occurred in Issue_Collision_PDU

No reasonable errors likely.

Issue_Detonation_PDU

IDPDU_ERROR

An undefined error occurred in Issue_Detonation_PDU

No reasonable errors likely.

Issue_Emission_PDU

IEPDU_ERROR

An undefined error occurred in Issue_Emission_PDU

No reasonable errors likely.

Issue_Entity_State_PDU

IESPDU_ERROR

An undefined error occurred in Issue_Entity_State_PDU

No reasonable errors likely.

Instantiate_Fly_Out_Model

IFOM_ERROR

An undefined error occurred in Instantiate_Fly_Out_Model

No reasonable errors likely.

Instantiate_Munition

IM_ERROR

An undefined error occurred in Instantiate_Munition

No reasonable errors likely.

Initialize_Network_Parameters

INP_ERROR

An undefined error occurred in Initialize_Network_Parameters

No reasonable errors likely.

Initialize_Simulation

IS_ERROR

An undefined error occurred in Initialize_Simulation

If you are running the ModSAF version, an error may have occurred trying to set up the connection to ModSAF; otherwise, no reasonable errors are likely.

Kinematic_Fly_Out_Model

KFOM_ERROR

An undefined error occurred in Kinematic_Fly_Out_Model

No reasonable errors likely.

Load_Configuration_File

LCF_CANNOT_OPEN_FILE_ERROR

Specified file in Load_Configuration_File could not be opened

The most likely cause for this error is a misspelling of the filename or an incorrect path.

LCF_ERROR

An undefined error occurred in Load_Configuration_File

No reasonable errors likely.

LCF_INCOMPLETE_LINE_IN_CONFIG_FILE_ERROR

An incomplete line exists in the configuration file being read by Load_Configuration_File

There is a line in the configuration file that does not contain a keyword at the beginning, is missing an equal sign, or does not have a value following the equal sign. The end of the keyword is considered to be where there is a space, tab or equal sign. Immediately following the equal sign the unit looks for a value. Spaces can appear between the equal sign and the value, but not tabs.

Link_Munition

LM_ERROR

An undefined error occurred in Link_Munition

The most likely cause would be something wrong with allocating memory for the entire linked list entry.

Modify_Entity_Hashing_Index

MEHI_ADD_ENTITY_ERROR

A call was made to Modify_Entity_Hashing_Index to add an entity to the munition hash table, but the entity was already in the hash table

Somehow a munition was fired with an entity id matching an already active munition.

MEHI_ERROR

An undefined error occurred in Modify_Entity_Hashing_Index

No reasonable errors likely.

MEHI_INFINITE_LOOP_ERROR

An infinite loop was detected while searching for a hashing index in Modify_Entity_Hashing_Index

A non-prime size was selected for the hash table and is a multiple of the hash table increment or else the entire hash table is full.

Map_Interface

MI_ERROR

An undefined error occurred in Map_Interface

No reasonable errors likely.

Move_Munition

MM_ERROR

An undefined error occurred in Move_Munition

No reasonable errors likely.

OS (Ordnance Server)

OS_ERROR

An undefined error occurred in OS (Ordnance Server)

No reasonable errors likely.

OS_Start_GUI

OSSGUI_ERROR

An undefined error occurred in OS_Start_GUI

No reasonable errors likely.

OSSGUI_EXECVE_ERROR

An error occurred when OS_Start_GUI called System_Exec.ExecVE.

A new program could not be loaded into the current process space.

Process_Configuration_Data

PCD_ERROR

An undefined error occurred in Process_Configuration_Data

If the value of a parameters was not of the expected type (i.e. an integer instead of a float), then this error would be set.

PCD_STRING_NOT_KEYWORD_ERROR

String in configuration file is not identified as a keyword by Process_Configuration_Data

The keyword string was most likely misspelled.

Process_Collision_PDU

PCPDU_ERROR

An undefined error occurred in Process_Collision_PDU

No reasonable errors likely.

Process_Detonation_PDU

PDPDU_ERROR

An undefined error occurred in Process_Detonation_PDU

No reasonable errors likely.

Process_Fire_PDU

PFPDU_ERROR

An undefined error occurred in Process_Fire_PDU

No reasonable errors likely.

Pursuit_Guidance

PG_ERROR

An undefined error occurred in Pursuit_Guidance

No reasonable errors likely.

Process_Remove_Entity_PDU

PREPDU_ERROR

An undefined error occurred in Process_Remove_Entity_PDU

No reasonable errors likely.

Process_Stop_Freeze_Entity_PDU

PSFEPDU_ERROR

An undefined error occurred in Process_Stop_Freeze_Entity_PDU

No reasonable errors likely.

Process_Stop_Freeze_Simulation_PDU

PSFSPDU_ERROR

An undefined error occurred in Process_Stop_Freeze_Simulation_PDU

No reasonable errors likely.

Process_Sim_Mgmt_PDU

PSMPDU_ERROR

An undefined error occurred in Process_Sim_Mgmt_PDU

No reasonable errors likely.

Process_Start_Resume_Entity_PDU

PSREPDU_ERROR

An undefined error occurred in Process_Start_Resume_Entity_PDU

No reasonable errors likely.

Process_Start_Resume_Simulation_PDU

PSRSPDU_ERROR

An undefined error occurred in Process_Start_Resume_Simulation_PDU

No reasonable errors likely.

Report_Error

RE_OVERFLOW_ERROR

The error was not placed on the queue because the queue is full in Report_Error

The error buffer is full and the incoming error could not be placed on the buffer. The GUI is simply behind in displaying errors.

Remove_Entity_By_Hashing_Index

REBHI_ERROR

An undefined error occurred in Remove_Entity_By_Hashing_Index

No reasonable errors likely.

Resume_Munition

RM_ERROR

An undefined error occurred in Resume_Munition

No reasonable errors likely.

Save_Configuration_File

SCF_ERROR

An undefined error occurred in Save_Configuration_File

No reasonable errors likely.

Search_for_Target

SFT_ERROR

An undefined error occurred in Search_for_Target

No reasonable errors likely.

Set_Timer

ST_ERROR

An undefined error occurred in Set_Timer

No reasonable errors likely.

ST_SETITIMER_FAILED_ERROR

An error occurred when Set_Timer called System_ITimer.SetITimer

An error occurred within the system call to SetITimer.

Unmap_Interface

UI_ERROR

An undefined error occurred in Unmap_Interface

No reasonable errors likely.

Unlink_Munition

UM_ERROR

An undefined error occurred in Unlink_Munition

The most likely cause would be something wrong with deallocating memory for the Munition_Data_Pointer or the entire linked list entry.

UM_MUNITION_NOT_FOUND_ERROR

The munition identified is not found on the specified list to be unlinked in Unlink_Munition

The munition either does not exist or is on the other list (Active vs. Frozen).

Update_GUI_Display

UGUID_ERROR

An undefined error occurred in Update_GUI_Display

No reasonable errors likely.

Update_Munition

UPM_ERROR

An undefined error occurred in Update_Munition

No reasonable errors likely.

UPM_OVERRUN

The timeslice was exceeded in Update_Munition, but processing is continuing smoothly

A variable timeslice is implemented, but if this error message occurs continuously, the cycle time should be lengthened.

Update_Target

UT_ERROR

An undefined error occurred in Update_Target

No reasonable errors likely.

APPENDIX Atc "APPENDIX A"\l 1§

WHAT DOES EACH UNIT IN THE ORDNANCE SERVER DO?tc "WHAT DOES EACH UNIT IN THE ORDNANCE SERVER DO?"\L 2§

Here is a break down of the Ordnance Server into packages and units. First the package name along with the filename of its spec and body is followed by the package's purpose. The "(_)" indicates a spec and body both exist; the spec filename uses an underscore and the body does not. In the event that only a spec or a body exists, its filename will be listed with or without the underscore as appropriate. Following the package information, the filename of each unit within the package, as well as its purpose, is included. Since some units are not part of a package, these independent units along with independent functions are mentioned first.

Independent units

OS OS.ada

The OS unit controls operation of the Ordnance Server based on the operation mode of the Ordnance Server (i.e. RUN, FREEZE, etc.) and on whether active munitions exist. Processing within the Ordnance Server is split between accepting incoming events and updating munitions.

Initialize Simulation (IS) Initialize_Simulation.ada

The IS unit logs the OS into the DG, sets up the hash table, loads the default configuration file and initiates the OS GUI.

OS_Start_GUI (OSGUI) OS_Start_GUI.ada

The OSGUI unit sets up an environment and spawns the OS GUI as a new process.

Independent functions

Is_Parent Is_Parent.ada

The IS_Parent function determines whether the entity ID provided corresponds to a parent entity of munitions within the OS.

Number_of_Articulation_Parameters Number_of_Articulated_Parts.ada

The Number_of_Articulation_Parameters functions returns the number of articulated parts.

Active_Frozen_Lists (AFL) Active_Frozen_Lists(_).ada

The AFL package maintains the active and frozen munition lists for the Ordnance Server. These doubly linked lists contain munition data records of each munition other the control of the OS. Each munition data record contains the Entity ID, Entity Type, and the Hashing Index of a munition.

Activate_Munition (AM) AFL_Activate_Munition.ada

The AM unit instantiates a new munition and places the munition on the active munition list.

Clear_List (CL) AFL_Clear_List.ada

The CL unit eliminates all munitions from the specified munition list.

Deactivate_Munition (DM) AFL_Deactivate_Munition.ada

The DM unit eliminates a munition such that it is no longer part of the simulation which includes requesting its removal from the munition hash table.

Find_Entity_Data_By_ID (FEDBID) AFL_Find_Entity_Data_By_ID.ada

The FEDBID unit loops through all the munitions on the specified list until the specified munition is found and then returns a pointer to the munition's data.

Freeze_Munition (FM) AFL_Freeze_Munition.ada

The FM unit places a munition data record on the frozen munition list, after removing it from the active munition list. This action causes processing of the munition to stop, but the data will be saved in case the munition is resumed in the future.

Link_Munition (LM) AFL_Link_Munition.ada

The LM unit links a munition data record to the top of the specified list.

Resume_Munition (RM) AFL_Resume_Munition.ada

The RM unit places a munition data record on the active munition list after removing it from the frozen munition list so that processing of the munition will resume.

Unlink_Munition (UM) AFL_Unlink_Munition.ada

The UM unit unlinks a linked list entry record from the specified list and returns a pointer to the munition data record which it contains.

Detonation_Event (DE) Detonation_Event(_).ada

The DE package determines when and what type of detonation occurs for each munition and initiates the issuing of the Detonation PDU and the deactivation of the munition.

Check_for_Detonation (CFD) DE_Check_for_Detonation.ada

The CFD unit performs tests to determine whether a detonation should occur and then initiates the detonation if one is required. Any condition which would ignite a fuse is cause for detonation as well as exceeding maximum range.

Determine_Detonation_Result (DDR) DE_Determine_Detonation_Result.ada

The DDR unit determines the result of a detonation and makes the necessary calls to detonate the munition (issue the Detonation PDU and deactivates the munition).

Errors (ERR) Errors(_).ada

The ERR package provides error handling in the event the munition must be removed from the simulation due to an error as well as error reporting and logging.

Detonate_Due_to_Error (DDTE) ERR_Detonate_Due_to_Error.ada

The DDTE unit attempts to detonate and deactivate a munition in the event of an error in order to satisfy the DIS requirement that every fire event have a corresponding detonation event.

Get_Error (GE) ERR_Get_Error.ada

The GE unit retrieves the next error, if one exists, from the error buffer. Although this unit exists in the Errors package, it is only called by the OS's GUI.

Report_Errors (RE) ERR_Report_Errors.ada

The RE unit reports an error message to the error buffer to be displayed to the user and writes a copy of the error to an error log if these capabilities are requested.

Fly_Out_Model (FOM) Fly_Out_Model(_).ada

The FOM package moves each munition along its trajectory by implementing the fly-out model specified for the munition and the specified guidance method if one is required.

Beam_Rider_Guidance (BRG) FOM_Beam_Rider_Guidance.ada

The BRG unit provides a guidance method where the munition proceeds toward the target based on information about the parent's, target's and munition's position and velocity. This method assumes that the munition needs to remain on the line between the parent and the expected target's position in order to pick up the parent's illumination of the target.

Collision_Guidance (CG) FOM_Collision_Guidance.ada

The CG unit provides a guidance method where the munition proceeds toward the position the target is expected to occupy.

DRA_FPW_Fly_Out_Model (DFFOM) FOM_DRA_FPW_Fly_Out_Model.ada

The DFFOM unit provides a fly-out model based on the FPW Dead Reckoning Algorithm defined in the DIS Standard.

Generic_Fly_Out_Model (GFOM) FOM_Generic_Fly_Out_Model.ada

The GFOM unit provides a generic fly-out model for all types of guided munitions.

Instantiate_Fly_Out_Model (IFOM) FOM_Instantiate_Fly_Out_Model.ada

The IFOM unit applies the Fly-Out Model ID of the corresponding munition type to the particular munition being instantiated and initializes data needed for the fly-out of the munition.

Kinematic_Fly_Out_Model (KFOM) FOM_Kinematic_Fly_Out_Model.ada

The KFOM unit provides a fly-out model based on the standard rectilinear equations of motion. This model is best suited for dropped bombs and rockets (unguided munitions).

Move_Munition (MM) FOM_Move_Munition.ada

The MM unit invokes the assigned fly-out model to advance the munition's position.

Pursuit_Guidance (PG) FOM_Pursuit_Guidance.ada

The PG unit provides a pursuit guidance model in which the munition attempts to fly to where the target currently is.

G_Utilities (GU) G_Utilities_.ada

The GU package allows for the manipulation of the g acceleration vector.

Find_G (FG) G_Utilities_.ada

The FG unit determines the components of a g acceleration vector along the World Coordinate axes given the vector along which g is acting.

Gateway_Interface (GI) Gateway_Interface(_).ada

The GI packages provides access to the DIS Gateway which allows DIS PDUs to be sent to or received from the network. This package also allows the Ordnance Server to request the most recent entity state data relative to a particular entity

Get_Entity_State_Data (GESD) GI_Get_Entity_State_Data.ada

The GESD unit requests, from the DIS Gateway, a pointer to data about a specified entity.

Get_Events (GE) GI_Get_Events.ada

The GE unit allows incoming events (from the DIS Gateway) to be processed by calling the appropriate unit for the incoming event.

Initialize_Network_Parameters (INP) GI_Initialize_Network_Parameters.ada

The INP unit places data related to PDUs in a hash table by copying the relevant data from the Fire PDU when a Fire PDU from a parent entity or site is received.

Issue_Acknowledge_PDU (IAPDU) GI_Issue_Acknowledge_PDU.ada

The IAPDU unit generates an Acknowledge PDU which it then passes to the DIS Gateway to send to the network.

Issue_Collision_PDU (ICPDU) GI_Issue_Collision_PDU.ada

The ICPDU unit generates a Collision PDU which it then passes to the DIS Gateway to send to the network.

Issue_Detonation_PDU (IDPDU) GI_Issue_Detonation_PDU.ada

The IDPDU unit generates a Detonation PDU which it then passes to the DIS Gateway to send to the network.

Issue_Emission_PDU (IEPDU) GI_Issue_Emission_PDU.ada

The IEPDU unit generates an Emission PDU which it then passes to the DIS Gateway to send to the network.

Issue_Entity_State_PDU (IESPDU) GI_Issue_Entity_State_PDU.ada

The IEPDU unit generates an Entity State PDU which it then passes to the DIS Gateway to send to the network.

Process_Collision_PDU (PCPDU) GI_Process_Collision_PDU.ada

The PCPDU unit processes incoming Collision PDUs to incorporate the effects of the collisions.

Process_Detonation_PDU (PDPDU) GI_Process_Detonation_PDU.ada

The PDPDU unit processes incoming Detonation PDUs to incorporate the effects of the detonations.

Process_Fire_PDU (PFPDU) GI_Process_Fire_PDU.ada

The PFPDU unit processes incoming Fire PDUs to determine whether a munition should be activated. If the munition was fired by a parent site or entity, the PFPDU unit initiates the initialization and activation of the munition.

Process_Remove_Entity_PDU (PREPDU) GI_Process_Remove_Entity_PDU.ada

The PREPDU unit removes an entity from the simulation in response to a Simulation Management PDU.

Process_Sim_Mgmt_PDU (PSMPDU) GI_Process_Sim_Mgmt_PDU.ada

The PSMPDU unit processes Simulation Management PDUs allowing the state of the simulation or the state of a single entity in the simulation to change.

Process_Start_Resume_Entity_PDU (PSREPDU) GI_Process_Start_Resume_Entity_PDU.ada

The PSREPDU unit restores a munition to the active munition list in response to a Simulation Management PDU.

Process_Start_Resume_Simulation_PDU (PSRSPDU) GI_Process_Start_Resume_Simulation_PDU.ada

The PSRSPDU unit sets the simulation state to RUN in response to a Simulation Management PDU.

Process_Stop_Freeze_Entity_PDU (PSFEPDU) GI_Process_Stop_Freeze_Entity_PDU.ada

The PSFEPDU unit moves a munition to the frozen munition list in response to a Simulation Management PDU.

Process_Stop_Freeze_Simulation_PDU (PSFSPDU) GI_Process_Stop_Freeze_Simulation_PDU.ada

The PSFSPDU unit sets the simulation state to FREEZE in response to a Simulation Management PDU.

Munition (MUN) Munition(_).ada

The MUN package controls al aspects of the munition's flight from the moment fired through detonation.

Add_Related_Entity_Data (ARED) MUN_Add_Related_Entity_Data.ada

The ARED unit updates information for a munition in the General Parameters list in the event of the user modifying this data.

Find_Related_Entity_Data (FRED) MUN_Find_Related_Entity_Data.ada

The FRED unit searches through the General Parameters list to find the closest match, if one exists, to the entity type being instantiated.

Instantiate_Munition (IM) MUN_Instantiate_Munition.ada

The IM unit initiates processes to define all parameters for the specified munition based on user-selected data and firing data.

Update_GUI_Display (UGUID) MUN_Update_GUI_Display.ada

The UGUID unit processes requests from the GUI to display the next or previous entry in the General Parameters list, or to add the currently displayed GUI information to the list.

Update_Munition (UM) MUN_Update_Munition.ada

The UPM unit manages all activity of each munition for the current timeslice.

OS_Configuration_Files (OSCF) OS_Configuration_Files(_).ada

The OSCF package controls the loading and saving of configuration files.

Load_Configuration_Files OSCF_Load_Configuration_Files.ada

The LCF unit reads the contents of a configuration file, discards comments, and parses other lines into keywords and values for processing.

Process_Configuration_Data OSCF_Process_Configuration_Data.ada

The PCD unit updates munition parameters and adds entries to the General Parameters list based on configuration keywords and values.

Save_Configuration_Files OSCF_Save_Configuration_Files.ada

The SCF unit writes the current settings of all OS parameters and all entries in the General Parameters list out to a file.

OS_GUI (OSG) OS_GUI(_).ada

The OSG package manages the shared memory interface.

Map_Interface OS_GUI.ada

The MI unit creates the shared memory interface.

Unmap_Interface OS_GUI.ada

The UI unit frees the shared memory interface.

OS_Hash_Table_Support (OSHTS) OS_Hash_Table_Support(_).ada

The OSHTS package adds, finds or removes entities from the munition hash table.

Modify_Entity_Hashing_Index (MEHI) OSHTS_Modify_Entity_Hashing_Index

The MEHI unit adds an entity to, finds an entity on or removes an entity from the munition hash table.

Remove_Entity_by_Hashing_Index (REBHI) OSHTS_Remove_Entity_by_Hashing_Index

The REBHI unit removes an entity from the munition hash table using the munition's hashing index.

OS_Timer (OST) OS_Timer(_).ada

The OST package provides timing information for the OS.

Cancel_Timer (CT) OST_Cancel_Timer.ada

The CT unit cancels the current interval timer if one is active.

Set_Timer (ST) OST_Set_Timer.ada

The ST unit specifies the duration of the time interval.

SIGALRM_Handler (SH) OST_SIGALRM_Handler.ada

The SH unit is the signal-catching routine associated with the SIGALRM signal. This signal is used internally by the OST routines to generate timing information.

Time_Remains OST_Time_Remains.ada

The Time_Remains function indicates if time remains in the interval given by the last Set_Timer call.

Target_Tracking (TT) Target_Tracking(_).ada

The TT package acquires, maintains and, when necessary, indicates loss of lock on a target for each munition active in the simulation and under the control of the Ordnance Server.

Check_for_Parent_Illumination (CFPI) TT_Check_for_Parent_Illumination.ada

The CFPI unit searches through the parent's Emission PDU to determine if the target entity is being illuminated.

Find_Closest_Entities (FCE) TT_Find_Closest_Entities.ada

The FCE unit finds the entities located in a sphere with a radius equal to the current maximum range of the munition. These entities are most likely to be targets during upcoming timeslices.

Search_for_Target (SFT) TT_Search_for_Target.ada

The SFT unit looks for a new target. The target must be within the cone of detection for the munition and within line of sight. The azimuth and elevation headings are also updated.

Update_Target (UT) TT_Update_Target.ada

The UT unit makes a call for the most recent position and velocity of the target and then determines whether this entity is still a reasonable target. If not, a call is made to find a new target.

Terrain_Database_Interface (TDI) Terrain_Database_Interface(_).ada

The TDI packages allows height above terrain and height above sea level data to be determined for the specified position of an entity. The current implementation of the TDI package interfaces with ModSAF.

Get_Height_Above_Terrain (GHAT) TDI_Get_Height_Above_Terrain.ada

The GHAT unit determines a height above terrain by interfacing with a terrain database to acquire height of the terrain and then calculating the difference between height of the terrain and height of the munition.

Table 1 Functions, Units and Packages and their Abbreviations and CorrelationsTC "Table 1 Functions, Units and Packages and their Abbreviations and Correlations"\f t§

Abbreviation	**Name**	**Package (if unit)**
	Is_Parent (Independent function)	N/A
	Number_of_Articulation_Parameters (Independent function)	N/A
	Time_Remains (Function)	OS_Timer
AFL	Active_Frozen_Lists	N/A
AM	Activate_Munition	Active_Frozen_Lists
ARED	Add_Related_Entity_Data	Munition
BRG	Beam_Rider_Guidance	Fly_Out_Model
CFD	Check_for_Detonation	Detonation_Event
CFPI	Check_for_Parent_Illumination	Target_Tracking
CG	Collision_Guidance	Fly_Out_Model
CL	Clear_List	Active_Frozen_Lists
CT	Cancel_Timer	OS_Timer
DDR	Determine_Detonation_Result	Detonation_Event
DDTE	Detonate_Due_to_Error	Errors

DE	Detonation_Event	N/A
DFFOM	DRA_FPW_Fly_Out_Model	Fly_Out_Model
DM	Deactivate_Munition	Active_Frozen_Lists
ERR	Errors	N/A
FCE	Find_Closest_Entities	Target_Tracking
FEDBID	Find_Entity_Data_By_ID	Active_Frozen_Lists
FG	Find_G	G_Utilities
FM	Freeze_Munition	Active_Frozen_Lists
FOM	Fly_Out_Model	N/A
FRED	Find_Related_Entity_Data	Munition
GE	Get_Events	Gateway_Interface
GER	Get_Error	Errors
GESD	Get_Entity_State_Data	Gateway_Interface
GFOM	Generic_Fly_Out_Model	Fly_Out_Model
GHAT	Get_Height_Above_Terrain	Terrain_Database_Interface
GI	Gateway_Interface	N/A
GU	G_Utilities	N/A
IAPDU	Issue_Acknowledge_PDU	Gateway_Interface

ICPDU	Issue_Collision_PDU	Gateway_Interface
IDPDU	Issue_Detonation_PDU	Gateway_Interface
IEPDU	Issue_Emission_PDU	Gateway_Interface
IESPDU	Issue_Entity_State_PDU	Gateway_Interface
IFOM	Instantiate_Fly_Out_Model	Fly_Out_Model
IM	Instantiate_Munition	Munition
INP	Initialize_Network_Parameters	Gateway_Interface
IS	Initialize_Simulation (Independent unit)	N/A
KFOM	Kinematic_Fly_Out_Model	Fly_Out_Model
LCF	Load_Configuration_File	OS_Configuration_Files
LM	Link_Munition	Active_Frozen_Lists
MEHI	Modify_Entity_Hashing_Index	OS_Hash_Table_Support
MM	Move_Munition	Fly_Out_Model
MUN	Munition	N/A
OS	OS (Independent control unit for the Ordnance Server)	N/A
OSCF	OS_Configuration_Files	N/A
OSHTS	OS_Hash_Table_Support	N/A
OSSGUI	OS_Start_GUI (Independent unit)	N/A

OST	OS_Timer	N/A
PCD	Process_Configuration_Data	OS_Configuration_Files
PCPDU	Process_Collision_PDU	Gateway_Interface
PDPDU	Process_Detonation_PDU	Gateway_Interface
PFPDU	Process_Fire_PDU	Gateway_Interface
PG	Pursuit_Guidance	Fly_Out_Model
PREPDU	Process_Remove_Entity	Gateway_Interface
PSFEPDU	Process_Stop_Freeze_Entity_PDU	Gateway_Interface
PSFSPDU	Process_Stop_Freeze_Simulation_PDU	Gateway_Interface
PSMPDU	Process_Sim_Mgmt_PDU	Gateway_Interface
PSREPDU	Process_Start_Resume_Entity_PDU	Gateway_Interface
PSRSPDU	Process_Start_Resume_Simulation_PDU	Gateway_Interface
RE	Report_Error	Errors
REBHI	Remove_Entity_by_Hashing_Index	OS_Hash_Table_Support
RM	Resume_Munition	Active_Frozen_Lists
SCF	Save_Configuration_File	OS_Configuration_Files
SFT	Search_for_Target	Target_Tracking
SH	SIGALRM_Handler	OS_Timer

ST	Set_Timer	OS_Timer
TDI	Terrain_Database_Interface	N/A
TT	Target_Tracking	N/A
UGUID	Update_GUI_Display	Munition
UM	Unlink_Munition	Active_Frozen_Lists
UPM	Update_Munition	Munition
UT	Update_Target	Target_Tracking

APPENDIX Btc "APPENDIX B"\l 1§

The following variables are available for additional fly-out models and guidance methods. To access these variables, which are located in the hash table, use the hashing index. See OS_Data_Types_.ada for a complete listing of the fields in these records. See the code for examples.

Network_Parameters

These parameters contain information for outgoing PDUs. Some of these parameters are included in Appendix C because they must be updated by the fly-out models. Entity_Orientation is currently set to the firing entity's orientation until orientation can be incorporated into the Fire PDU. Force_ID is set to the firing entity's force ID.

Aerodynamic_Parameters

These parameters are required for fly-out models which implement target tracking or guidance methods.
Flight_Parameters

All parameters needed to fly-out a munition, including target data if needed, is located in this record. If the fly-out model uses entity coordinates, the parameters for the munition's location, velocity and velocity magnitude are all in entity coordinates and the firing data is defined. Otherwise, location, velocity, and firing data are left empty and velocity magnitude is in world coordinates.

Termination_Parameters

To determine if the fuse ignites or detonation contains exist, the Ordnance Server uses these parameters.

Simulation_Parameters

All data required for the simulation as a whole are located in this record.

APPENDIX Ctc "APPENDIX C"\ l 1§

These parameters must be set by the fly-out models so that detonation checks can be performed and accurate PDUs can be generated.

Network_Parameters

Location_in_WorldC
Velocity_in_WorldC
Target_Entity_ID (if applicable)

Aerodynamic_Parameters

These parameters only depend on the fly-out model you implement.

Flight_Parameters

Time_in_Flight
Velocity_Magnitude